Knowage

Knowage Labs

Apr 14, 2023

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 7 . . . 8
      - 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 . . . 9
  - 2.2.7 SlimerJS requirements . . . 9
- 2.3 Software release . . . 9
- 2.4 Manual installation . . . 10
  - 2.4.1 How to populate metadata database . . . 10
  - 2.4.2 Using Tomcat . . . 10
    - 2.4.2.1 Dependencies . . . 10
    - 2.4.2.2 File system resources . . . 10
    - 2.4.2.3 Metadata database connection . . . 10
    - 2.4.2.4 Connection to business data . . . 11
    - 2.4.2.5 Environment variables definition . . . 12
    - 2.4.2.6 Applications deploy . . . 12
    - 2.4.2.7 Thread pool defintion . . . 12
    - 2.4.2.8 Advanced memory settings . . . 12
  - 2.4.3 Datasource link within the applications . . . 13
  - 2.4.4 Configuration of the metadata db dialect . . . 13
  - 2.4.5 Modification of the Quartz configuration . . . 14
    - 2.4.5.1 Clustering . . . 14

2.4.6 Logging . . . 14
2.5 Knowage CE Installer . . . 14
2.5.1 Server-side requirements . . . 15
2.5.1.1 Operating system . . . 15
2.5.1.2 Java platform . . . 15
2.5.1.3 Memory . . . 15
2.5.1.4 Disk usage . . . 15
2.5.1.5 Database . . . 15
2.5.1.6 Application server . . . 15
2.5.1.7 Proxy settings . . . 16
2.5.2 Client-side requirements . . . 16
2.5.2.1 Browser . . . 16
2.5.2.2 Proxy settings . . . 16
2.5.3 Launching . . . 16
2.5.3.1 Windows . . . 16
2.5.3.2 Linux/macOS . . . 16
2.5.4 Managing Knowage CE . . . 17
2.5.4.1 Windows . . . 17
2.5.4.2 Windows (embedded MariaDB option) . . . 17
2.5.4.3 Linux/macOS . . . 17
2.6 R installation . . . 17
2.7 Python installation . . . 19
2.7.1 JPY installation . . . 19
2.8 CAS installation . . . 21
2.8.1 Deploy of the CAS application . . . 21
2.8.2 HTTPS certificate . . . 21
2.8.3 Configuration of the HTTPS protocol for Tomcat . . . 22
2.8.4 Knowage configuration . . . 22
2.9 Advanced configuration . . . 24
2.9.1 Thread manager . . . 24
2.9.2 Cache parameters . . . 25
2.9.3 Logging . . . 26
2.9.4 Mail server . . . 27
2.9.5 Maximum file size . . . 27
2.9.6 Date format . . . 28
2.9.7 Language . . . 28
2.9.8 LDAP security connectors . . . 29

3 Administration Manual **31**
3.1 Knowage at a glance . . . 31
3.1.1 Discovering Knowage . . . 31
3.1.2 Modules . . . 31
3.1.3 Layers . . . 33
3.1.3.1 Delivery layer . . . 33
3.1.3.2 Analytical layer . . . 35
3.1.3.3 Data layer . . . 35
3.1.4 What you can do with Knowage . . . 35
3.1.4.1 Analytical and operational functionalities . . . 36
3.1.4.2 Administrative tools and cross services . . . 36
3.2 User Interface . . . 36
3.2.1 Main menu . . . 36
3.3 Configure data sources . . . 39
3.3.1 Connect to your data . . . 39
3.3.2 Big Data and NoSQL . . . 42

3.3.2.1 Hive . . . 43
3.3.2.2 Spark SQL . . . 44
3.3.2.3 Impala . . . 44
3.3.2.4 MongoDB . . . 44
3.3.2.5 Cassandra . . . 45
3.4 Behavioural Model . . . 46
3.4.1 Roles, users and attributes . . . 46
3.5 Analytical Model . . . 50
3.5.1 Repository structure and rights . . . 50
3.5.2 Menu configuration . . . 50
3.6 Operational engines . . . 54
3.6.1 ETL . . . 54
3.6.1.1 KnowageTalendEngine . . . 54
3.6.1.2 Job design . . . 54
3.6.1.3 Template building . . . 55
3.6.1.4 Creating the analytical document . . . 56
3.6.1.5 Job execution . . . 56
3.6.1.6 Job scheduling . . . 56
3.6.2 External processes . . . 56
3.6.2.1 Class definition . . . 56
3.6.2.2 Template definition . . . 58
3.7 Scheduler . . . 59
3.7.1 Create an Activity . . . 59
3.7.1.1 Save as snapshot . . . 63
3.7.1.2 Save as file . . . 64
3.7.1.3 Save as document . . . 64
3.7.1.4 Send to Java class . . . 65
3.7.1.5 Send mail . . . 67
3.7.1.5.1 Static list . . . 67
3.7.1.5.2 Dynamic list with mapping dataset . . . 67
3.7.1.5.3 Dynamic List with script . . . 69
3.7.2 Schedulation panel . . . 69
3.7.3 Scheduler Monitor . . . 70
3.8 Server Manager . . . 70
3.8.1 Tenants Management . . . 72
3.8.2 Template Management . . . 72
3.8.3 Import\Export . . . 74
3.8.3.1 Documents . . . 74
3.8.3.2 Menu . . . 78
3.8.3.3 Users . . . 78
3.8.3.4 Catalogs . . . 80
3.8.3.5 KPIs . . . 81
3.8.3.6 Analytical Drivers . . . 84
3.8.3.7 Glossary . . . 86
3.8.3.8 Catalog . . . 86
3.9 Server Settings . . . 87
3.9.1 Configuration Management . . . 88
3.9.2 Domain Management . . . 89
3.9.3 Metadata . . . 89

4 User Guide **91**
4.1 Introduction . . . 91
4.2 User Interface . . . 91
4.2.1 Preliminary information . . . 91

4.2.2 Main menu . . . 92
4.2.3 Document Browser overview . . . 95
4.3 Document Execution . . . 97
4.3.1 Parameters management . . . 97
4.3.2 Document Toolbar . . . 97
4.3.2.1 Exporters . . . 99
4.3.2.2 Business and structural metadata . . . 99
4.3.2.3 Notes . . . 101
4.3.2.4 Rate document . . . 101
4.3.2.5 Scheduled Execution . . . 101

5 Functionalities **103**
5.1 Basic Data Access . . . 103
5.1.1 Dataset . . . 104
5.1.1.1 My dataset . . . 104
5.1.1.2 CKAN integration . . . 106
5.1.1.2.1 CKAN datasets access method . . . 107
5.1.1.2.2 Export dataset . . . 107
5.1.1.2.3 Save and handle dataset . . . 107
5.1.2 Models . . . 109
5.1.3 Dataset federation . . . 109
5.2 Advanced Data Access . . . 110
5.2.1 My first dataset . . . 110
5.2.1.1 New dataset creation . . . 111
5.2.1.2 File Dataset . . . 114
5.2.1.3 Query Dataset . . . 114
5.2.1.4 Java Class Dataset . . . 116
5.2.1.5 Script . . . 116
5.2.1.6 QbE . . . 116
5.2.1.7 Custom Dataset . . . 117
5.2.1.7.1 Flat Dataset . . . 118
5.2.1.7.2 Ckan . . . 118
5.2.1.7.3 Federated . . . 119
5.2.1.7.4 Rest . . . 119
5.2.1.7.5 Big Data - NoSQL . . . 123
5.2.2 Parameters and profile attributes . . . 125
5.2.3 Further operations on a dataset . . . 126
5.2.3.1 Script option . . . 126
5.2.3.2 Transformations . . . 128
5.2.3.3 Dataset persistence . . . 130
5.2.3.4 Preview . . . 131
5.3 Behavioural Model . . . 131
5.3.1 Roles, users and attributes . . . 131
5.3.2 Analytical drivers . . . 132
5.3.2.1 Creating a List Of Value . . . 135
5.3.2.2 Parametrizing LOVs . . . 139
5.3.2.3 Creating a validation rule . . . 140
5.3.2.4 Creating an analytical driver . . . 142
5.3.2.5 Creating an analytical driver for a spatial filter . . . 145
5.3.2.6 Analytical driver's use modes . . . 145
5.3.2.7 Behavioural Model Lineage . . . 146
5.4 Analytical Document . . . 149
5.4.1 Main concepts . . . 149
5.4.1.1 Document types . . . 149

5.4.2 Register an analytical document . . . 151
5.4.2.1 Analytical documents on Server . . . 151
5.4.2.1.1 Document lifecycle . . . 155
5.4.2.1.2 Template Versioning . . . 155
5.4.2.1.3 Document Visibility . . . 155
5.4.3 Visibility rules . . . 156
5.4.4 Association with analytical drivers . . . 156
5.4.4.1 Associating a Spatial driver . . . 157
5.4.4.2 Correlation between parameters . . . 159
5.4.4.3 Correlation through LOV and drivers . . . 161
5.4.4.4 Controlled visibility . . . 161
5.4.5 Cross Navigation . . . 163
5.4.5.1 Declaration of the output parameters . . . 164
5.4.5.2 Cross navigation definition . . . 164
5.5 Chart . . . 167
5.5.1 My first Chart . . . 167
5.5.1.1 Structure . . . 171
5.5.1.2 Configuration . . . 171
5.5.2 Chart types in detail . . . 171
5.5.2.1 Traditional charts . . . 173
5.5.2.2 Scatter chart . . . 176
5.5.2.3 Sunburst chart . . . 176
5.5.2.4 Wordcloud chart . . . 179
5.5.2.5 Treemap chart . . . 179
5.5.2.6 Parallel chart . . . 180
5.5.2.7 Heatmap chart . . . 181
5.5.2.8 Chord chart . . . 184
5.5.2.9 Gauge chart . . . 184
5.5.3 A short comment on chart drill down . . . 185
5.5.4 Stand alone charts . . . 185
5.5.5 Cross Navigation . . . 189
5.6 Cockpit . . . 189
5.6.1 My first Cockpit . . . 191
5.6.1.1 Text widget . . . 192
5.6.1.2 Image widget . . . 194
5.6.1.3 Chart widget . . . 194
5.6.1.4 Table widget . . . 196
5.6.1.5 Cross Table widget . . . 203
5.6.1.6 Document section . . . 208
5.6.1.7 Selection widget . . . 210
5.6.1.8 Selector Widget . . . 211
5.6.1.9 HTML Widget . . . 211
5.6.1.10 Widget properties . . . 215
5.6.2 General configuration . . . 217
5.6.3 Data configuration . . . 218
5.6.3.1 Source . . . 218
5.6.3.1.1 Parametric sources management . . . 218
5.6.3.2 Associations . . . 220
5.6.3.3 Frequency . . . 221
5.6.3.4 Template . . . 221
5.6.4 Selections . . . 221
5.6.5 Clear cache . . . 223
5.6.6 Save . . . 223
5.6.7 Multisheet functionality . . . 223

5.7 Free Inquiry . . . 224
5.7.1 My first Query By Example . . . 224
5.7.1.1 Query design and execution . . . 225
5.7.1.1.1 Datamart Schema . . . 225
5.7.1.1.2 Calculated fields management . . . 227
5.7.1.1.3 Range management . . . 227
5.7.1.1.4 Query Editor . . . 229
5.7.1.1.5 Select Fields . . . 229
5.7.1.1.6 Filters . . . 231
5.7.1.1.7 Filters on Groups . . . 233
5.7.1.1.8 Query Preview . . . 233
5.7.1.2 Advanced QbE functionalities . . . 234
5.7.1.2.1 Spatial fields usage . . . 234
5.7.1.2.2 Temporal dimension . . . 239
5.7.1.2.2.1 The PARALLEL_YEAR function. . . . 242
5.7.1.2.2.2 The LAST_YEAR function . . . 243
5.7.1.2.2.3 The LAST_MONTH function . . . 246
5.7.1.2.2.4 The YTD function . . . 246
5.7.1.2.2.5 The MTD function . . . 249
5.7.1.2.2.6 Catalogues . . . 254
5.7.1.2.2.7 Queries catalogue and subqueries . . . 254
5.7.1.2.3 Multiple relationships . . . 255
5.7.1.2.4 Aliases and relationships . . . 257
5.8 Meta Web . . . 260
5.8.1 Metamodel creation . . . 261
5.8.1.1 Create an empty model . . . 261
5.8.1.2 Editing the metamodel . . . 263
5.8.1.3 Create a new relationship . . . 265
5.8.1.4 Create a new business class . . . 266
5.8.1.5 Table property list . . . 268
5.8.1.6 Column property list . . . 268
5.8.1.7 Generate the datamart . . . 270
5.8.1.8 Additional functions for business model . . . 270
5.9 Registry . . . 273
5.9.1 Registry features . . . 273
5.9.1.1 JPivot Registry characteristics . . . 276
5.9.2 Registry development . . . 276
5.9.2.1 JPivot Registry instance . . . 278
5.10 BIRT reporting . . . 278
5.10.1 Introduction to Knowage Studio . . . 279
5.10.1.1 Metadata definition . . . 282
5.10.1.2 Data set definition . . . 283
5.10.2 Developing a BIRT report . . . 283
5.10.2.1 Using an external Data Set . . . 289
5.10.2.2 Adding parameters to reports . . . 289
5.10.3 Download and deploy . . . 292
5.10.4 Cross Navigation for BIRT Reports . . . 293
5.11 Jasper reporting . . . 296
5.11.1 Document defintion* . . . 296
5.12 Location Intelligence . . . 300
5.12.1 Basic concepts . . . 300
5.12.2 More on GIS and Spatial Data* . . . 301
5.12.2.1 Spatial Data . . . 301
5.12.2.2 GIS . . . 302

5.12.3 Analytical document execution . . . 303
5.12.3.1 Extra functionalities . . . 307
5.12.4 Template building with GIS designer . . . 307
5.12.5 Designer sections . . . 309
5.12.5.1 Layer section . . . 309
5.12.5.2 Dataset join columns . . . 309
5.12.5.3 Indicators . . . 310
5.12.5.4 Filters . . . 310
5.12.5.5 Map menu configuration . . . 311
5.12.5.6 Layer filters . . . 311
5.12.5.7 Edit map . . . 312
5.12.6 GEOReport Engine* . . . 312
5.12.7 Template building with GIS designer for technical user* . . . 315
5.12.8 Cross navigation definition* . . . 315
5.13 SVG document visualization . . . 317
5.13.1 My first SVG Map or design . . . 317
5.13.1.1 Technical activities . . . 320
5.13.1.1.1 SVG Catalogue . . . 320
5.13.1.1.2 SVG Format . . . 320
5.13.1.1.3 Datasets definition . . . 322
5.13.1.1.4 Template building . . . 322
5.13.1.1.5 Advanced functionalities . . . 328
5.14 Data Mining . . . 328
5.14.1 Data Mining document interface . . . 329
5.14.2 Functions Catalog . . . 334
5.14.2.1 The General tab* . . . 338
5.14.2.2 The Input tab* . . . 338
5.14.2.3 The Script tab* . . . 339
5.14.2.4 The Output tab* . . . 343
5.14.3 Engine description . . . 346
5.14.3.1 Data . . . 346
5.14.3.2 Parametrization and customization . . . 346
5.14.3.3 Outputs . . . 346
5.14.4 My first data mining document* . . . 346
5.14.5 Create a new function in the Function Catalog . . . 348
5.15 Network . . . 350
5.15.1 Template . . . 351
5.15.2 An example* . . . 353
5.16 Multidimensional Analysis . . . 354
5.16.1 OLAP user manual step by step . . . 354
5.16.1.1 The filter panel . . . 356
5.16.1.2 The filter cards . . . 356
5.16.1.3 Axes panel . . . 357
5.16.1.4 Pivot table . . . 357
5.16.1.5 Side bar . . . 359
5.16.2 Functionalities . . . 365
5.16.2.1 Placing hierarchies on axes . . . 365
5.16.2.2 Swaping axes . . . 365
5.16.2.3 Selecting different hierarchies on dimension . . . 367
5.16.2.4 Slicing . . . 368
5.16.2.5 Filtering . . . 369
5.16.2.6 Drill down and drill up . . . 369
5.16.2.7 Drill through . . . 372
5.16.2.8 Refreshing model . . . 377

5.16.2.9 Showing MDX . . . 377
5.16.2.10 Sending MDX . . . 377
5.16.2.11 Showing parent members . . . 378
5.16.2.12 Hiding/showing spans . . . 378
5.16.2.13 Showing properties . . . 378
5.16.2.14 Suppressing empty colunms/rows . . . 378
5.16.2.15 Sorting . . . 378
5.16.2.16 Calculated members and sets . . . 383
5.16.3 Creation of an OLAP document* . . . 389
5.16.3.1 About the engine . . . 389
5.16.3.2 Development of an OLAP document . . . 389
5.16.3.2.1 Schema modelling . . . 390
5.16.3.2.1.1 Engine catalogue configuration . . . 391
5.16.3.2.2 OLAP template building . . . 391
5.16.3.2.3 Creating the analytical document . . . 393
5.16.3.3 OLAP Designer* . . . 393
5.16.3.3.1 Profiled access . . . 396
5.16.3.4 Cross Navigation . . . 400
5.16.3.4.1 Cross navigation on members . . . 401
5.16.3.4.2 Cross navigation from a cell of the pivot table . . . 401
5.17 What-if analysis . . . 402
5.17.1 Interface details . . . 402
5.17.2 Meta-language description . . . 404
5.17.3 What-if analysis implementation . . . 407
5.17.3.1 Workflow description* . . . 407
5.17.3.2 Schema definition* . . . 408
5.17.3.3 Changes in the mondrian schema* . . . 408
5.18 Glossary and data lineage . . . 413
5.18.1 Glossary management . . . 413
5.18.2 Glossary Usage . . . 416
5.18.3 Help Online functionality . . . 419
5.19 Key Performance Indicator . . . 421
5.19.1 KPI development . . . 421
5.19.2 Creation of a KPI document . . . 436
5.20 Scorecard . . . 439
5.20.1 Scorecard development . . . 439
5.20.2 Creation of a Scorecard document . . . 447
5.21 Alert . . . 448
5.21.1 Alert implementation . . . 451

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.

CHAPTER 1

Before starting

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.

CHAPTER 2

Installation Manual

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	6 64-bit
Windows	7

Table 2.2: Compatible environments

Compatible Environments	
Operating System	**Version**
RHEL Red Hat Enterprise	6.4
Ubuntu	16 LST,18 LST
Windows server	2012, 2008

2.2.2 Disk usage

The Knowage installation requires 2 GB of free 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 enviroment 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.

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.

```
export JAVA_HOME=<root path of the Java installation>
export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_60/
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 server

Support type	Application Server	Version
Certified	Apache Tomcat	7.0.65

Important: Roadmap

JBoss support will be available on Q4 2018

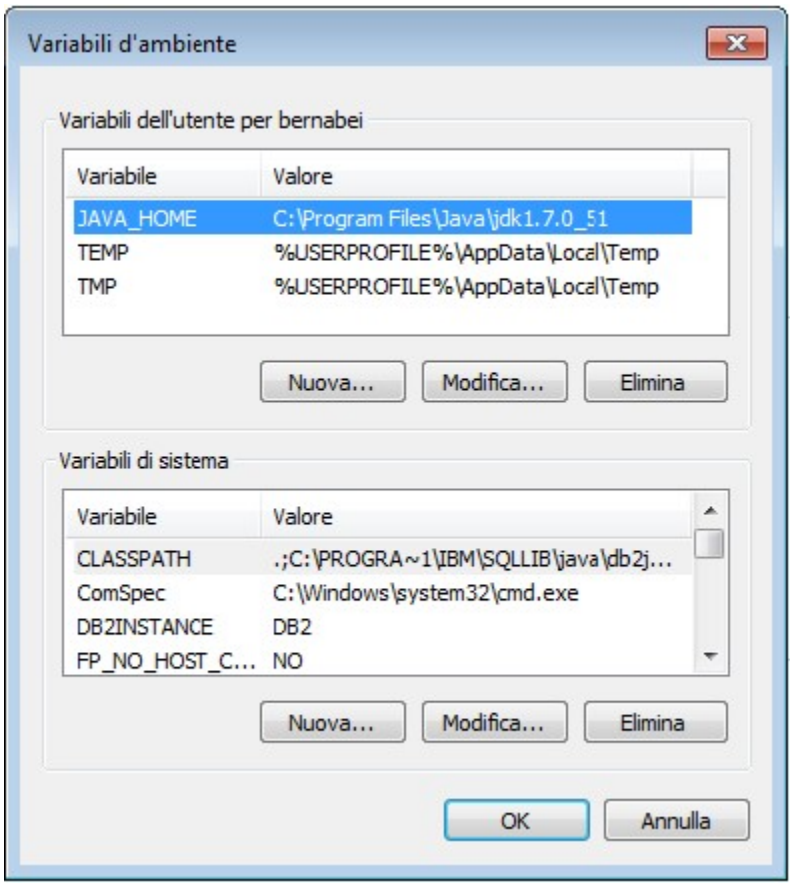


Fig. 2.1: Setting the path for the JAVA_HOME variable for Windows

For each application server installation please refer to the official documnetation.

2.2.4.1 Tomcat 7

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

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.

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

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

```
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.

```
export CATALINA_PID=<root folder of the Tomcat installation>/logs/tomcat7.
pid 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:

```
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-7.0-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	5.2, 5.5, 5.6
Certified	PostgreSQL	8.2, 9.1
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 SlimerJS requirements

Knowage includes a standalone edition of SlimerJS 0.9 to export some contents to PDF and image files. Usually SlimerJS runs out-of-the-box on Windows, but requires OS-dependent libraries on Unix-like operating systems.

- In order to fulfill all **SlimerJS requirements** please refer to its official documentation at https://docs.slimerjs.org/0.9/installation.html#requirements.
- **Xvfb** may be required on Unix-like operating systems if no suitable X display server is available; install it with package manager.

Troubleshooting missing requirements may be difficult on Unix-like operating systems. Executing SlimerJS manually with **debug option** may help to investigate further:

Listing 2.2: Executing SlimerJS with debug option via Xvfb

```
xvfb-run ./slimerjs --debug=yes
```

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 How to populate metadata database

The metadata database must contains a schema which will collect all Knowage metadata. For configuring 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 2.3: 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 module which includes the commonj library with its dependencies (if any):
  - Apache Geronimo,
  - Concurrency JSR-166,
  - Foo 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 Metadata database connection

To create a new connection, 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 and URL. We need to create two connection:

- `jdbc/knowage`: Knowage metadata

```
<Resource auth="Container"
                driverClassName="<JDBC driver>"
                name="jdbc/knowage"
                password="<password>"
                type="javax.sql.DataSource"
                url="<JDBC URL>"
                username="<user name>"
                maxActive="10"
                maxIdle="1"
                validationQuery="<validation query>"
                removeAbandoned="true"
                removeAbandonedTimeout="3600"
                logAbandoned="true"
                testOnReturn="true"
                testWhileIdle="true"
                timeBetweenEvictionRunsMillis="10000"
                minEvictableIdleTimeMillis="60000" />
```

- `jdbc/ds_cache`: Knowage cache, this must be an empty schema

```
<Resource auth="Container"
                driverClassName="<JDBC driver>"
                name="jdbc/ds_cache"
                password="<password>"
                type="javax.sql.DataSource"
                url="<JDBC URL>"
                username="<user name>"
                maxActive="10"
                maxIdle="1"
                validationQuery="<validation query>"
                removeAbandoned="true"
                removeAbandonedTimeout="3600"
                logAbandoned="true"
                testOnReturn="true"
                testWhileIdle="true"
                timeBetweenEvictionRunsMillis="10000"
                minEvictableIdleTimeMillis="60000" />
```

2.4.2.4 Connection to business data

Edit the `TOMCAT_HOME/conf/server.xml` and add the information related to the metadata database inside the `GlobalNamingResources` tag, specifying username, password, driver class name and URL.

```
<Resource auth="Container"
                driverClassName="<JDBC driver>"
                name="jdbc/dwh"
                password="<password>"
                type="javax.sql.DataSource"
                url="<JDBC URL>"
                username="<user name>"
                maxWait="-1"
                maxActive="10"
                maxIdle="1"
                validationQuery="<validation query>"
                removeAbandoned="true"
                removeAbandonedTimeout="3600"
                logAbandoned="true"
                testOnReturn="true"
                testWhileIdle="true"
                timeBetweenEvictionRunsMillis="10000"
                minEvictableIdleTimeMillis="60000"
                factory="org.apache.tomcat.jdbc.pool.DataSourceFactory" />
```

2.4.2.5 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.

Listing 2.4: Tomcat environment variables configuration.

```
<Environment name="resource_path" type="java.lang.String" value="${catalina.home}/resources"/>
<Environment name="sso_class" type="java.lang.String" value="it.eng.spagobi.services.common.JWTSsoService"/>
<Environment name="service_url" type="java.lang.String" value="http://localhost:8080/knowage"/>
<Environment name="host_url" type="java.lang.String" value="<server URL which is hosting knowage>"/>
<Environment name="hmacKey" description="HMAC key" type="java.lang.String" value="<PUT ANY RANDOM STRING HERE>
↪"/>
```

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.

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.

2.4.2.6 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.

2.4.2.7 Thread pool defintion

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.

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

2.4.2.8 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
```

2.4.3 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" />
  </Context>
```

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

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

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

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

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

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:

```
# Hsqldb delegate class
#org.quartz.jobStore.driverDelegateClass=org.quartz.impl.jdbcjobstore.HSQLDBDelegate
# Mysql delegate class org.quartz.jobStore.driverDelegateClass=org.quartz.impl.jdbcjobstore.StdJDBCDelegate
# Postgres delegate class
#org.quartz.jobStore.driverDelegateClass=org.quartz.impl.jdbcjobstore.PostgreSQLDelegate
# Oracle delegate class
#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:

```
org.quartz.jobStore.isClustered = true
org.quartz.jobStore.clusterCheckinInterval = 20000
org.quartz.scheduler.instanceId = AUTO
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/knwoageXXXXXEngine.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.

2.5 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.

2.5.1 Server-side requirements

2.5.1.1 Operating system

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

2.5.1.2 Java platform

Knowage CE Installer requires:

- JDK 1.8
- `JAVA_HOME` environment variable

2.5.1.3 Memory

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

2.5.1.4 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.

2.5.1.5 Database

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

- MySQL Server 5.5 or 5.6 or 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.

2.5.1.6 Application server

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

2.5.1.7 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.

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

2.5.2 Client-side requirements

2.5.2.1 Browser

Enable your browser to execute JavaScript.

2.5.2.2 Proxy settings

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

2.5.3 Launching

2.5.3.1 Windows

Important: The installer has to be run as administrator.

2.5.3.2 Linux/macOS

Extract the installer SH file typing the command in shell:

```
unzip Knowage-6_2_0-CE-Installer-Unix-20180719.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:

```
chmod +x Knowage-6_2_0-CE-Installer-Unix-20180719.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:

```
./Knowage-6_2_0-CE-Installer-Unix-20180719.sh
```

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

```
./Knowage-6_2_0-CE-Installer-Unix-20180719.sh -c
```

2.5.4 Managing Knowage CE

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

2.5.4.1 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

2.5.4.2 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

2.5.4.3 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

2.6 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 2.5: Commands for the rpm file

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

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:

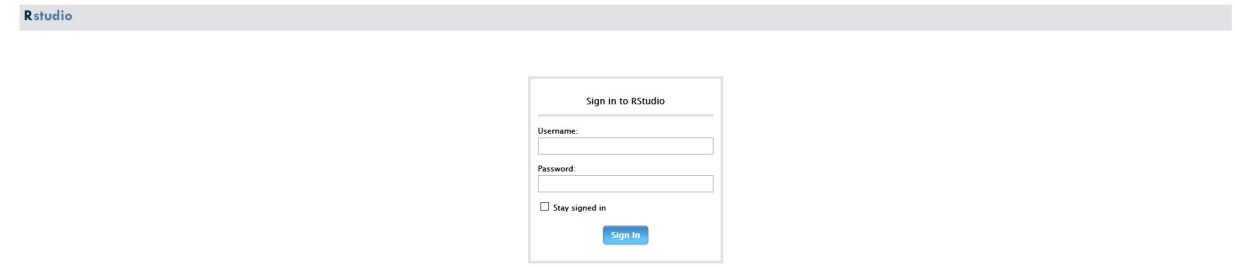
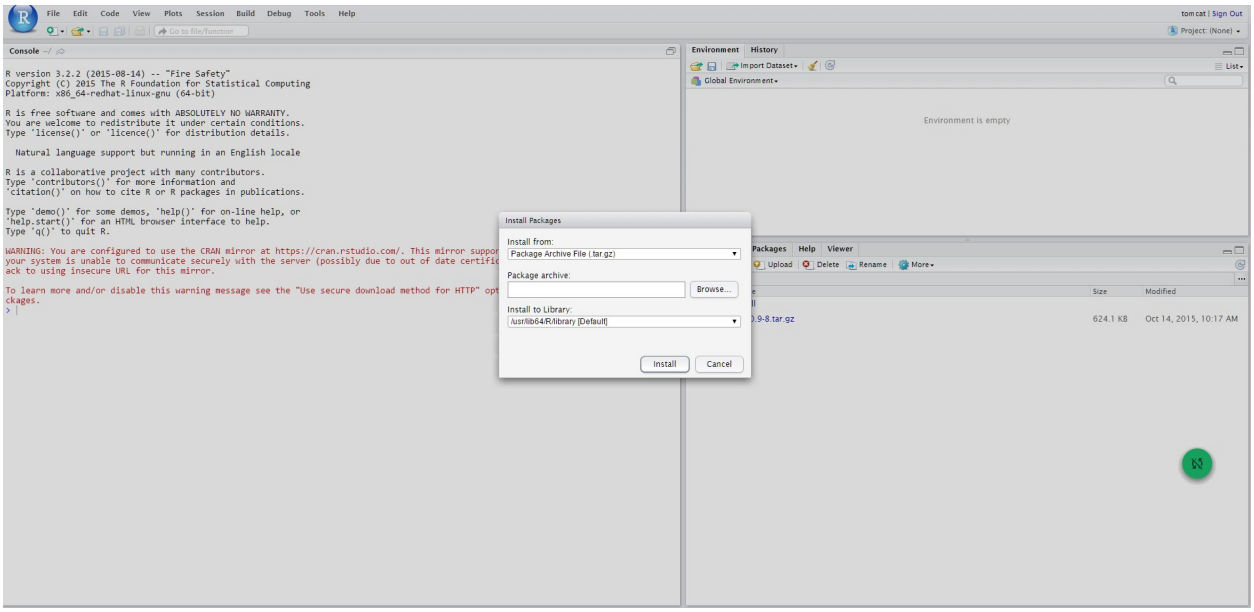
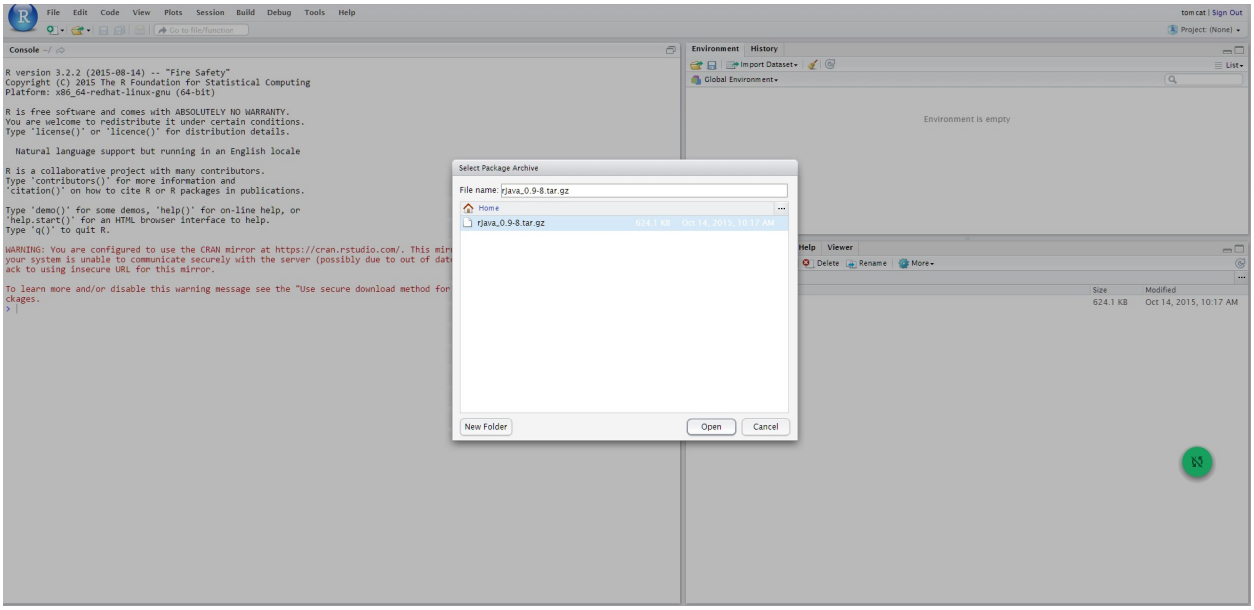
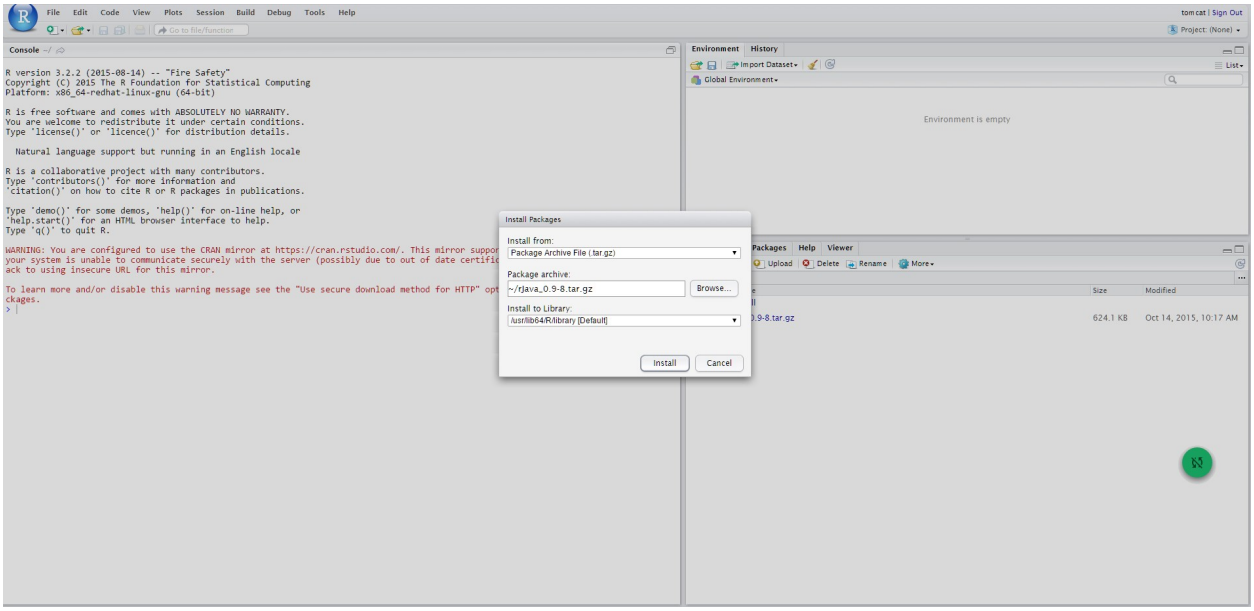


Fig. 2.2: Sign in RStudio.







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:

```
export R_HOME=/usr/lib64/R
export LD_LIBRARY_PATH=/usr/lib64/R/library/rJava/jri
```

2.7 Python installation

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).

2.7.1 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 by 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:

```
SET DISTUTILS_USE_SDK=1
    SET JAVA_HOME=%JDK_HOME%
    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:

```
    pip install pandas
    pip install numpy
    pip install scipy
    pip install matplotlib.
```

Listing 2.6: Example of a Knowage Data Mining engine template which uses a Python script

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<DATA_MINING>
   <LANGUAGE name="Python"/>
   <DATASETS>
       <DATASET name="df" readType="csv" type="file" label="HairEyeColor" canUpload="true"><![CDATA[sep=',']]>
       </DATASET>
   </DATASETS>
   <SCRIPTS>
       <SCRIPT name="test01" mode="auto" datasets="df" label="HairEyeColor" libraries="csv,os,pandas,numpy">
        <![CDATA[ print(df.ix[0,0]) y=df.ix[0,0] ]]>
       </SCRIPT>
   </SCRIPTS>
   <COMMANDS>
               <COMMAND name="testcommand" scriptName="test01" label="test01"  mode=" auto">
```

(continues on next page)

(continued from previous page)

```
        <OUTPUTS>
                <OUTPUT type="text" name="first_element" value="y" function=""  mode="manual" label="first_
↪element"/>
        </OUTPUTS>
    </COMMAND>
   </COMMANDS>
</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.

2.8 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.

2.8.1 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 metadata database of Knowage,as following

Listing 2.7: Connection parameters for Knowage metadata db.

```
        spagobi.datasource.driver=<driver JDBC>
        spagobi.datasource.url=<URL JDBC>
        spagobi.datasource.user=<user_name>
        spagobi.datasource.pwd=<password> encript.password=true
```

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

2.8.2 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 2.8: keystore.jks creation.

```
        $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 2.9: Importing the certificate into JDK CA repository.

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

2.8.3 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 2.10: Export of the certificate.

```
        <Connector acceptCount="100"
                               maxHttpHeaderSize="8192"
                               clientAuth="false"
                               debug="0"
                               disableUploadTimeout="true"
                               enableLookups="false"
                               SSLEnabled="true"
                               keystoreFile="conf/keystore.jks"
                               keystorePass="<keystore password>"
                               maxSpareThreads="75"
                               maxThreads="150"
                               minSpareThreads="25"
                               port="8443"
                               scheme="https"
                               secure="true"
                               sslProtocol="TLS"
        />
```

2.8.4 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 2.11: Values of the SBI_CONFIG table to change.

```
        SPAGOBI_SSO.ACTIVE:
        set valueCheck to true

```

(continues on next page)

(continued from previous page)

```
        CAS_SSO.VALIDATE-USER.URL:
        set valueCheck to https://<URL of the CAS application>/cas

        CAS_SSO.VALIDATE-USER.SERVICE:
        set valueCheck to https://<URL of the Knowage server >:8443/knowage/proxyCallback

        SPAGOBI_SSO.SECURITY_LOGOUT_URL:
        set valueCheck to https://<URL of the CAS application>/cas/logout
```

Then set the `sso_class` environment variable as below:

```
        <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 2.12: Setting the CAS filters for sso_class variable.

```
<filter>
  <filter-name>CAS Authentication Filter</filter-name>
  <filter-class>org.jasig.cas.client.authentication.AuthenticationFilter</filter-class>
  <init-param>
   <param-name>casServerLoginUrl</param-name>
    <param-value>https://<nome del server CAS>/cas/login</param-value>
  </init-param>
  <init-param>
   <param-name>serverName</param-name>
    <param-value><dominio di knowage, incluso il protocollo e la porta, se non standard></param-value>
  </init-param>
</filter>

<filter>
  <filter-name>CAS Validation Filter</filter-name>
  <filter-class>org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter</filter-class>
  <init-param>
        <param-name>casServerUrlPrefix</param-name>
        <param-value>https://<nome del server CAS>/cas/</param-value>
        </init-param>
  <init-param>
        <param-name>serverName</param-name>
        <param-value><dominio di Knowage Server, incluso il protocollo e la porta, se non standard></param-
↪value>

  </init-param>
  <init-param>
        <param-name>proxyReceptorUrl</param-name>
        <param-value>/proxyCallback</param-value>
  </init-param>

[Nelle web application knowageXXXengine presente anche questo parametro:

<init-param> <param-name>proxyCallbackUrl</param-name>
<param-value>
   <dominio di knowage Server, incluso il protocollo e la porta, se  non standard>/< knowageXXXengine>/
↪proxyCallback </param-value>
</init-param>]

</filter>

```

(continues on next page)

(continued from previous page)

```
<filter>
  <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>
  <filter-class>org.jasig.cas.client.util.HttpServletRequestWrapperFtilter</filter-class>

</filter>...

<filter-mapping>
 <filter-name>CAS Authentication Filter</filter-name>
 <url-pattern>/servlet/*</url-pattern>
</filter-mapping>

<filter-mapping>
 <filter-name>CAS Validation Filter</filter-name>
 <url-pattern>/servlet/*</url-pattern>
</filter-mapping>
 <filter-mapping>
 <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>
 <url-pattern>/servlet/*</url-pattern>
</filter-mapping>

[Nelle web application knowageXXXengine presente anche questo mapping:
 <filter-mapping>
  <filter-name>CAS Validation Filter</filter-name>
  <url-pattern>/proxyCallback</url-pattern>
  </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 abow reports.

2.9 Advanced configuration

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

2.9.1 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 2.13: Configuration of the pool of thread for Tomcat.

```
<Resource auth="Container" factory="de.myfoo.commonj.work.FooWorkManagerFactory"
    maxThreads="5"
    minThreads="1"
    queueLength="10"
    maxDaemons="10"
    name="wm/SpagoWorkManager"
    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 2.14: Configuration of the pool of thread for JBoss.

```
<object-factory name="java:global/SpagoWorkManager" module="de.myfoo.commonj"
  class="de.myfoo.commonj.work.MyFooWorkManagerFactory">
  <environment>
  <property name="maxThreads" value="5"/>
  <property name="minThreads" value="1"/>
```

(continues on next page)

(continued from previous page)

```
  <property name="queueLength" value="10"/>
  <property name="maxDaemons" value="10"/>
  </environment>
</object-factory>
```

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

- minThreads: the mininum number of threads in the thread pool. Default: 2;
- maxThreads: the maximum number of threads in the thread pool. Default: 10;
- queueLenght: 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.

2.9.2 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 : 1073741824)
- **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 empy. The allowable values are: EVERY_10_MINS, EVERY_15_MINS, EVERY_20_MINS, EVERY_30_MINS, HOURLY,DAILY,WEEKLY,MONTHLY,YEARLY. Every value different from those just listed unenable the periodic cleaning. (Defualt: 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 toolow 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.CREATE_AND_PERSIST_TABLE.TIMEOUT**: 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 persistance attempt. (Default : 10)
- **SPAGOBI.WORKMANAGER.SQLDBCACHE.TIMEOUT**: It represents the maximum waiting time (in-milliseconds) of an asynchronus 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)

2.9.3 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 analize the log file of Knowage. In the first part we can set the generation mechanism of the log file, while ih the second one the level of tracing.

Listing 2.15: Logg appender.

```
log4j.rootLogger=ERROR, SpagoBI

# SpagoBI Appender
log4j.appender.SpagoBI=org.apache.log4j.RollingFileAppender
log4j.appender.SpagoBI.File=${catalina.base}/logs/knowage.log
log4j.appender.SpagoBI.MaxFileSize=10000KB
log4j.appender.SpagoBI.MaxBackupIndex=0
log4j.appender.SpagoBI.layout=org.apache.log4j.PatternLayout
log4j.appender.SpagoBI.layout.ConversionPattern=[%t] %d{DATE} %5p %c.%M:%L - %m %n

log4j.appender.SpagoBI.append=false

log4j.appender.Quartz=org.apache.log4j.RollingFileAppender
log4j.appender.Quartz.File=${catalina.base}/logs/Quartz.log
log4j.appender.Quartz.MaxFileSize=10000KB
log4j.appender.Quartz.MaxBackupIndex=10
log4j.appender.Quartz.layout=org.apache.log4j.PatternLayout
log4j.appender.Quartz.layout.ConversionPattern= [%t] %d{DATE} %5p %c.%M:%L - %m  %n

log4j.appender.SpagoBI_Audit=org.apache.log4j.FileAppender
log4j.appender.SpagoBI_Audit.File=${catalina.base}/logs/knowage_[1]\_OperatorTrace.log

log4j.appender.SpagoBI_Audit.layout=org.apache.log4j.PatternLayout
log4j.appender.SpagoBI_Audit.layout.ConversionPattern=%m%n

log4j.appender.CONSOLE = org.apache.log4j.ConsoleAppender
log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
log4j.appender.CONSOLE.layout.ConversionPattern=%c.%M: %m%n #


log4j.logger.Spago=ERROR, SpagoBI log4j.additivity.Spago=false

log4j.logger.it.eng.spagobi=ERROR, SpagoBI, CONSOLE
log4j.additivity.it.eng.spagobi=false

log4j.logger.it.eng.spagobi.commons.utilities.messages=ERROR, SpagoBI
log4j.logger.it.eng.spagobi.commons.utilities.urls.WebUrlBuilder=ERROR,SpagoBI
log4j.logger.org.quartz=ERROR, Quartz, CONSOLE
log4j.logger.org.hibernate=ERROR, SpagoBI

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

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

in

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

For further details we refer to the official Log4J documents.

2.9.4 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.

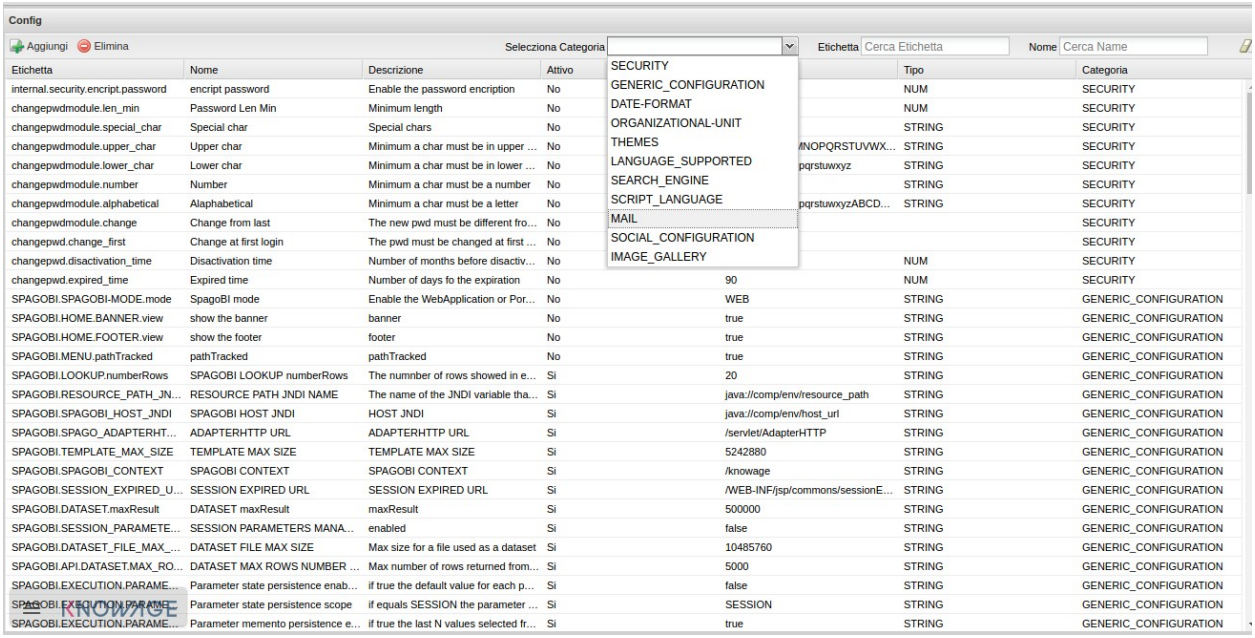


Fig. 2.3: Mail server configuration.

Moreover, each profile has the following values:

- smtphost: the smpt 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,
- useSSL: in case the SSl is in use.

2.9.5 Maximum file size

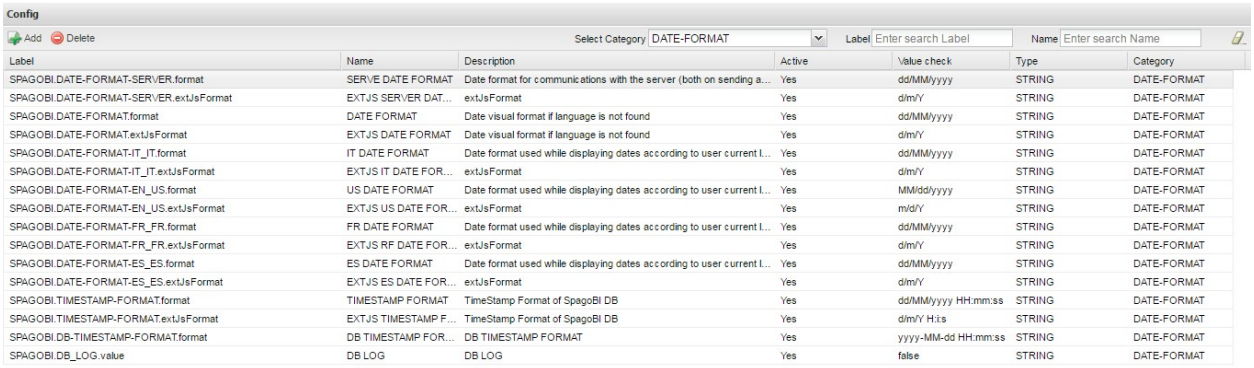
For security reasons, Knowage has a series of parameters which manage the maximum file size that can be loaded on the server through the web GUI. To modify those parameters, it is required to enter the Knowage server application as

administrator and access the ”server settings“ section and then ”configuration management“. The parameters at issue are the following:

- **SPAGOBI.TEMPLATE_MAX_SIZE** : TEMPLATE MAX SIZE: it is the maximum template dimension of an anlytical document, expressed in bytes; the default value is 5MB;
- **SPAGOBI.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;
- **SPAGOBI.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;

2.9.6 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:i:s	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 valorize the parameters in compliance with each other, according to a local data.

The parameters **SPAGOBI.DATE-FORMAT-SERVER.forma**t 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.

2.9.7 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:sup:‘_‘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].

2.9.8 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 cohesist 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 mantain 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"
```

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.

CHAPTER 3

Administration Manual

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.

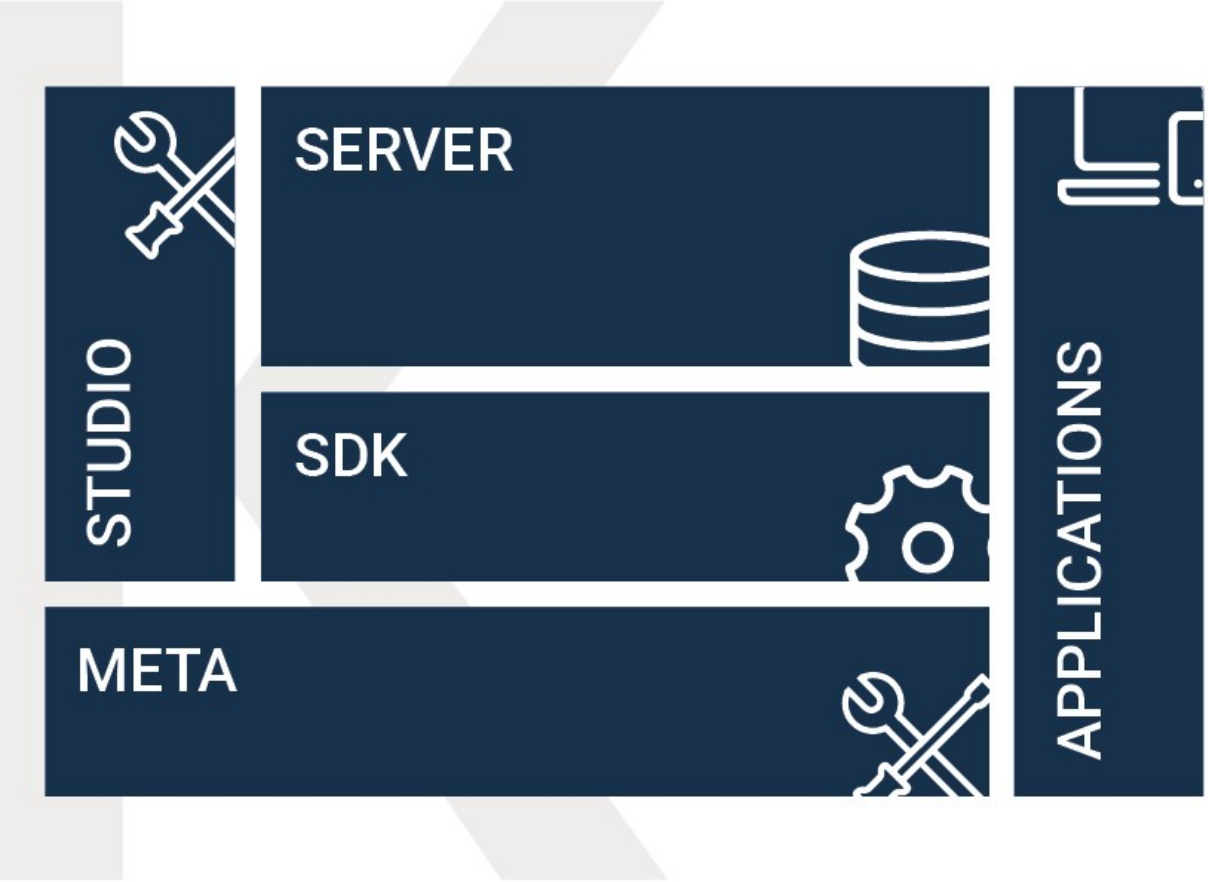


Fig. 3.1: Knowage modules.

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.

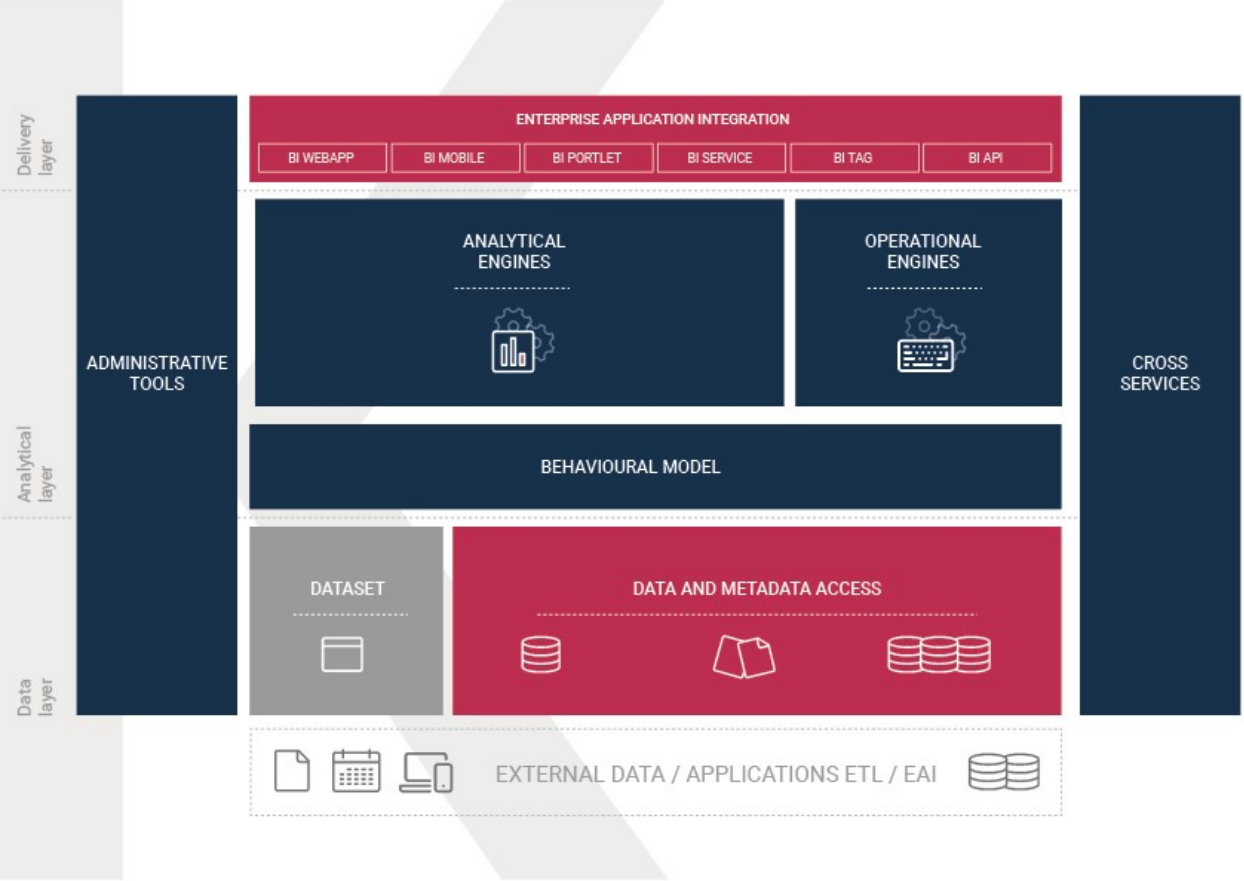


Fig. 3.2: Knowage Server architecture structure.

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.

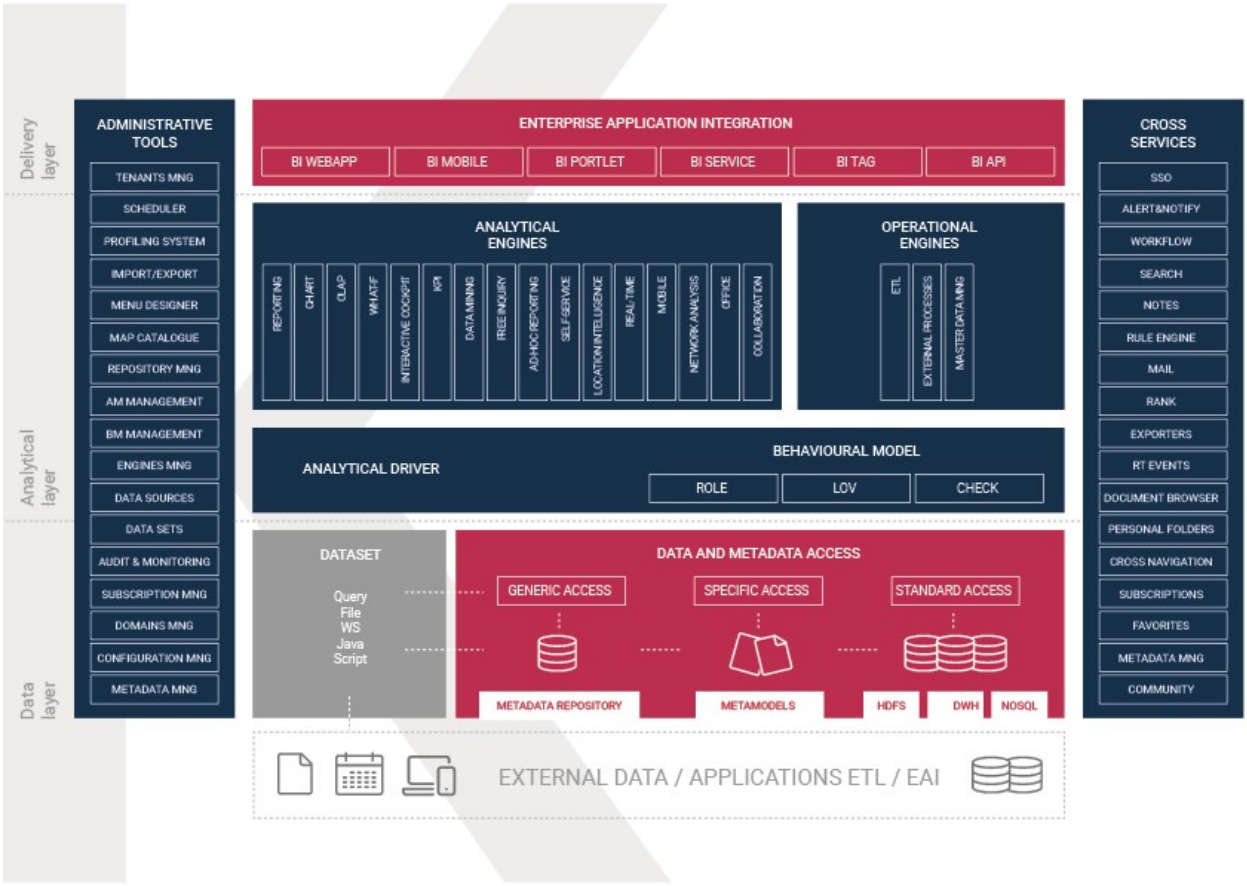


Fig. 3.3: Knowage Server architecture detail.

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 exiting environments; high flexibility and modularity; high scalability, with minimum economic, infrastructural and applicationlevel 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 intance, 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 funciontalities, 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

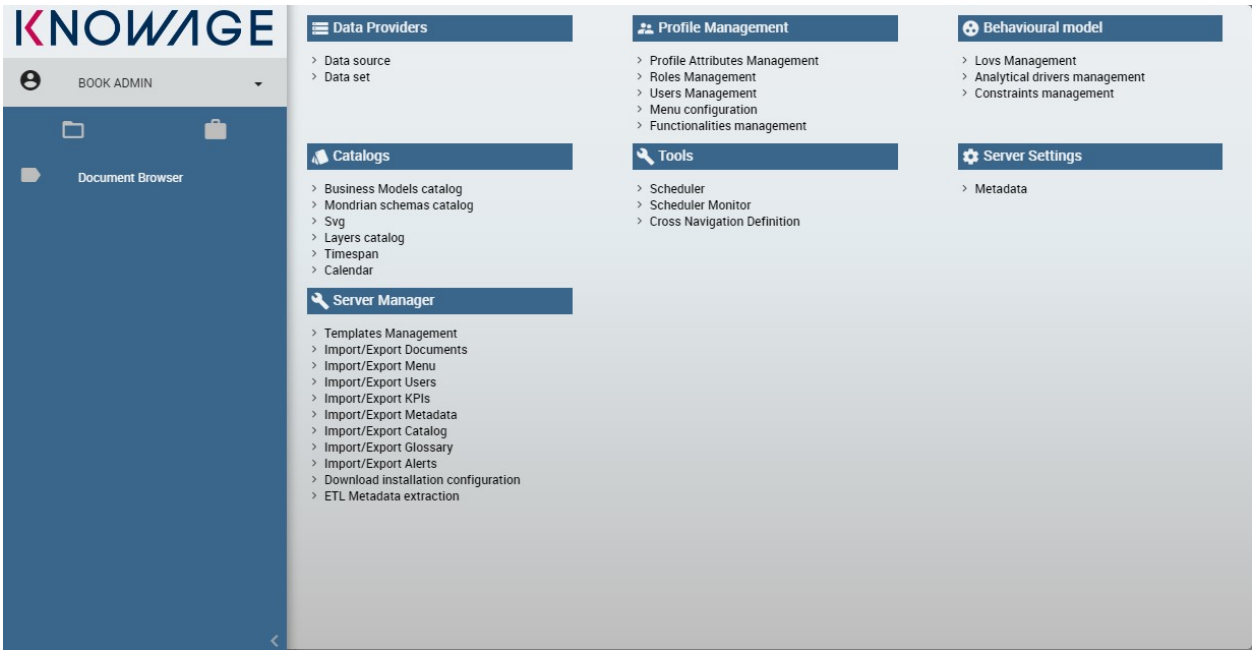


Fig. 3.5: Menu

- **Select role** 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.
- **Info** To view the details of current Knowage version.
- **Log out** To go back to log in page.

Table 3.1: Menu components - User menu

Icon	Name	Description
	Knowage user	It opens a hidden menu with extra functionalities.
	Select role	To select the default role (available if the user is associated to more than one role).
	Languages	Language options.
	Info	Info on Knowage version.
	Log out	log out.

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 3.2: Menu components - BI Functionalities Menu.

Icon	Name	Description
	Documents development	Document creation and acces to the archive folders.
	Workspace	Inquiry your business models, navigate and create your datasets

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

- **Data Providers** Here you can set and manage Data Sources.
- **Profile Management** In this panel you can organize the users profilation, authorizations and attributes, but also organize the Analitycal model. It means you can create/manage Roles, Users and Attributes as well as configure the functionality tree and the menu, i.e. the list of quick access link to analytcal document or other resources provided to the users.
- **Behavioural model** Here you manage all the Behavioural model, which means create analytical drivers and lov. In this area you can access the contraints configuration too.
- **Catalogs** In this area you manage different catalogs, that may vary from product to product: the **Business Model** catalogues used for QbE; the **Layer Catalogs** for the creation of GIS analitycal documents and so on.
- **Tools** In this area you can access the different scheduler options.

- **Server Settings** In this panel you have access to all server settings configuration options, such as configuration or domain management.
- **Server Manager** This is an optional package. It gives you access to different server functionalities, such as template management and all the import/export features.

Table 3.3: Menu components - Administrator Menu.

Icon	Name	Management areas
|image27|	Data Providers	Data source settings.
|image28|	Profile Management	Profile Attributes, Roles, User and Menu configuration
|image29|	Behavioural model	Lovs, Analytical Driver and Contraints
|image30|	Catalogs	Business Models and Layers
|image31|	Tools	Scheduler
|image32|	Server Settings	User Data Properties, Configuration, Domains and Metadata
|image33|	Server Manager	Template manager and Import-Export options

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

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:

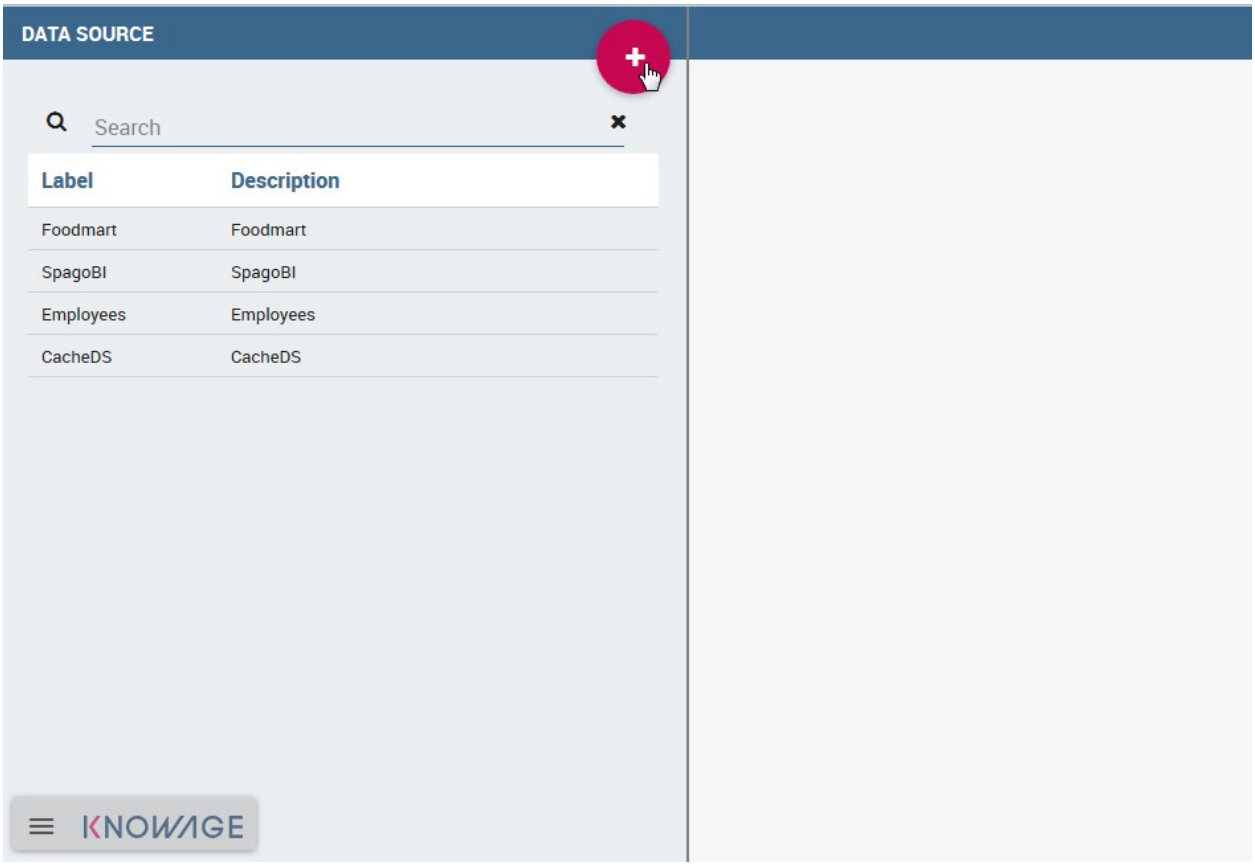


Fig. 3.6: Add a new data source

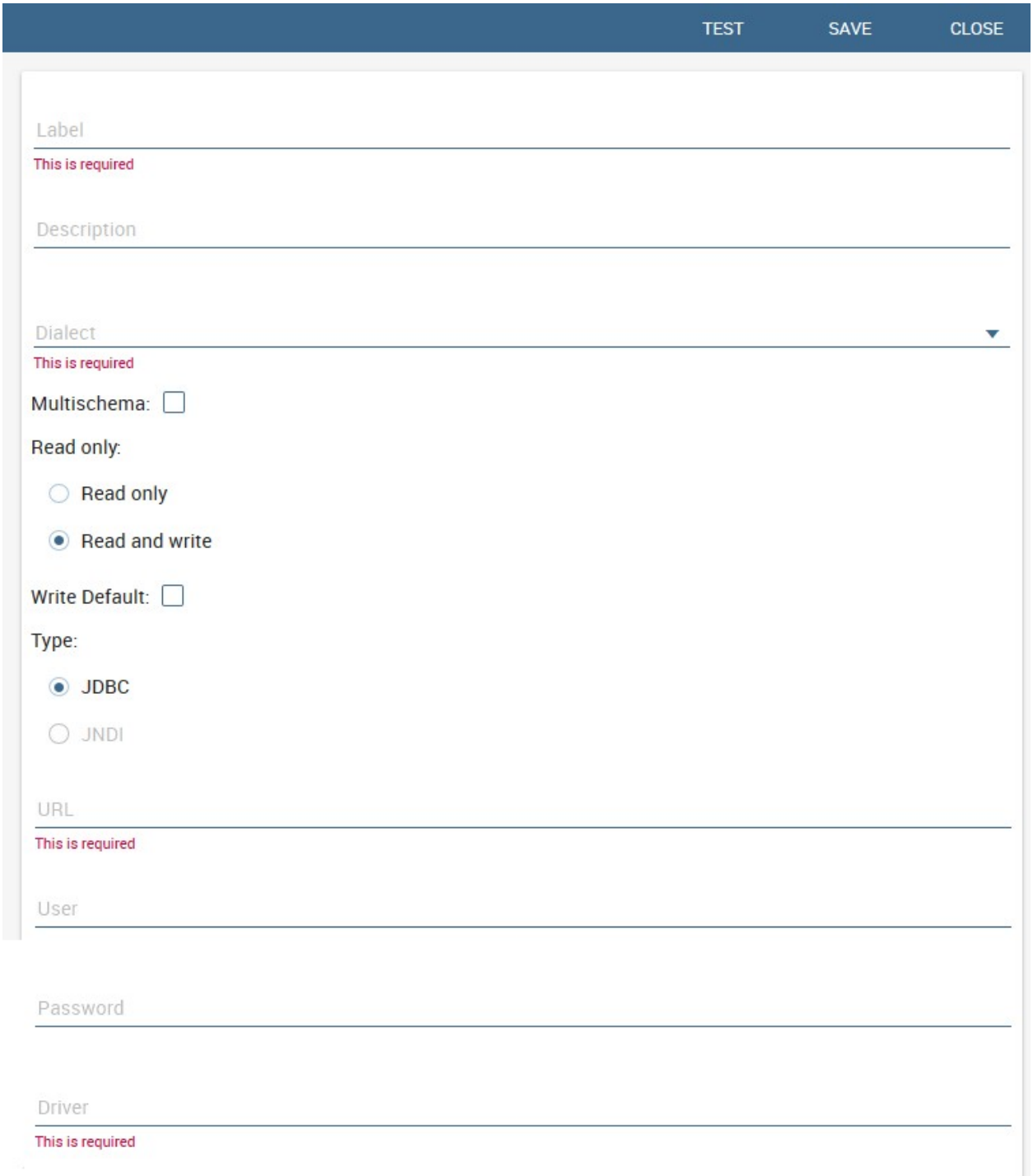


Fig. 3.7: Data source details.

Table 3.4: Certified Data Sources

Certified Data Sources	
Oracle	11, 12
MySQL	5.2, 5.5, 5.6
PostgreSQL	8.2, 9.1
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

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.

Fig. 3.8: Libraries to include in the apache driver.

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

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 connetction 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 doesnt 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 priviledges 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" }  ]
```

(continues on next page)

(continued from previous page)

```
    }
)
```

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 .

```
use admin
db.createRole( { role: "executeFunctions", privileges: [ { resource: { anyResource: true }, actions: [
↪"anyAction" ] } ], roles: [] } )
use foodmart
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#.W59wiaYzaUl)

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 bevioural 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 bevioural 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 3.5: 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. 3.9: 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.

Fig. 3.10: 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 is intended to assign specific categories to each role, in a way that each user can only see the business models, datasets or KPI that belong to the categories associated with his role.

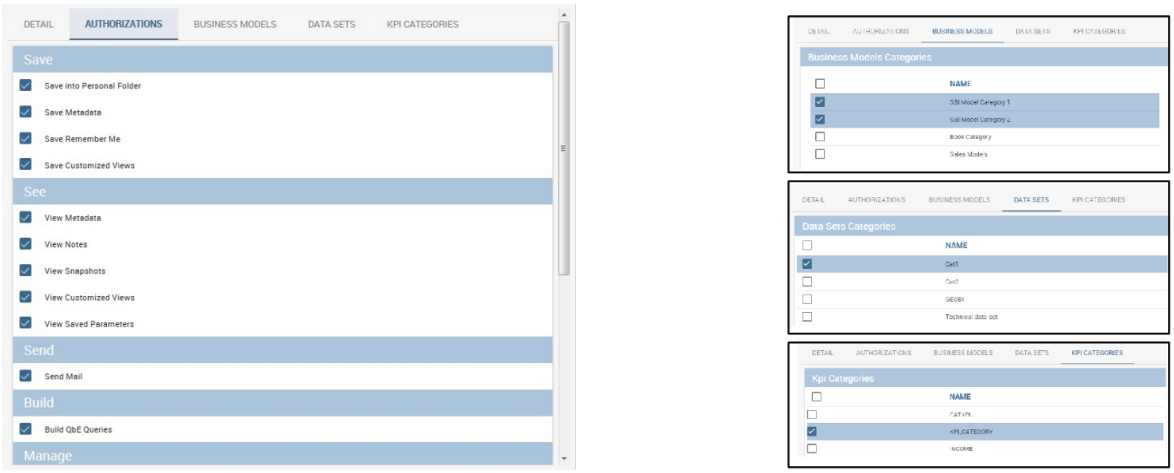


Fig. 3.11: Roles Management.

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.

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. 3.12: Users Management.

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.

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

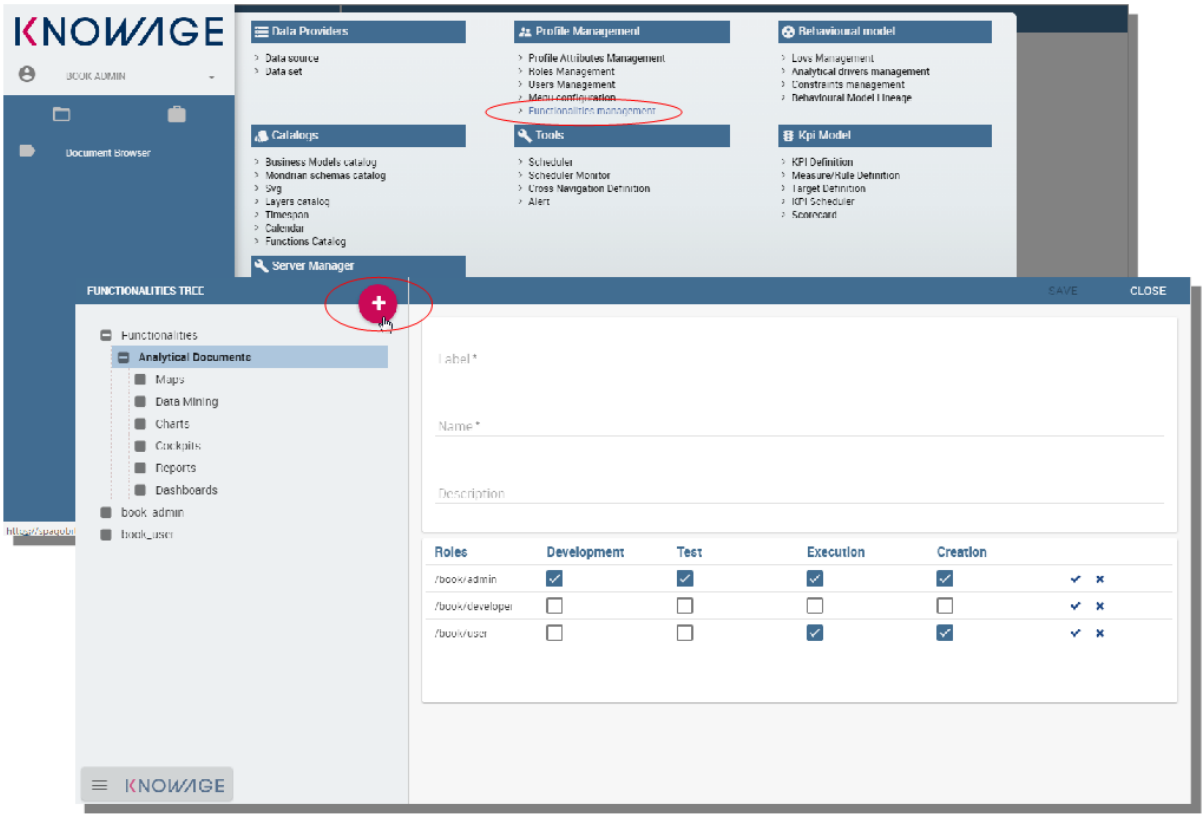


Fig. 3.13: Create a new folder and assign permissions.

(as a folder) or nothing (empty node). Every node can be associated to different roles. This menu structure can be created and modified exclusively by the administrator in the **Tools** area. To access the Menu configuration area, go to **Profile Management > Menu Configuration** from the Main Menu.

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

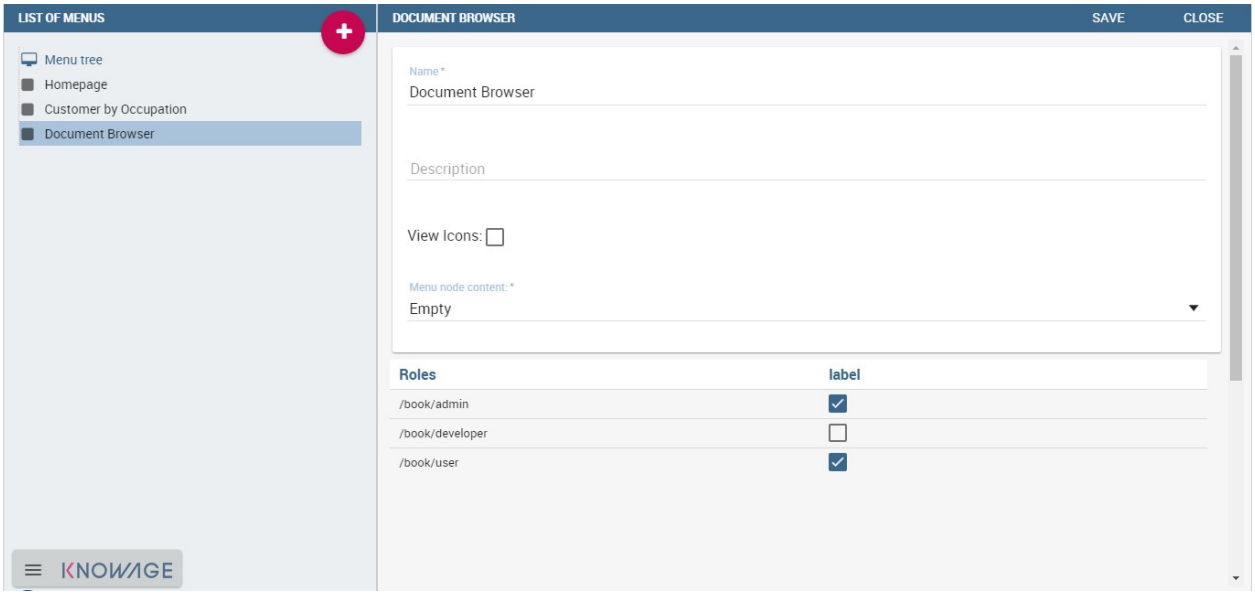


Fig. 3.14: Menu tree actions.

In general you can:

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

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

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

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.

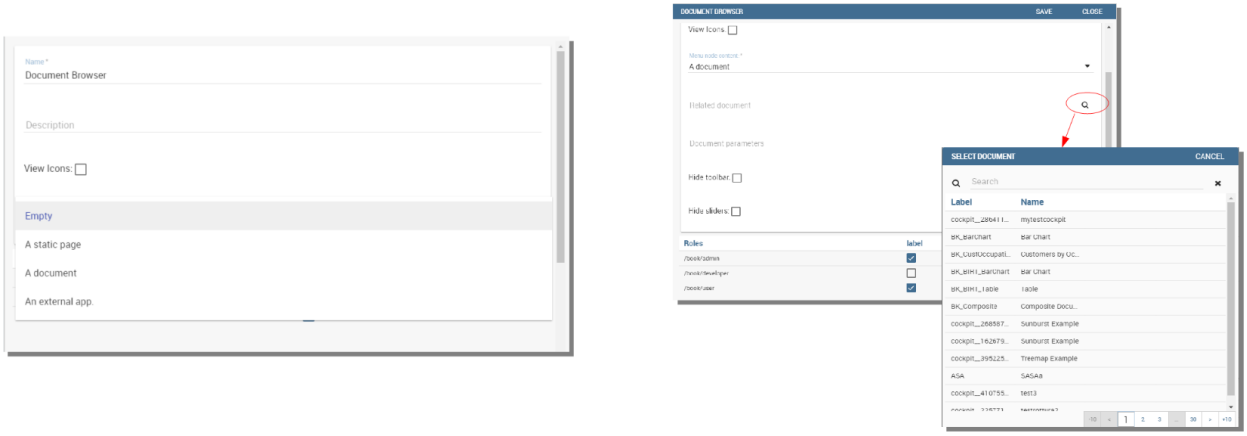


Fig. 3.15: **Empty** (left) and **document** (right) content type.

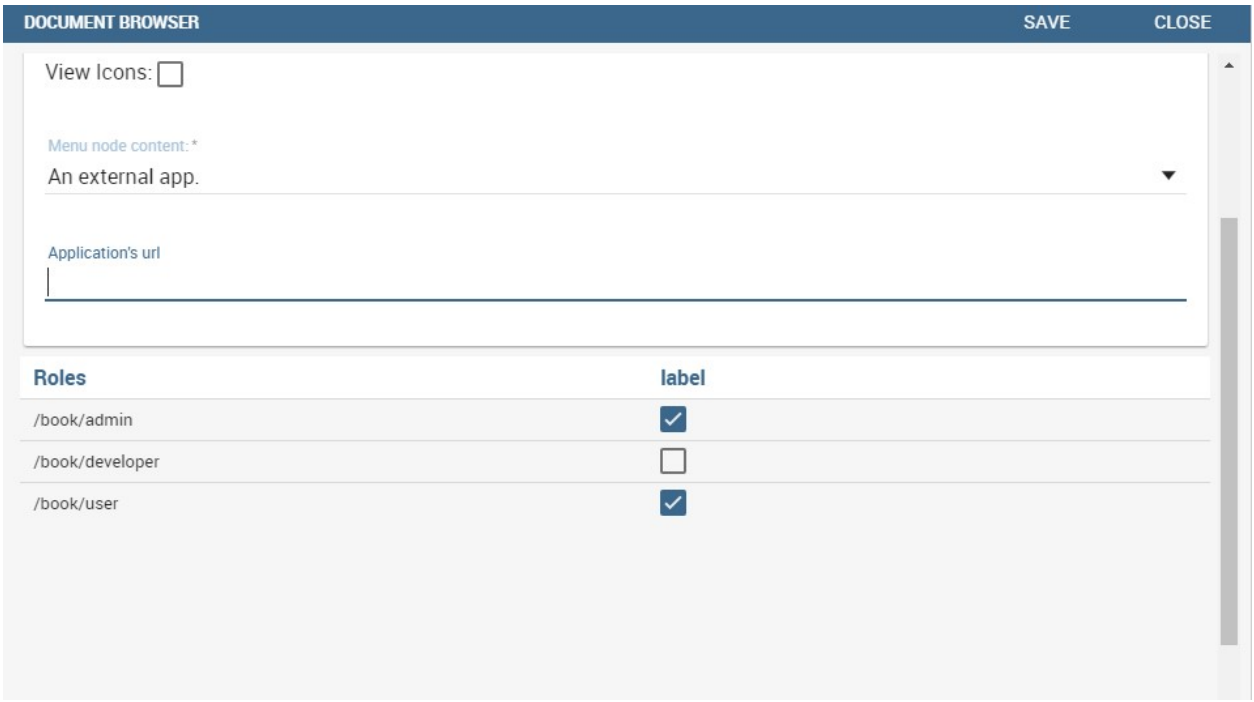


Fig. 3.16: External application content type.

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.

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 > Taned > Import/export** from within Talend. Then set connection options as described below.

Table 3.6: 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.

```
<etl>
      <job    project="Foodmart"
              jobName="sales_by_month_country_product_familiy"
              context="Default"
              version="0.1"
              language="java"
      />
</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

```
package it.eng.spagobi.job;

        import java.util.Iterator;
        import it.eng.spagobi.engines.commonj.process.SpagoBIWork;

        public class CommandJob extends SpagoBIWork {
            @Override
            public boolean isDaemon() {
                return true;
            }

    @Override
    public void release() {
        System.out.println("Release!!");
        super.release();
    }

    @Override public void run() {
        super.run();
        System.out.println("Job started! ");
        java.util.Map parameters = getSbiParameters();
        for (Iterator iterator = parameters.keySet().iterator(); iterator.hasNext();) {
            String type = (String) iterator.next();
            Object o = parameters.get(type);
            System.out.println("Parameter " + type + " value" + o.toString());
        }
        for(int i = 0; i < 50 && isRunning(); i++) {
            System.out.println("job is running!");
            try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        System.out.println("Job finished!");
    }
}
```

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.

```
package it.eng.spagobi.job;
import it.eng.spagobi.engines.commonj.process.CmdExecWork;
import java.io.IOException;
public class CommandJob extends CmdExecWork{
public boolean isDaemon() {
```

(continues on next page)

(continued from previous page)

```
   return true;}
   public void release() {
   super.release();}
   public void run() {
   super.run();
   if(isRunning()){
   try {
   execCommand();
   } catch (InterruptedException e) {
   } 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

```
package it.eng.test;
   import java.io.FileNotFoundException;
   import java.io.FileOutputStream;
   import java.io.PrintStream;
   public class ProcessTest {
   public static void main(String[] args) {
   FileOutputStream file=null;
   try {
   file = new FileOutputStream("C:/file.txt");
   } catch (FileNotFoundException e) {
   // TODO Auto-generated catch block
   e.printStackTrace();}
   PrintStream output = new PrintStream(file);
   while (true){
   output.println("New row");
   output.flush();
   try {
   Thread.currentThread().sleep(5000l);
   } catch (InterruptedException e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
   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

```
<COMMONJ>
   <WORK workName='JobTest' className='it.eng.spagobi.job.CommandJob'>
   <PARAMETERS>
   <PARAMETER name='cmd'value='C:/Programmi/Java/jdk1.5.0_16/bin/java'/>
   <PARAMETER name='classpath'
   value='C:/resources/commonj/CommonjRepository/JobTest/process.jar'/>
   <PARAMETER name='cmd_par' value='it.eng.test.ProcessTest'/>
   <PARAMETER name='sbi_analytical_driver' value='update'/>
```

(continues on next page)

(continued from previous page)

```
    <PARAMETER name='sbi_analytical_driver' value='level'/>
    </PARAMETERS>
    </WORK>
  </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 proces (as defined above).
- The tag <PARAMETERS> contains all parameters. Each <PARAMETER> tag includes a parameter. Some of them are mandatory

Table 3.7: 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 CmdWorkExec or KnowageWork).
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

```
C:/Programmi/Java/jdk1.5.0_16/bin/java 'it.eng.test.ProcessTest'
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.

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.

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

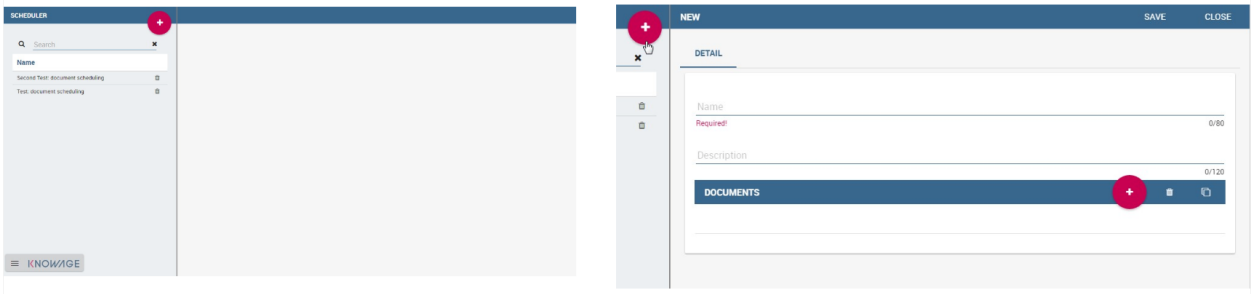


Fig. 3.17: Left: scheduler main page. Right: New activity GUI

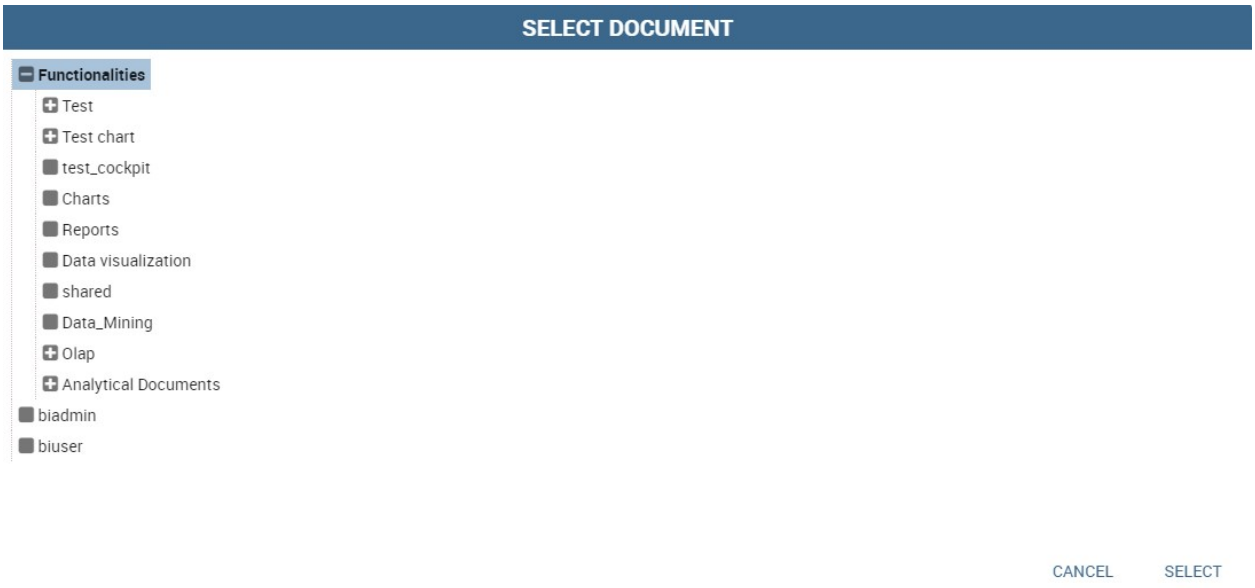


Fig. 3.18: Adding a document to an activity.

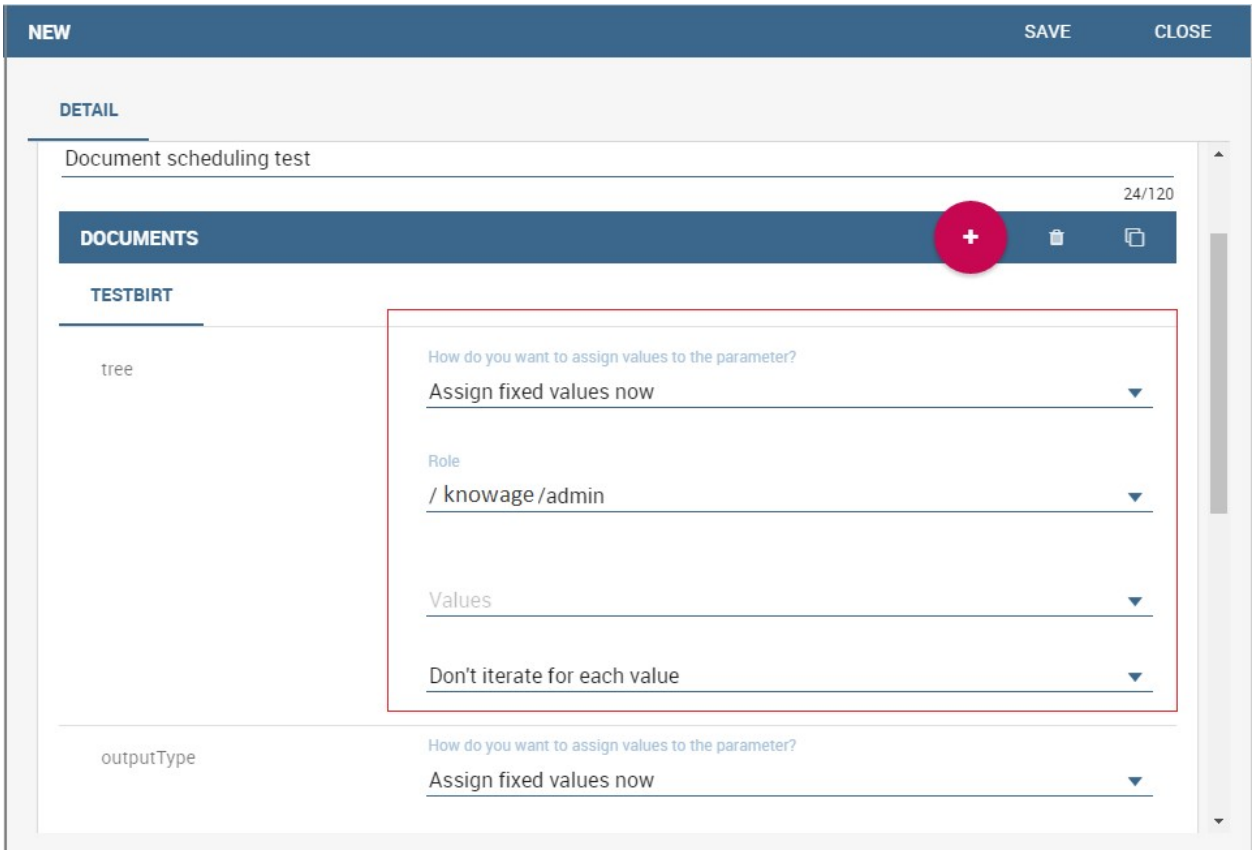


Fig. 3.19: Manage parameters.

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 esired documents have been 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 schedulation area in the bottom right side. Similarly, click on the same button to associate schedules to newly created activities.

Schedulation details opens (Figure below).

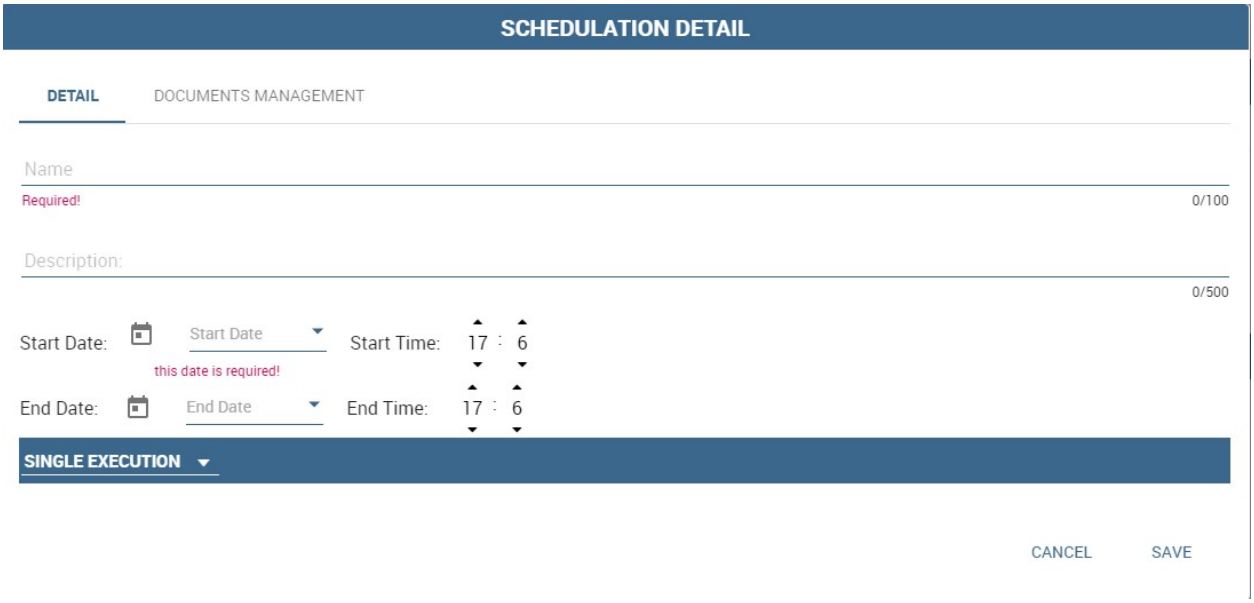


Fig. 3.20: Schedulation detail panel.

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

You can decide it, by choosing the schedulation type.

Available schedulation type are:

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

A **Single Execution** is a fixed execution for a date and a time which happens only once. A **Customized Execution** repeats your schedulation perodically 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 messge,
- 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 schedulation is trigged otherwise it isn't.

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

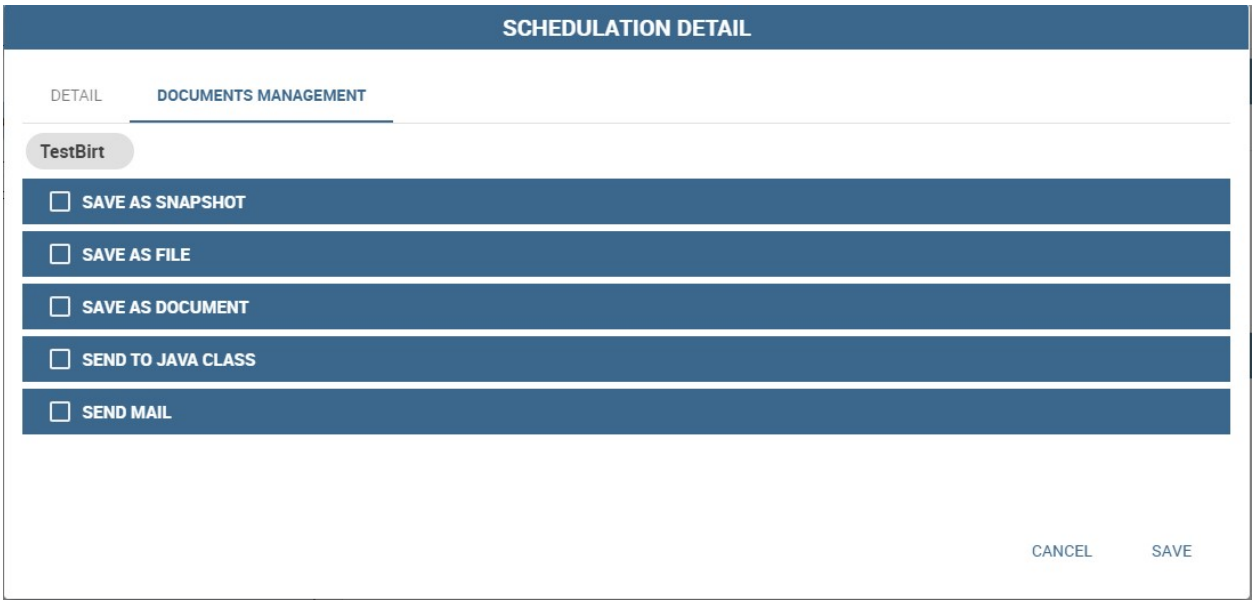


Fig. 3.21: Document management.

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 cliking 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.

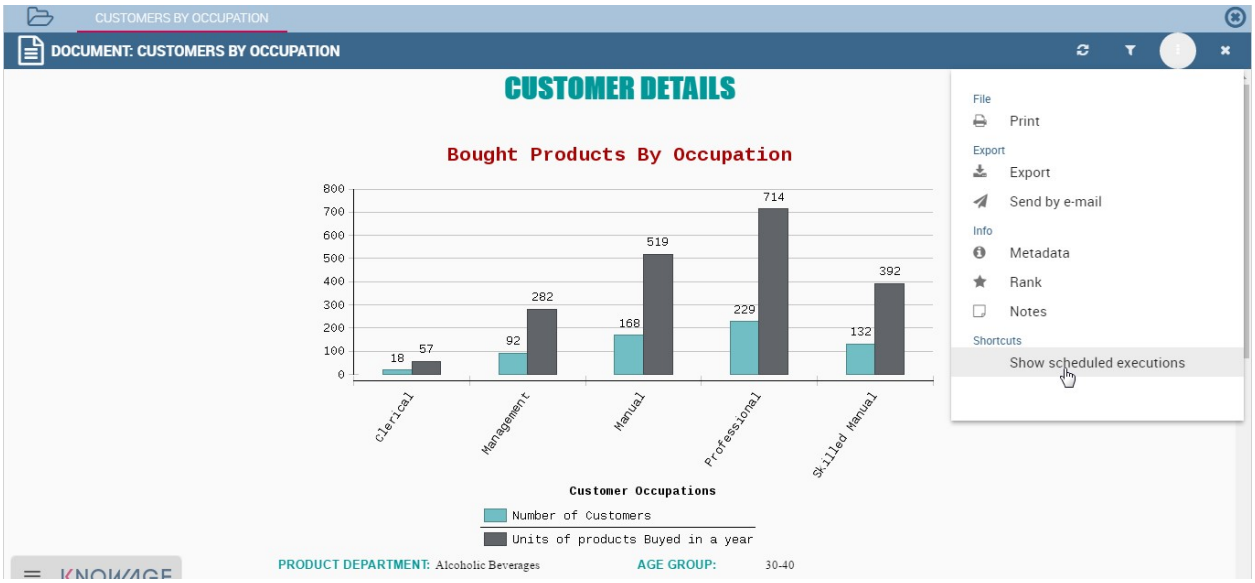


Fig. 3.22: Steps to open saved snapshots

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 containg 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.

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 yous 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.

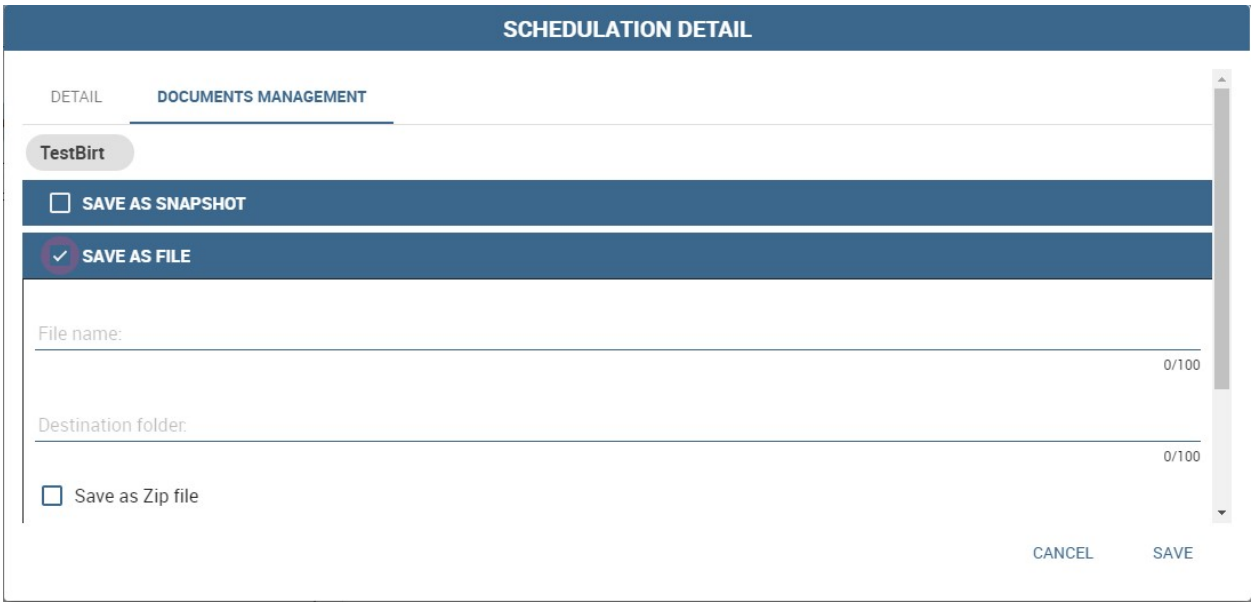


Fig. 3.23: Save as File form.

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.

```
package it.eng.spagobi.tools;
import it.eng.spagobi.analiticalmodel.document.bo.BIObject
public abstract class JavaClassDestination
implements IJavaClassDestination {
BIObject biObj=null;
byte[] documentByte=null;
public abstract void execute();
public byte[] getDocumentByte() {
return documentByte;
} public void setDocumentByte(byte[] documentByte) {
this.documentByte = documentByte;
}
public BIObject getBiObj() {
return biObj;
}
public void setBiObj(BIObject biObj) {
this.biObj = biObj;
}
}
```

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.

```
public class FileDestination extends JavaClassDestination {
public static final String OUTPUT_FILE_DIR = "D:\\ScheduledRpts\\";
public static final String OUTPUT_FILE_NAME = "output.dat";
private static transient Logger logger = Logger.getLogger(FileDestination.class);
public void execute() {
File outputDir;
File outputFile;
OutputStream out;
byte[] content = this.getDocumentByte();
String outputFileName;
logger.debug("IN");
outputFile = null;
out = null;
try {
outputFileName = getFileName();
logger.debug("Output dir [" + OUTPUT_FILE_DIR + "]");
logger.debug("Output filename [" + outputFileName + "]");
outputDir = new File(OUTPUT_FILE_DIR);
outputFile = new File(outputDir, outputFileName);
if(!outputDir.exists()) {
logger.debug("Creating output dir [" + OUTPUT_FILE_DIR + "] ...");
if(outputDir.mkdirs()) {
logger.debug("Output dir [" + OUTPUT_FILE_DIR + "] succesfully created");
} else {
throw new SpagoBIRuntimeException( "Impossible to create outputd dir
[" + OUTPUT_FILE_DIR + "]");
}
} else {
if(!outputDir.isDirectory()) {
throw new SpagoBIRuntimeException( "Outputd dir [" + OUTPUT_FILE_DIR + "]
is not a valid directory");
}
}
try {
out = new BufferedOutputStream( new FileOutputStream(outputFile));
} catch (FileNotFoundException e) {
throw new SpagoBIRuntimeException(
"Impossible to open a byte stream to file
[" + outputFile.getName() + "]", e);
} try {
out.write(content);
} catch (IOException e) {
throw new SpagoBIRuntimeException( "Impossible to write on file
[" + outputFile.getName() + "]", e);
}
} catch(Throwable t) {
throw new SpagoBIRuntimeException( "An unexpected error occurs while saving
document" + " to file [" + outputFile.getName() + "]", t);
} finally {
if(out != null) {
try {
out.flush(); out.close();
} catch (IOException e) {
throw new SpagoBIRuntimeException( "Impossible to properly close file
[" + outputFile.getName() + "]", e);
}
}
logger.debug("OUT");
}
}
private String getFileName() {
String filename = "";
BIObject analyticalDoc;
List analyticalDrivers;
```

(continues on next page)

(continued from previous page)

```
BIObjectParameter analyticalDriver;
String extension = "pdf";
analyticalDoc = getBiObj();
analyticalDrivers = analyticalDoc.getBiObjectParameters();
for(int i = 0; i < analyticalDrivers.size(); i++) {
analyticalDriver = (BIObjectParameter)analyticalDrivers.get(i);
String parameterUrlName = analyticalDriver.getParameterUrlName();
List values = analyticalDriver.getParameterValues();
if(!parameterUrlName.equalsIgnoreCase("outputType")){
filename += values.get(0);
} else {
extension = "" + values.get(0);
}
}
filename = filename.replaceAll("[^a-zA-Z0-9]", "_");
filename += "." + extension;
return filename;
}
}
```

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 avaiable 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:

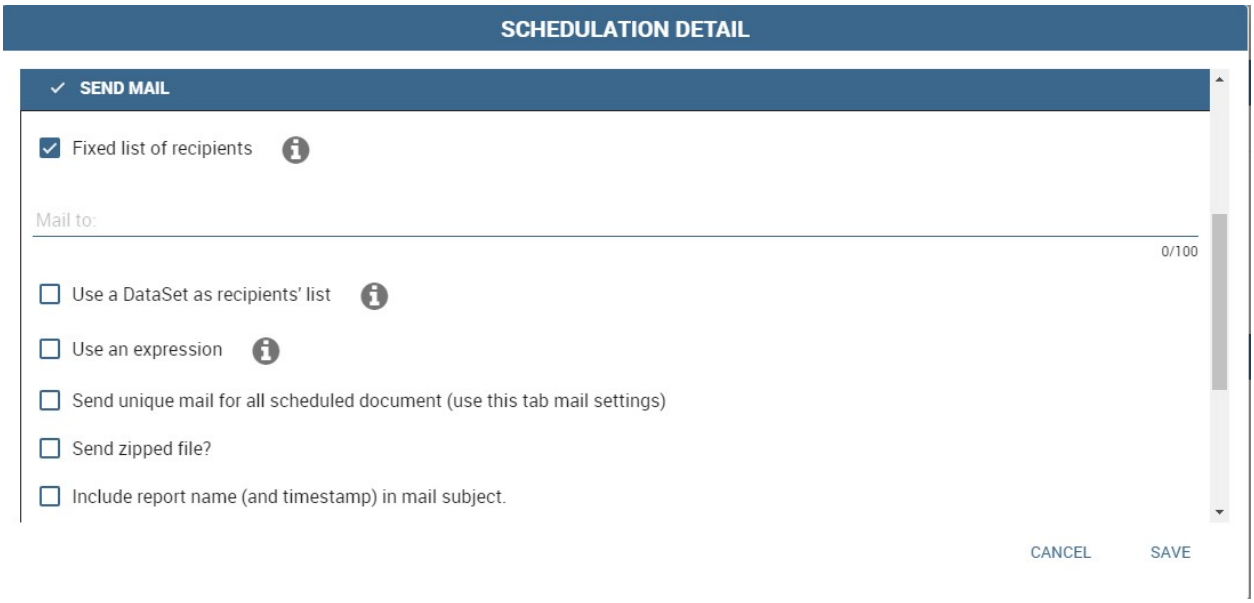


Fig. 3.24: Sending mail form.

- 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.

	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

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 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 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 Schedulation 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.

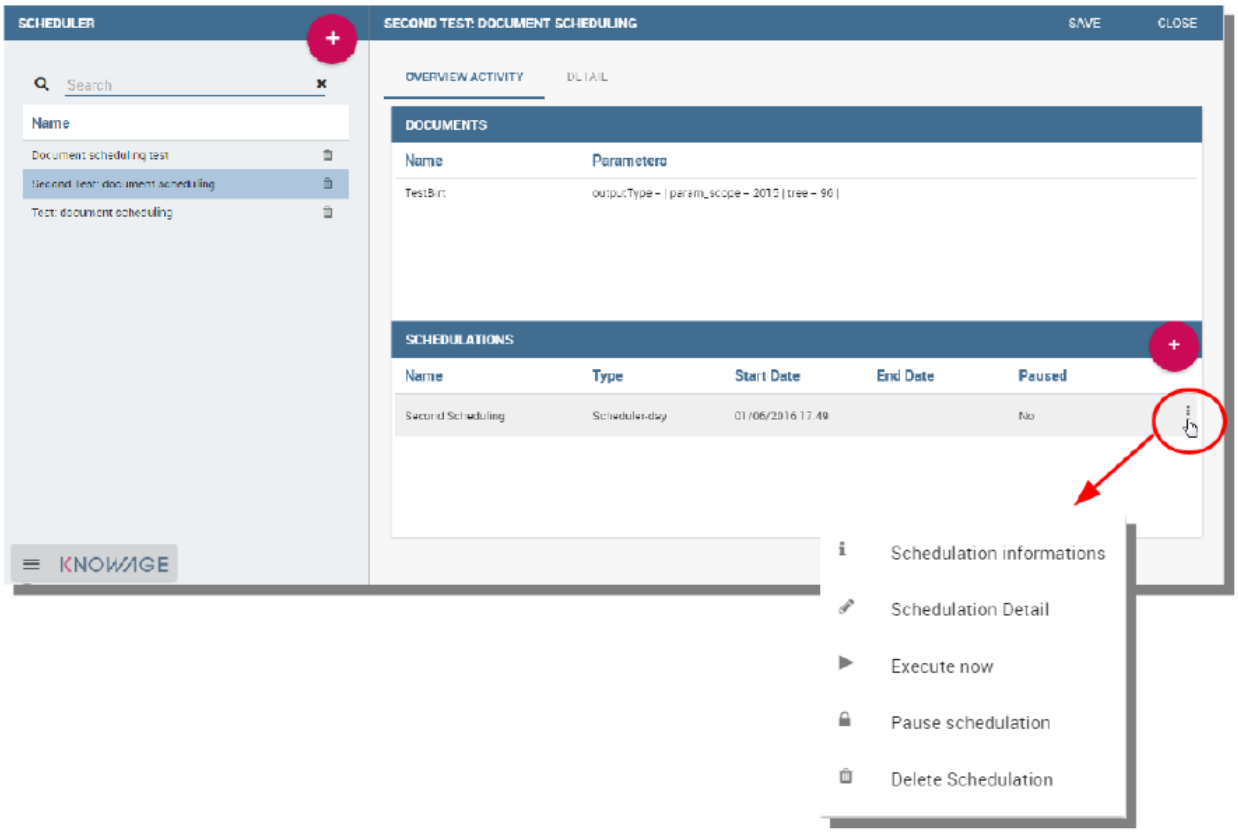


Fig. 3.25: Exploring the detailed of a scheduled activity.

A general overview of the selected schedulation 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 schedulation 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 Exectution), 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:

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

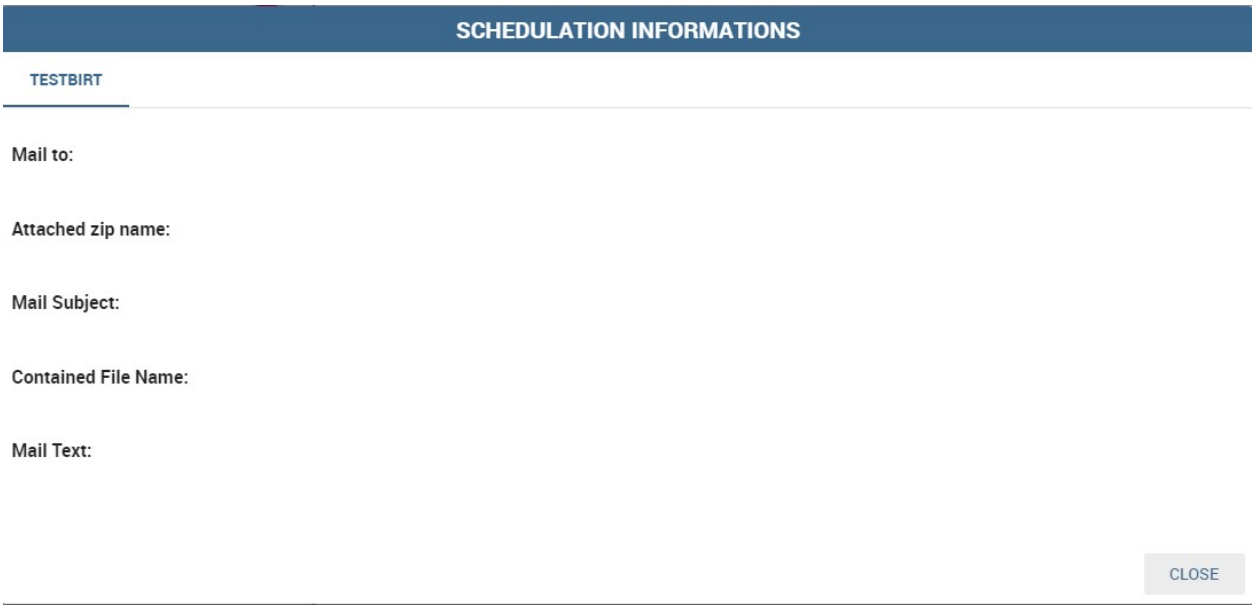


Fig. 3.26: Schedulation information pop up example

- **Execute now**, by clicking it you immediatly 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 Schedulation**, 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

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.

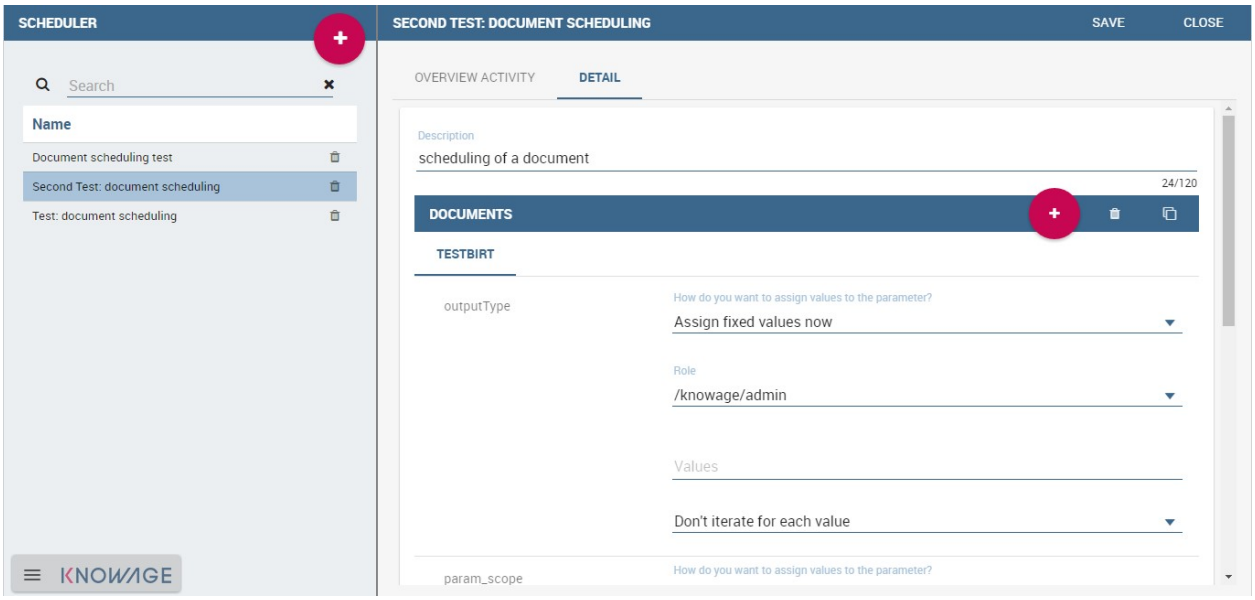


Fig. 3.27: Schedulation detail tab

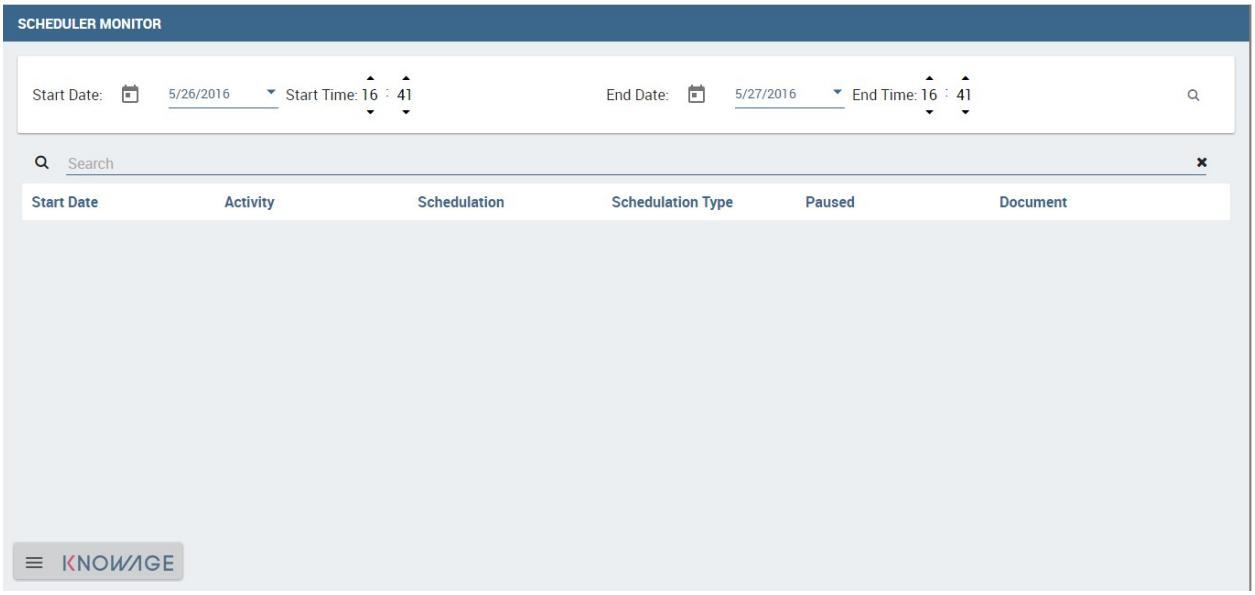


Fig. 3.28: Schedulation detail tab

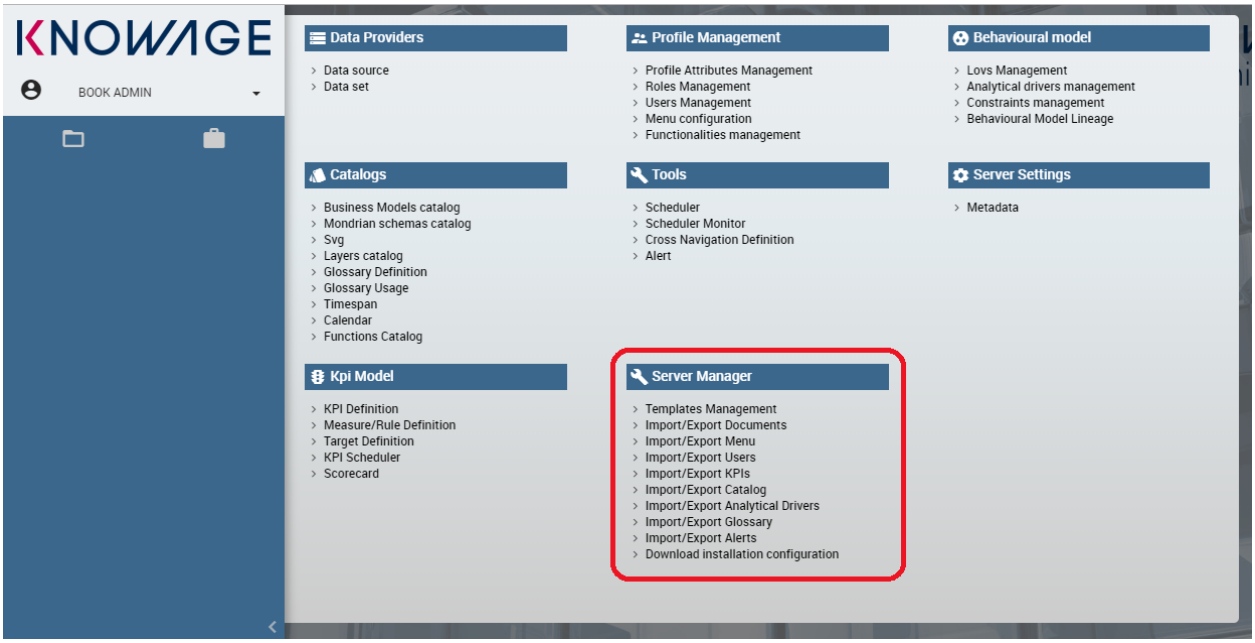


Fig. 3.29: Server Manager Panel

Those about **Import/Export** let you export some configurations or elements from one istallation 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 funtionalities 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-tentant 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 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.

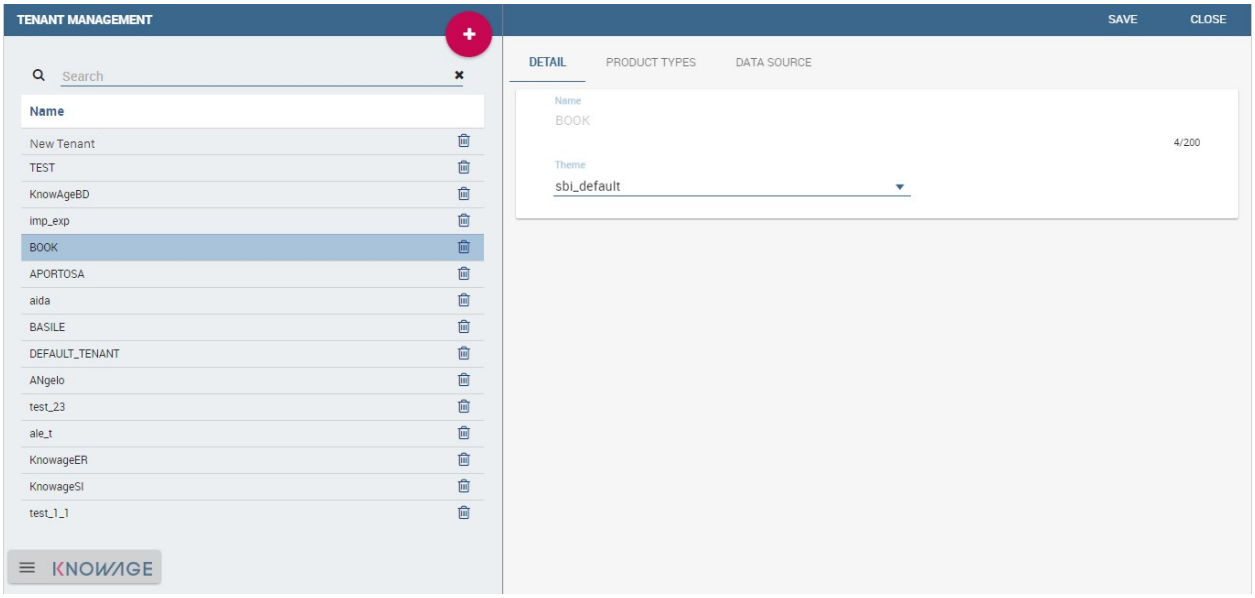


Fig. 3.30: Tenants Management window.

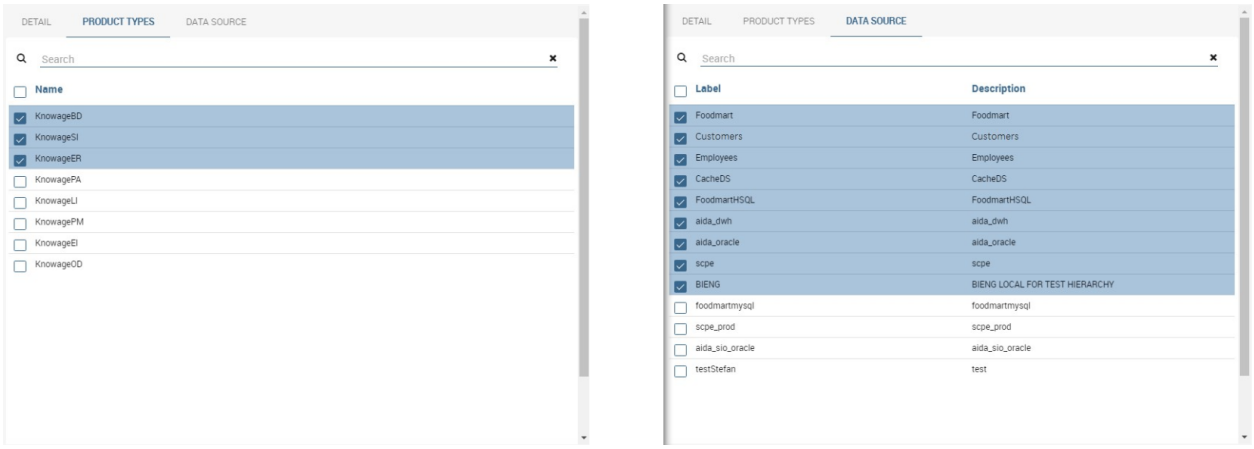


Fig. 3.31: Product types tab (Left) Datasources tab (Right).

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 childer folders/leaves.

When you have chose a name and select some documents the export icon change colour from gray 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 **CrossNavigation** and/or **BIRT Translation** and/or **Schedule configurations** .

- **Olap customized View** Clicking on this functionalities 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** Clicking on this functionalities 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:
- **CrossNavigation** Clicking on this functionalities the export will include all the cross navigation associated to the chosen documents and the documents related to navigation.
- **BIRT Translation** Clicking on this functionalities the export will include all the translation added into 'Localization' functionalities of the BIRT templates.
- **Schedule configurations** Clicking on this functionalities 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.

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

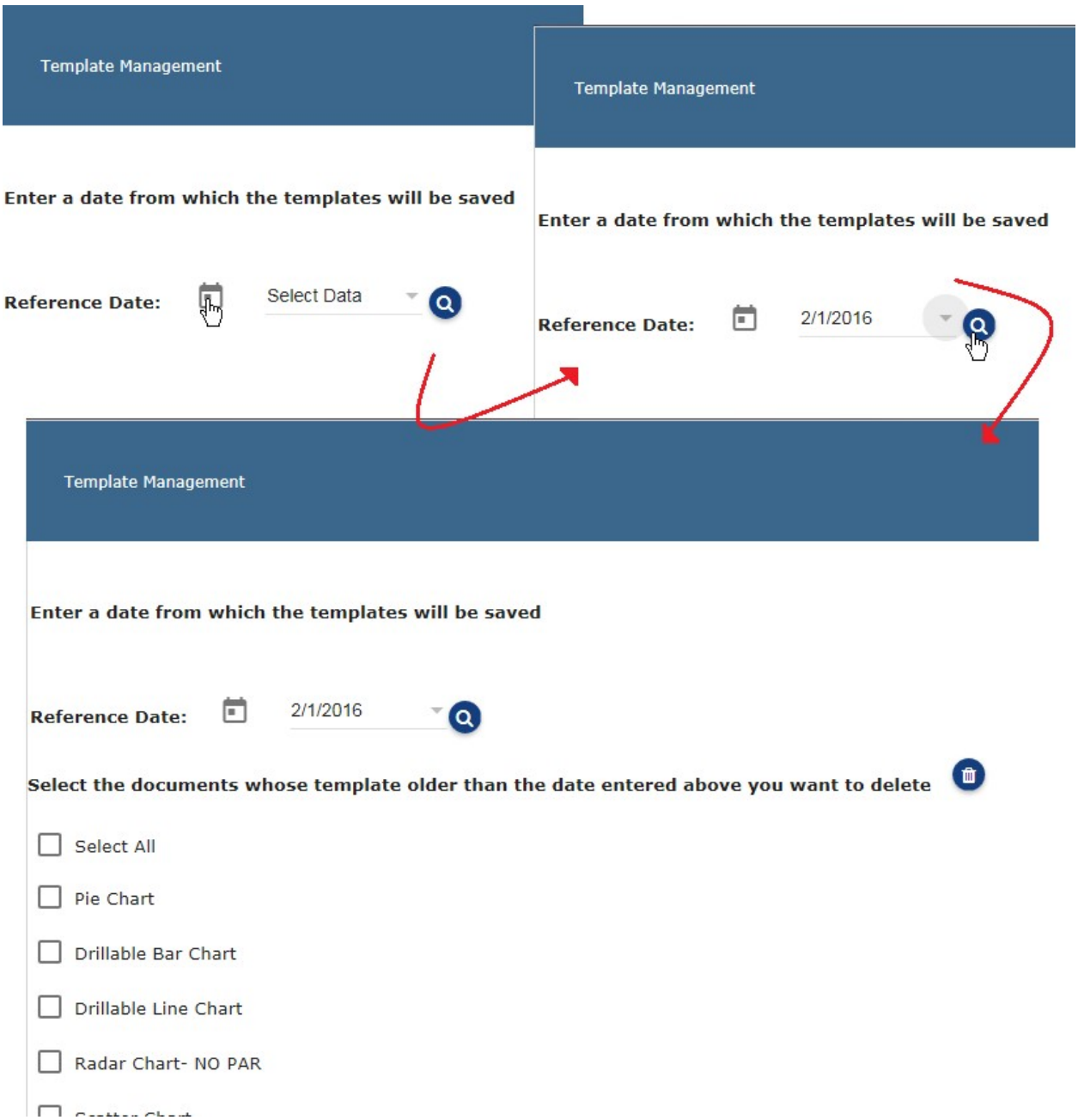


Fig. 3.32: Deleting templates

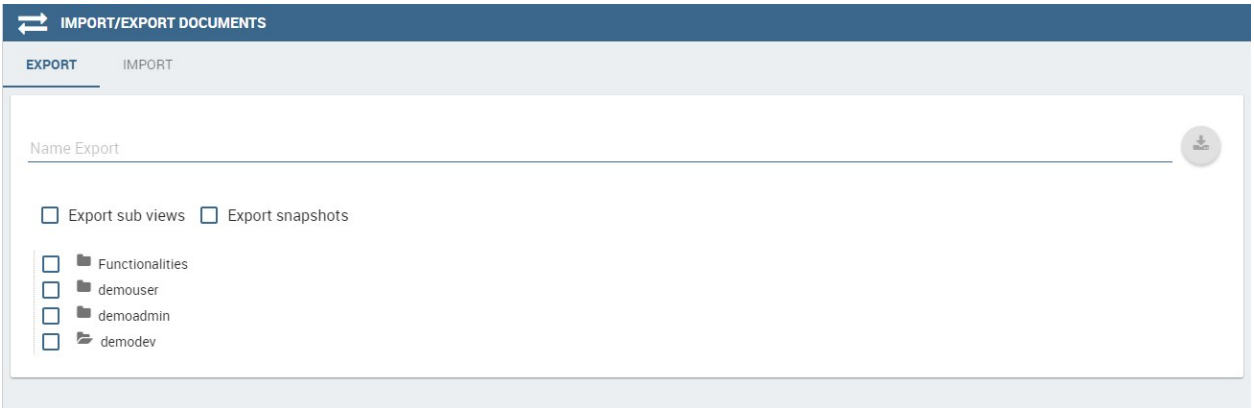


Fig. 3.33: Document Export

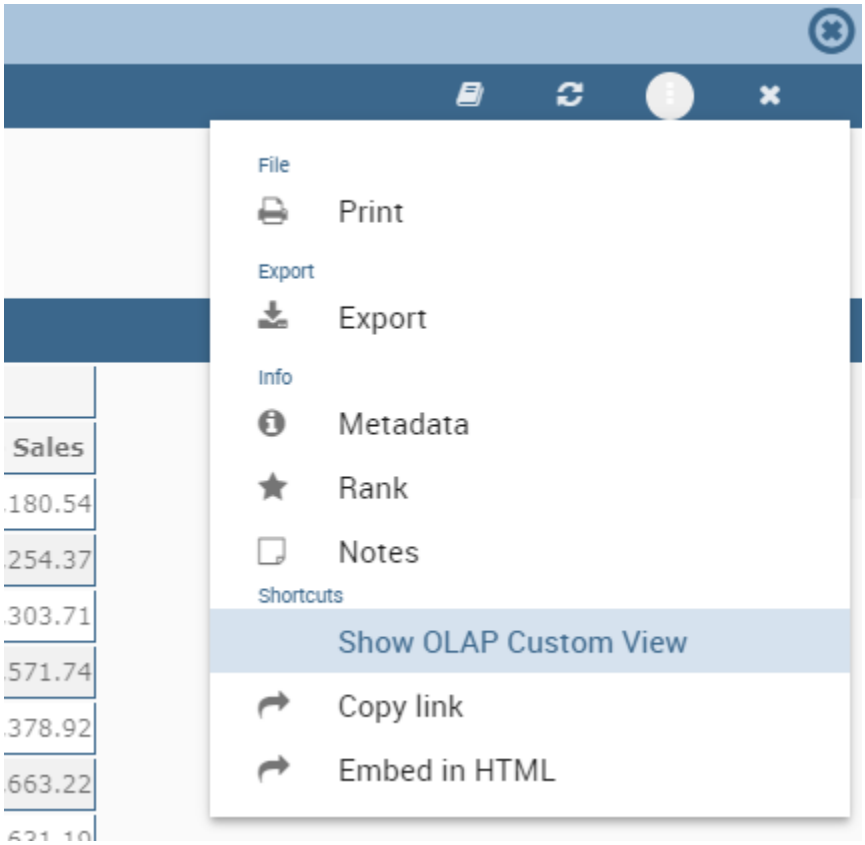


Fig. 3.34: Olap customized view

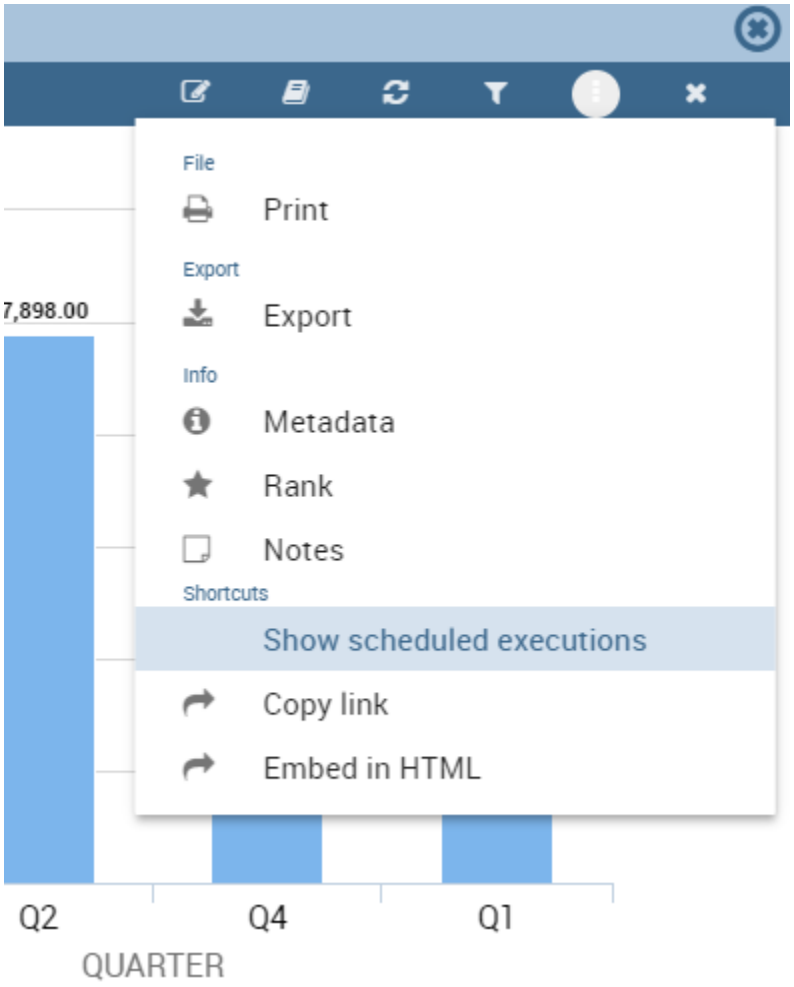


Fig. 3.35: Scheduled documents

Switch to the **Import** tab and click on **Browse** to accede your personal folders. In Figure below we show the document import interface.

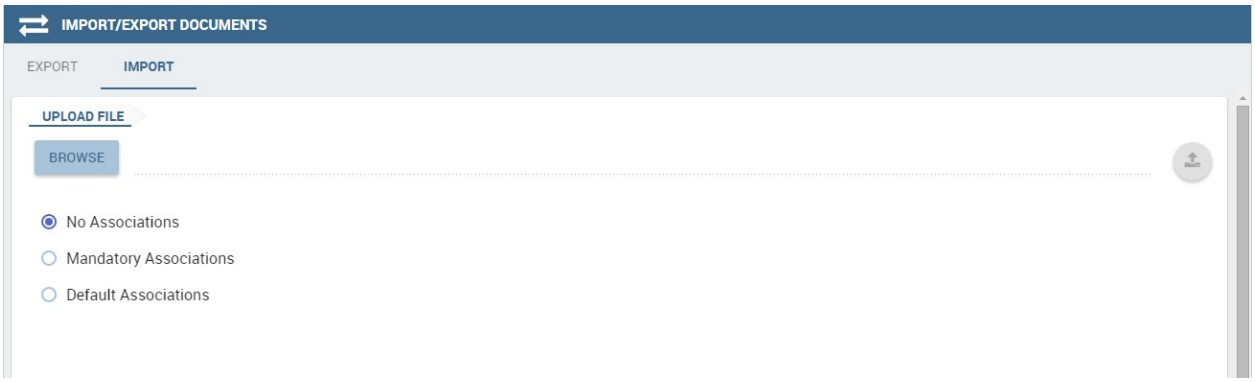


Fig. 3.36: Document Import

Choose the .zip obtained from the **Export** phase and click on the import icon. Few steps guide you trought importation. You are asked to map from source to target: the Roles, the Engines and the Metadata. If a role doesn't map any of the existing among the target one, it will be created. Please keep attention during the metadata step beacause you can choose to overwrite or don't the target metadata. By default this option is set to false. If you change to yes documents, lov, driver, etc. which has the same label of the exported ones will have metadata overwritten at the end of import procedure.

3.8.3.2 Menu

This feature let you export the menu structure.

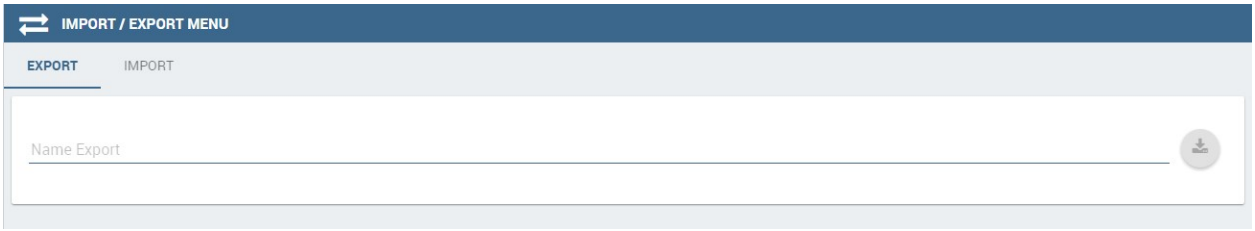


Fig. 3.37: Menu Export

To start the export you need only to insert the Export name. Once inserted the name, the export icon changes colour 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.

To upload it in another istallation, 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.

Then choose between the two import modes: **Override** and **Add Missing**. If you choose **Override**, the menu items which match with existing ones will be override by the imported. If you choose **Add missing** only the menu items which don't match with the existing one will be added. You are ready to start importation by clicking on **Start Import**.

3.8.3.3 Users

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

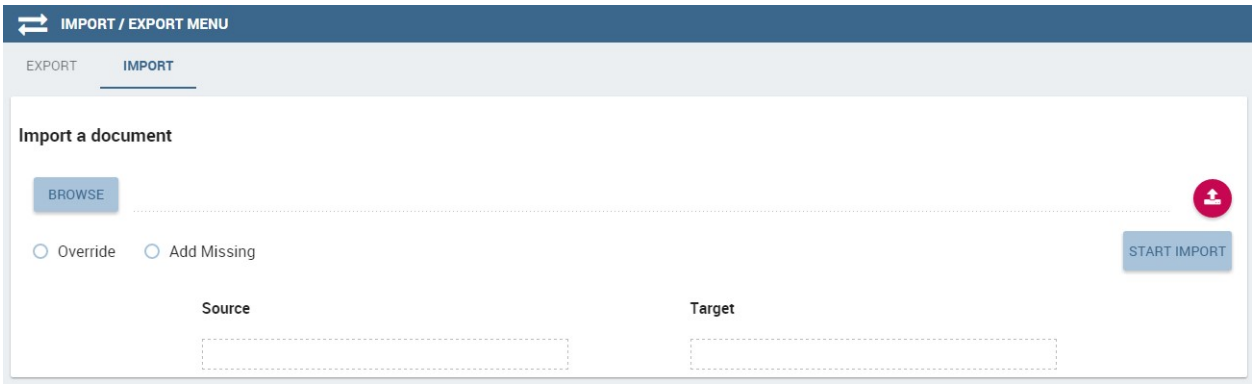


Fig. 3.38: Menu Import

Fig. 3.39: 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.

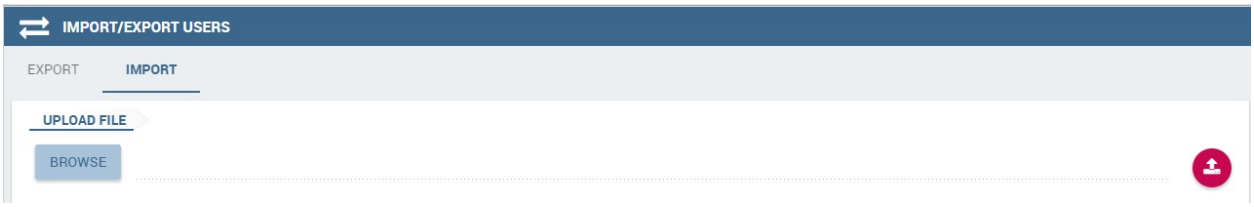


Fig. 3.40: User Import

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.

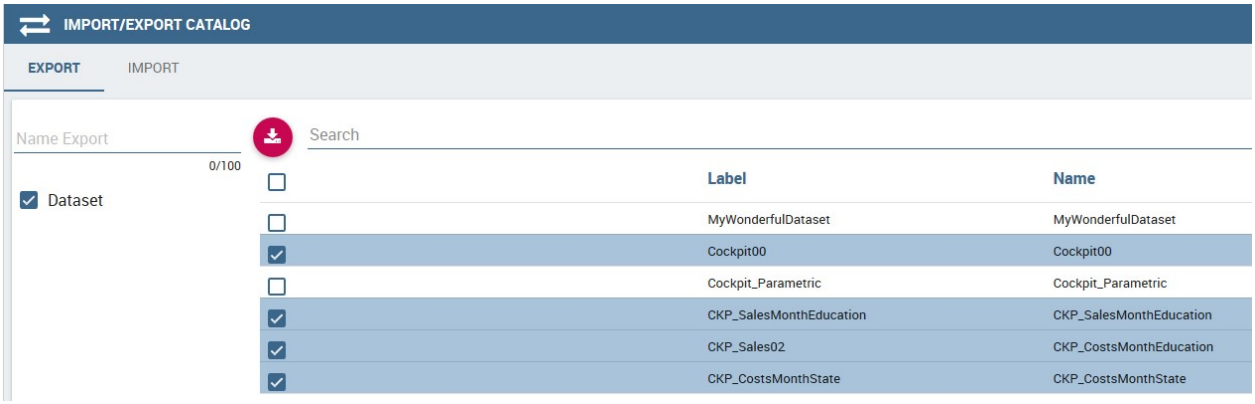


Fig. 3.41: Catalogs Export

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.

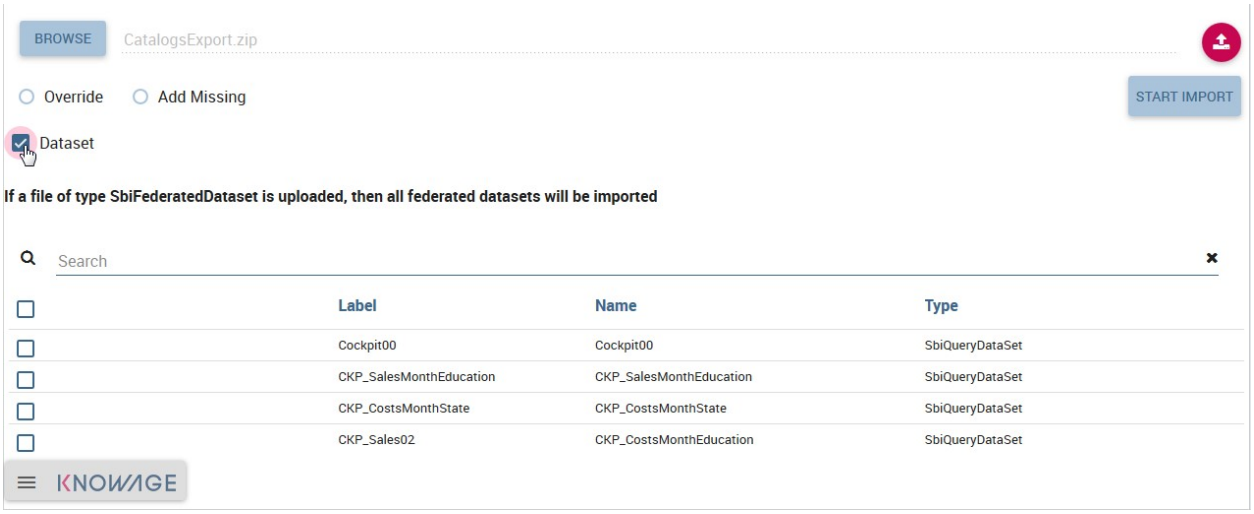


Fig. 3.42: Catalogs Import

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.

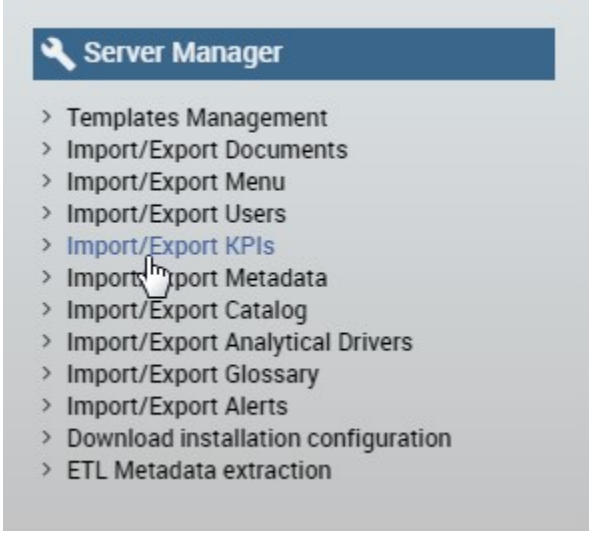


Fig. 3.43: KPIs Import/Export from menu

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:

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

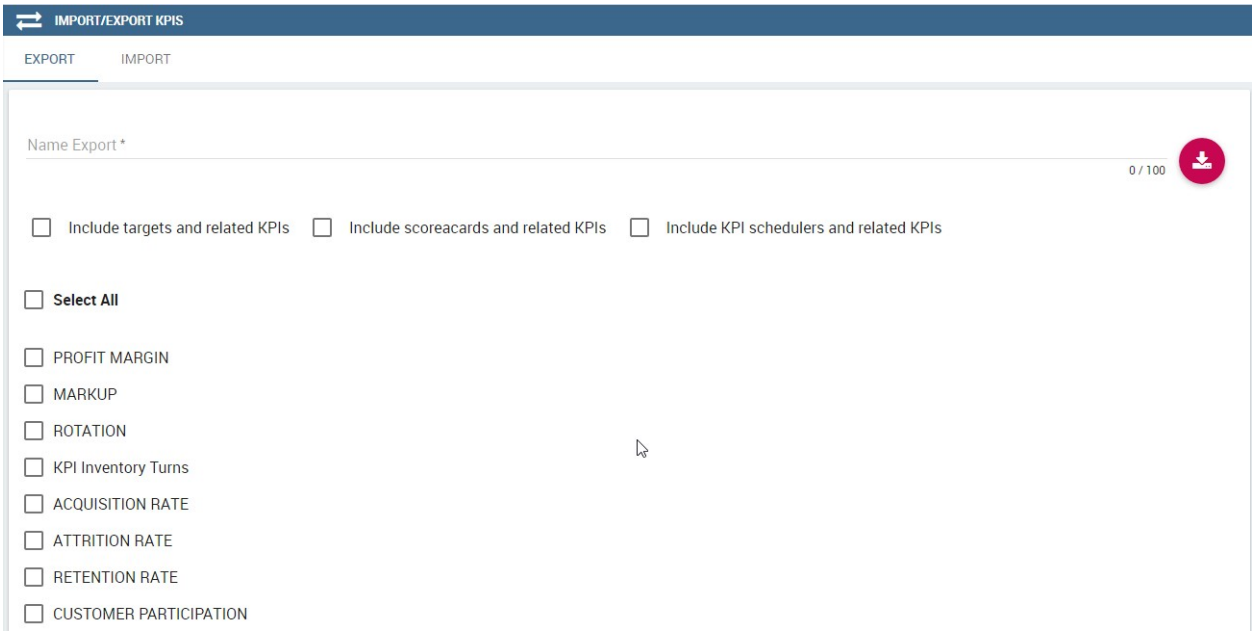


Fig. 3.44: KPIs Import window

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

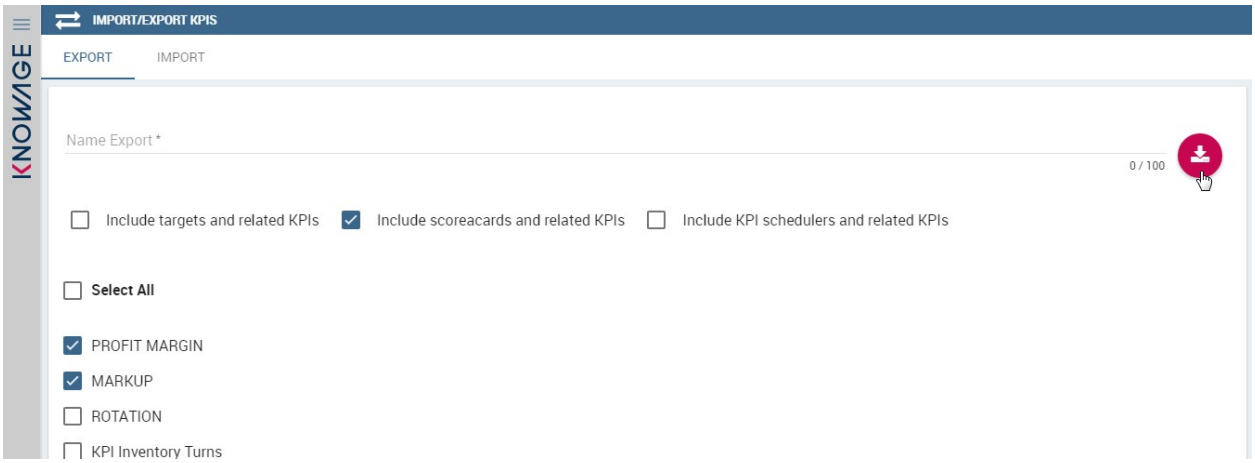


Fig. 3.45: Start export button

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

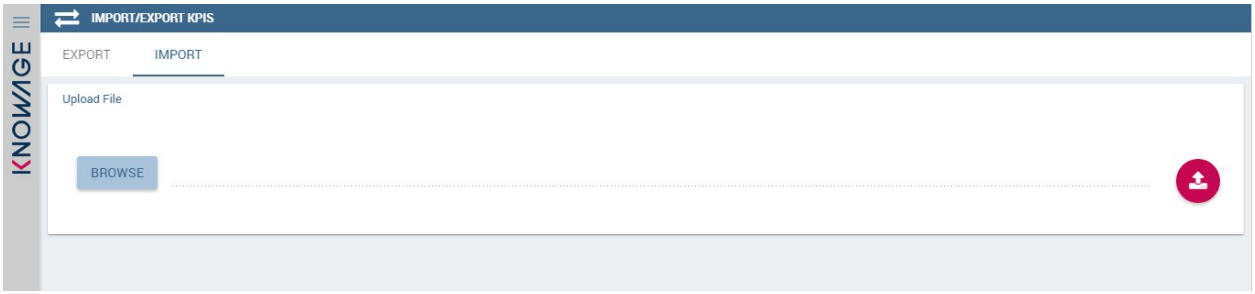


Fig. 3.46: Import tab

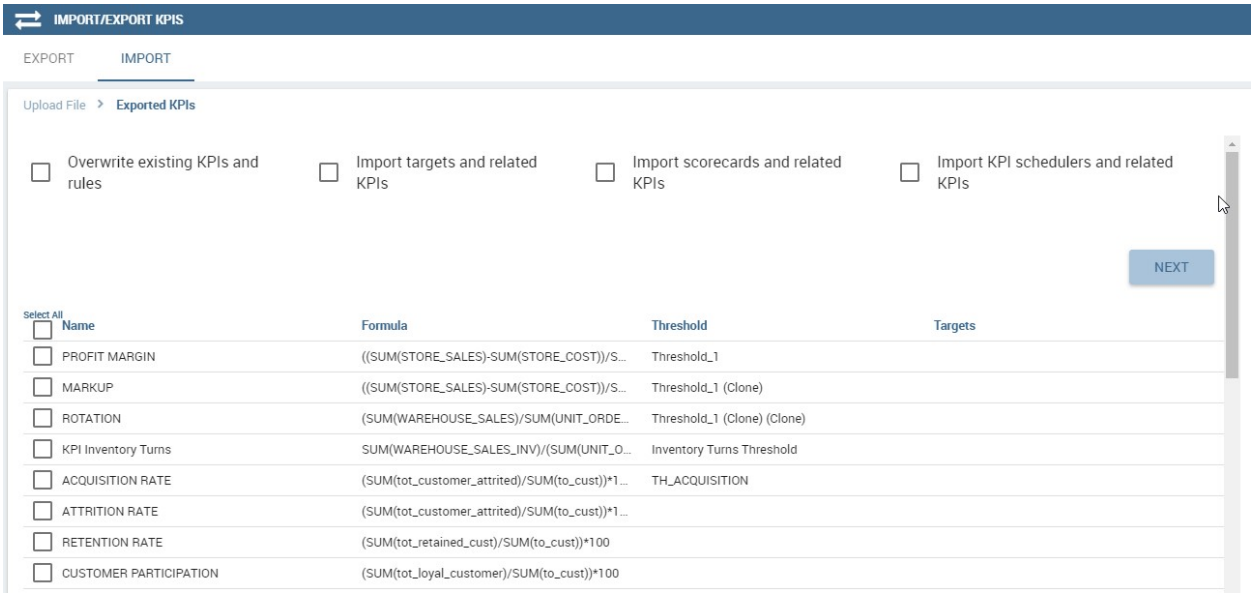


Fig. 3.47: Import KPIs settings

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

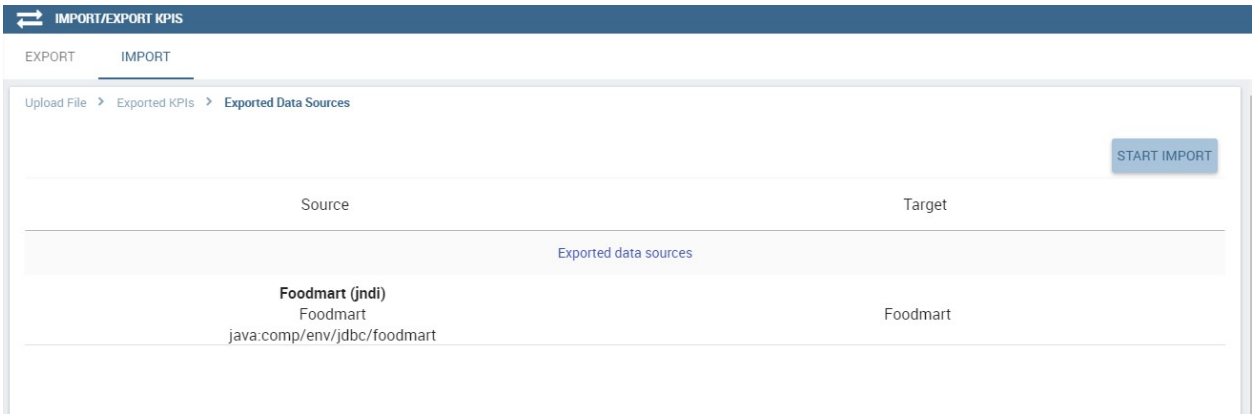


Fig. 3.48: Mapping data sources

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

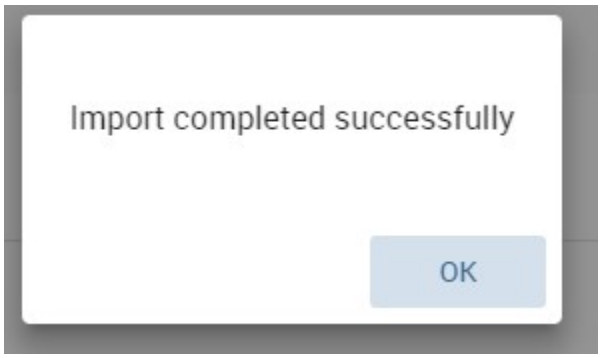


Fig. 3.49: Import KPIs ended successfully

3.8.3.6 Analytical Drivers

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

As shown in Figure anowe, 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 suce a file, the user has to log in as administrator of the source tentant. 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.

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

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

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

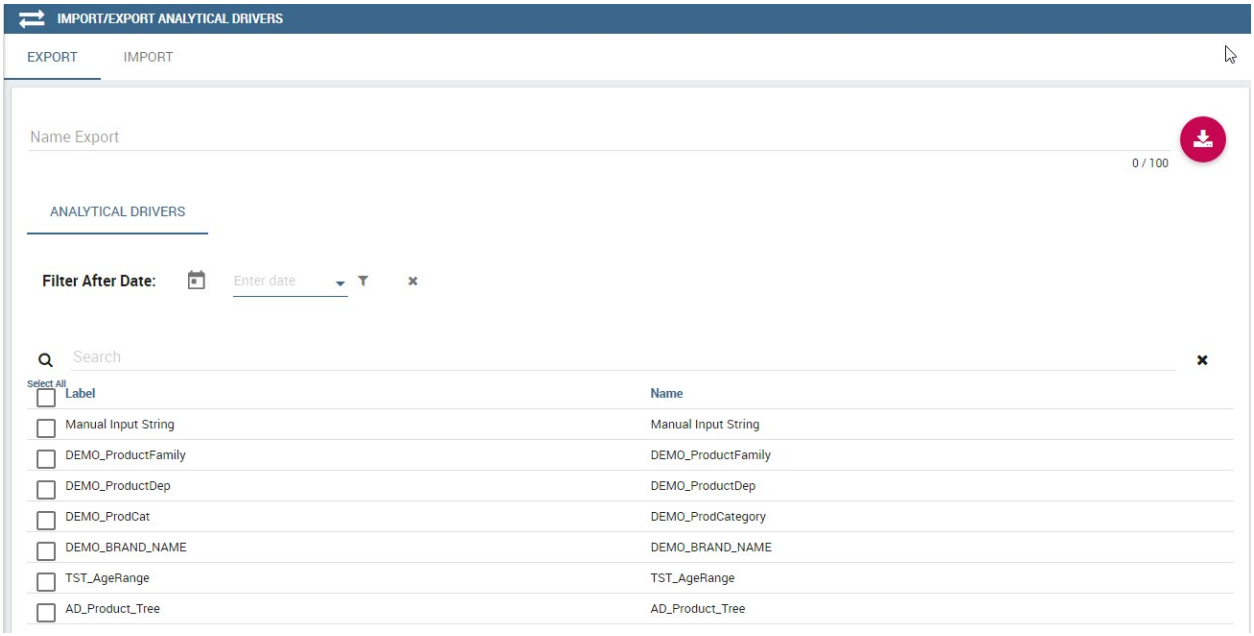


Fig. 3.50: Import/Export of analytical drivers

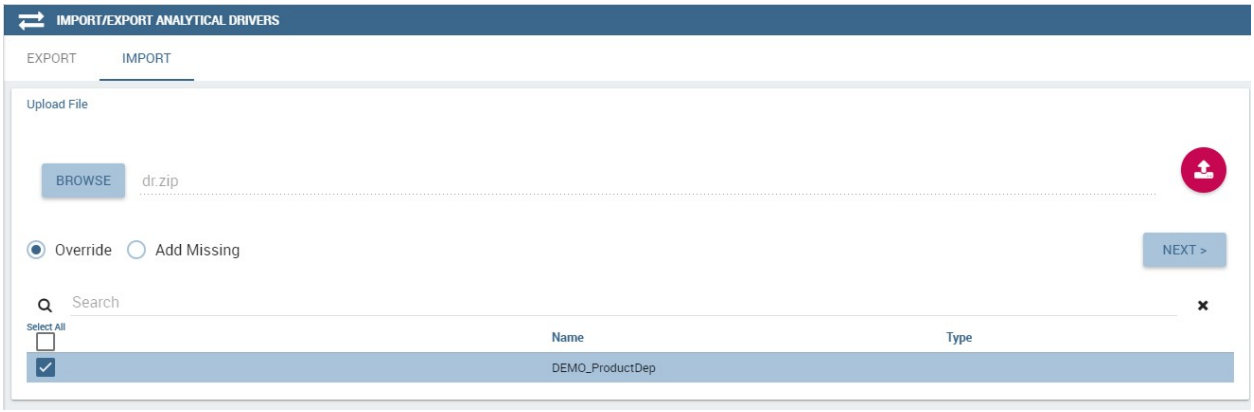


Fig. 3.51: Import of analytical drivers

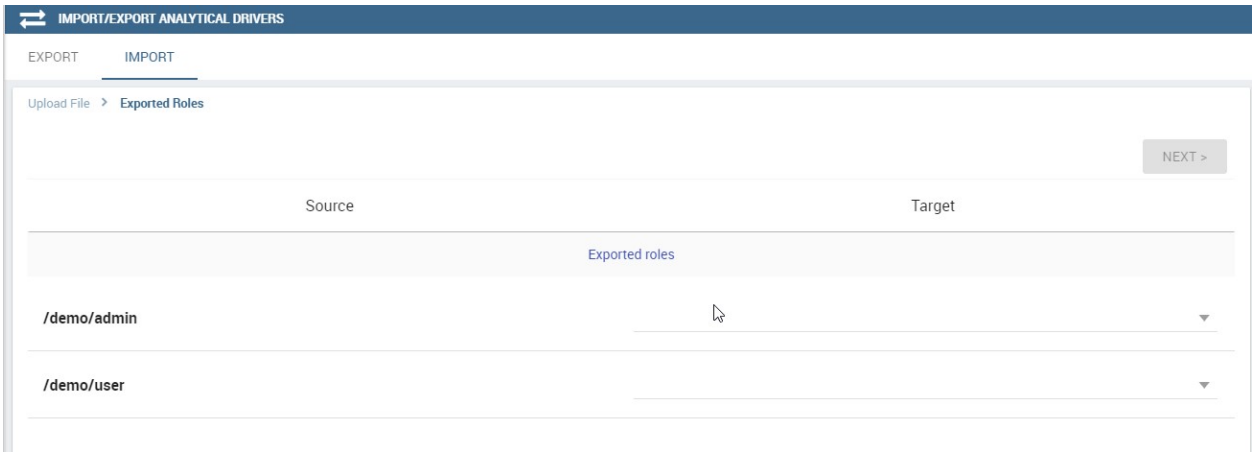


Fig. 3.52: Import of analytical drivers

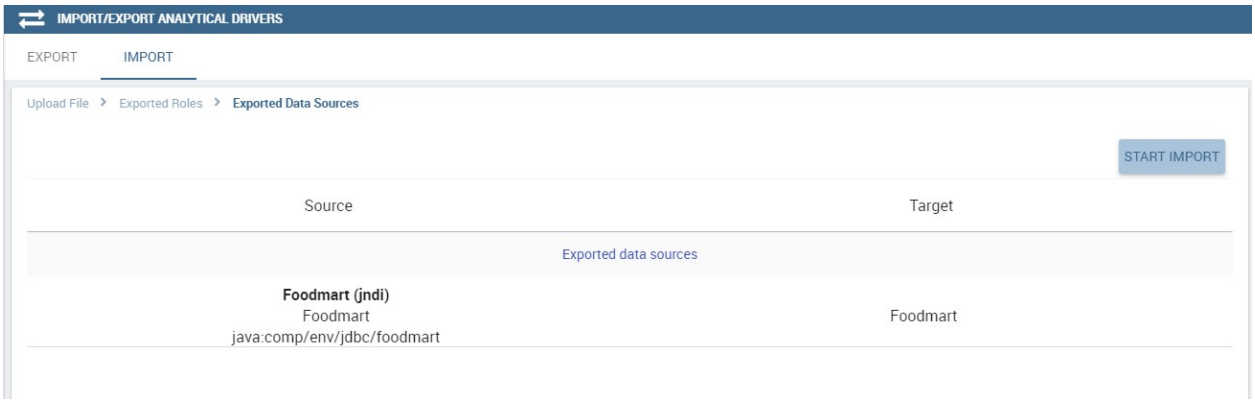


Fig. 3.53: Import of analytical drivers

3.8.3.7 Glossary

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

Fig. 3.54: Export/Import of glossaries window

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 finalise the import.

3.8.3.8 Catalog

This functionality allows to Export/Import the following elements

- Data sets,
- Business models,
- Mondrian catalogs,

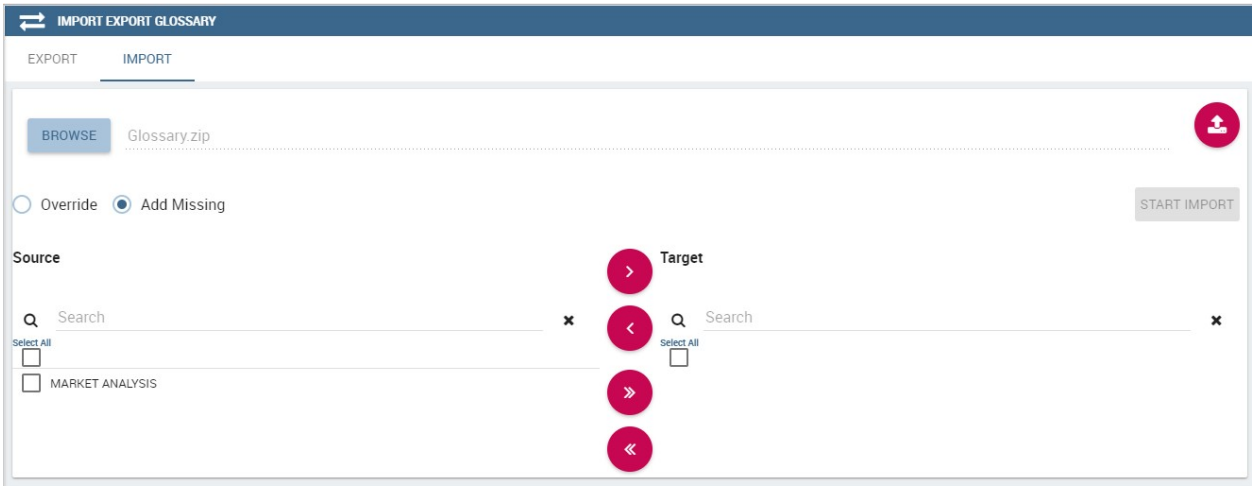


Fig. 3.55: Import of glossaries

- 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.

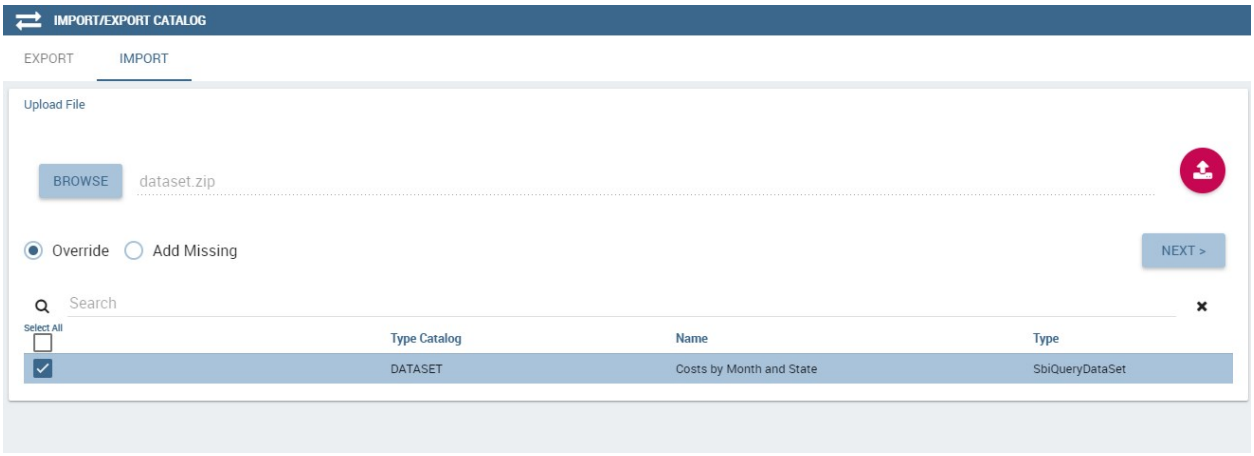


Fig. 3.56: Import of catalog

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

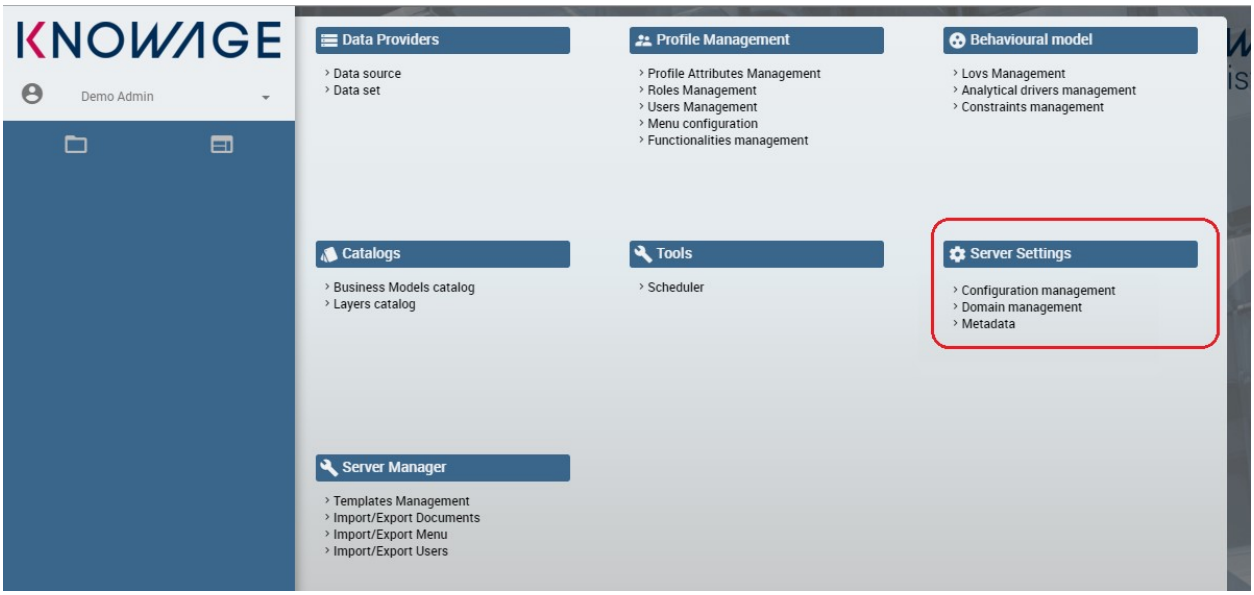


Fig. 3.57: Server Settings Panel.

let you understand how thier 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.

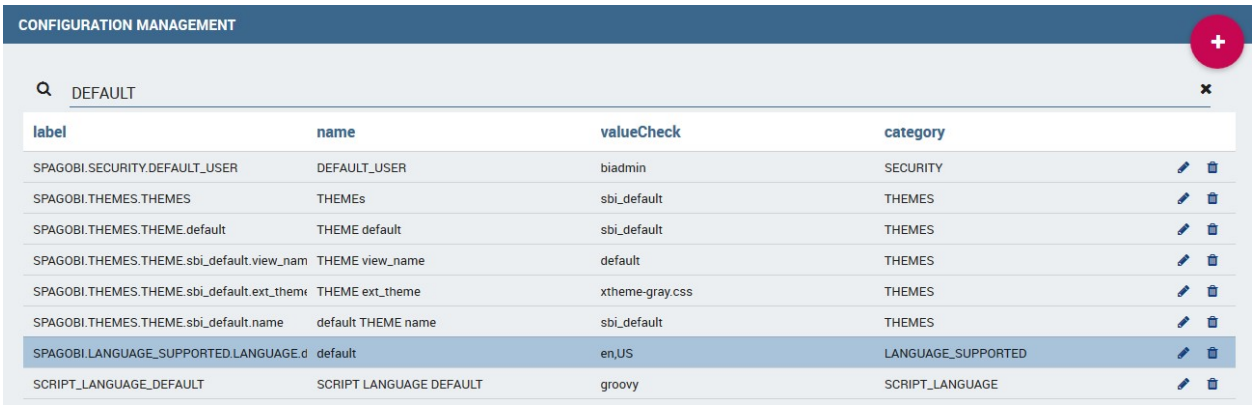


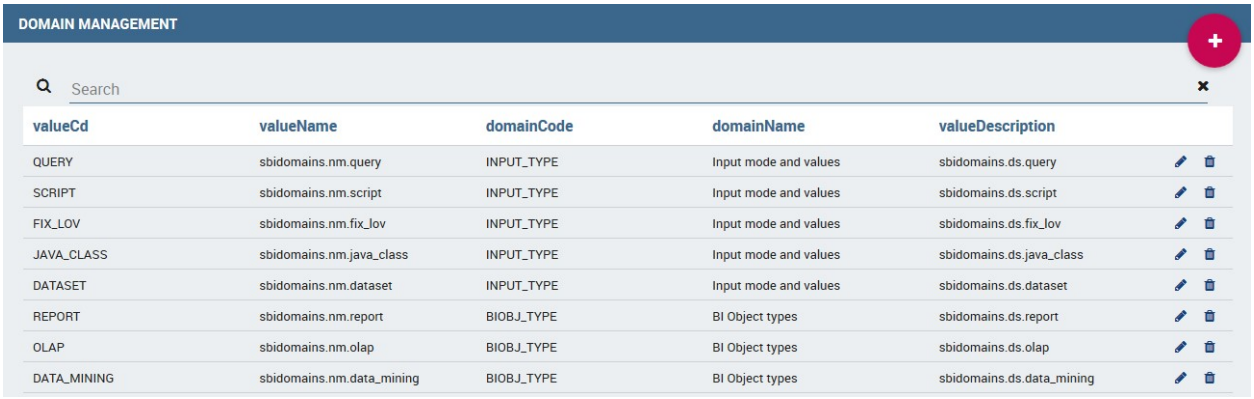
Fig. 3.58: Configuration categories list.

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.

3.9.2 Domain Management

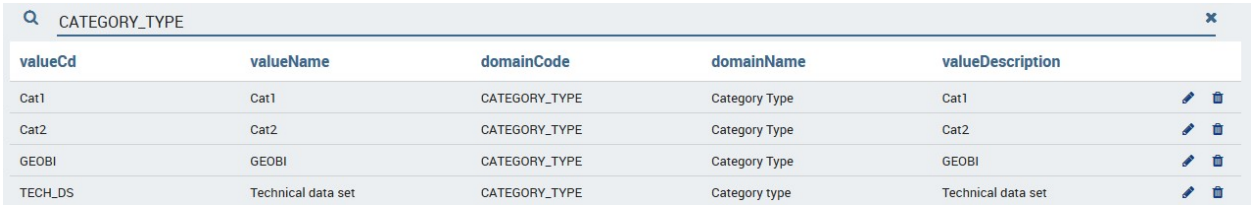
By clicking on **Domains Management** item menu, you can manage categories. In the figure below we show Domain Management editor. You can add for example new categories for a business model, for a dataset and for all domain you can see in the **Domain name column**.



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. 3.59: 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.



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. 3.60: 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
- **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 Metadata

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

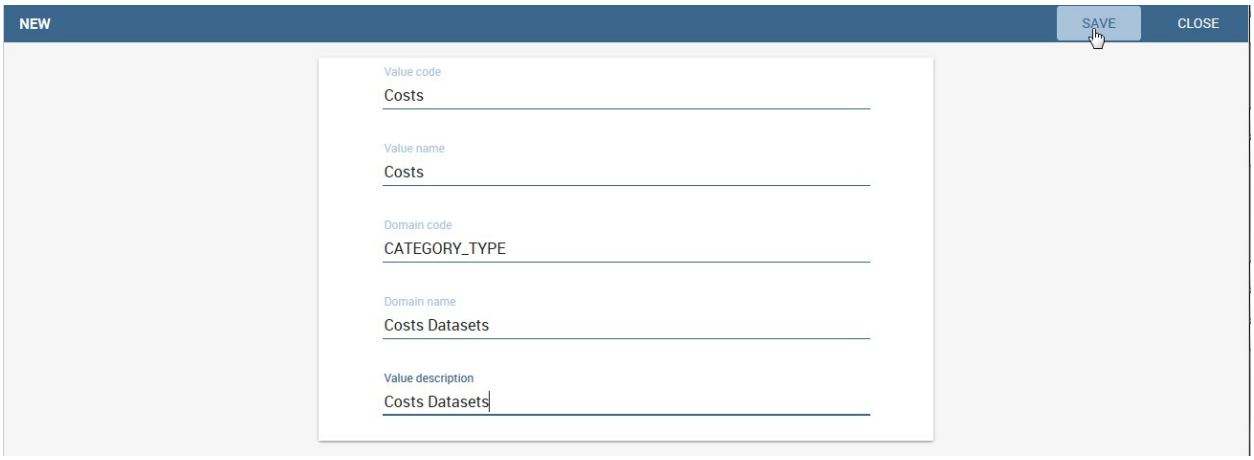


Fig. 3.61: Form to be filled to create a new category.

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.

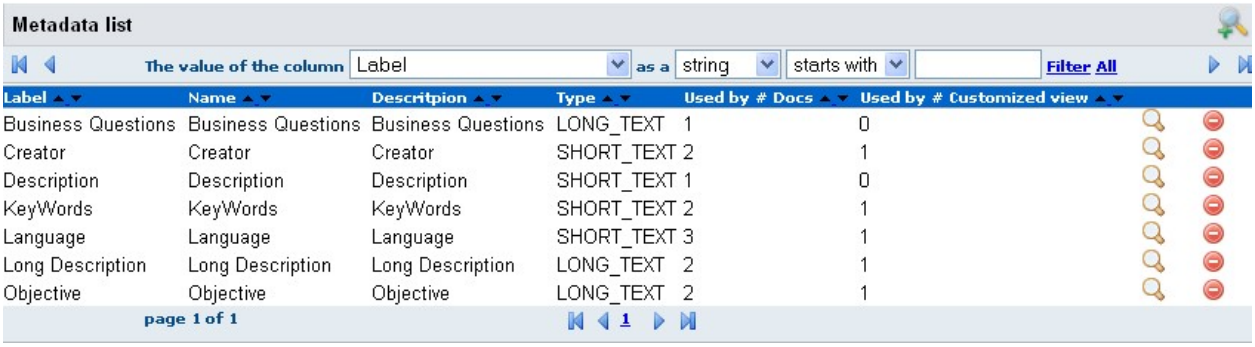


Fig. 3.62: List of existing metadata.

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**.

CHAPTER 4

User Guide

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 visiblity 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.

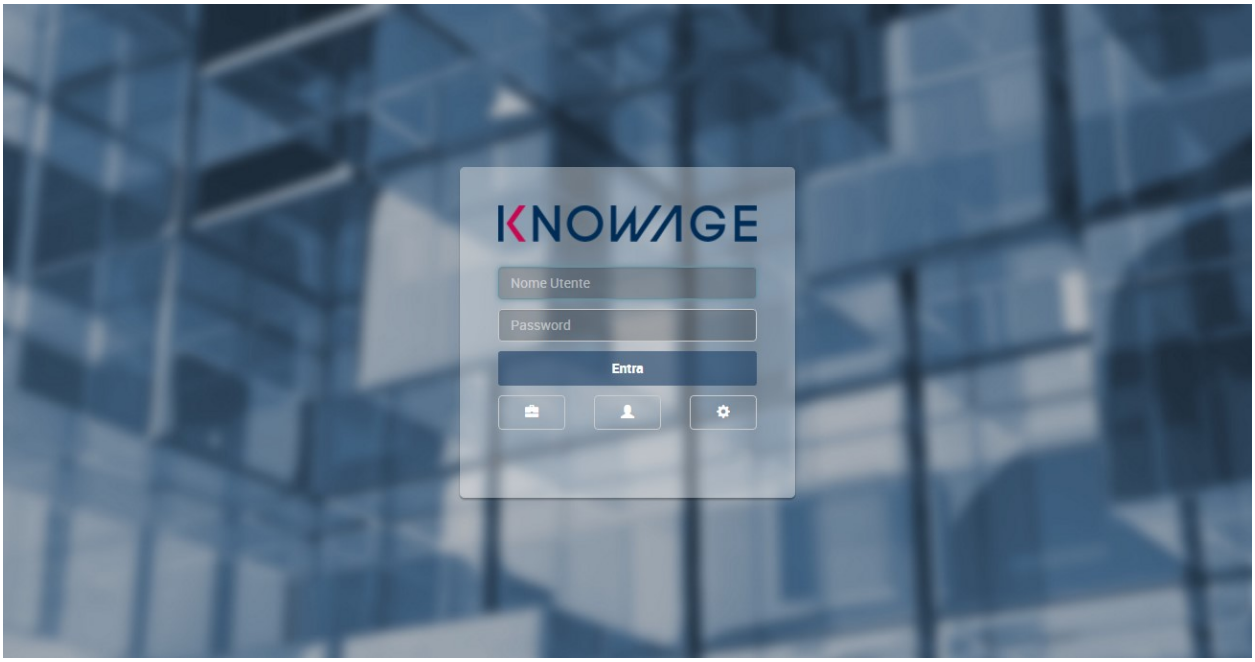


Fig. 4.1: Login page

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: clik 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.

Fig. 4.2: Home page

Table 4.2: Menu components - General menu

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

Table 4.3: Menu components - BI functionalities menu

Icon	Name	Description
	Document browser	Show the archive folders and related documents.
	Workspace	Inquiry, navigate and create your data. Available only for KnowageBD and KnowageSI.
	Functions catalog	Access data mining functions.
	Glossary definition	Access the glossary area. Visualise the existing words and glossaries. Create new words and glossaries.
	Calendar	Access the calendar list. Create a new calendar.

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 previuously 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.

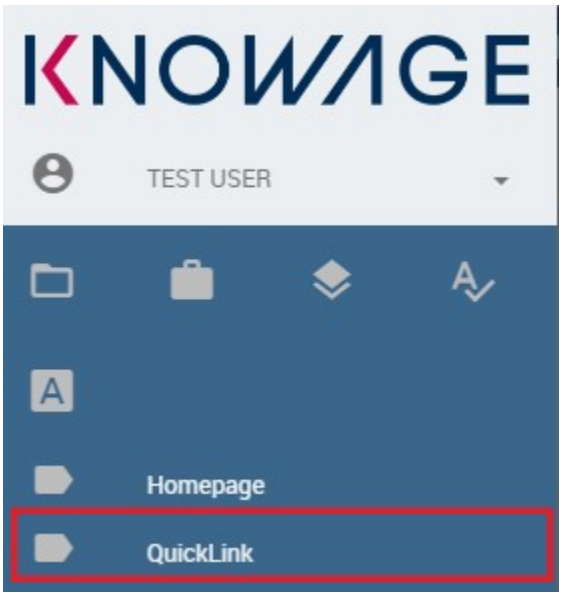


Fig. 4.3: Quick links

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. 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 (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.

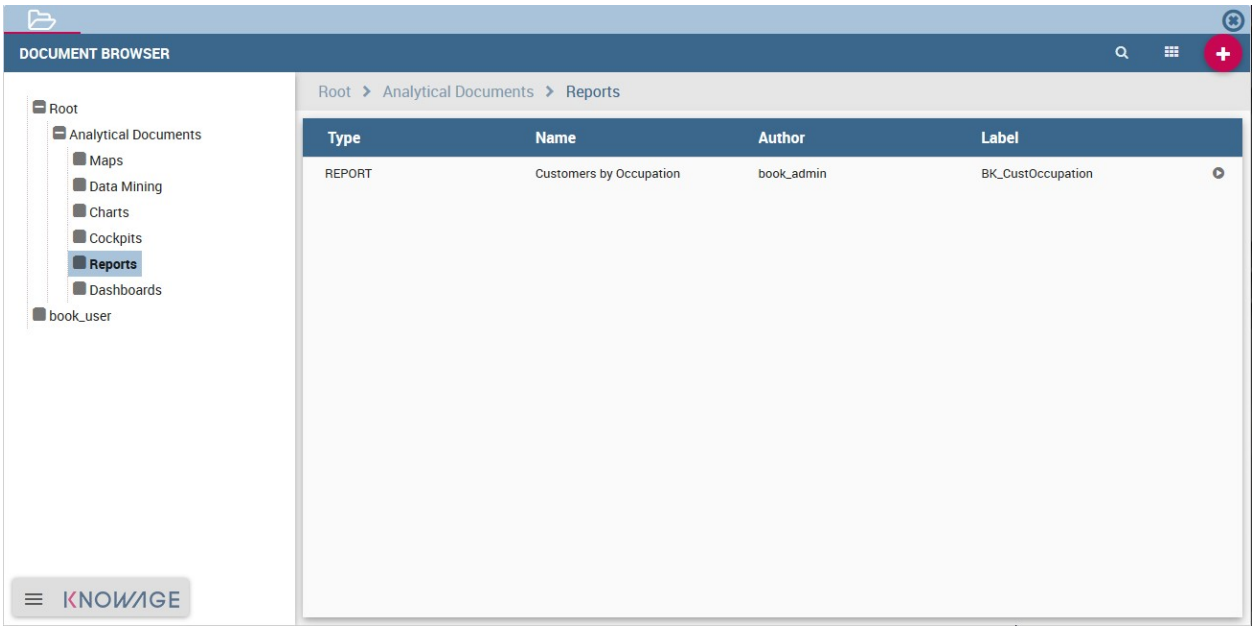


Fig. 4.4: Document Browser

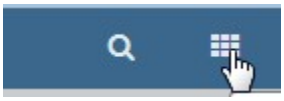


Fig. 4.5: Changing documents view

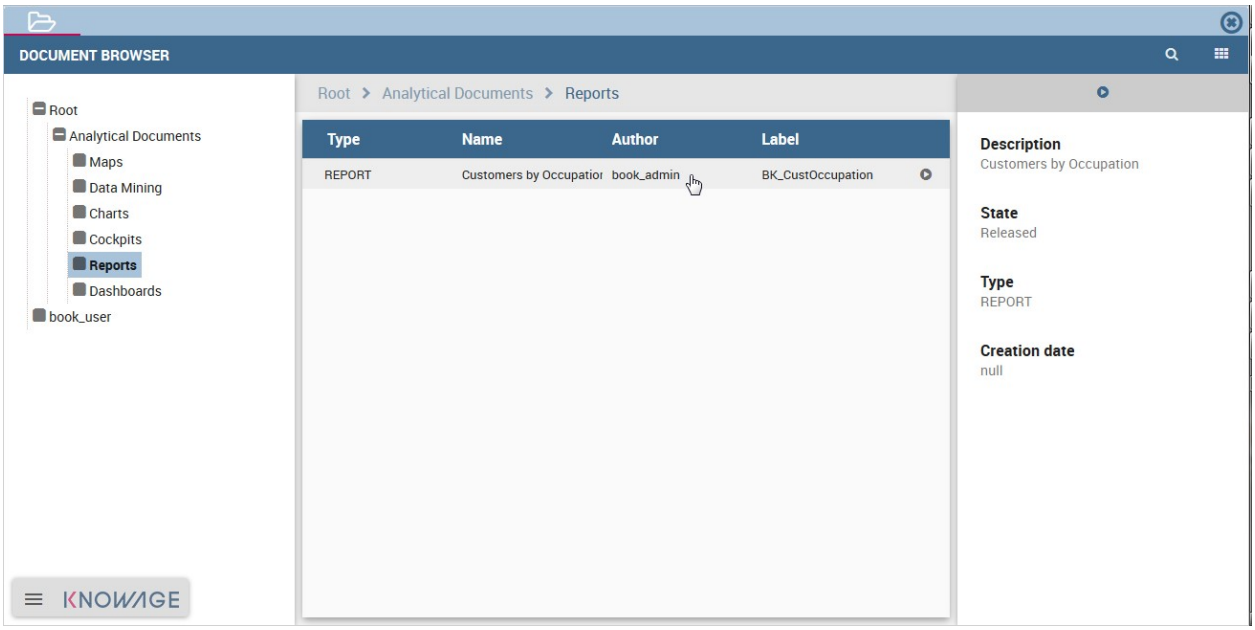


Fig. 4.6: Documents details expanded.

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.

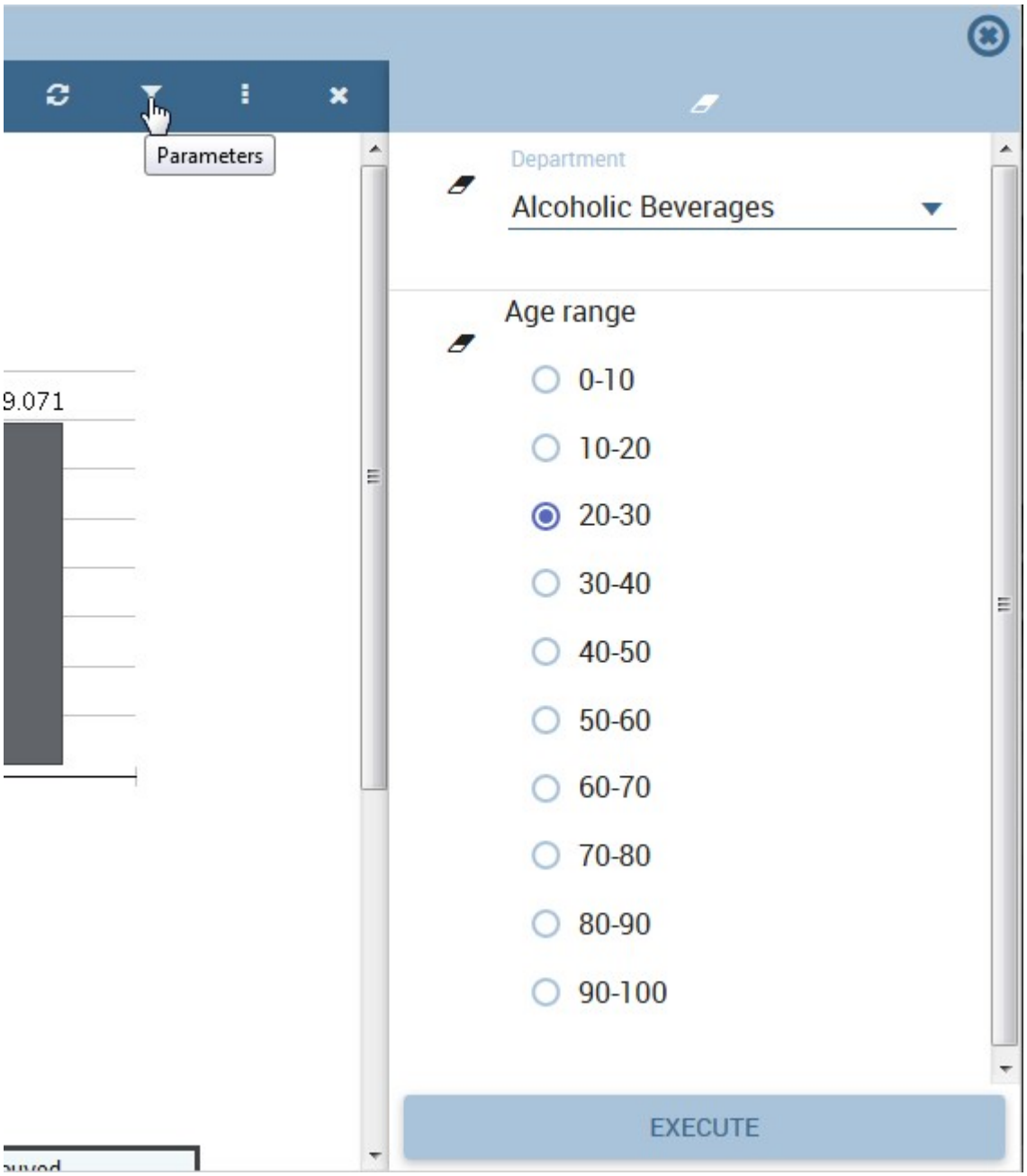


Fig. 4.8: Parameters panel

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.

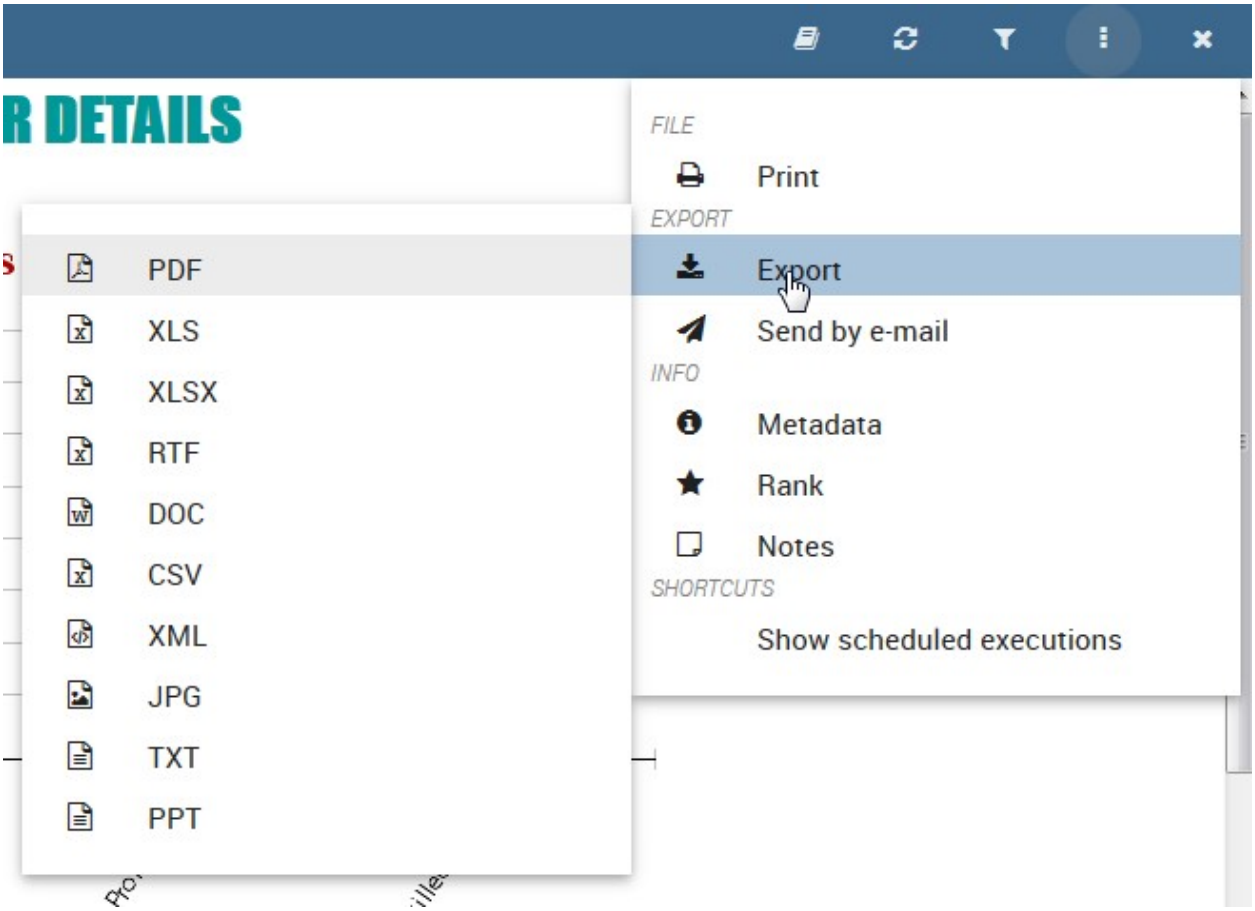


Fig. 4.10: Toolbar contextual menu

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 Notes 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 clickin the corresponding item in the contextual menu as shown in the following figure.

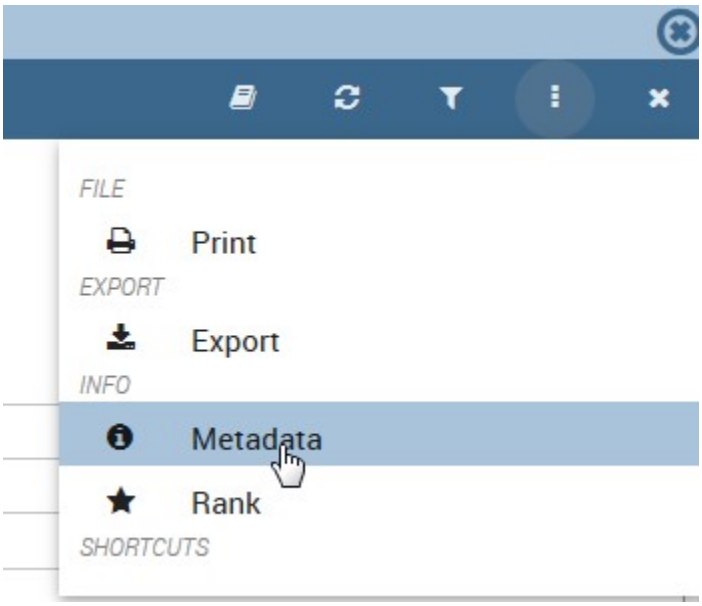


Fig. 4.11: View document metadata

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.

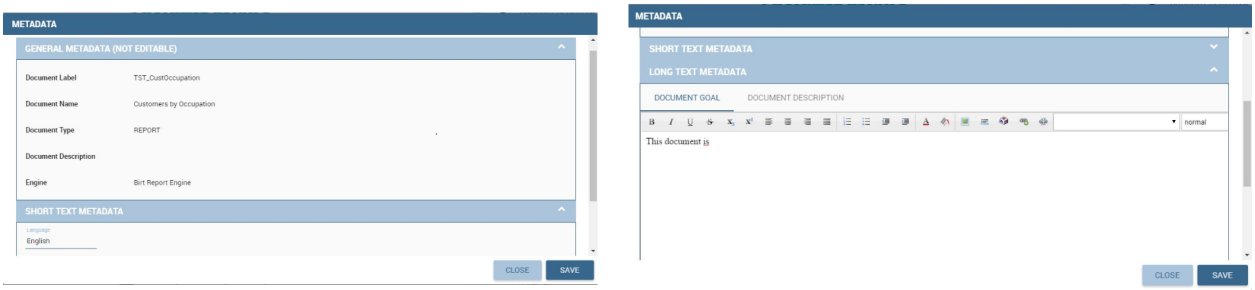


Fig. 4.12: Left: General and short metadata, Right: editing long metadata

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.

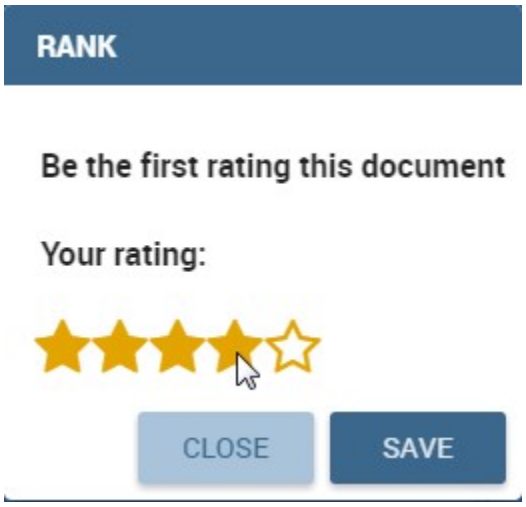


Fig. 4.13: Rating a document

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**.

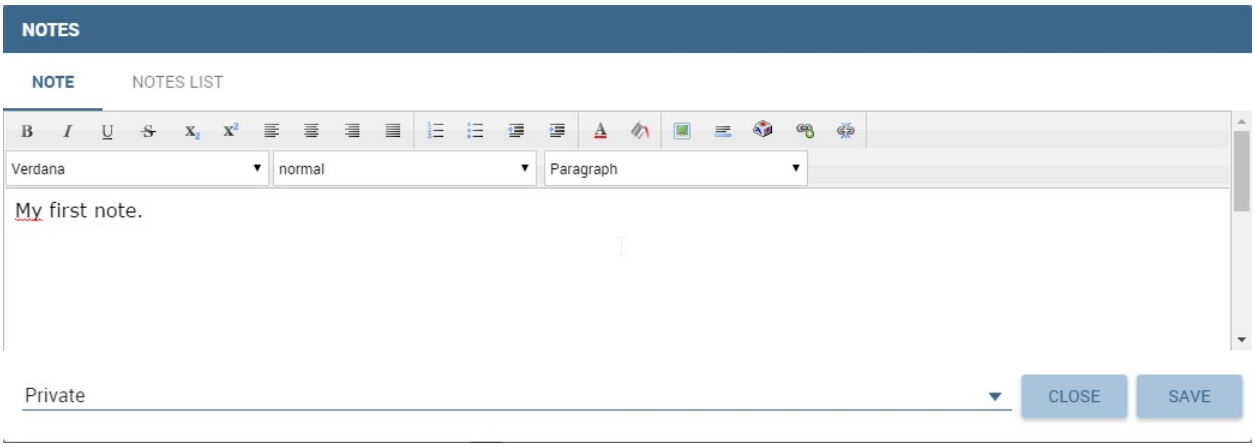


Fig. 4.14: Share notes and comments

CHAPTER 5

Functionalities

5.1 Basic Data Access

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

Important: Enterprise Edition

If you purchased Knowage EE, the following features are available only in KnowageBD and KnowageSI products

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

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.

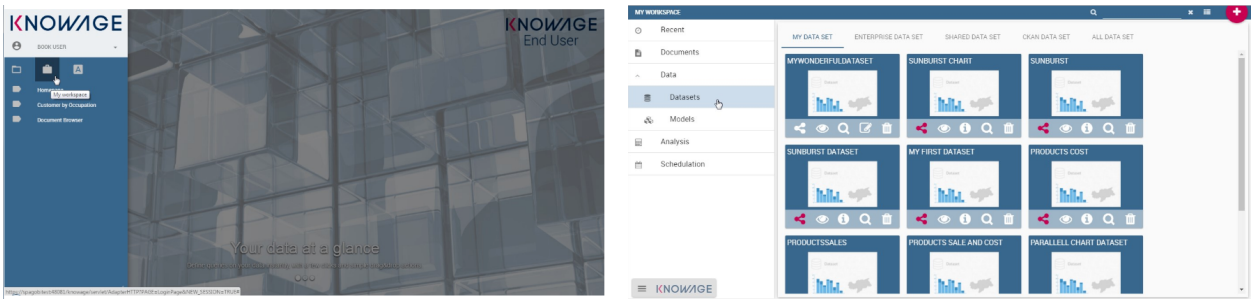


Fig. 5.1: Access to **My Data** area

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 yourself uploading a CSV or XLS file or creating a query on a business model using the Qbe interface;
- **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;
- **CKAN dataset**: in this area you can download public datasets and visualize your CKAN datasets;
- **All dataset**: in this folder are stored all the available datasets, namely all datasets contained in the classes just described.

5.1.1.1 My dataset

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

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

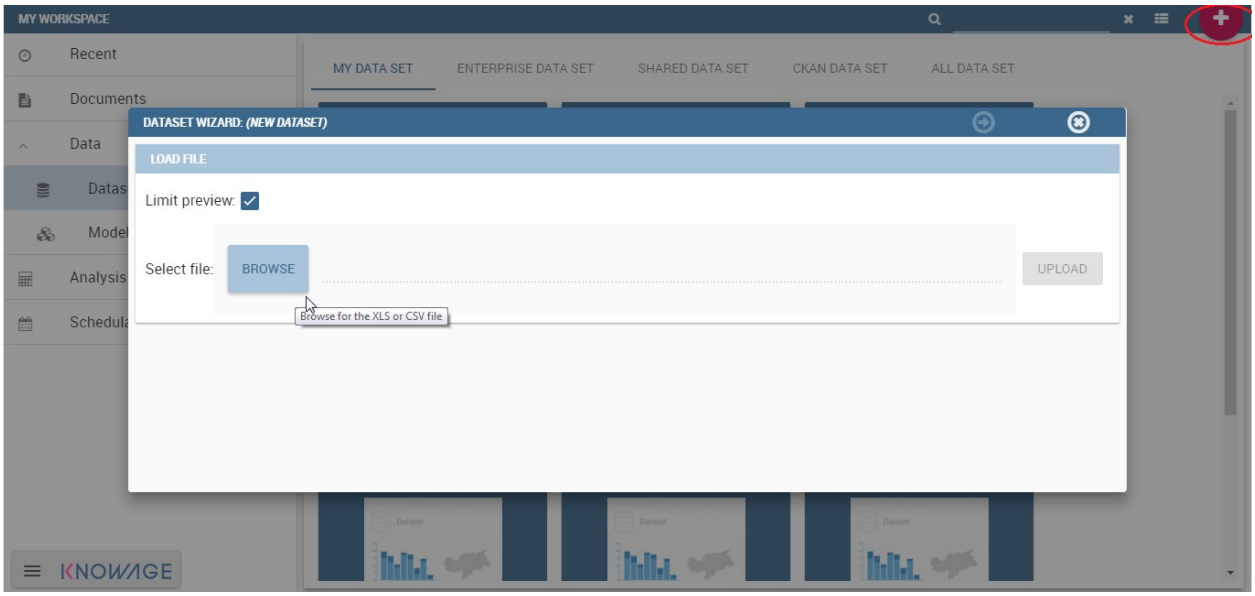


Fig. 5.2: Dataset creation.

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

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.

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

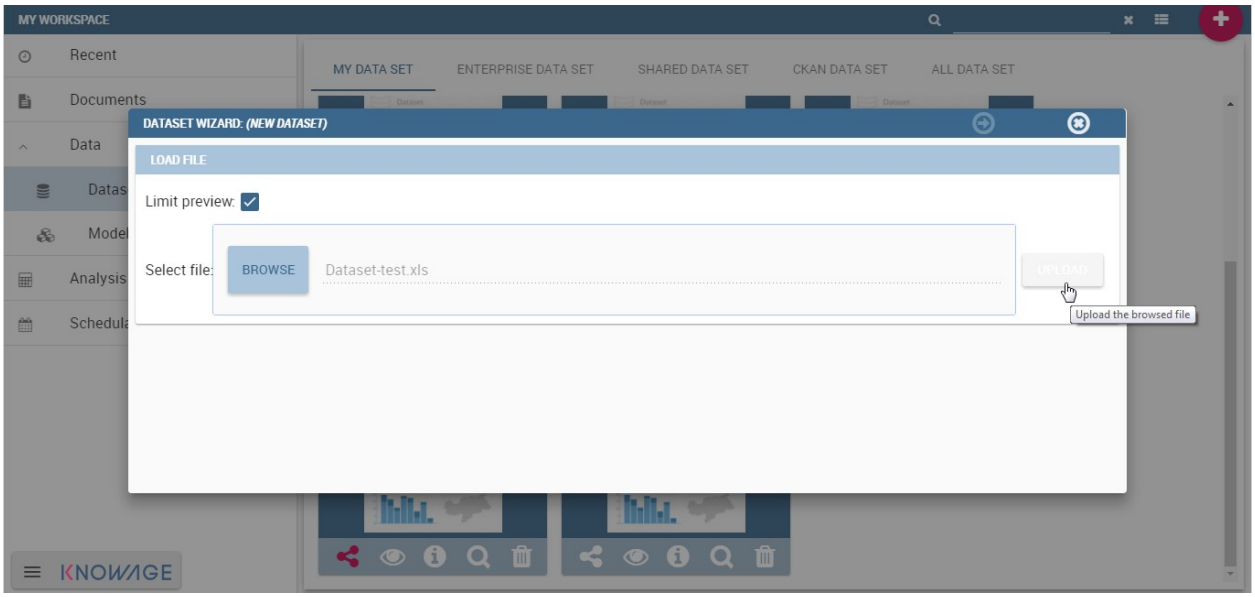


Fig. 5.3: Uploading XLS for dataset.

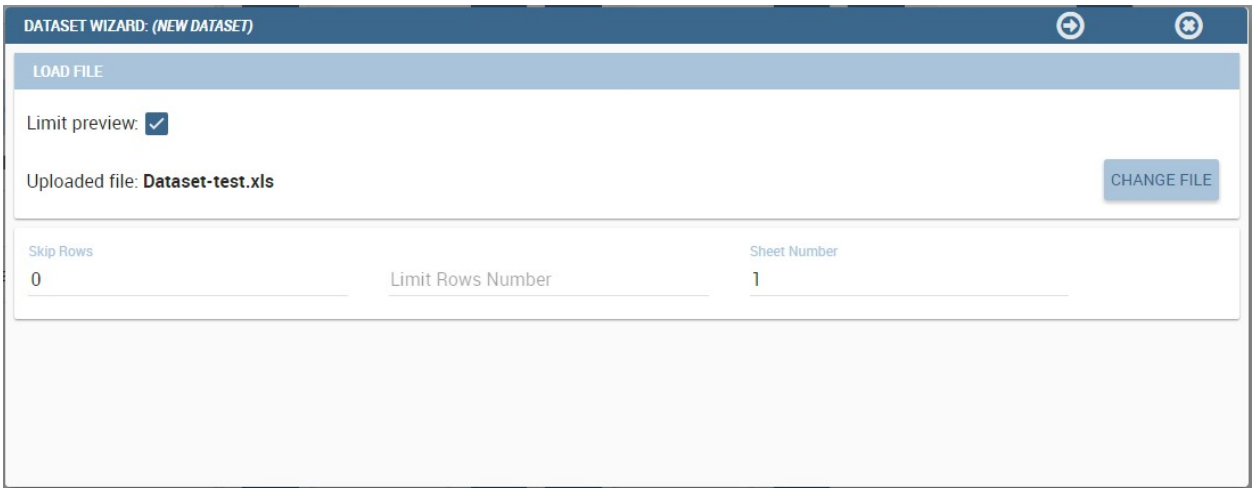


Fig. 5.4: Configuration features.

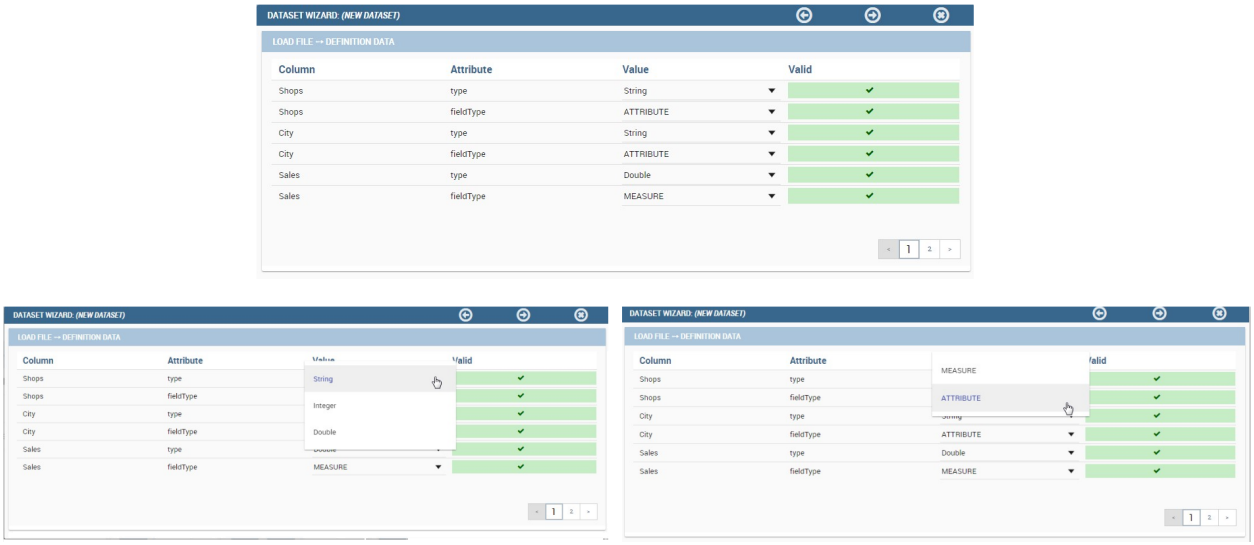


Fig. 5.5: Change metadata.

Just few steps before saving the dataset:

- Check the data preview in order to verify the accuracy of your data;
- enable or disable the persistance 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.

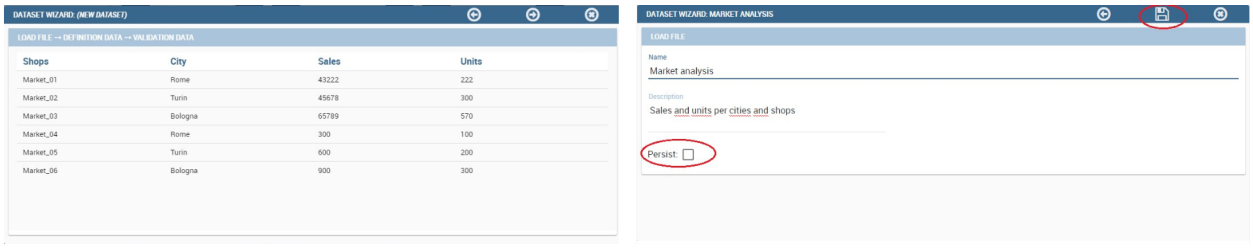


Fig. 5.6: Saving dataset.

As we discussed previously, you find all created datasets under **My dataset** area. You can share/unshare them by clicking on the **share** icon (have a look at 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 role.

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

5.1.1.2 CKAN integration

Thanks to CKAN integration you can easily 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.

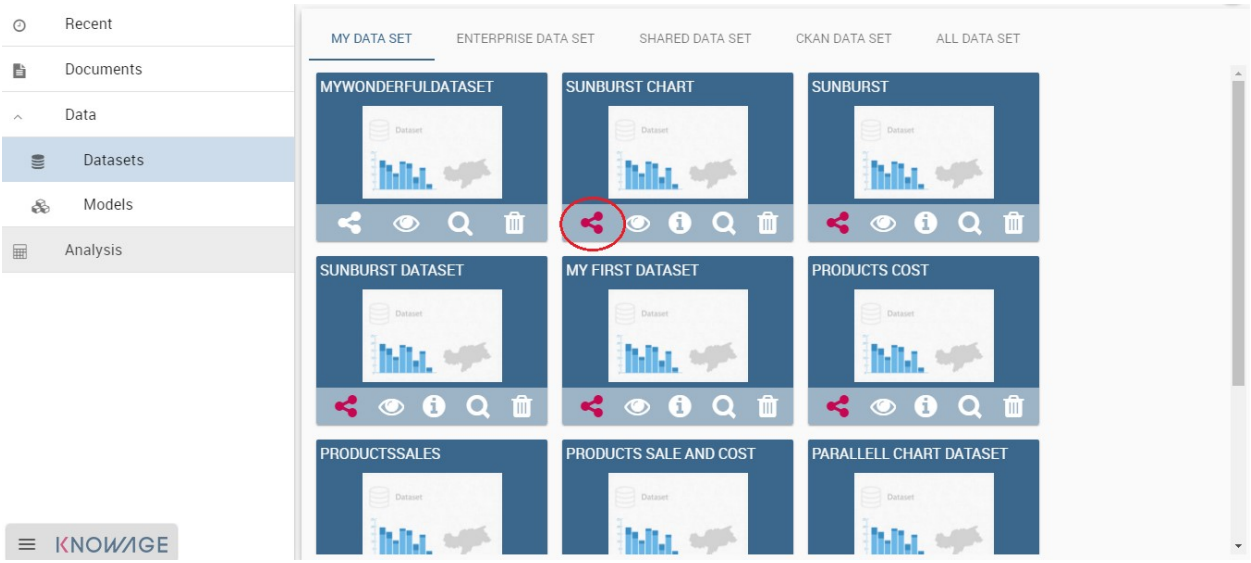


Fig. 5.7: Share a dataset.

Warning: CKAN Datasets

CKAN datasets can be divided in four main categories: "Public", "Organization private", "Acquired", "User private". You can 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, choose from the combobox the repository that you are interested in and then click on the repository name to access it.

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

These are not usable yet, but you 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 status (e.g. visibility, last modification date) ill be displayed by Knowage, as in the following figure. To use one of them you have 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 you want to use a dataset not used yet, any action on it will start the metadata import wizard. You access it by clicking the magnifier icon. As a first step, you have to insert some mandatory parameters to set the parser configuration.

As a second step the user have 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

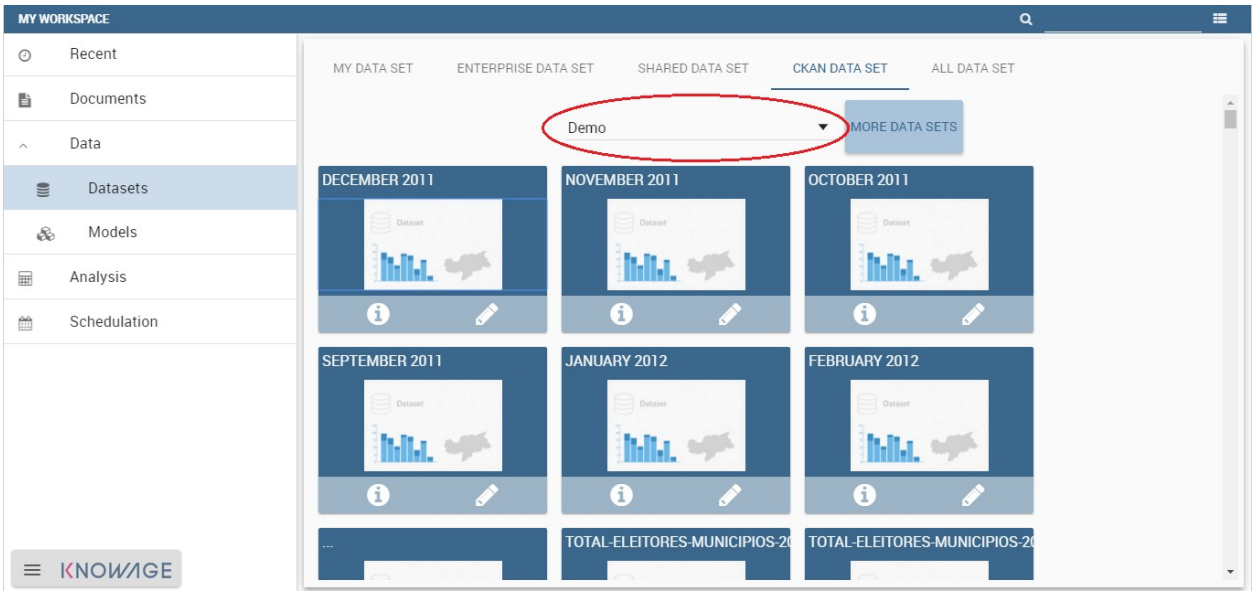


Fig. 5.8: CKAN Repositories.

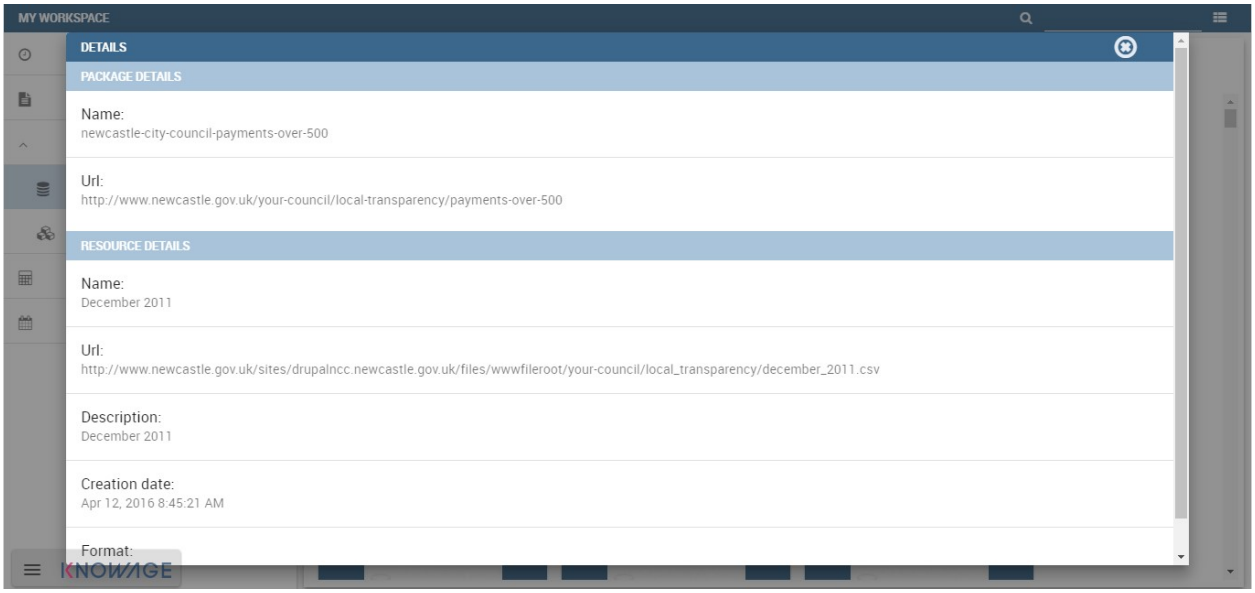


Fig. 5.9: CKAN dataset details.

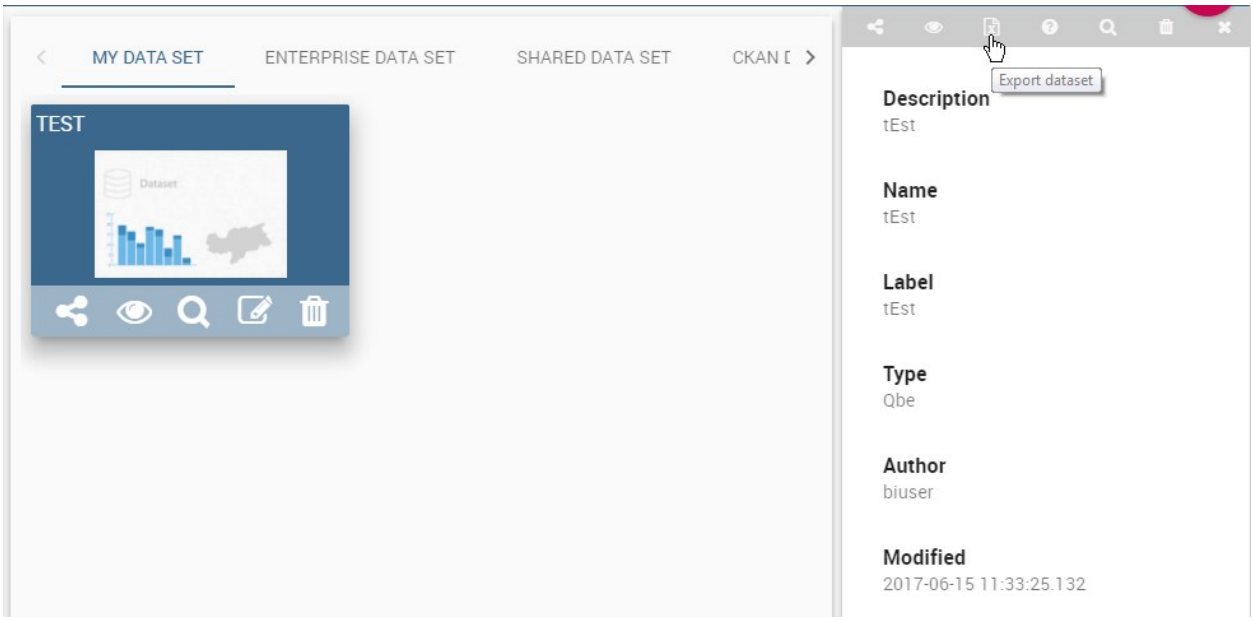


Fig. 5.10: Export dataset.

validation results, confirm and finalize dataset import. Once completed the dataset importation, the selected dataset will appear in the **DataSet** tab too. These actions just listed on the dataset change for downloded datasets. In particular you have the eye-shaped icon to refresh the dataset or change metadata by repeating the download process and the magnifier icon to inquire it trought the QbE interface.

5.1.2 Models

Here you find the models that the a technical user has built for you. You can query it using the QbE interface and create your own dataset from them.

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.

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.

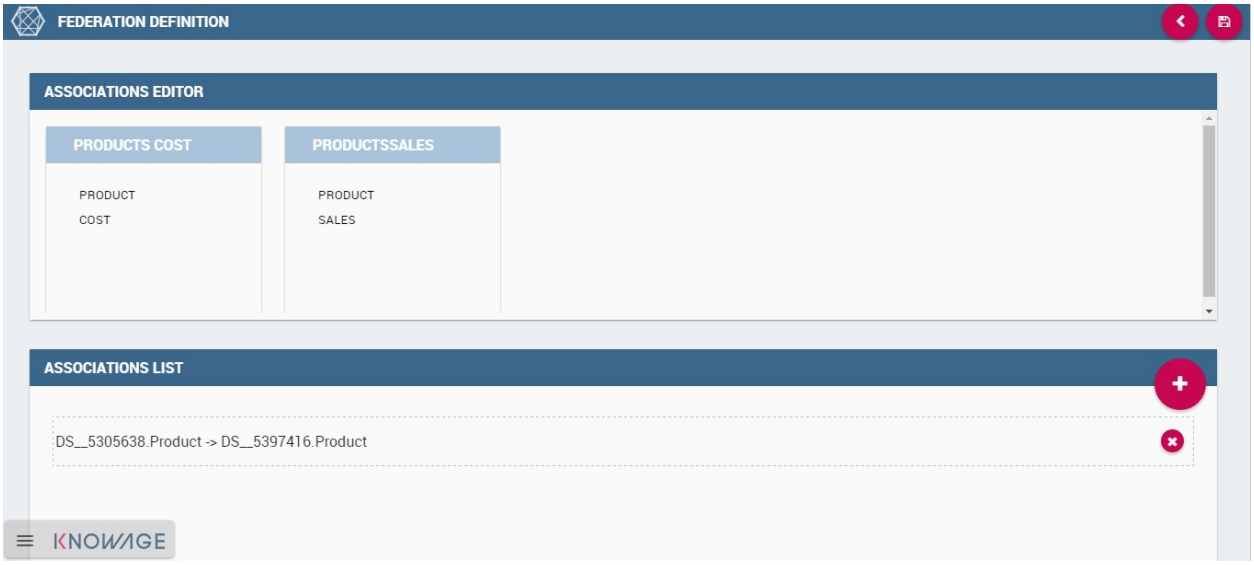


Fig. 5.11: Federated dataset details.

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.

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,

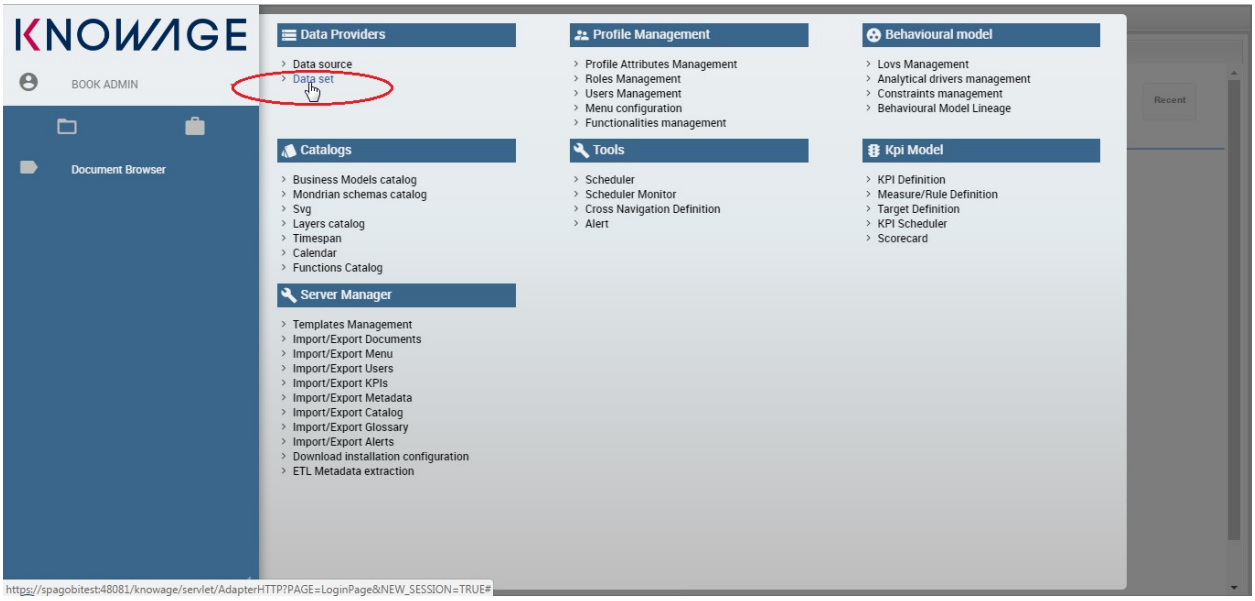


Fig. 5.12: Access data set creation area.

- Big Data.

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).

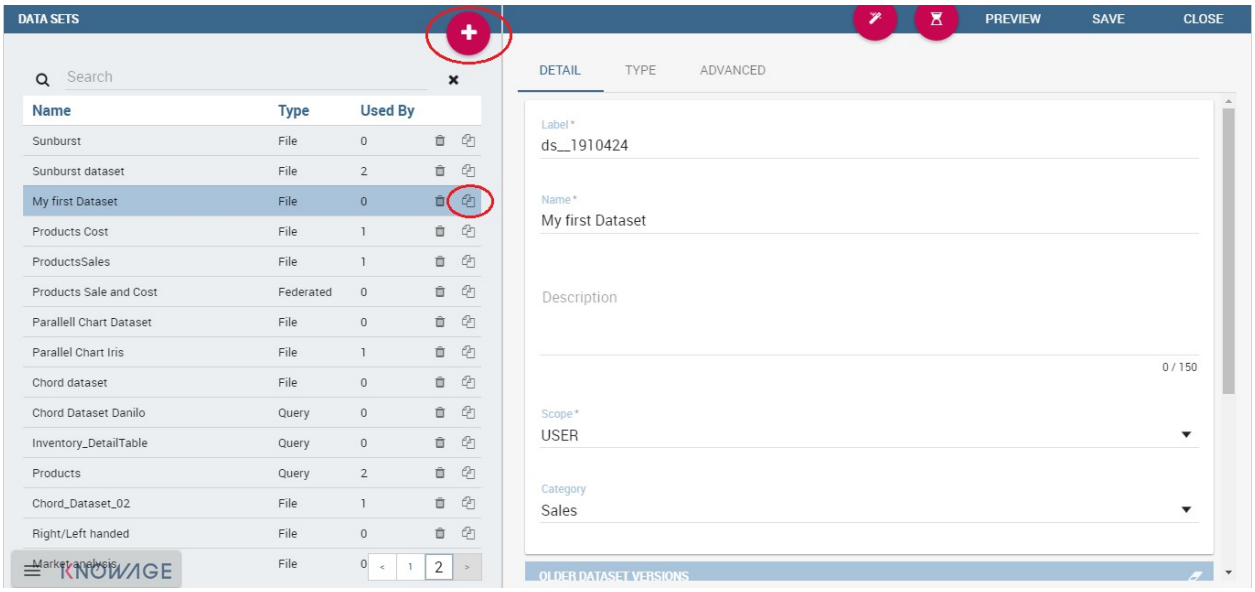


Fig. 5.13: Dataset Panel.

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.

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.

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.

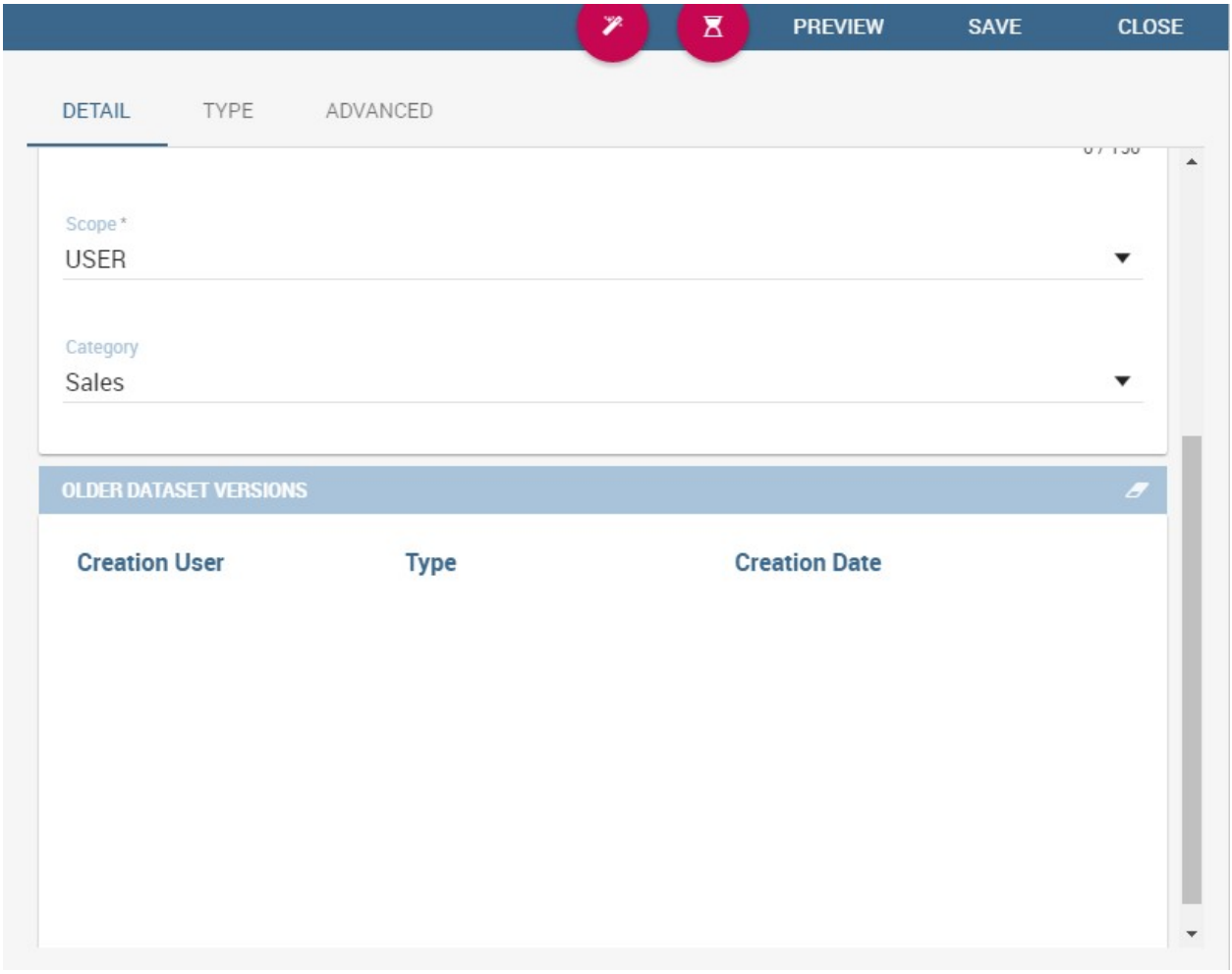


Fig. 5.14: The dataset versioning.

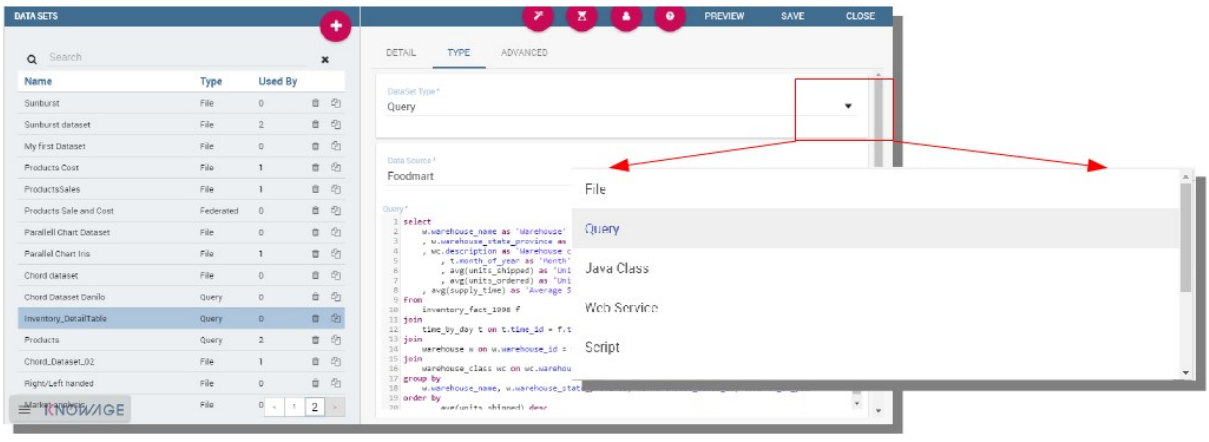


Fig. 5.15: The dataset type definition.

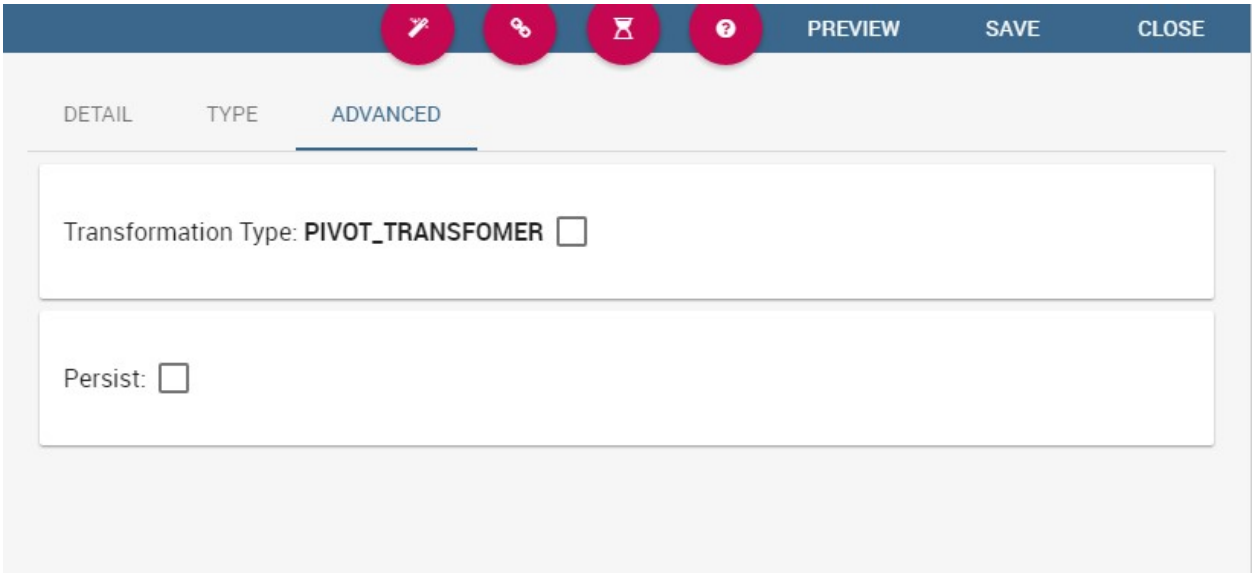


Fig. 5.16: The dataset trasformation tab.

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.

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

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

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).

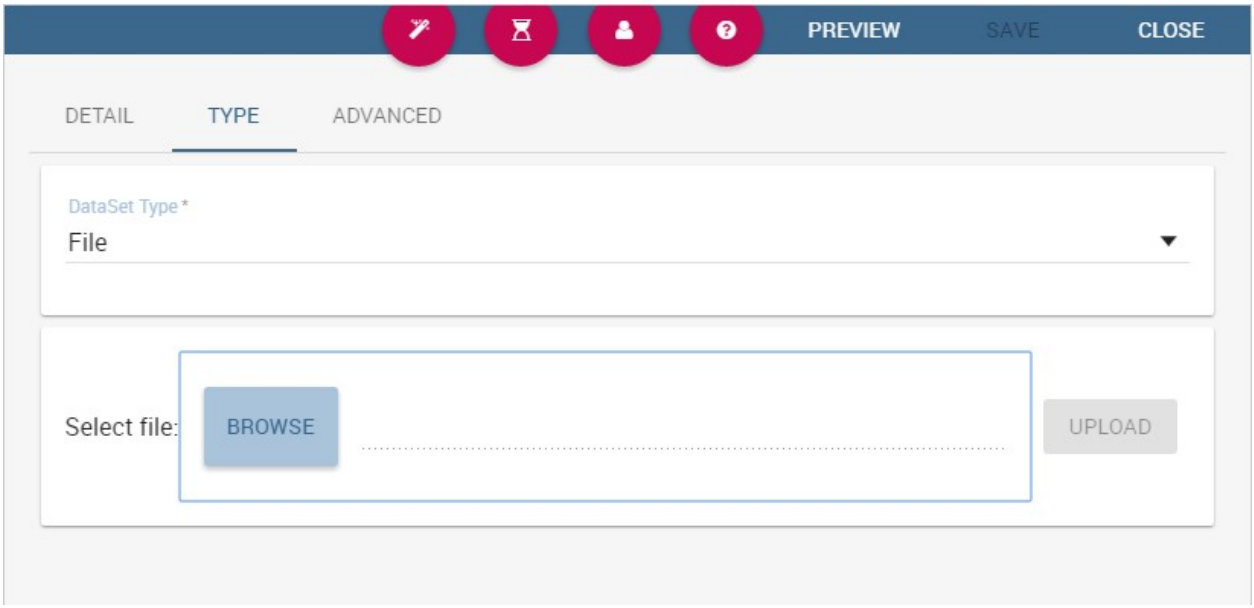


Fig. 5.17: File Dataset.

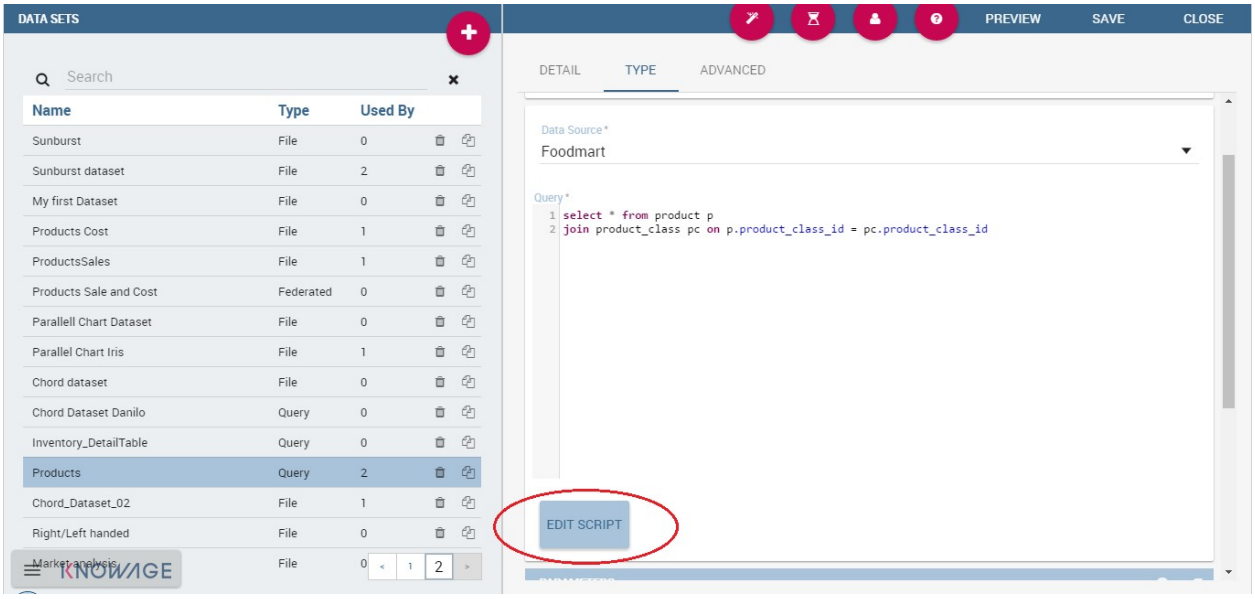


Fig. 5.18: Script editing for dataset.

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

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

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:

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

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

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.

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

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

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.

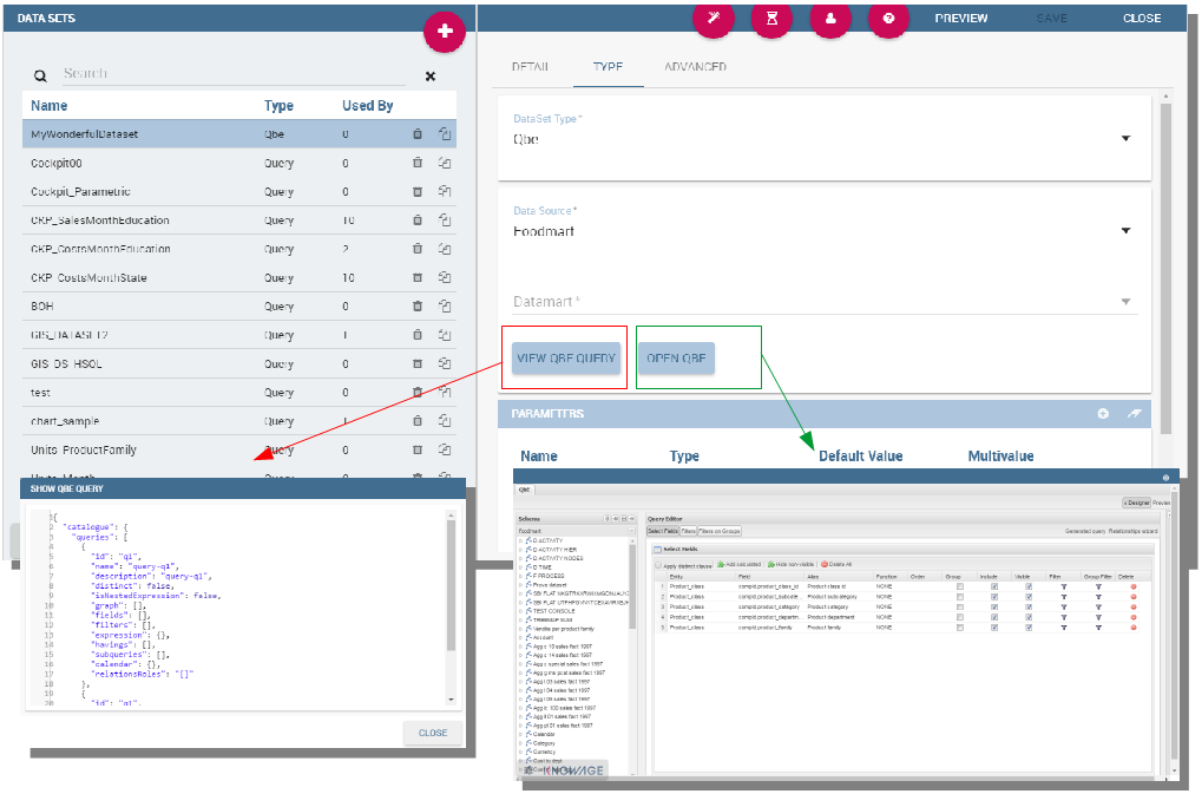


Fig. 5.19: QbE Dataset.

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.

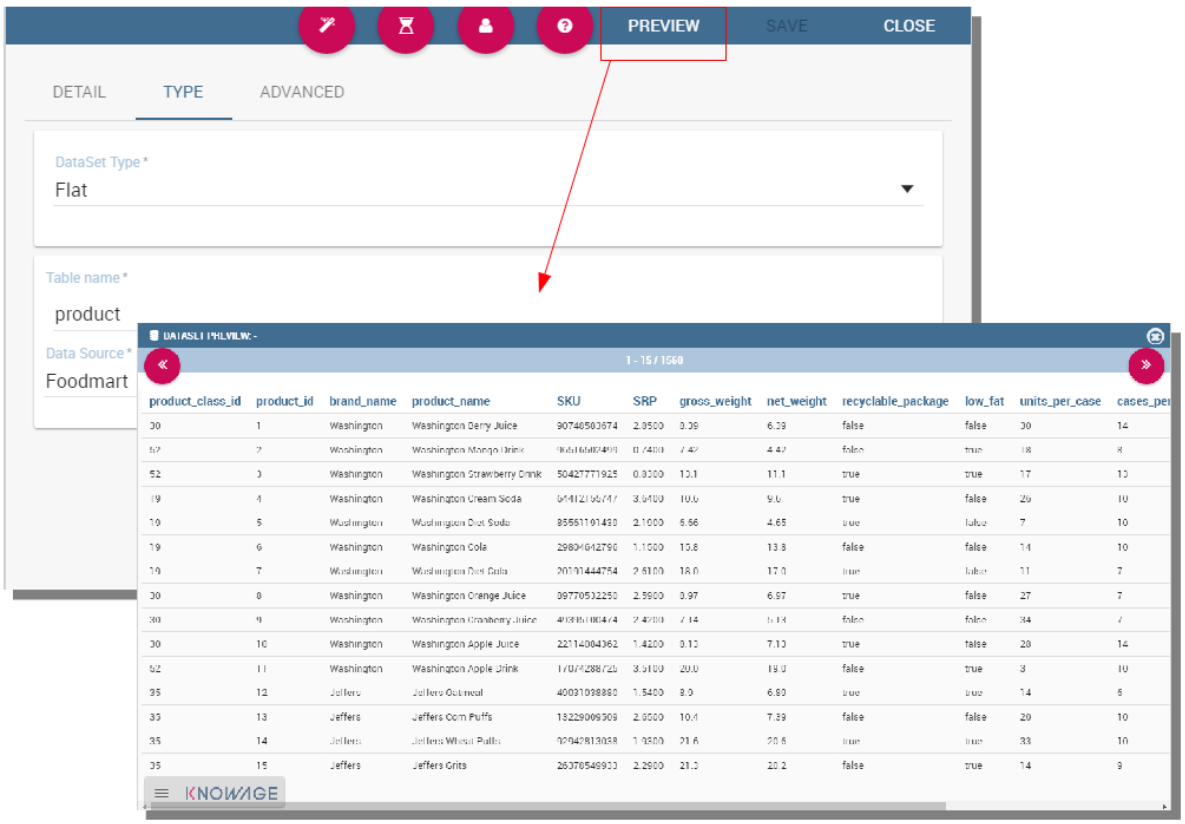


Fig. 5.20: Flat Dataset.

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

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

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 **Federatation 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

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

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

Listing 5.4: RJSON response code

```
{
       "contextResponses": [
    {
       "contextElement": {
       "id": "pros6_Meter",
       "type": "Meter",
       "isPattern": "false",
       "attributes": [
           {
             "name": "atTime",
             "type": "timestamp",
              "value": "2015-07-21T14:49:46.968+0200"
            },
            {
             "name": "downstreamActivePower",
             "type": "double",
             "value": "3.8"
            },
           {
             "name": "prosumerId",
             "type": "string",
             "value": "pros3"
           },
           {
             "name": "unitOfMeasurement",
             "type": "string",
             "value": "kW"
            },
            {
             "name": "upstreamActivePower",
             "type": "double",
             "value": "3.97"
             }
           ]
           },
      "statusCode": {
              "reasonPhrase": "OK",
              "code": "200"
                    }
        },
           {
       "contextElement": {
              "id": "pros5_Meter",
              "type": "Meter",
              "isPattern": "false",
              "attributes": [
           {
              "name": "atTime",
              "type": "timestamp",
              "value": "2015-08-09T20:29:45.698+0200"
           },
           {
```

(continues on next page)

(continued from previous page)

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

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

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

Let's see how to define a Knowage dataset:

We specified

- the URL of the REST service;
- 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).

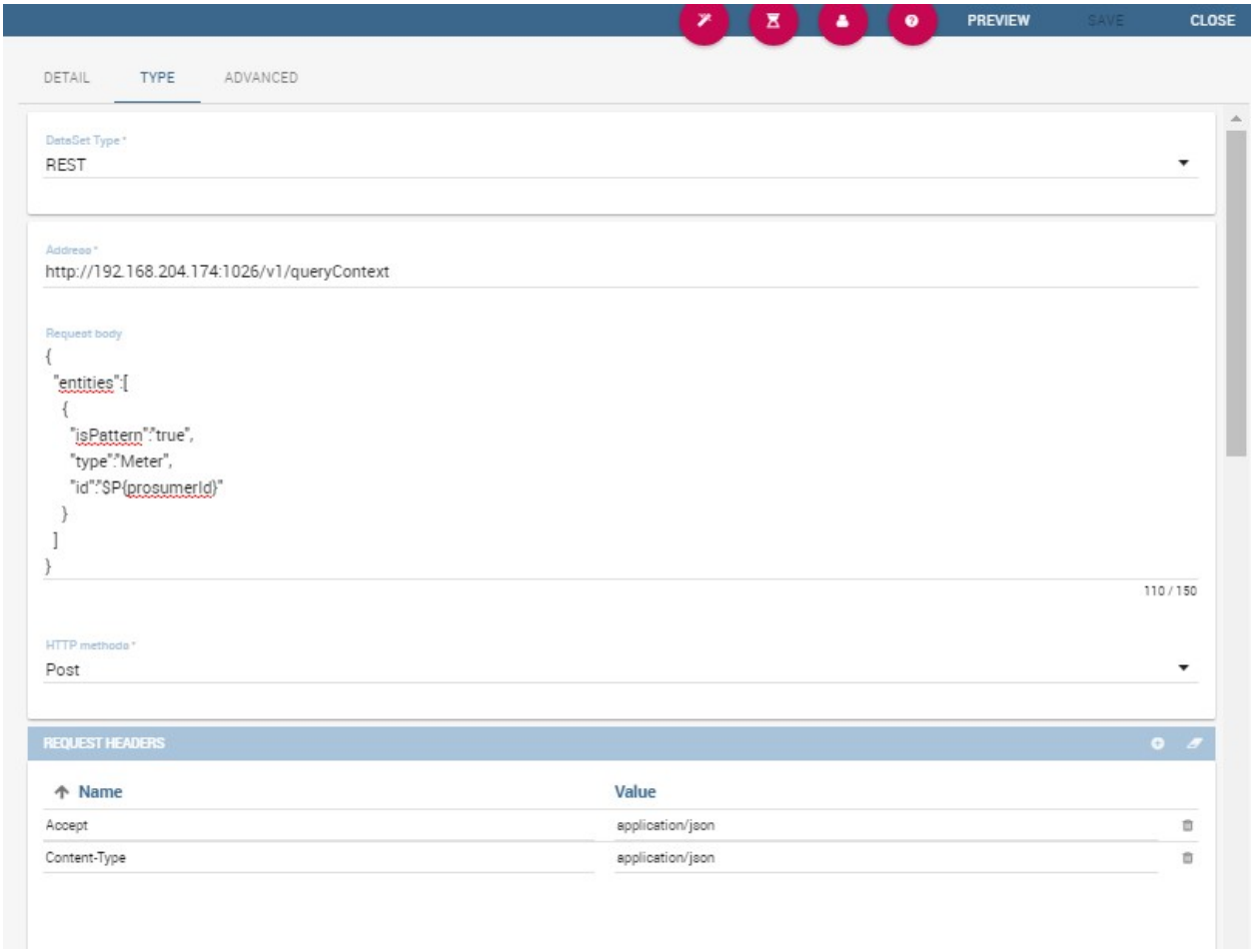


Fig. 5.21: REST dataset interface.

When checking the **Use directly JSON attributes** checkbox, yon can skip the definition of the JSONPath attributes, since the JSON structure is presumed to be fixed as in the following example:

Listing 5.5: Use directly JSON attributes

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

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

```
{
 "entities":[
   {
    "isPattern":"true",
    "type":"Meter",
    "id":"$P{prosumerId}"
   }
           ]
}
```

5.2.1.7.5 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 accordingly to the datasource dialect).

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

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

Listing 5.7: Request body for prosumerId parameter

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

- if the return value doesn't come from a query, for example it's a js variable, than it must be assigned to a variable with name `sbiDatasetfixedResult`. The result will be managed by Knowage accordingly to the type of the variable:
  - if it's a primitive type the resulting dataset contains only a columns with name `result` and value equal to the value of the variable `sbiDatasetfixedResult`;
  - if it's an object, the resulting dataset contains a column for each property of the object.

    For example, if we consider the query `sbiDatasetfixedResult = {a:2, b:3}` the dataset is as shown in Table 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 istance, 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:

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

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

```
var retVal= "SINGLE_DOCUMENT_QUERY";
```

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

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/

5.2.2 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 is, with no futher 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

```
SELECT  s.customer_id as CUSTOMER
, sum(s.store_sales) as SALES
, c.yearly_income as INCOME
, p.media_type as MEDIA
FROM sales_fact_1998 s, customer c, promotion p
WHERE
s.customer_id=c.customer_id and s.promotion_id=p.promotion_id and
p.media_type in ($P{MediaType})
GROUP BY
s.customer_id,
c.yearly_income,
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 behavioral model and choose the one(s) you wish to insert in the dataset query/script text, as shown below.

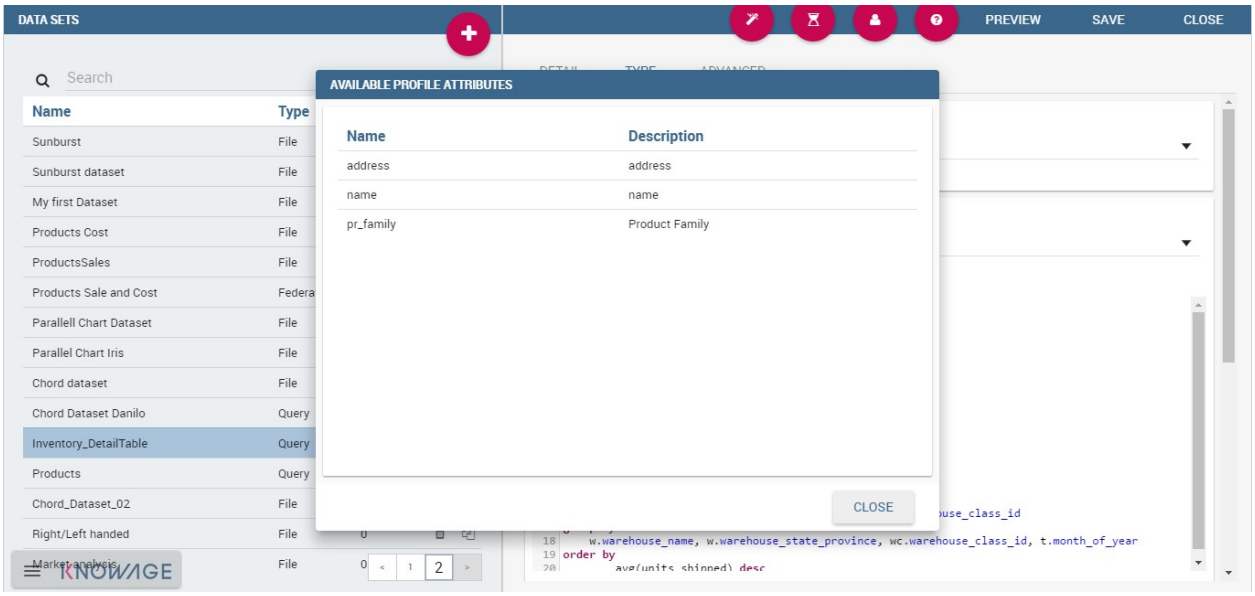


Fig. 5.22: Profile Attributes assignment.

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

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.

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

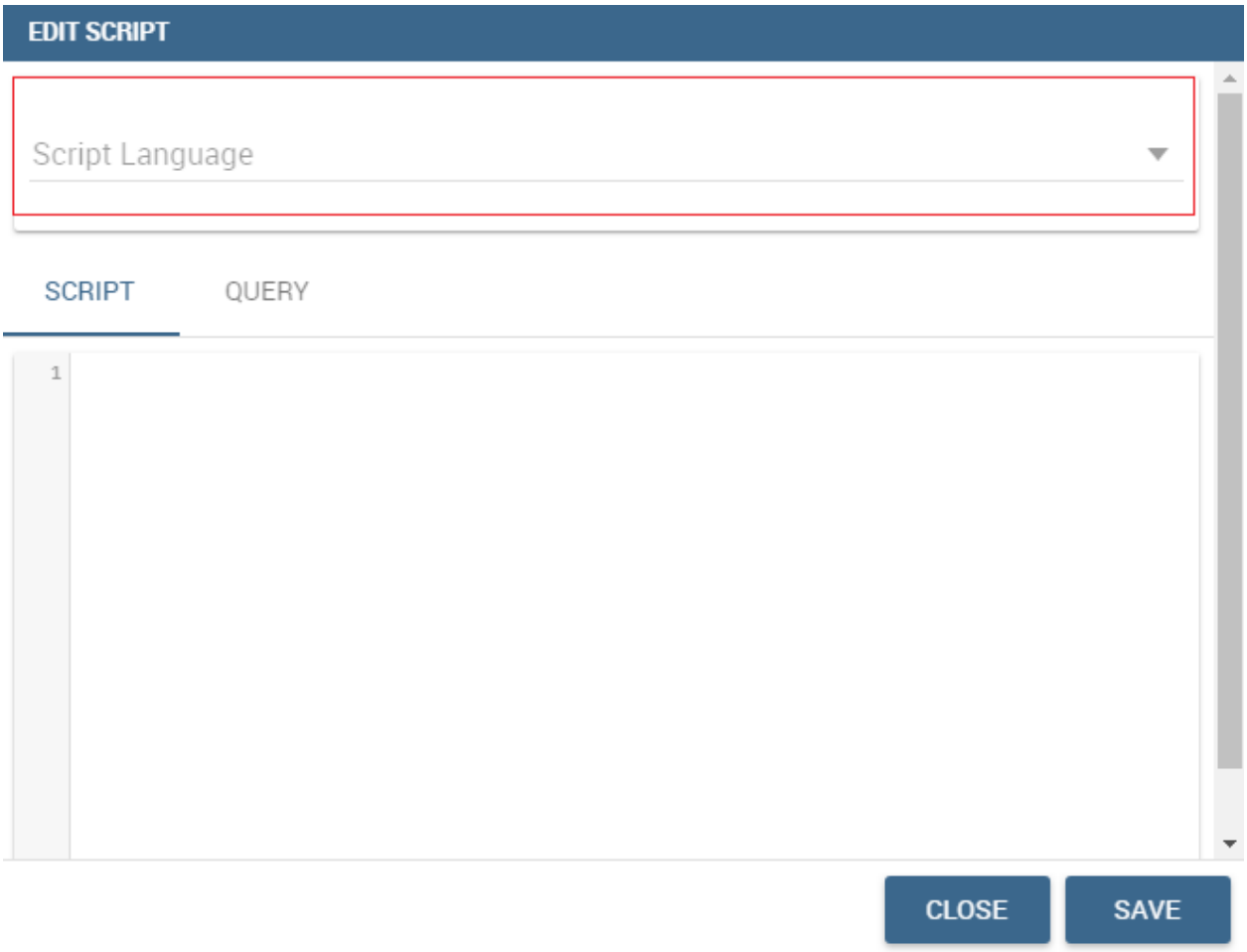


Fig. 5.23: Editing script.

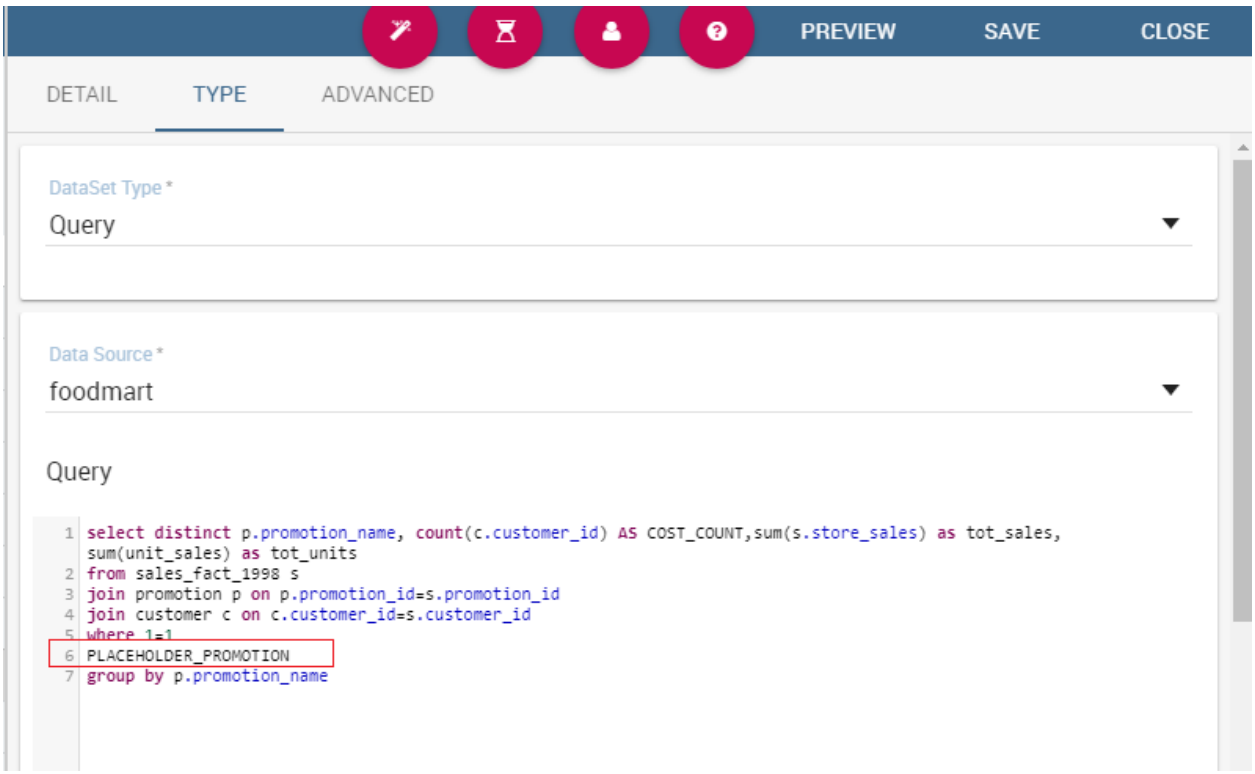


Fig. 5.24: Setting placeholder using script.

parameters, are satisfied, the variable is assigned an SQL code string. Then, the JS method ".replace" will substitute the placehoder with the content the variable.

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.

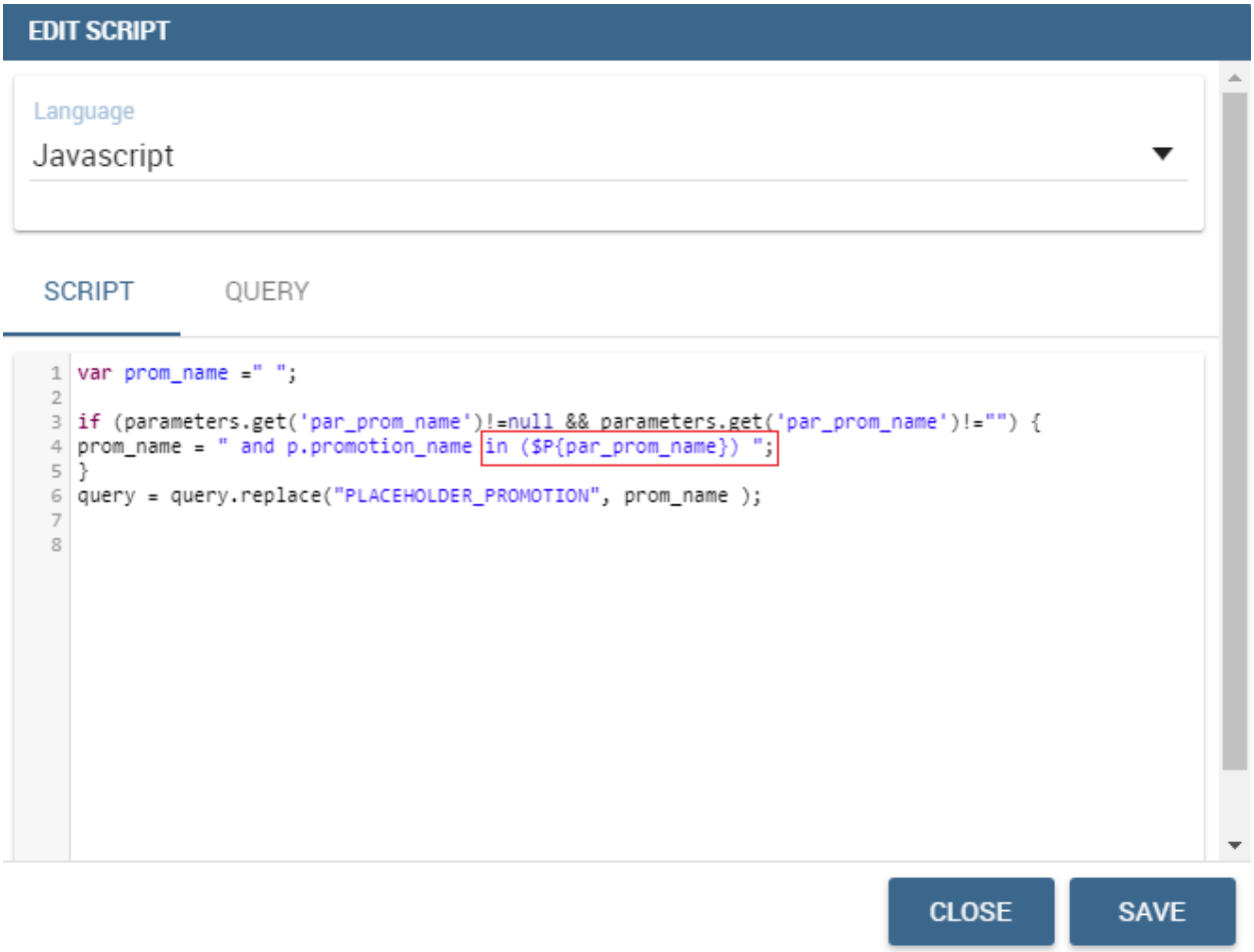


Fig. 5.25: Editing script.

- 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.

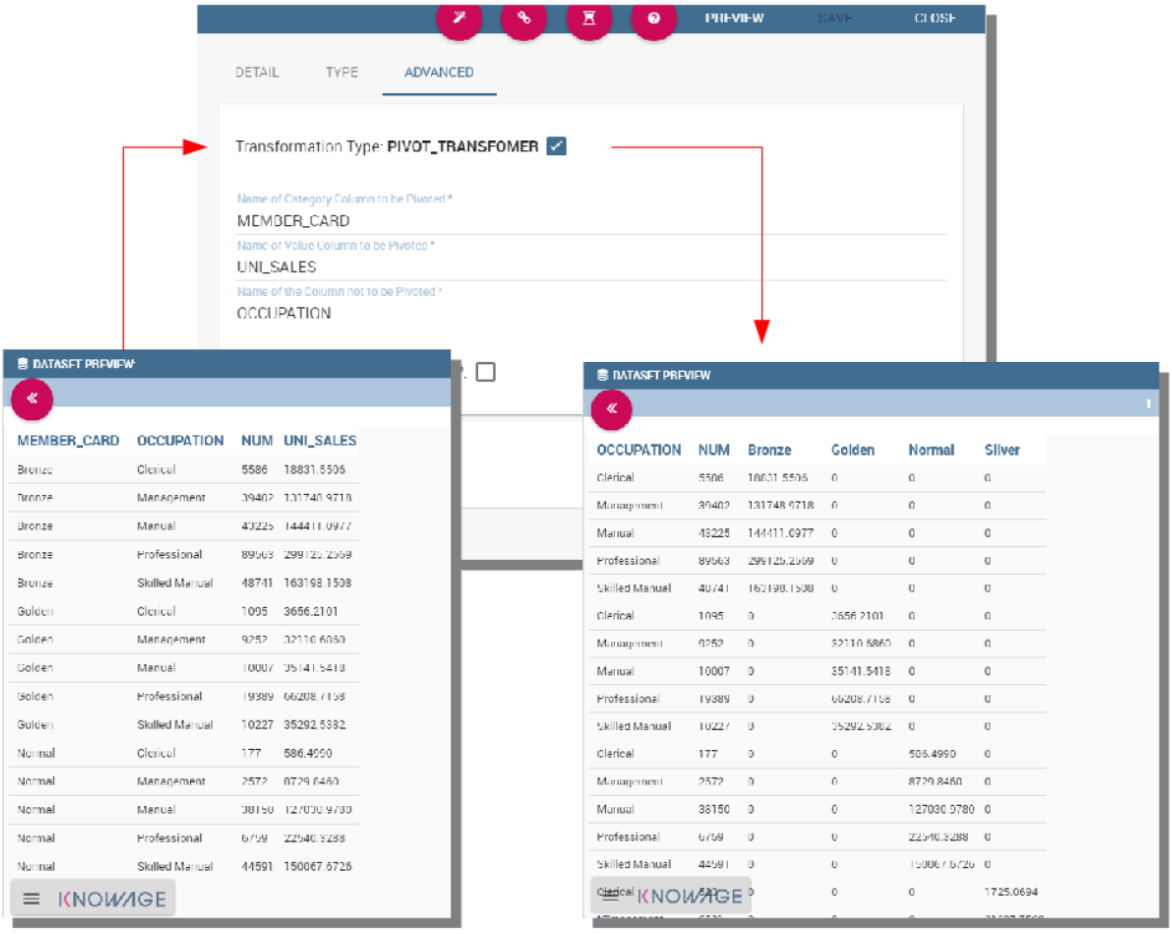


Fig. 5.26: Pivot transformation.

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 persisent 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.

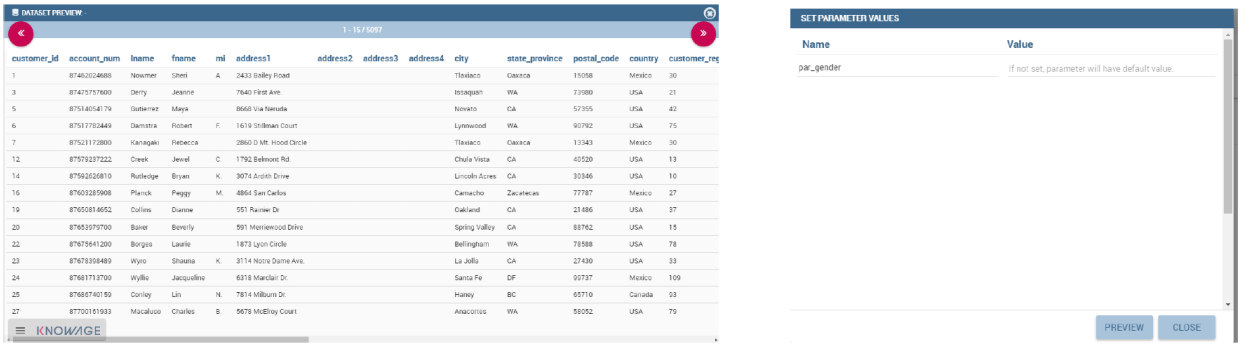


Fig. 5.27: Dataset preview (left) and parameters prompt window (right).

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 bevioural 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 conlcude 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.

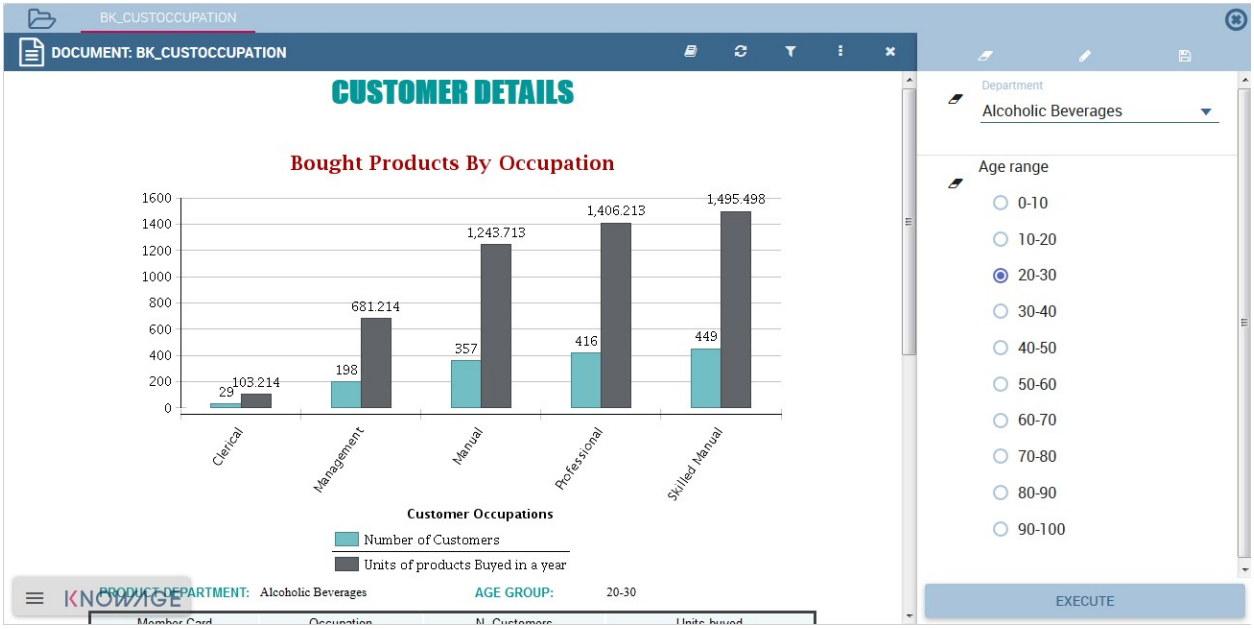


Fig. 5.28: Parametric Report.

The Figure above represents a report with two parameters:

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

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

- Who is involved in a specific use mode, in terms of a list of end user roles, considering that a role can be associated to a single use mode only.
- What data he can access and how they are presented to the end user for his potential selection. This information is provided by the so called *List of Value (LOV)*.
- How to check the validity of the chosen values. This information is provided by the so called *Check*.

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

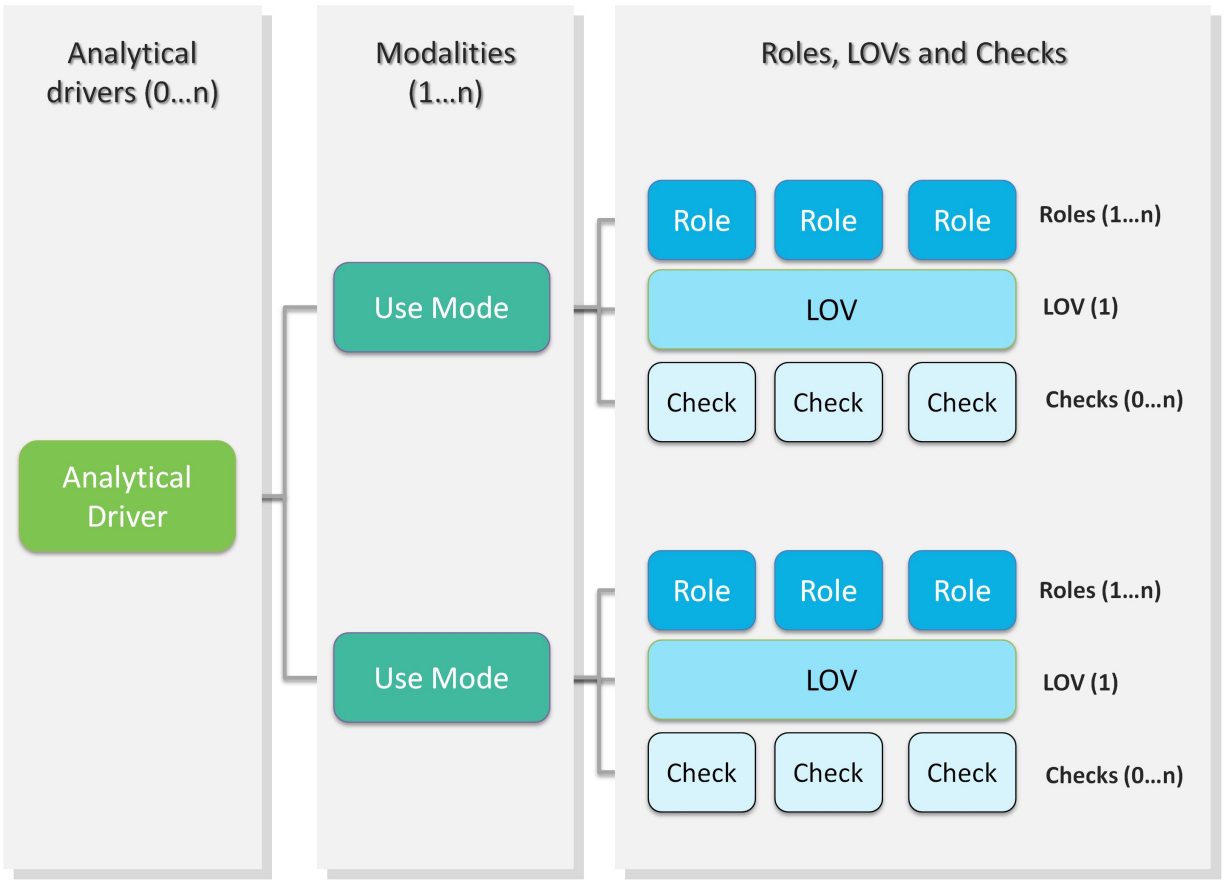


Fig. 5.29: Analytical driver schema.

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

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

Once defined, a driver can be related to many documents, driving their behaviour and filters in a common way. This

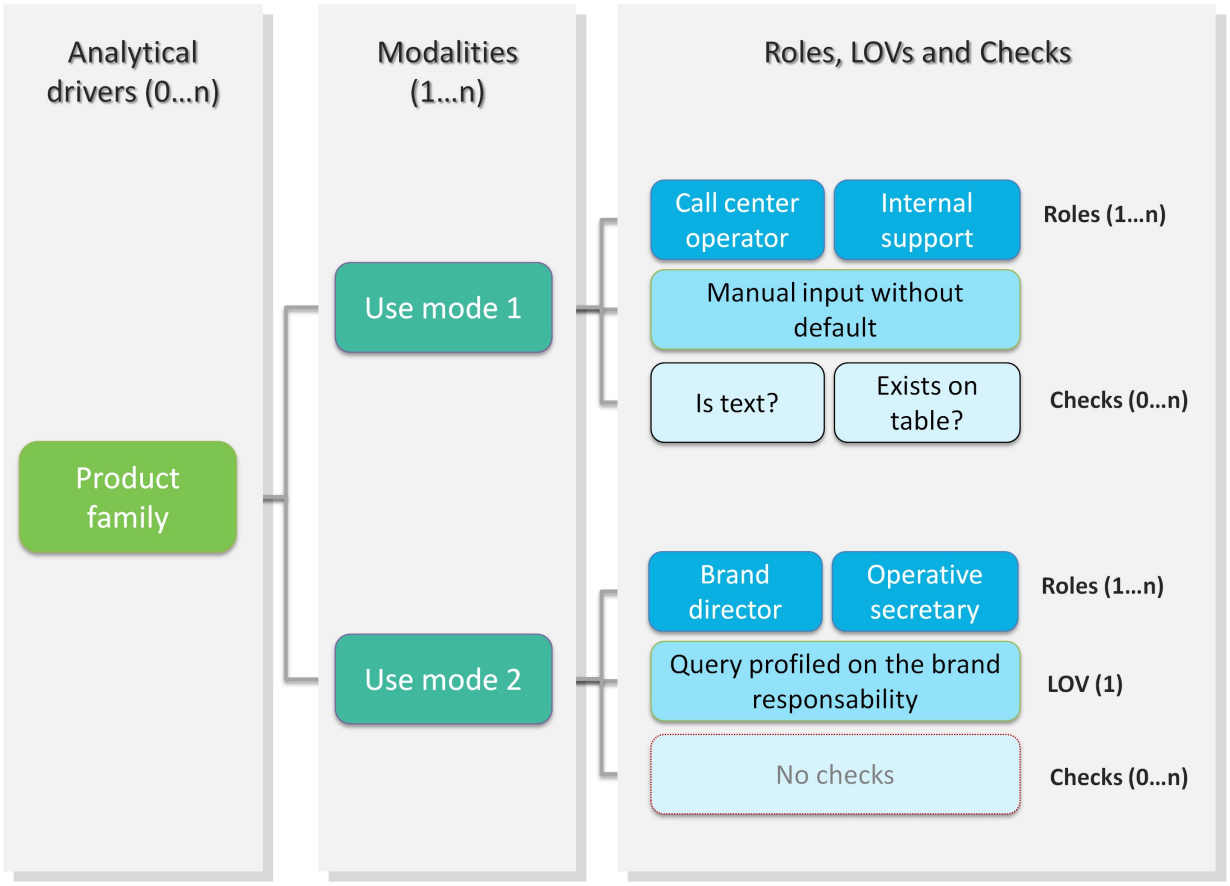


Fig. 5.30: Analytical driver schema - Example.

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.

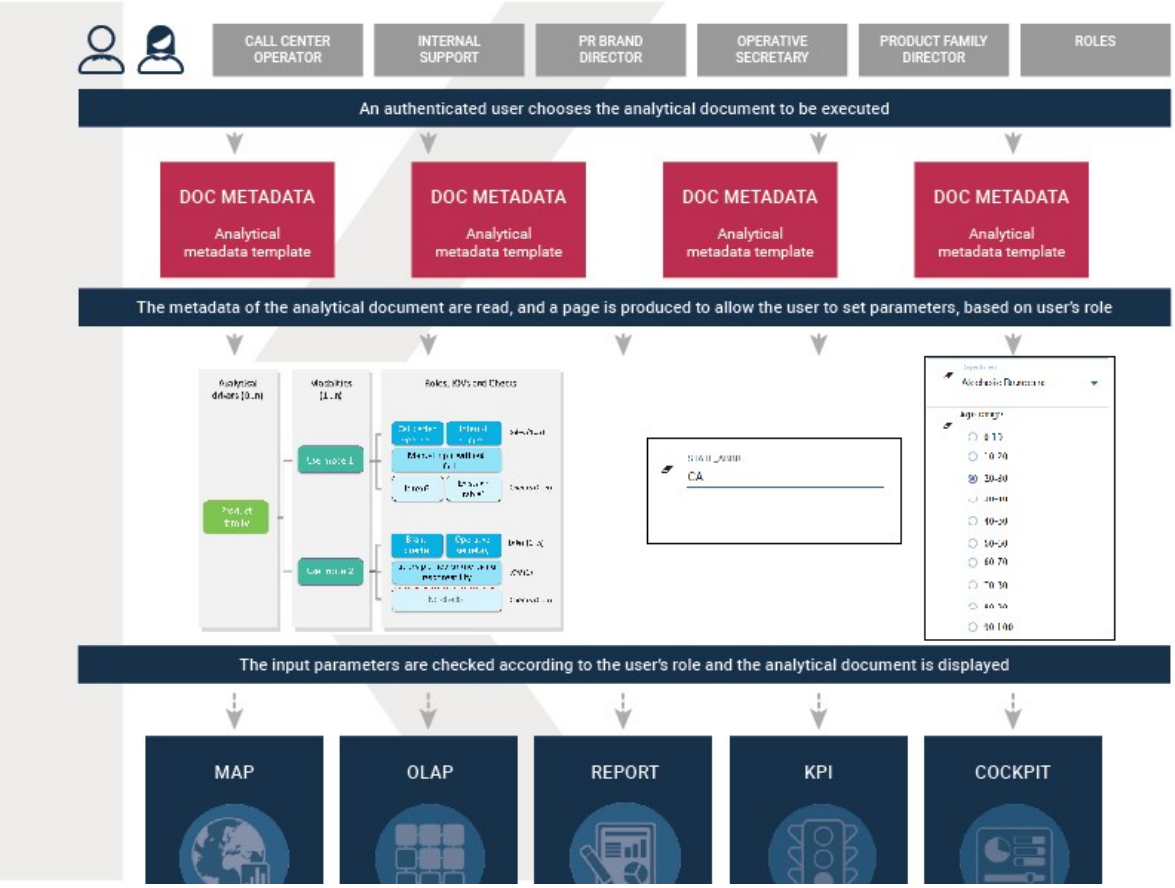


Fig. 5.31: Overall process.

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

```
{195, High Top Almonds, Food};
{522, Tell Tale Walnuts, Food};
{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.

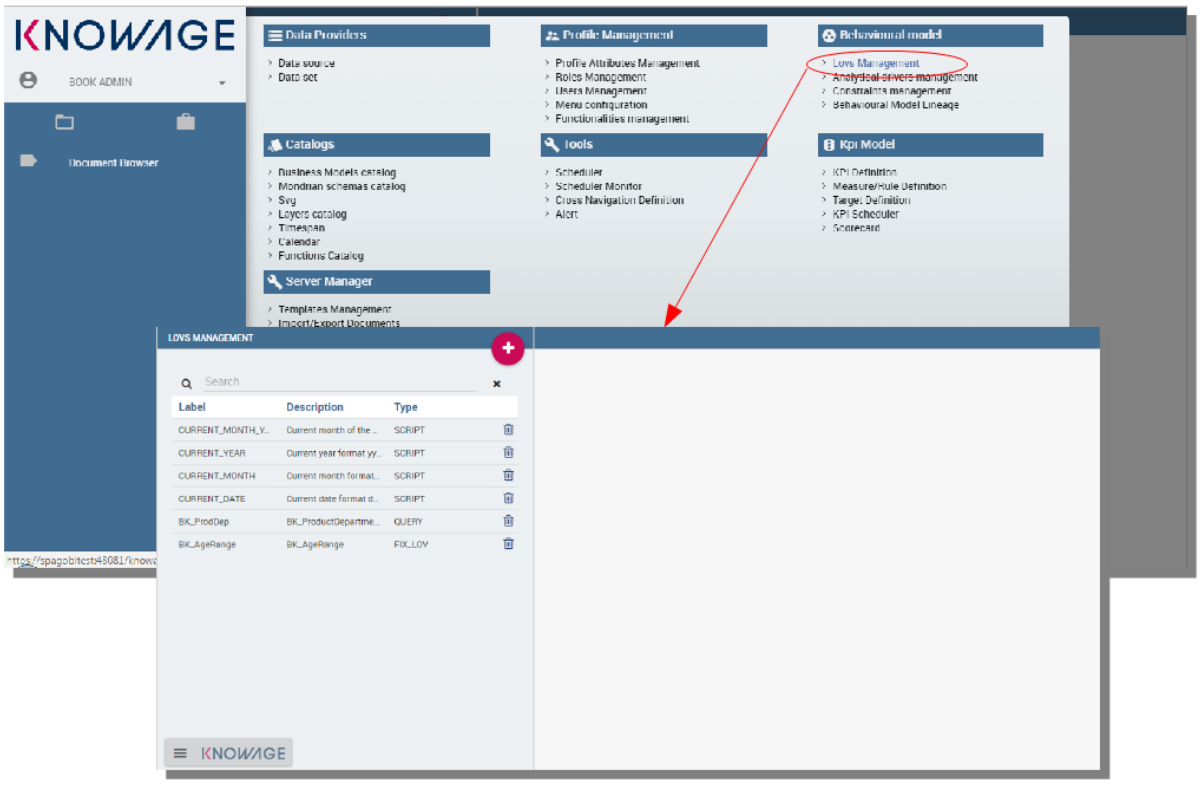


Fig. 5.32: LOV list.

To create and manage LOVs, select **Behavioural Model > Lovs Management** from the developer menu. The entire

list of available LOVs appears, as shown in figure 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.

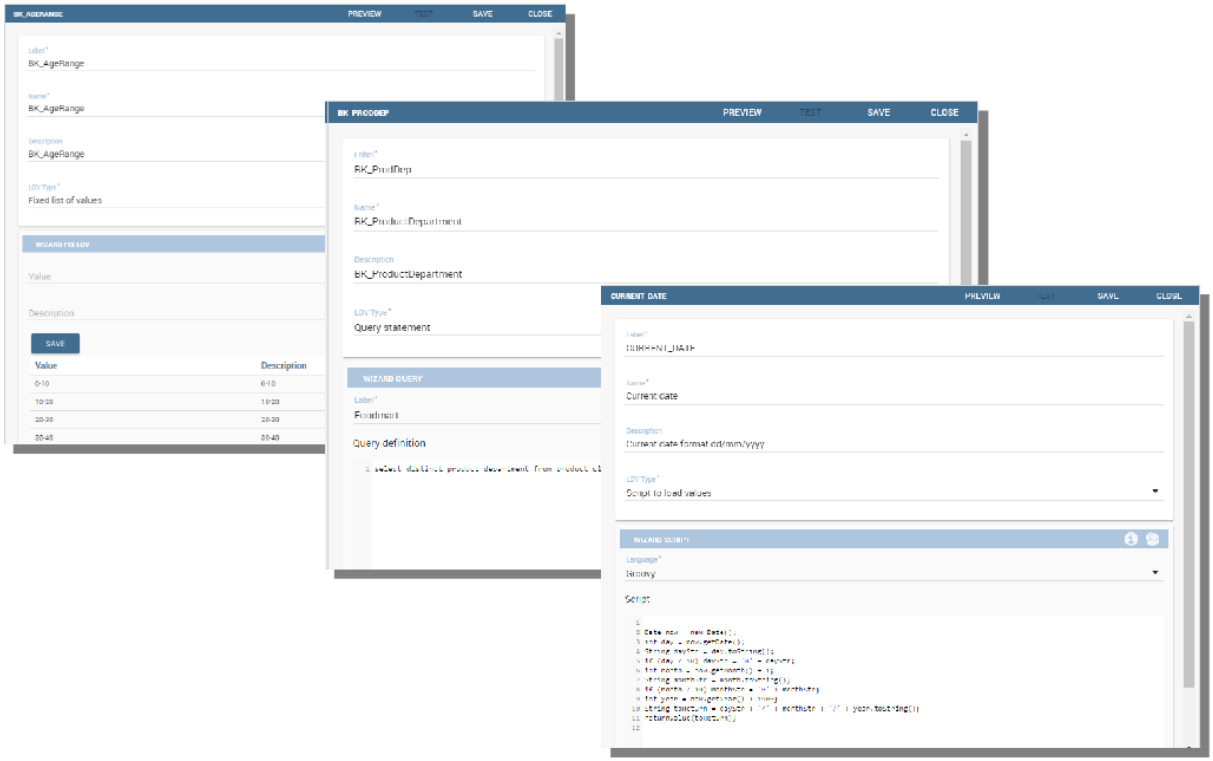


Fig. 5.33: LOV Creation interface.

Once completed the form, click on **Preview** button to enable the **Test** button. Notice that you cannot save the LOV without testing it, since this allows to detect errors before the LOV is actually used in a driver and associated to a document. After testing, you will be able to define which column is the actual value of the LOV, i.e., which value will be passed to the analytical driver using this LOV. Only *one* column can be the value attribute and only *one* column can be chosen as Descriptive attribute, while the others can be visible. 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.

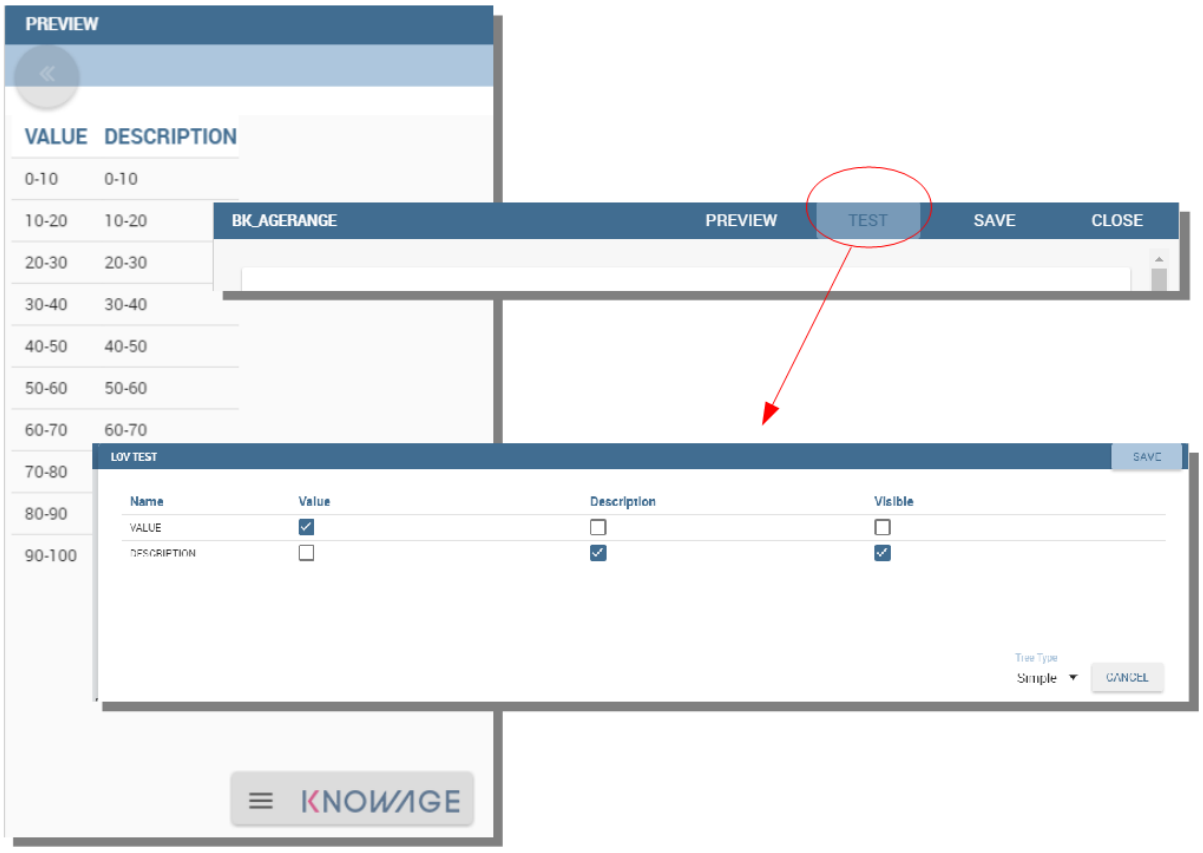


Fig. 5.34: Preview and Test of the LOV.

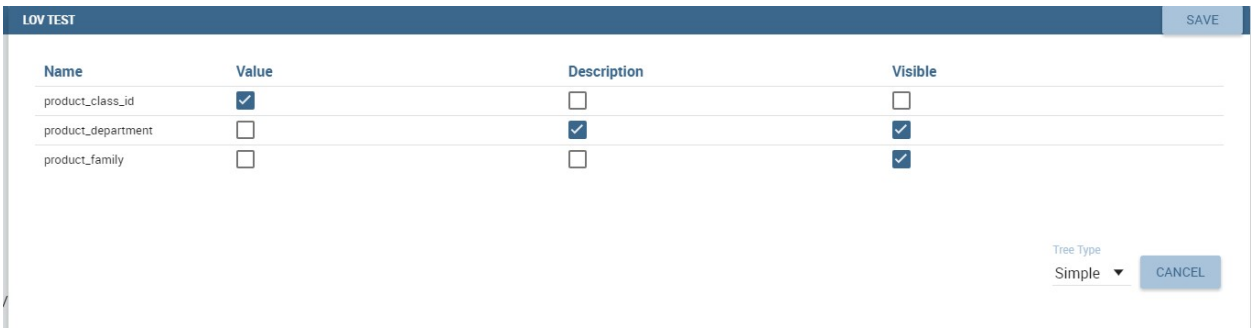


Fig. 5.35: Preview and Test of the LOV.

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.

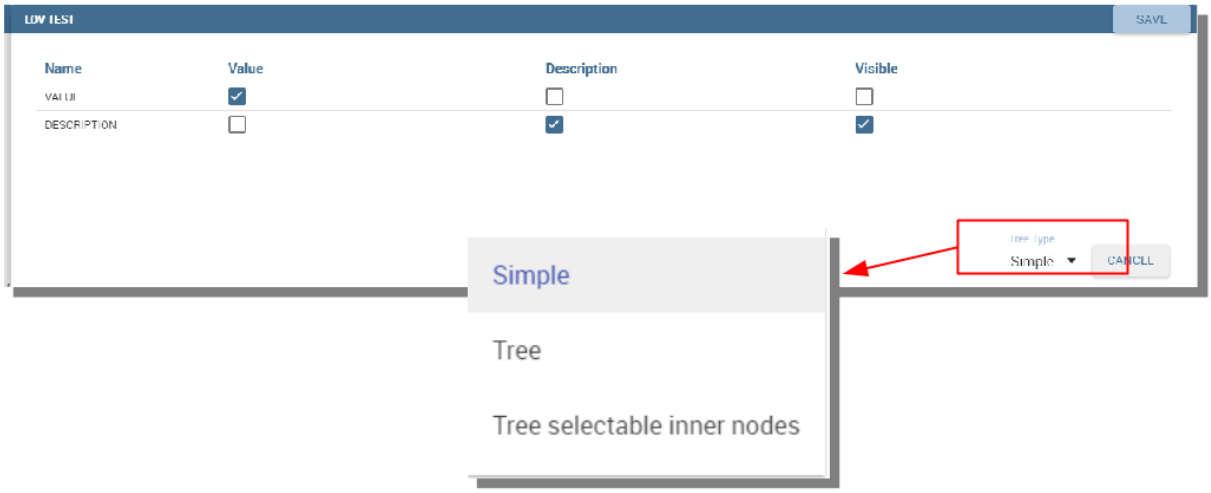


Fig. 5.36: Hierarchical LOV definition.

Note: Create a LOV for the default value of an analytical driver of type Manual Input Date

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.

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

```
SELECT DISTINCT PRODUCT_FAMILY, BRAND_NAME
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 standar syntax `${profile_attribute_name}`, as shown in Parametric query.

Listing 5.11: Parametric query

```
SELECT DISTINCT PRODUCT_FAMILY, BRAND_NAME
FROM PRODUCT
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

```
SELECT DISTINCT PRODUCT_FAMILY, BRAND_NAME
FROM PRODUCT
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 avaiable that dont' 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.

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:

- **Alfanumeric**: it checks if the parameter is alfanumeric;

Fig. 5.37: Assistance in retrieving syntax and profile attributes.

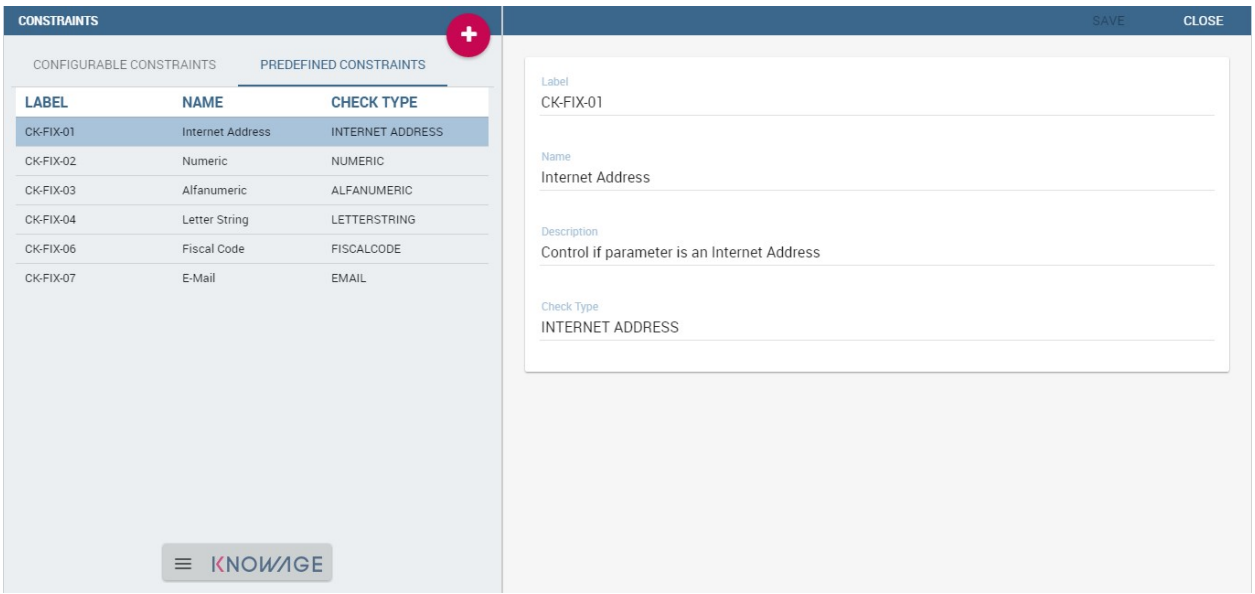


Fig. 5.38: Contraints Management.

- **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 costumized format type of date;
- **Regular Expression**: to set a regular expression validation rule;
- **Max/Min Length**: it lets you set the maximum and/or minimum character parameters length;
- **Range**: to set a range the parameters value has to satisfy;
- **Decimal**: to set a maximal decimal places for the parameters.

5.3.2.4 Creating an analytical driver

As explained at the beginning of this section, analytical drivers use nformation 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.

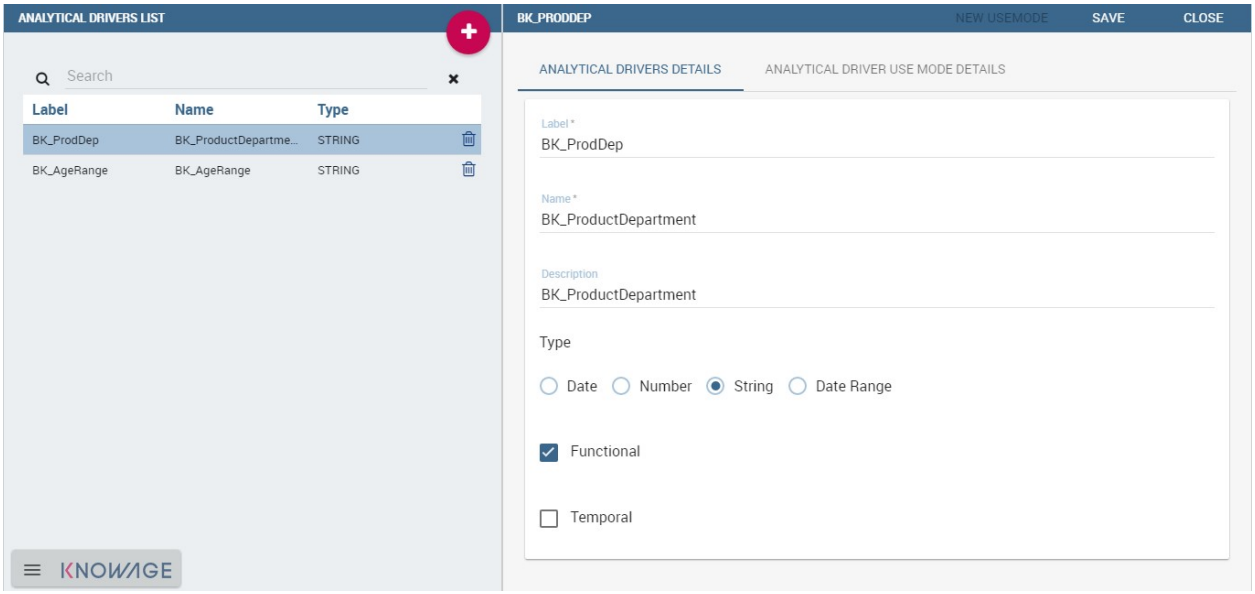


Fig. 5.39: Analytical Driver Management.

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 adocument is currently using it.

To create a new driver, click on at the top right corner. The driver creation interface will open. At first execution only the upper part of the window is visible, as shown in 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 botton, enabled as soon as the form is filled in, will save the driver and let the section below appear.

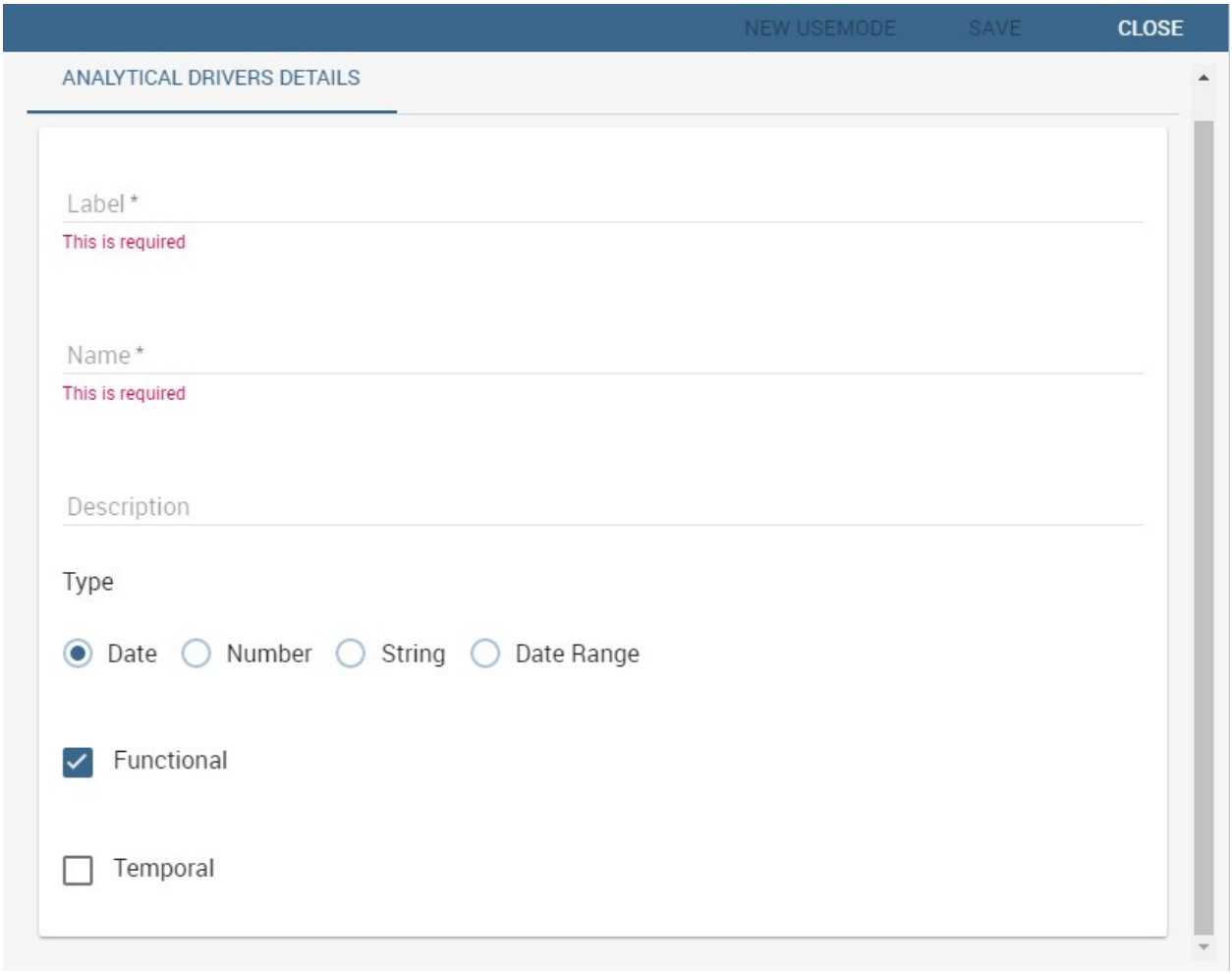


Fig. 5.40: Driver creation.

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

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

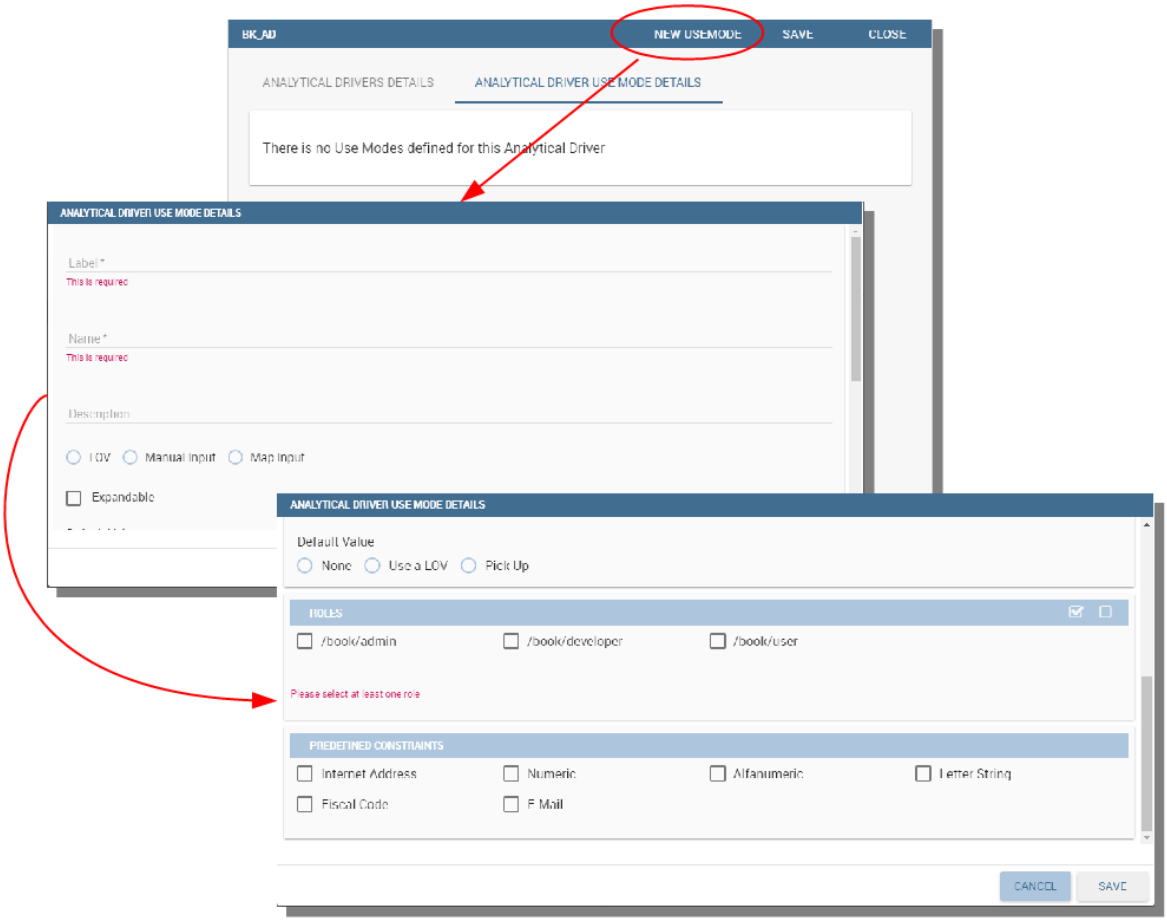


Fig. 5.41: Detail panel of LOV creation, second step.

- **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).

Note: Analytical driver of type Manual Input Date with a default value

In the case you want to use an analytical driver of type Manual Input Date with a particular date as default 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.

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.

These few steps will implent 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 familiy.

Fig. 5.42: Spatial analytical driver settings.

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.

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, analyitical 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.

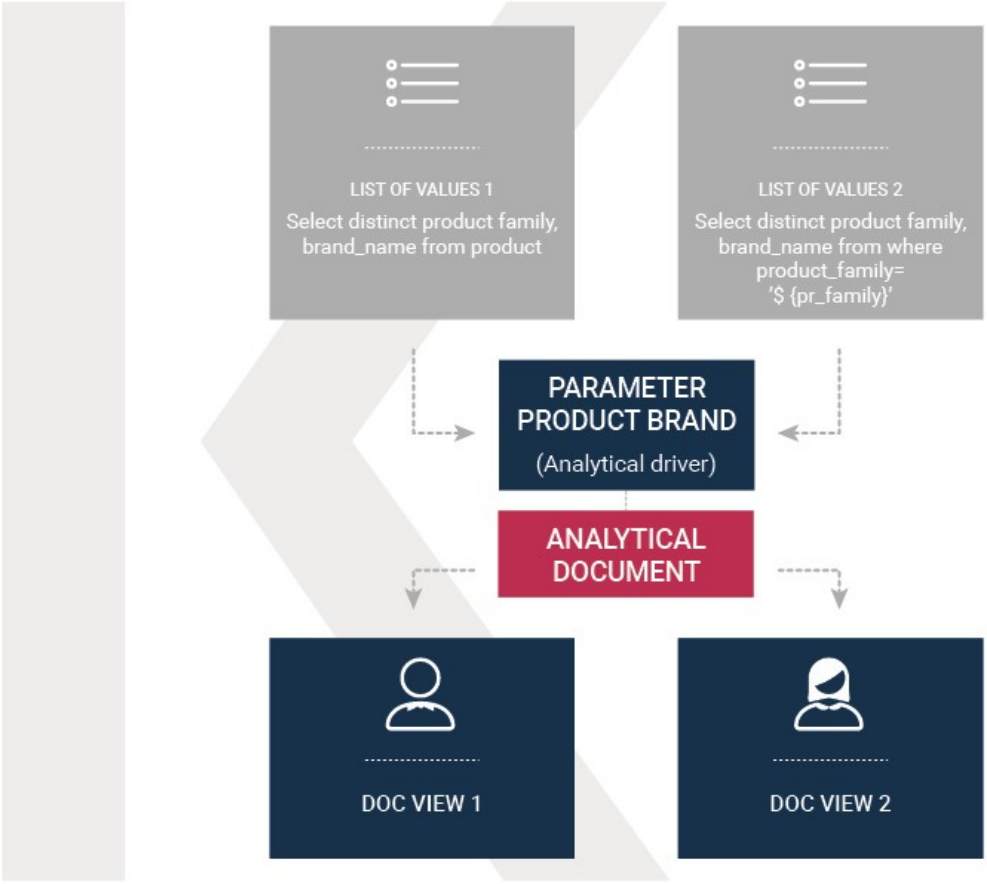


Fig. 5.43: Behavioural Model Schema.

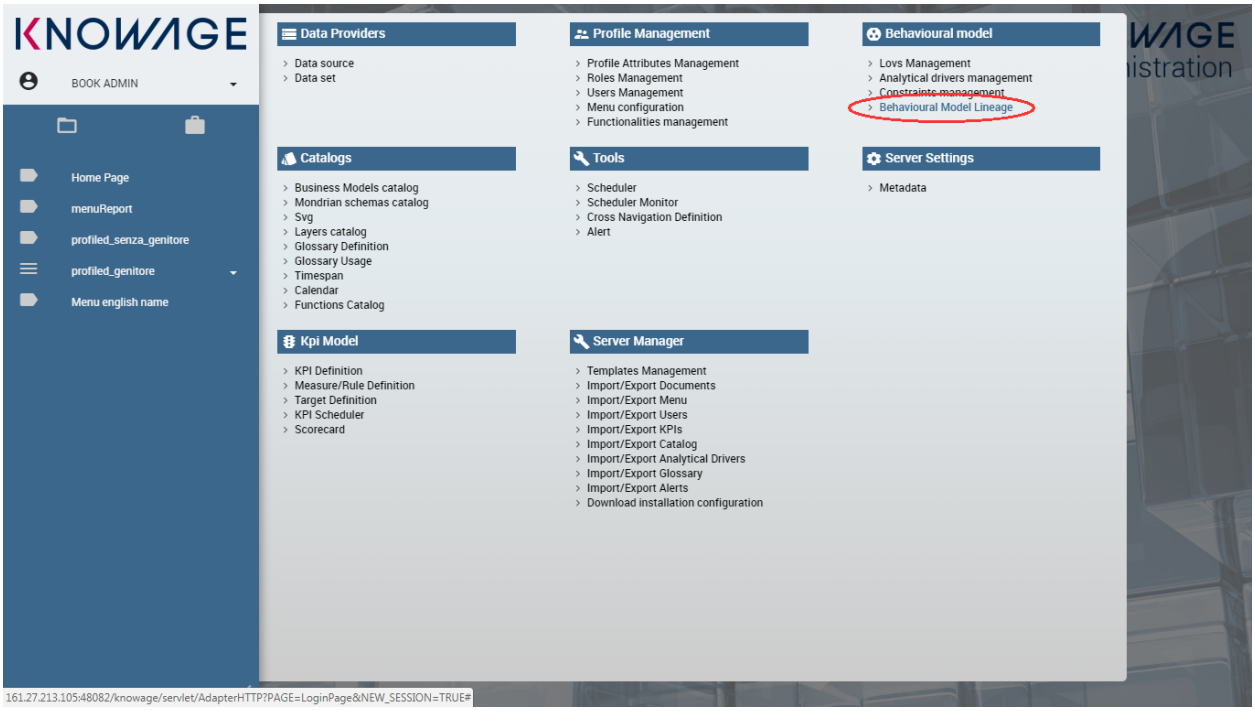


Fig. 5.44: Behavioural Model Lineage.

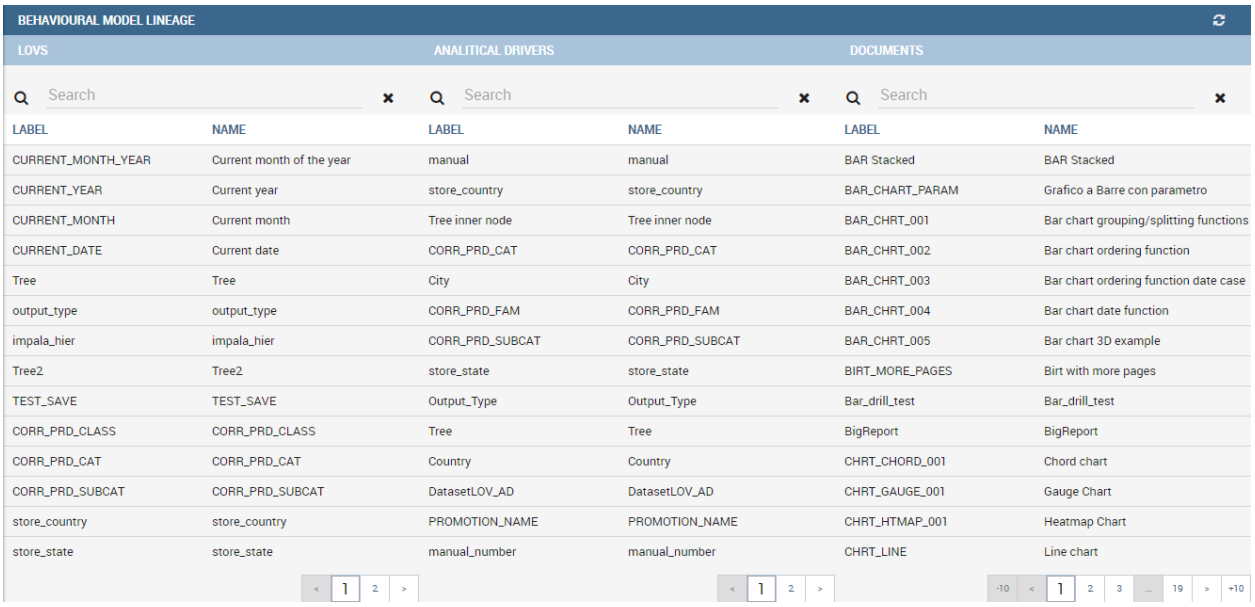


Fig. 5.45: List of LOVs, analyitical driver and documents.

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;

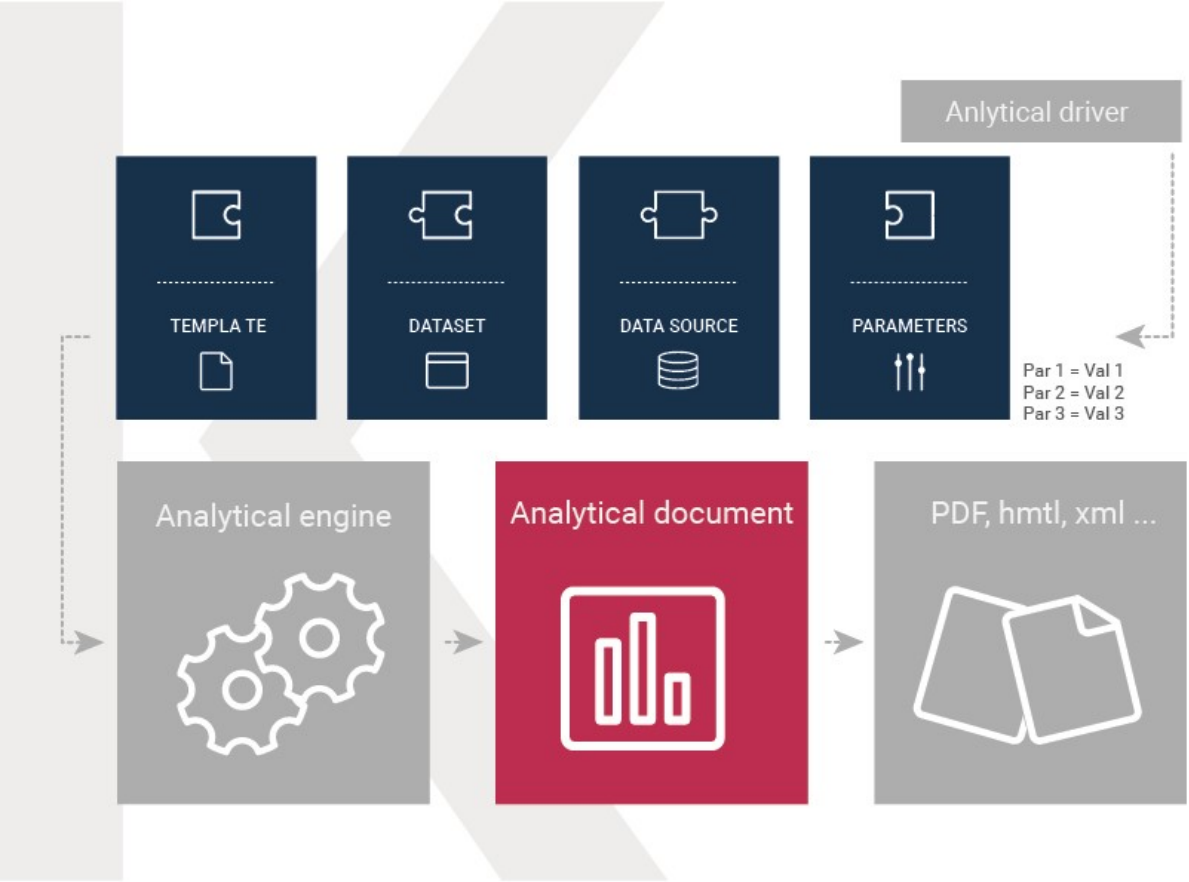


Fig. 5.46: Components of Knowage analytical document.

- 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 .

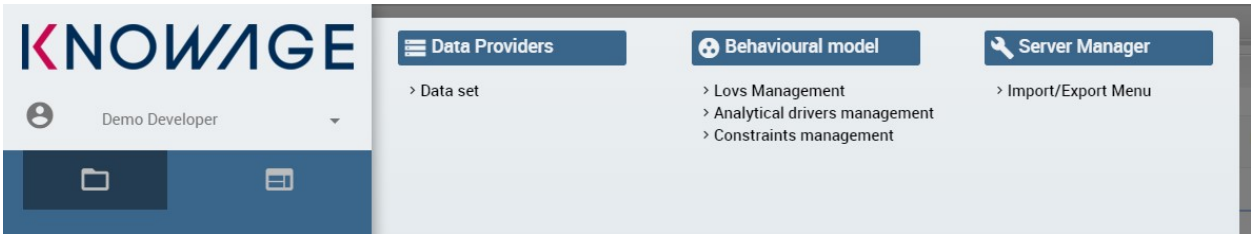


Fig. 5.47: Documents Development button.

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;

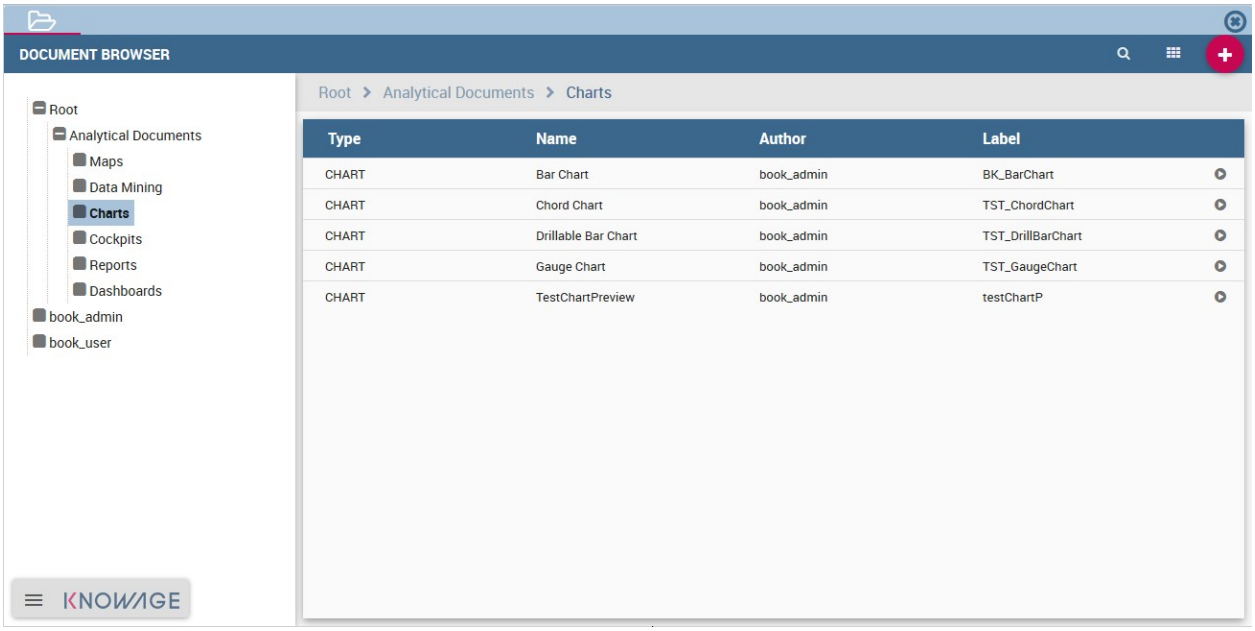


Fig. 5.48: Documents Development section.

Fig. 5.49: Changing documents view.

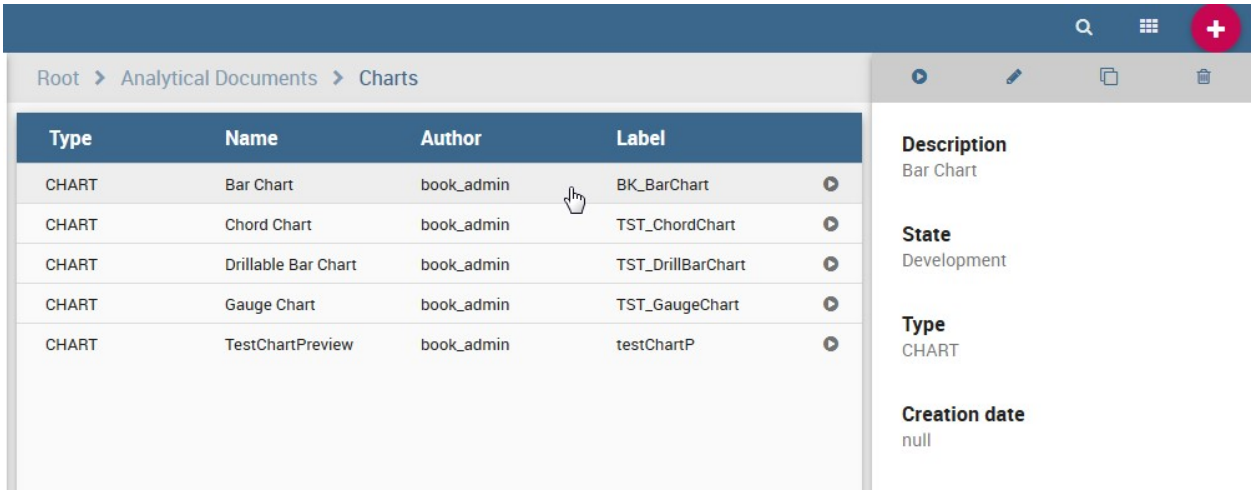


Fig. 5.50: Side panel.

- 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 alternatively see either the history of document templates or the functionality tree and the document position. If you want to copy or move a document from a folder into another, check or uncheck the corresponding folders (see the last figure of the *Document Visibility* paragraph).

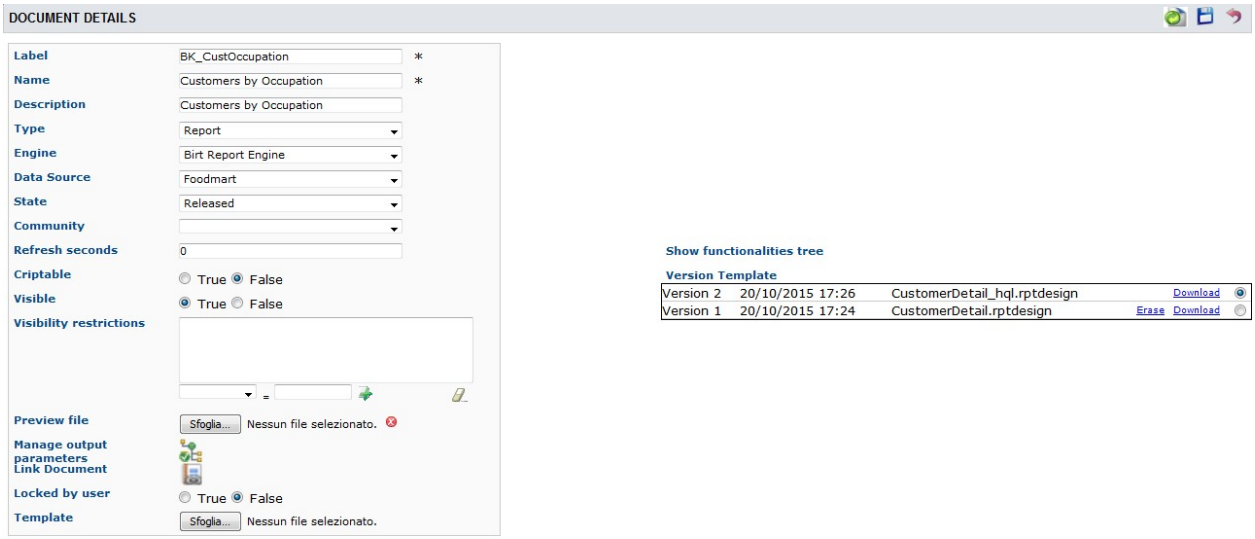


Fig. 5.51: Detail panel of Knowage analytical document.

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: **Geo-Referenced Analysis**, **Cockpit** and **Generic Document**. Please note that not all of them are available in all Knowage products.

To create a new generic document click the above-mentioned button and select **Generic Document**. You will be shown a window like the one in figure 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 data source that will feed your document with data, see the following figure .

Both should have already been defined in the corresponding sections for Knowageto show them in the available options of the menus. Select the data source from the drop down menu. Then click on the green icon and select the dataset from the lookup window.

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

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

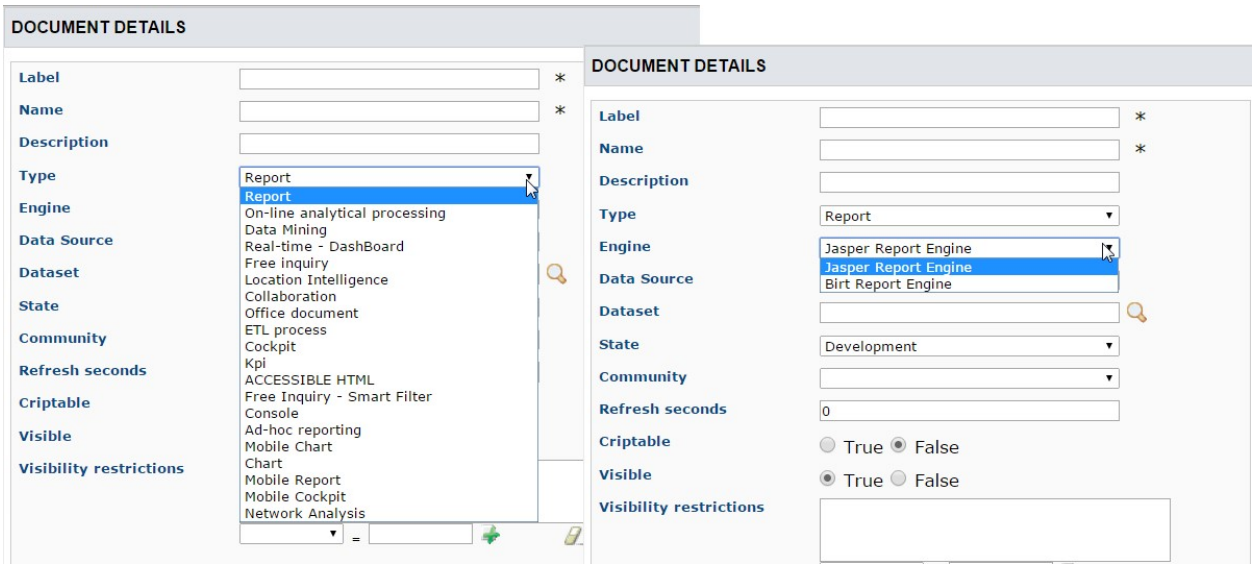


Fig. 5.52: Select Type and Engine for a new document.

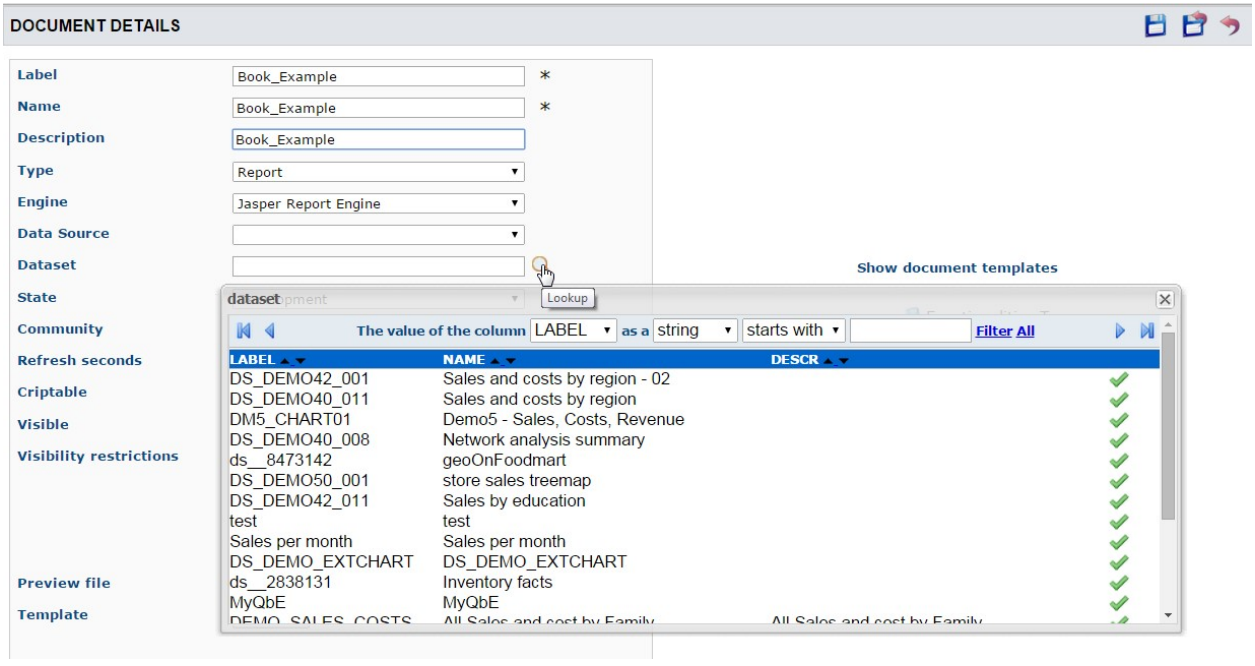


Fig. 5.53: Selecting a dataset for the document.

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 **Browse** next to **Template** and upload the template from your local file system. 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 in the document detail window, click on **Show document templates** in the right panel. All templates are saved with their date and name, and can be easily restored or deleted. To restore a template, choose it in the list by clicking on the selector, then remember to save: the new template will be uploaded. Using the same list you can download or delete a template.

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 maneged by administrator, who is in charge to profile user visibility too.

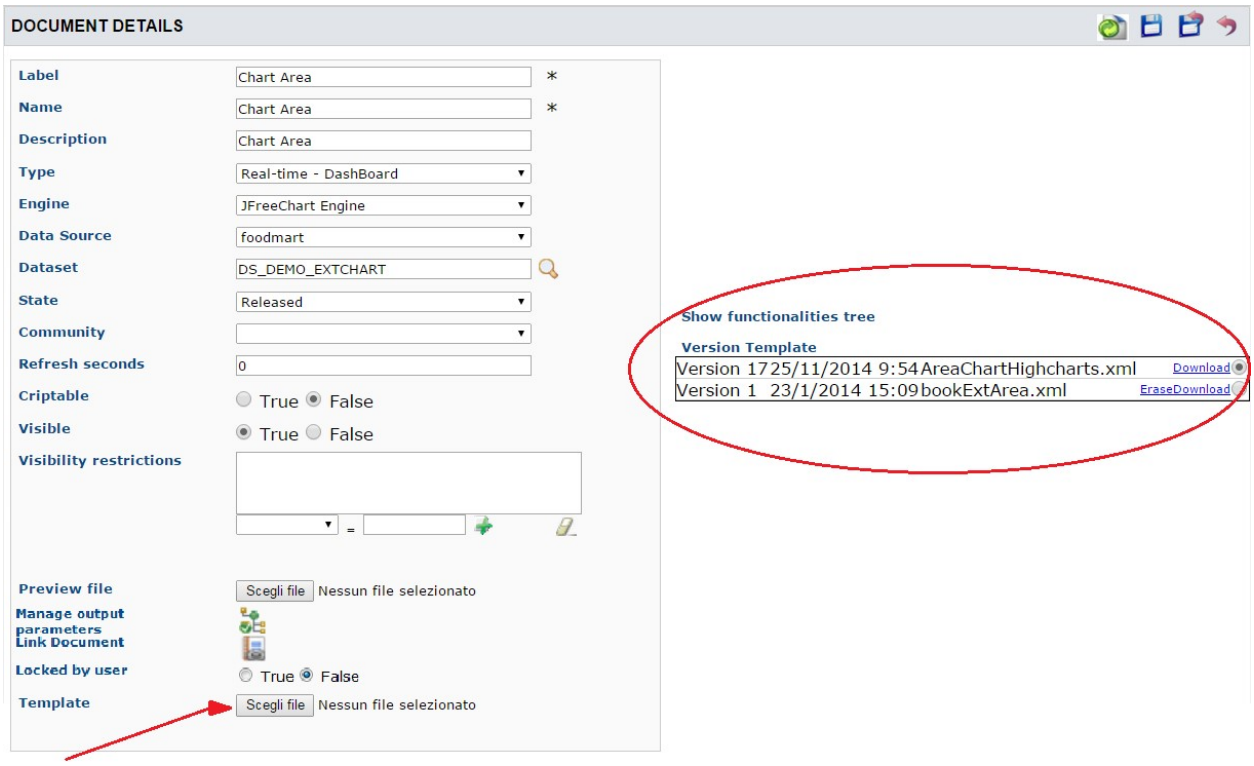


Fig. 5.54: Template versioning for analytical documents.

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

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 fine-grained level, beyond simple repository-based policies.

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

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.

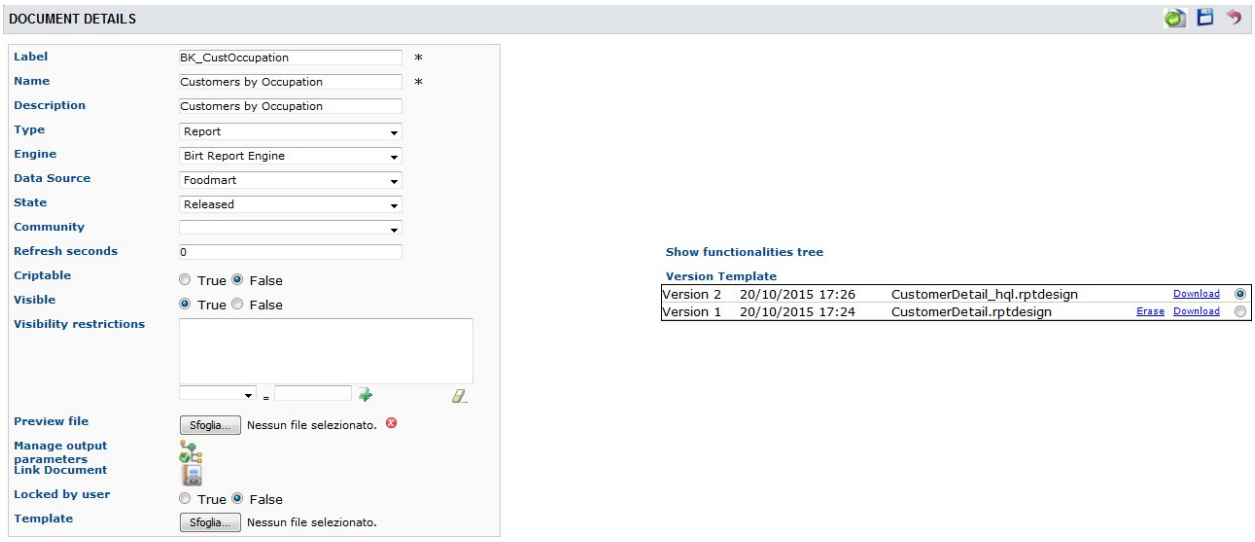


Fig. 5.55: Functionality Tree, document saving settings.

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

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

Choose a human readable name for the title. Then click on the lookup icon to choose the driver you wish to associate to the document. This will open the driver lookup window, where you can select the driver by clicking on the green icon . You can also inspect or delete a driver from here.

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

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

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

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

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

5.4.4.1 Associating a Spatial driver

As just seen, to filter on data visualization a user needs to associate an ananlytical 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

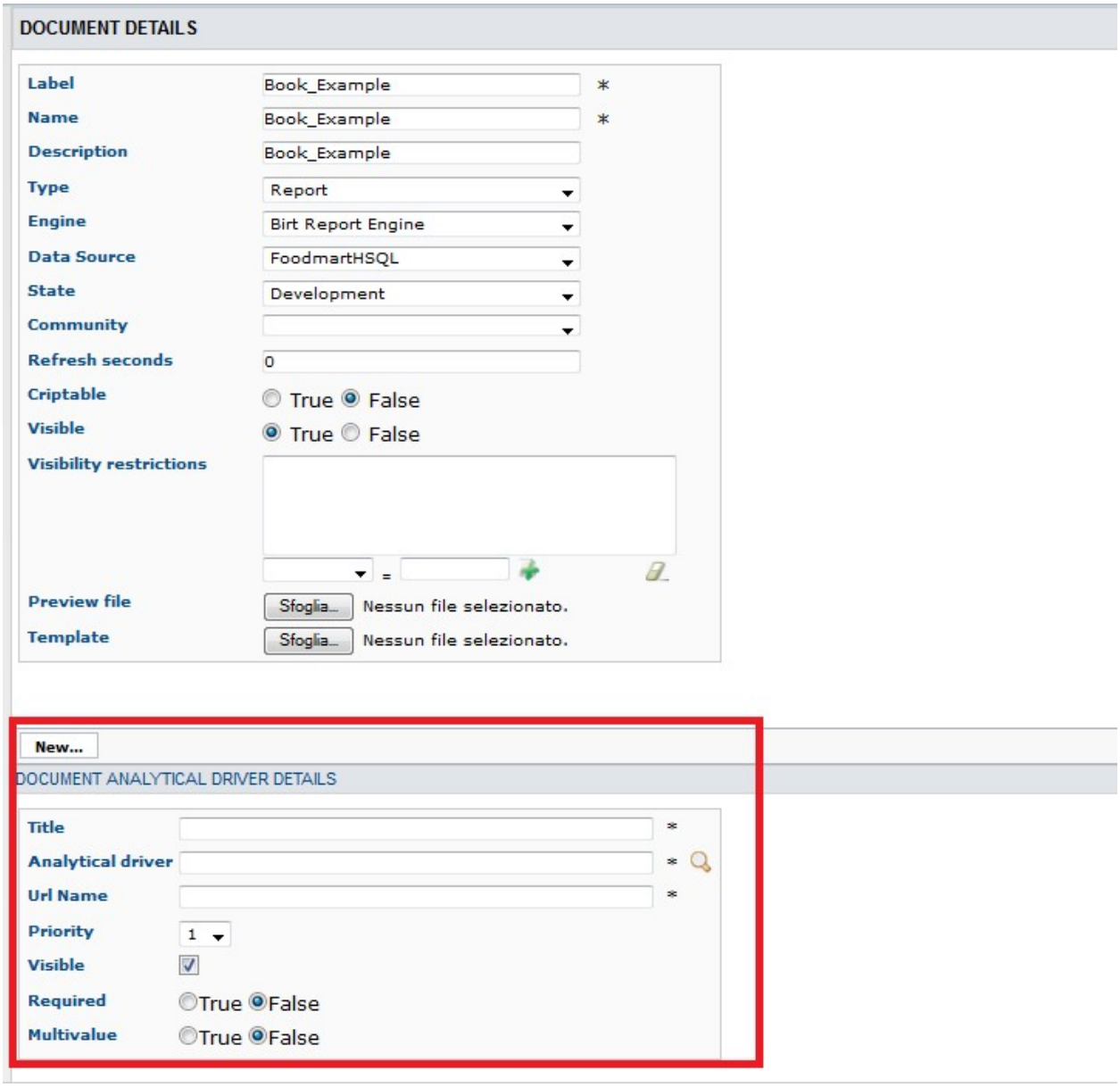


Fig. 5.56: Association with analytical driver panel.

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.

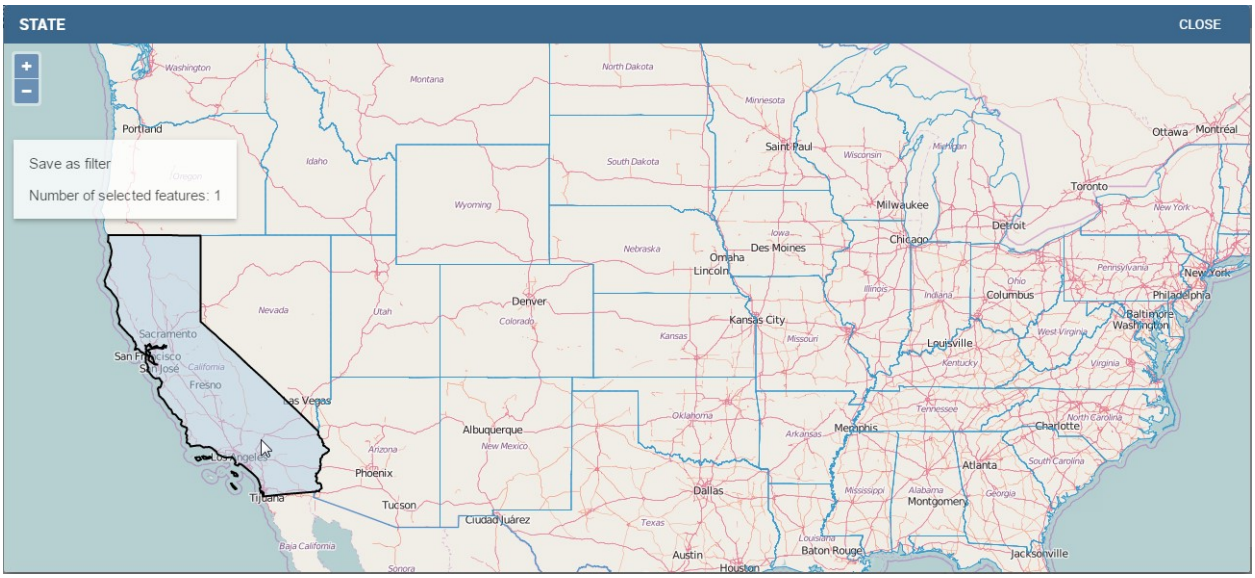


Fig. 5.57: Map filtering selection.

Click on "Close" button to confirm your selection. An example of the output is shown in the following figure.

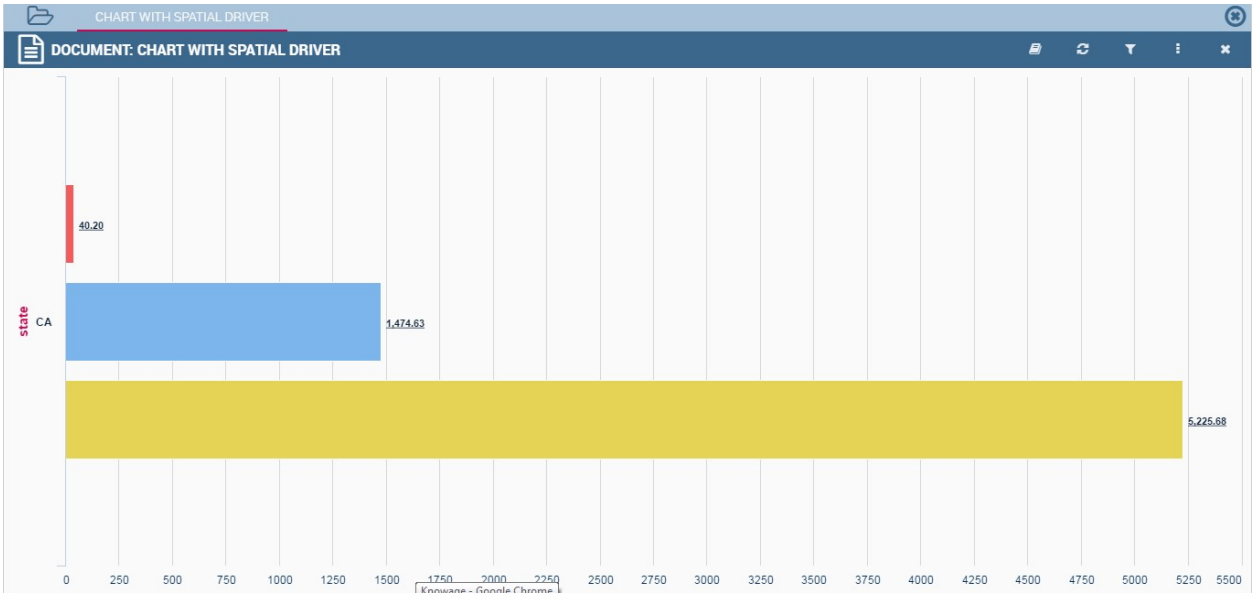


Fig. 5.58: Example of document run with a spatial filter.

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.

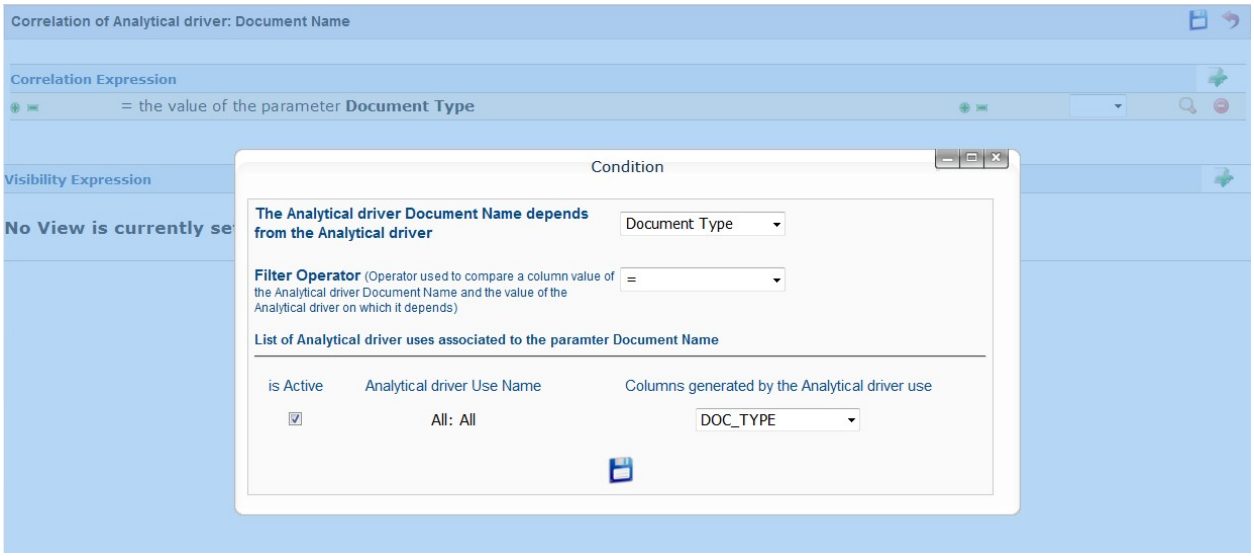


Fig. 5.59: Definition of the correlation.

To set the correlation, click the tab of the child parameter and displaying the details click on the correlation button . Here you can add a new correlation rule by clicking on . Here you need to define:

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

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

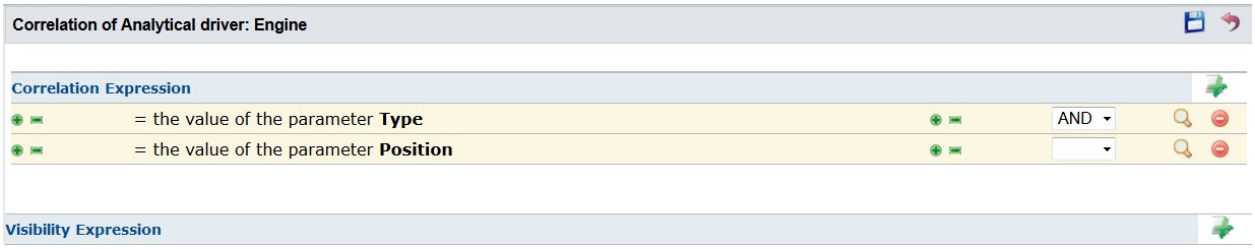


Fig. 5.60: Multiple correlations.

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

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

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}
```

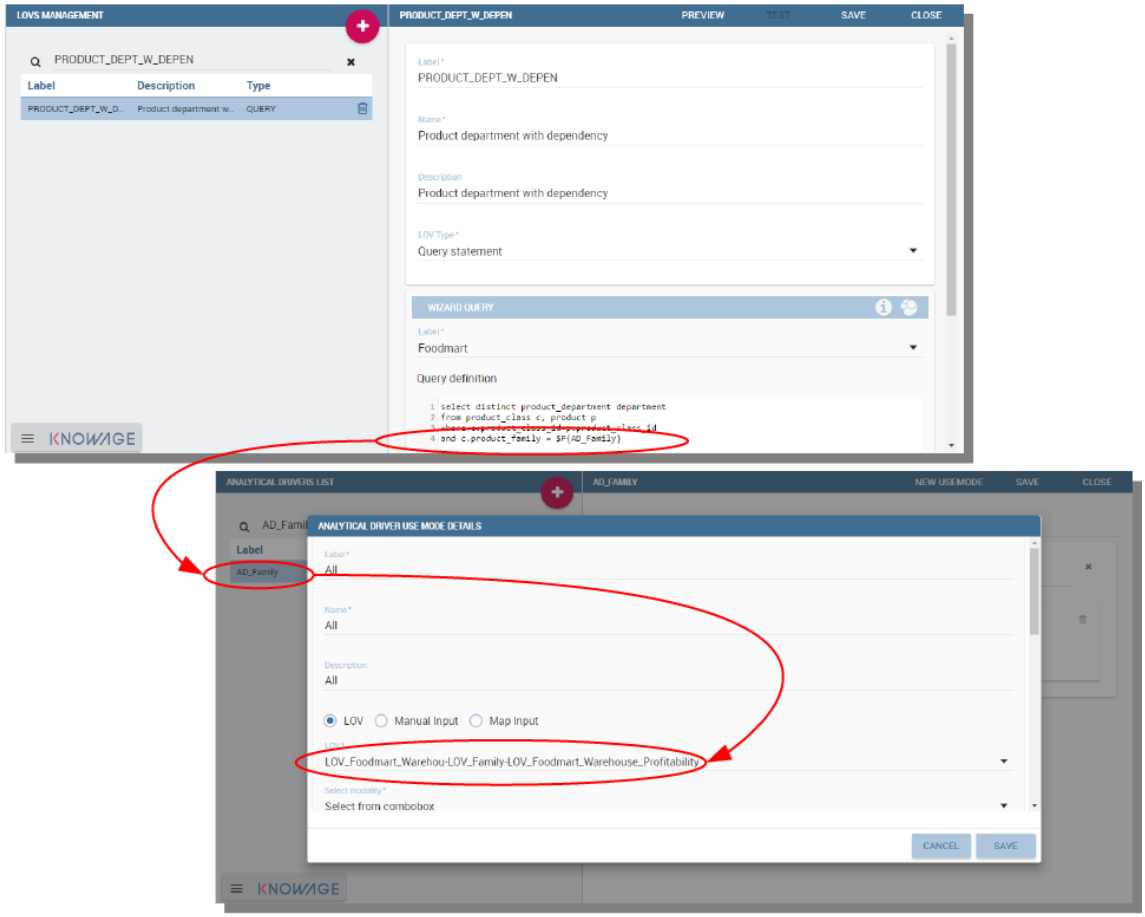


Fig. 5.61: Correlation passing driver values to LOV query .

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 the hiding or showing of a child parameter in the parameters mask. Note that in the first case, the child parameter is hidden by default, while in the second case the parameter is shown by default.

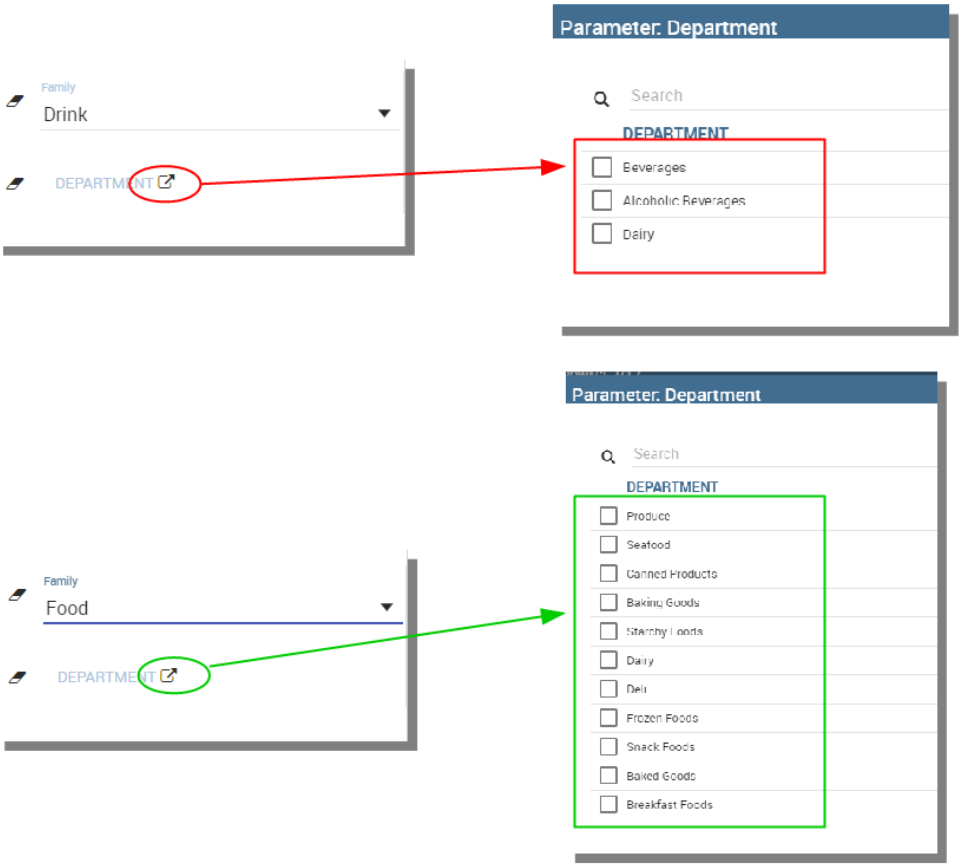


Fig. 5.62: Filtering with correlation.

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

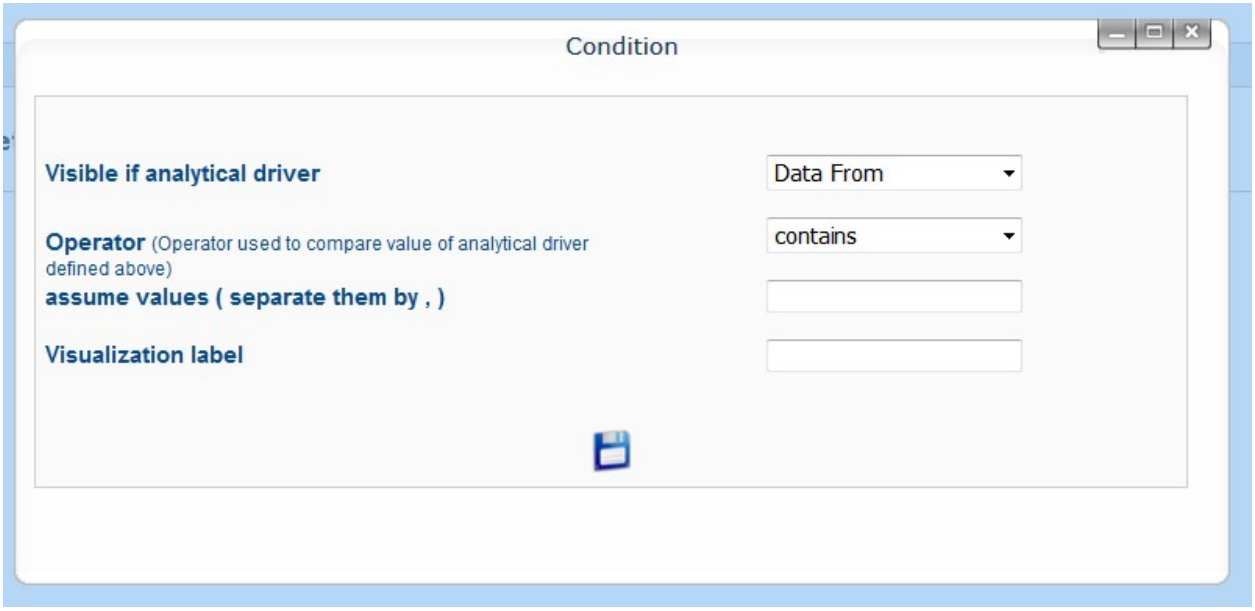


Fig. 5.63: Visibilty expressions.

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 necessarly 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 **Manage outputparameters** and the **Output parameters** dialog will open.

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**).

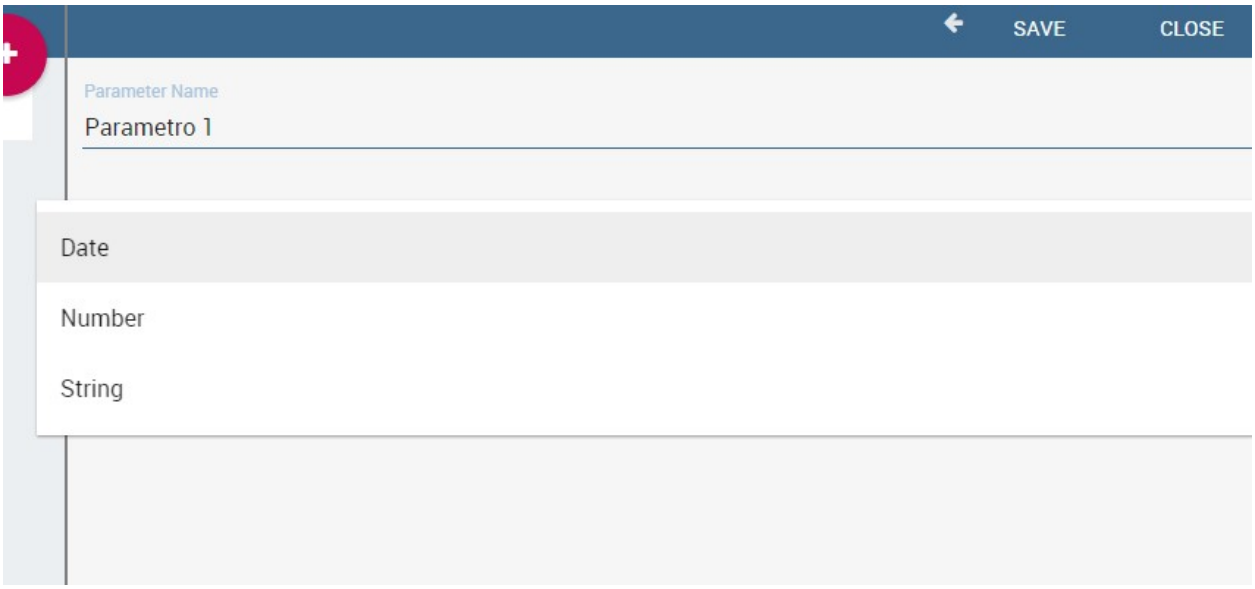


Fig. 5.64: Setting an output parameter.

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.

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 betweeen input/output parameters by simply dragging and dropping a parameter from the left column on another of the right column.

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

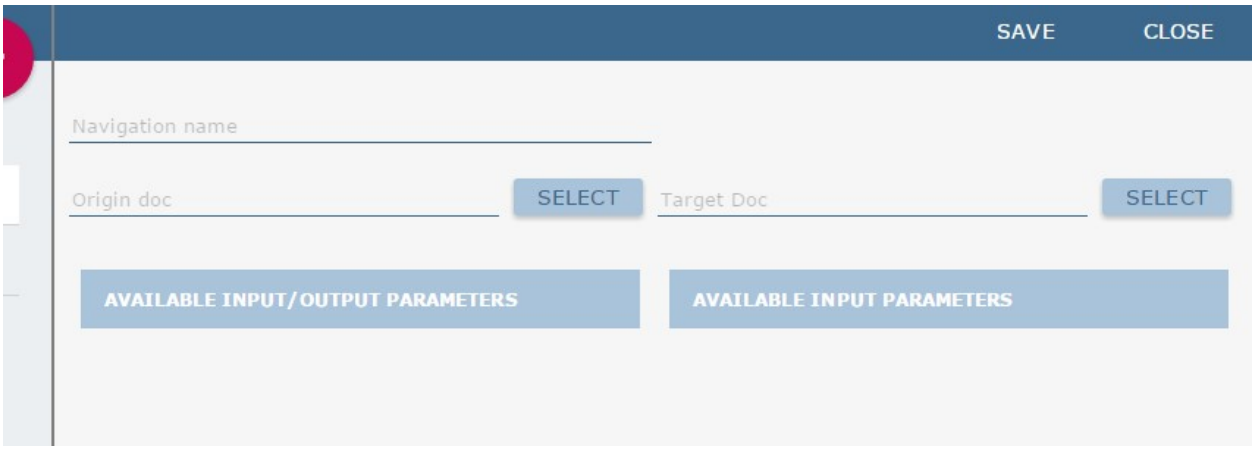


Fig. 5.65: Cross navigation GUI.

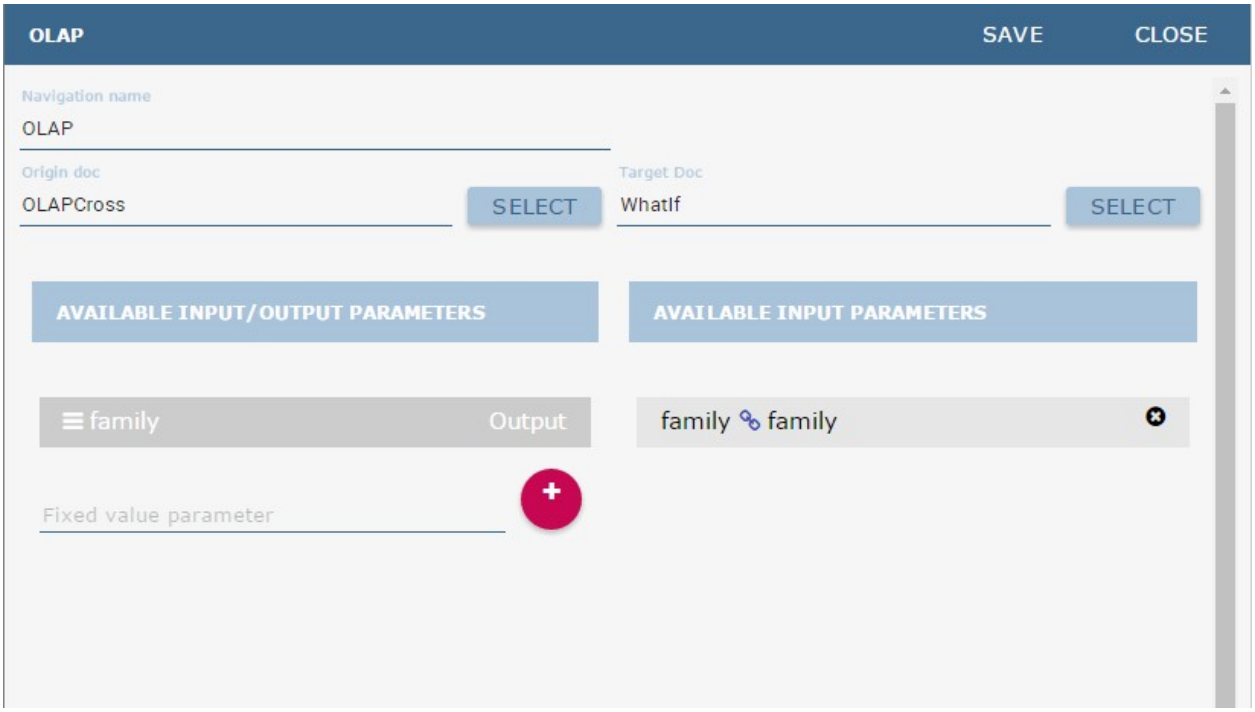


Fig. 5.66: Setting the cross navigation through the menu item.

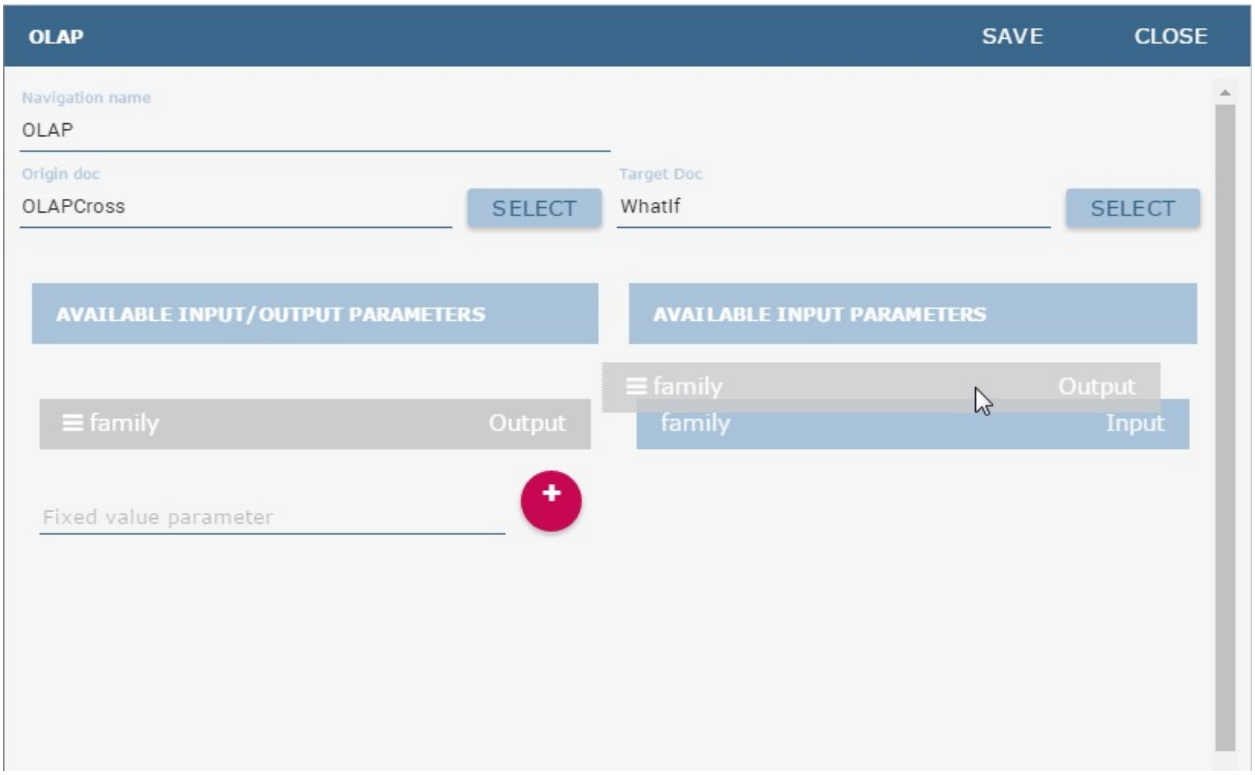


Fig. 5.67: Relating parameters.

Fig. 5.68: Association betweeen parameters.

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

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
- 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.

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

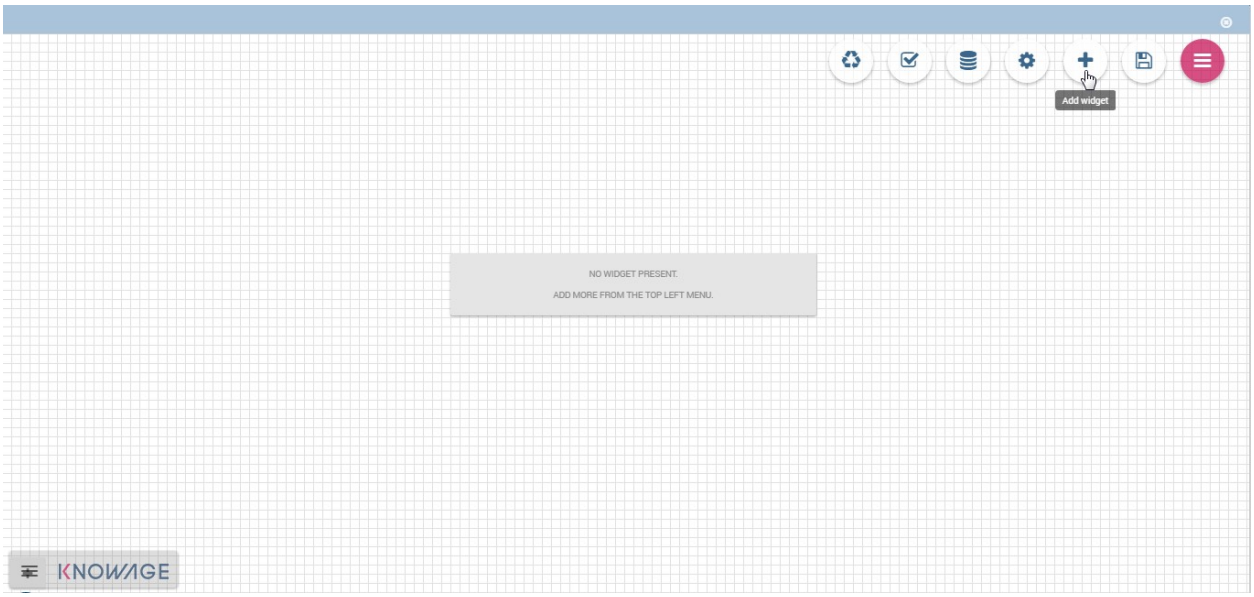


Fig. 5.69: Add a chart to a cockpit.

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 found seven 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**

These seven 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 and the Configuration tabs are described in a more specific way.

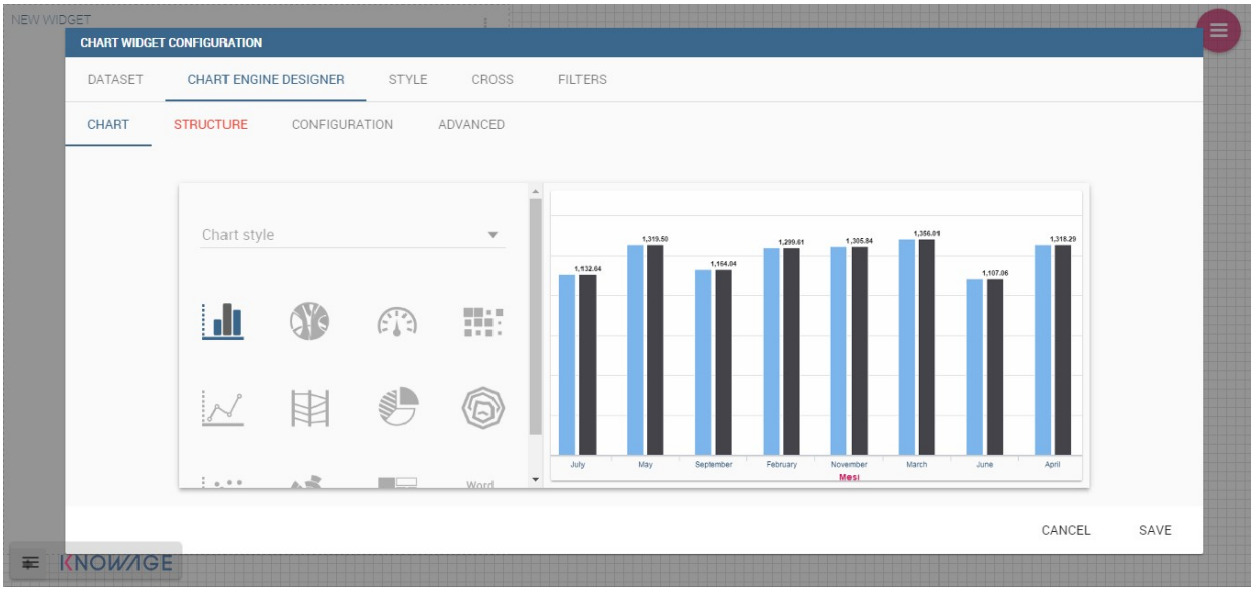


Fig. 5.70: Chart editor.

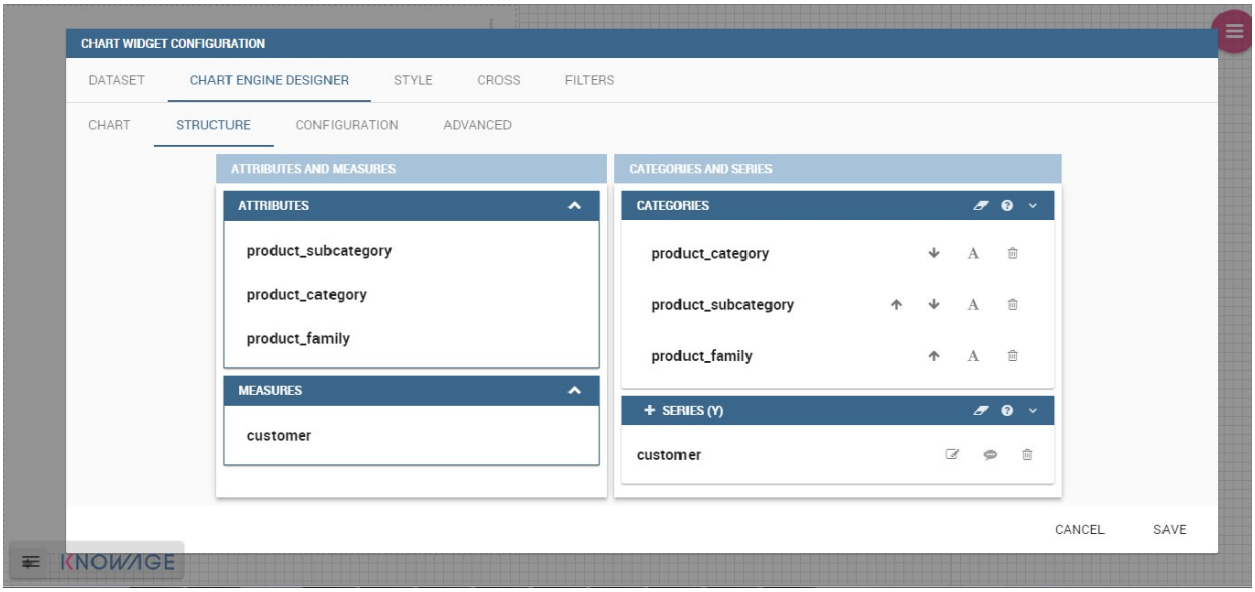


Fig. 5.71: Chart structure.

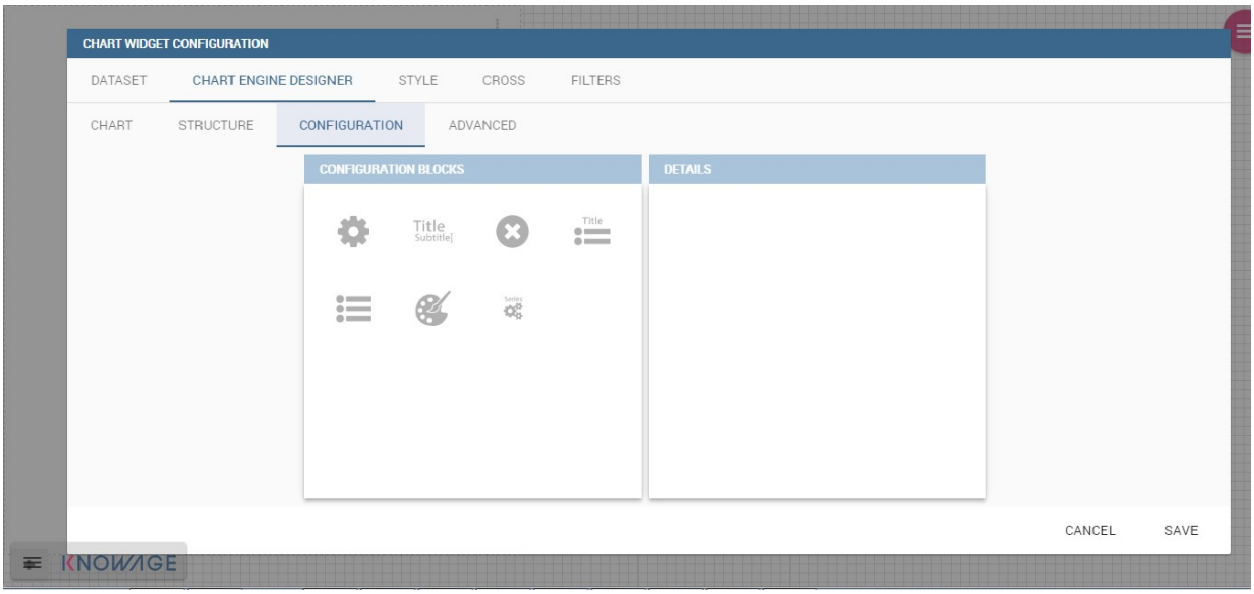


Fig. 5.72: Chart configuration.

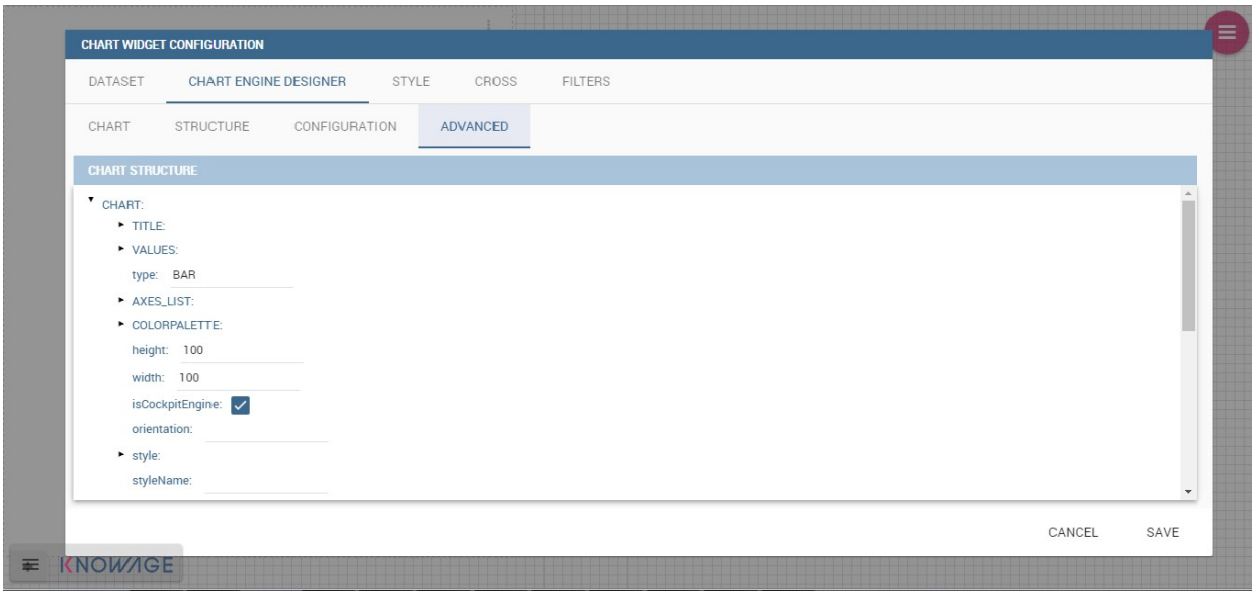


Fig. 5.73: Chart Advanced Features.

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, titleand 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:

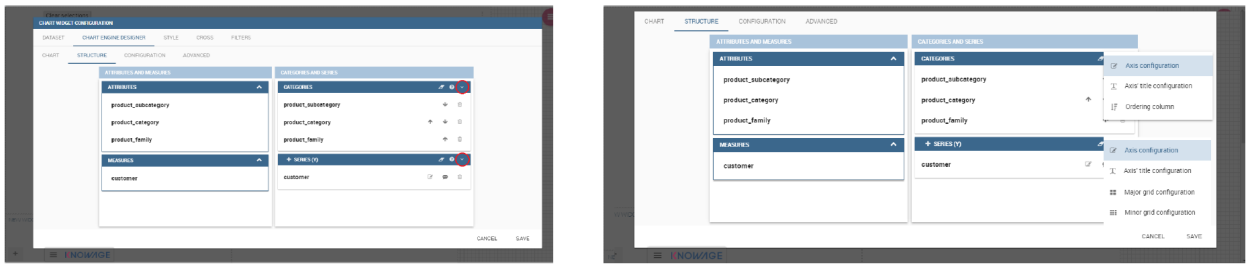


Fig. 5.74: From left to right: (a) Generic configuration axis (the specific arrow). (b) Generic configuration axis.

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, 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.

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.

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.

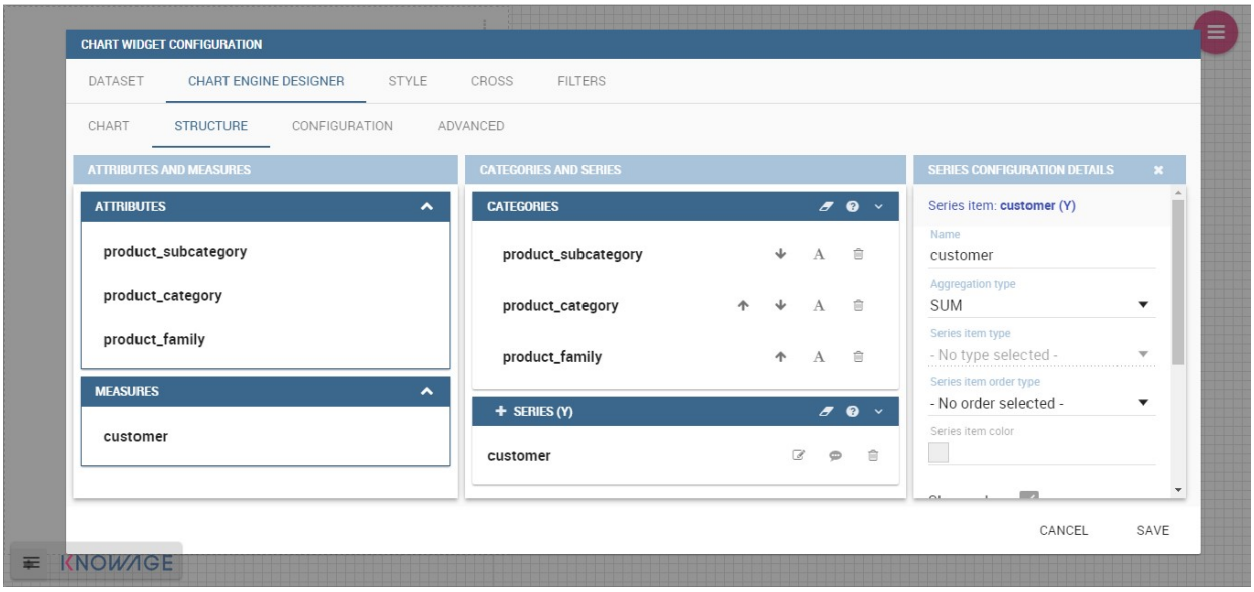


Fig. 5.75: Series style configuration.

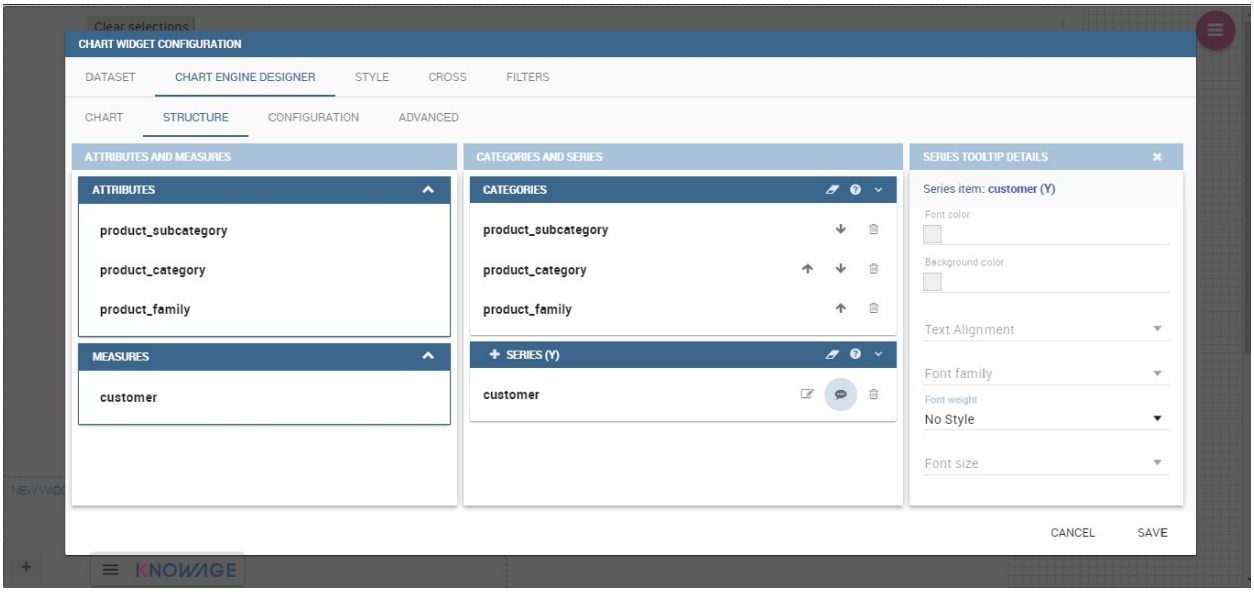


Fig. 5.76: Series tooltip details.

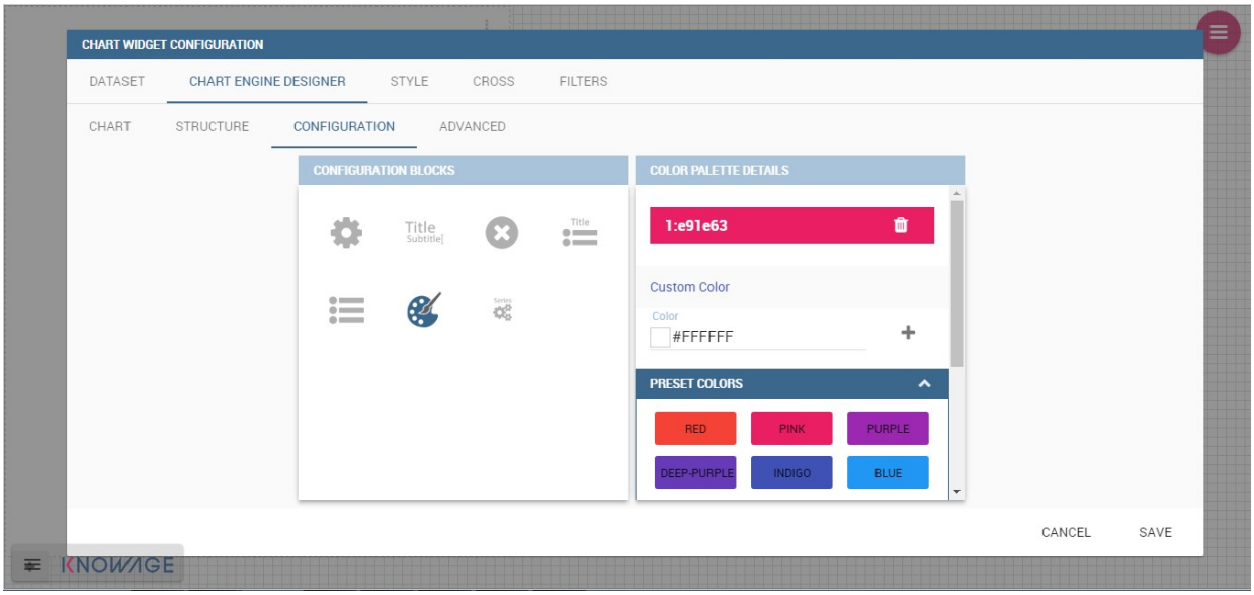


Fig. 5.77: Color box editing.

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 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.

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.

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

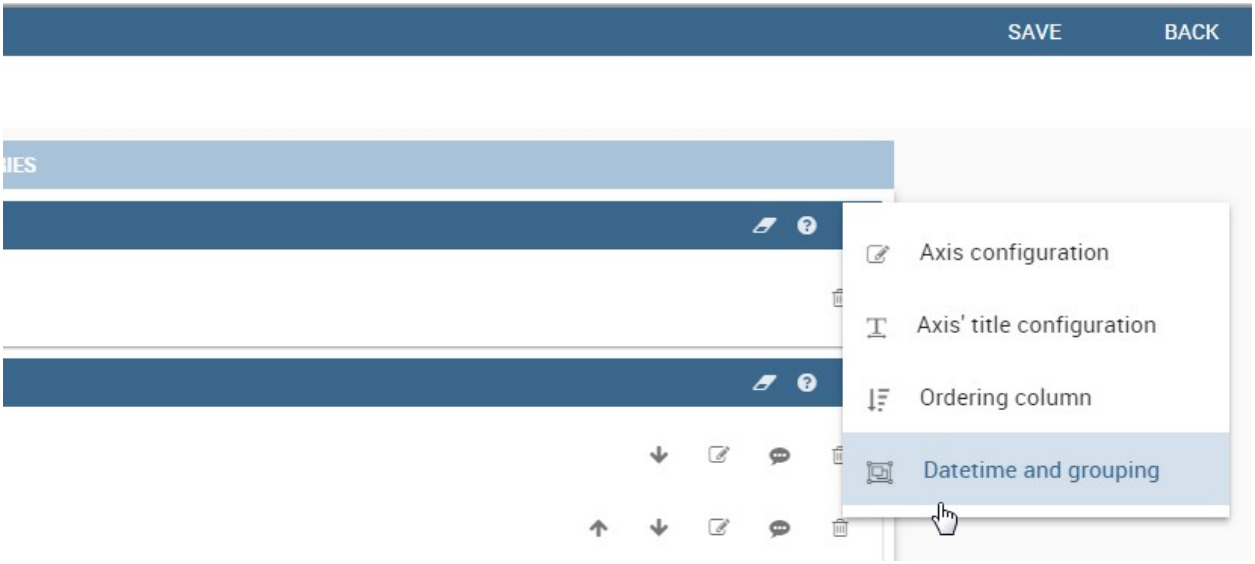


Fig. 5.78: Datetime and grouping function.

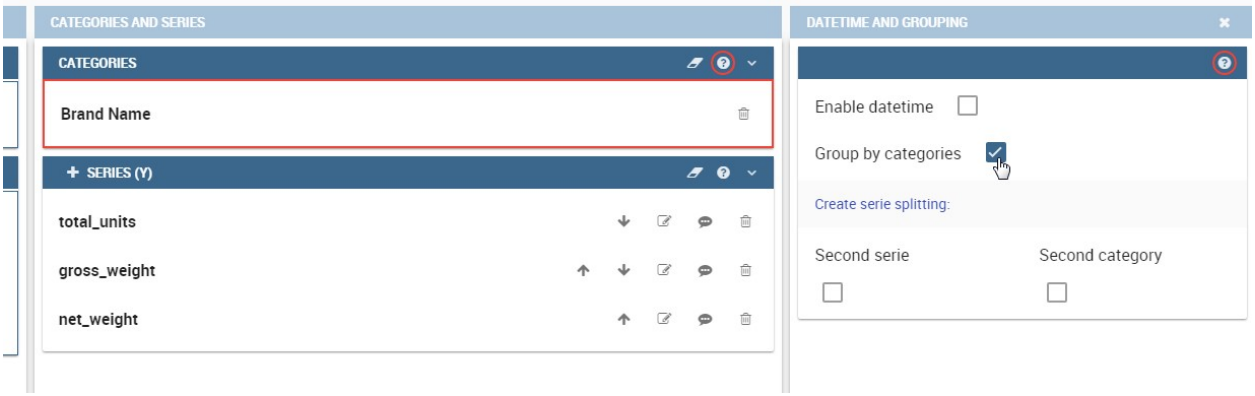


Fig. 5.79: Error alarm when enabling the grouping function.

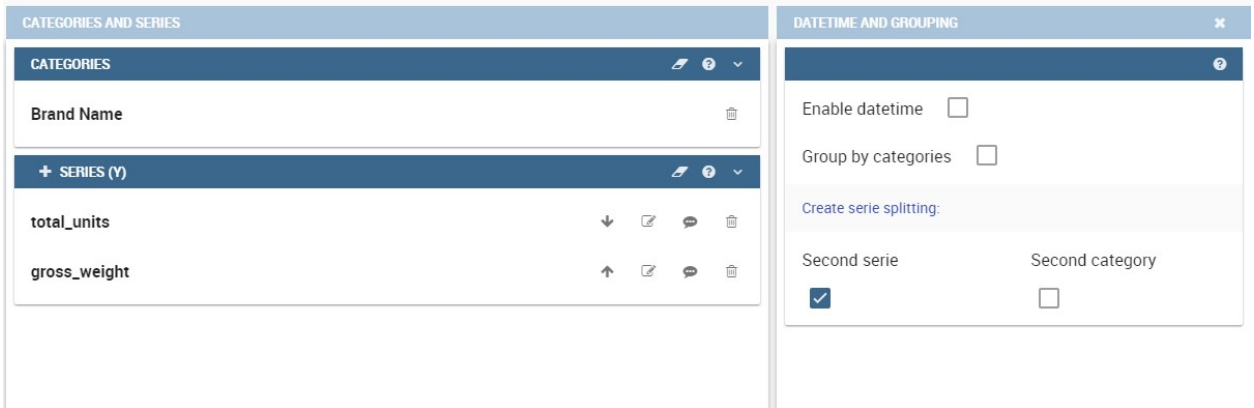


Fig. 5.80: Split over second series.

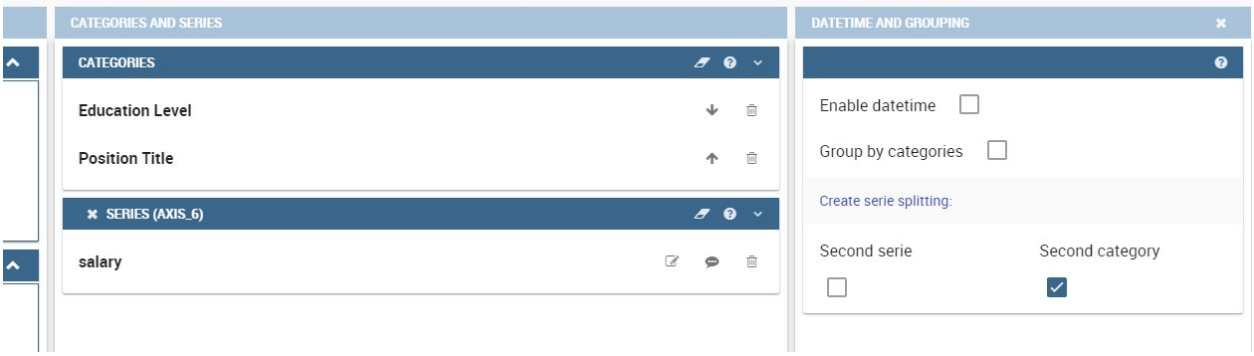


Fig. 5.81: Split over second category.

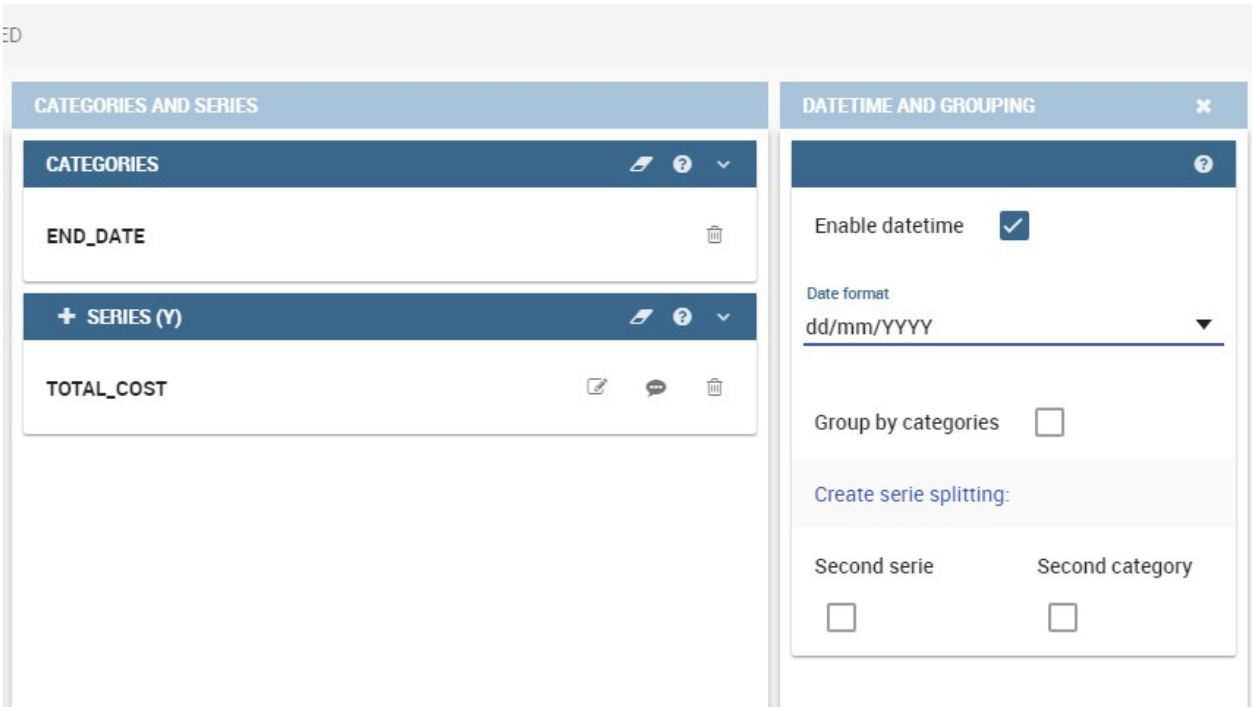


Fig. 5.82: Datetime function usage.

(zero) value. It is check box **alignAxis** where checked means that axises 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 toolip: to show/hide absolute value and/or percentage. Inside **tooltip** property of serie object you can find properies **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.

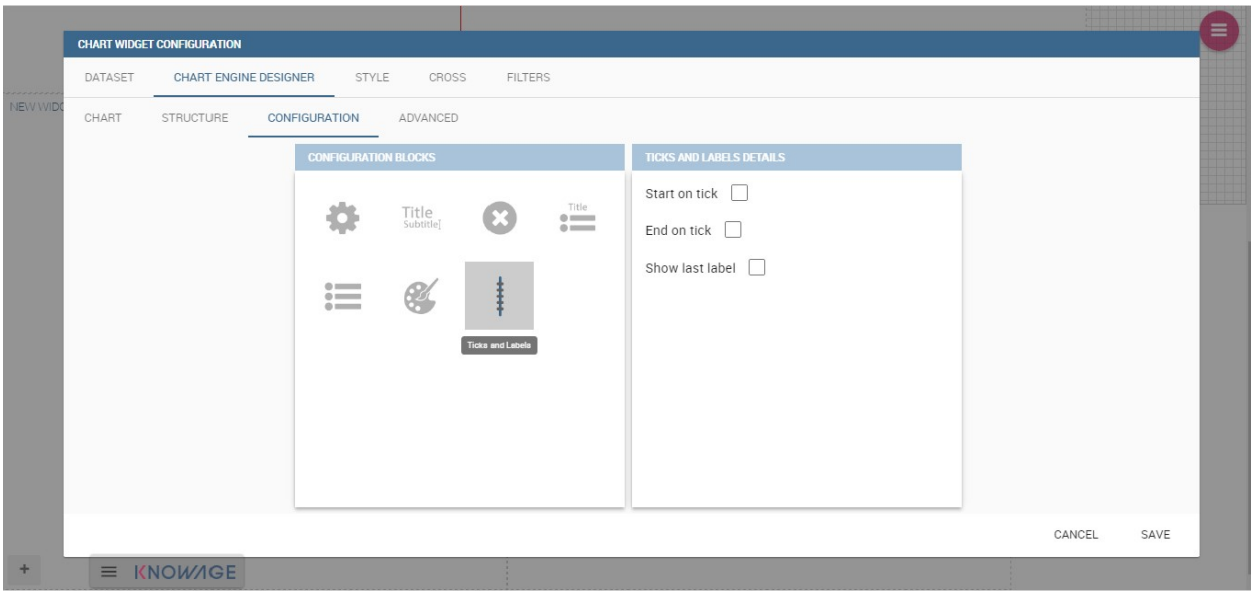


Fig. 5.83: Scatter Chart, ticks and labels details.

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 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.

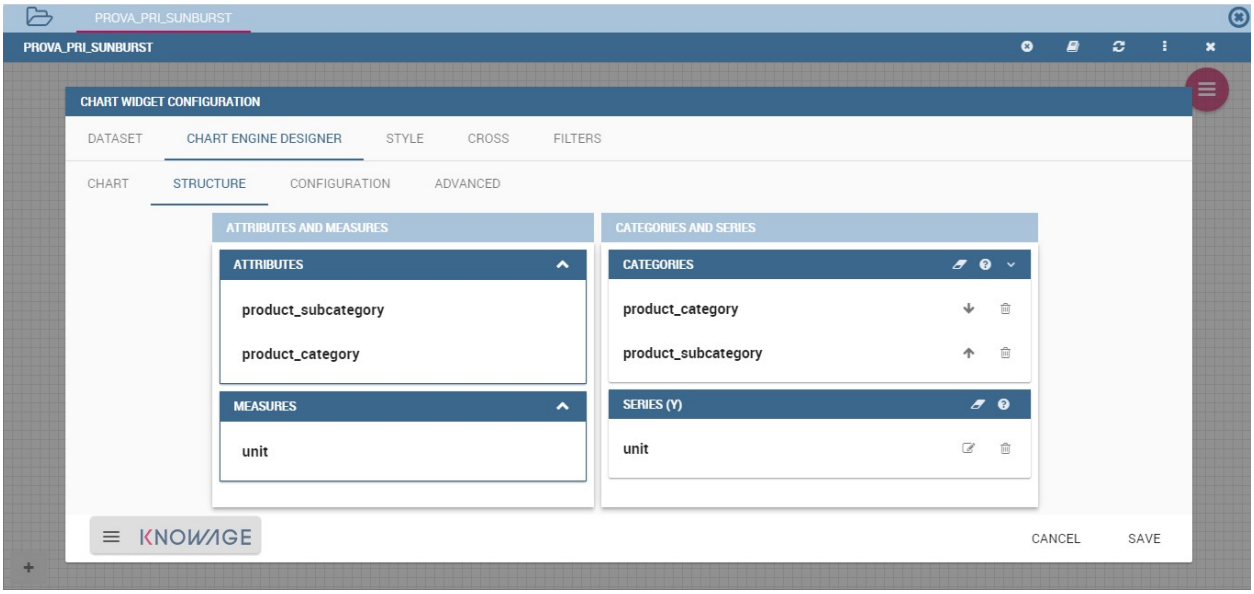


Fig. 5.84: Sunburst configuration.

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. To custom the root-node path, click on the **Sequence** icon and choose position, label tail size and text style. 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. 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 value for property **scale**.

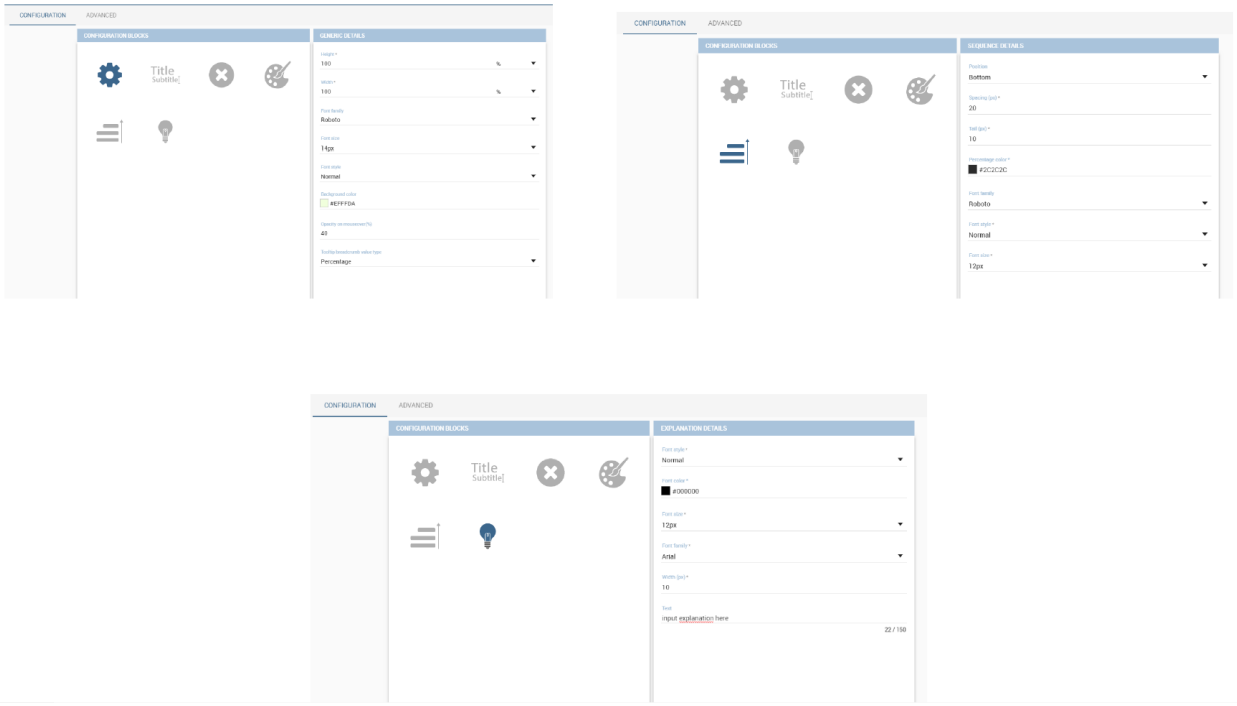


Fig. 5.85: Generic, Sequence and Explanation configuration

Fig. 5.86: From left to right: (a) Sunburst. (b) Sunburst category.(c) Sunburst subcategory.

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 datails**. Here the use 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.

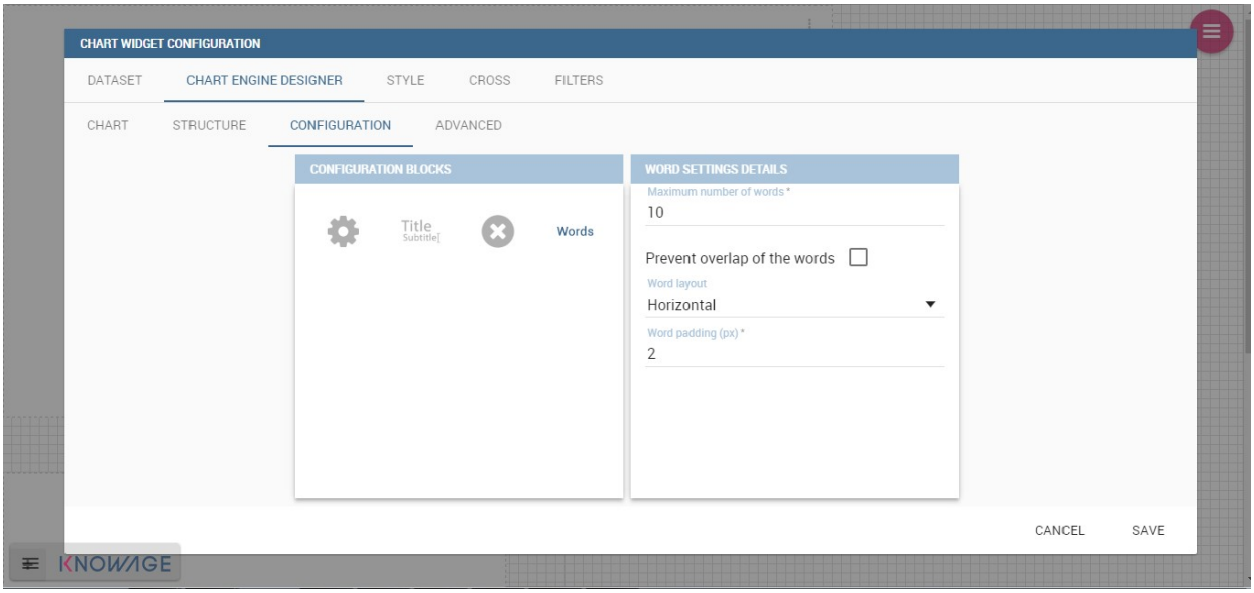


Fig. 5.87: Wordcloud chart specific configuration.

5.5.2.5 Treemap chart

The treemap is a graphical representation of hierarchical data, which are displayed as nestled 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 usefull 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 reflects 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. 5.88: 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 lenght, 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.

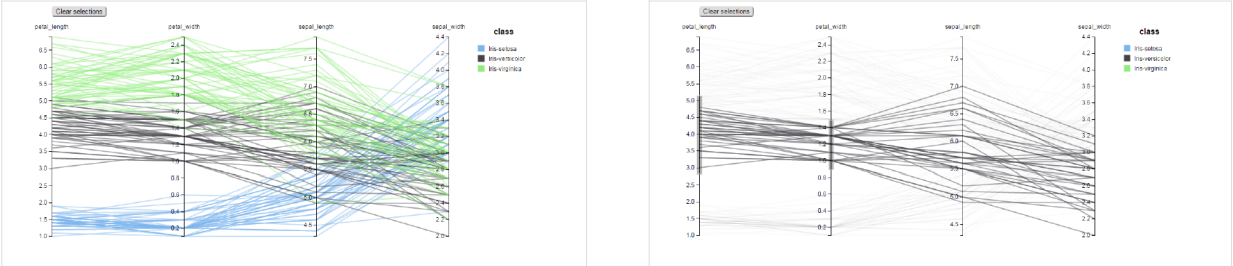


Fig. 5.89: From left to right: (a) Parallel. (b) Parallel chart selection.

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”.

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.

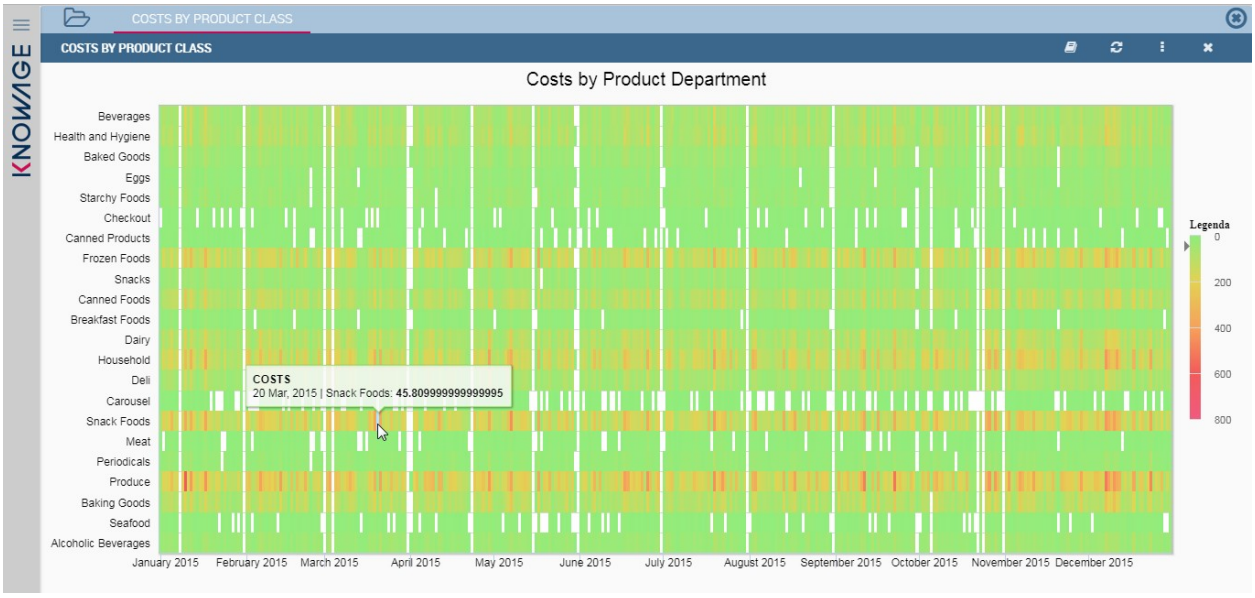


Fig. 5.90: Heatmap example.

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.

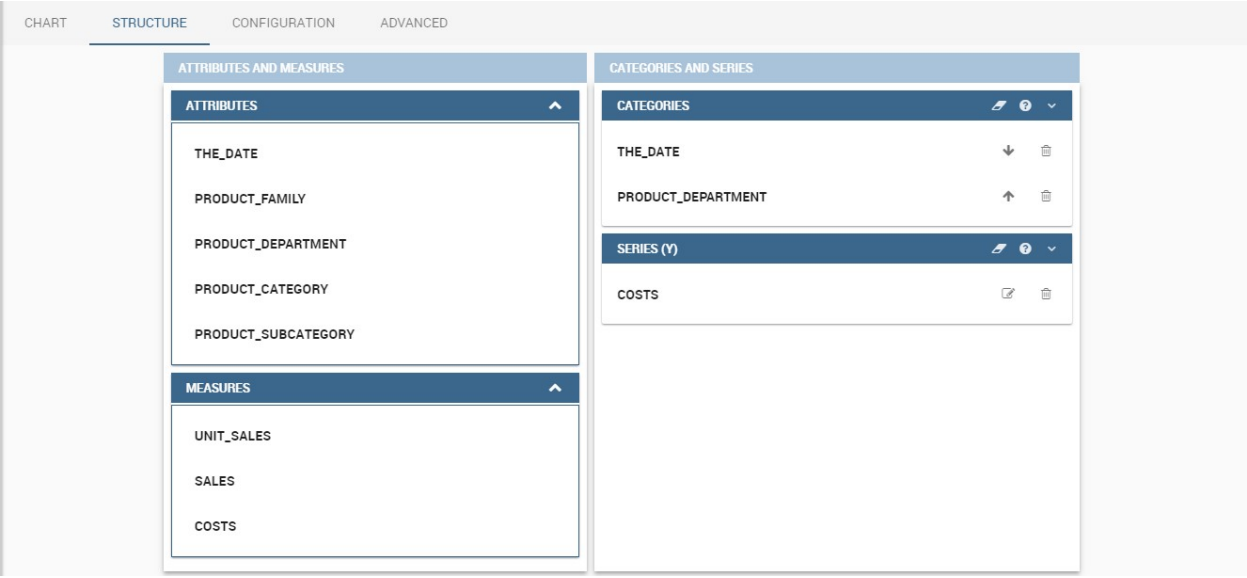


Fig. 5.91: Configuring the attributes and the series for the heatmap chart.

Note that for series axis it is possible to specify the values' range by assigning a minimun and the maximum value, as shown in figure below. Otherwise, the engine will automatically link the axis scale to dataset results set.

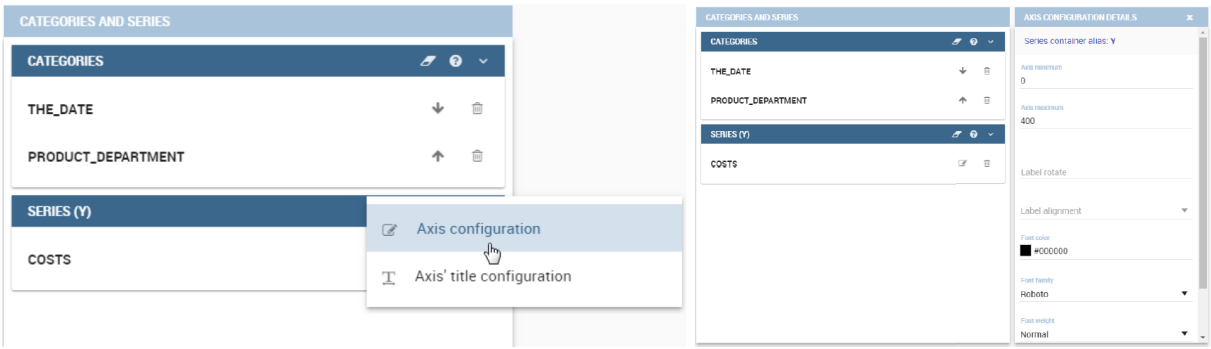


Fig. 5.92: Configure min and max values for series.

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.

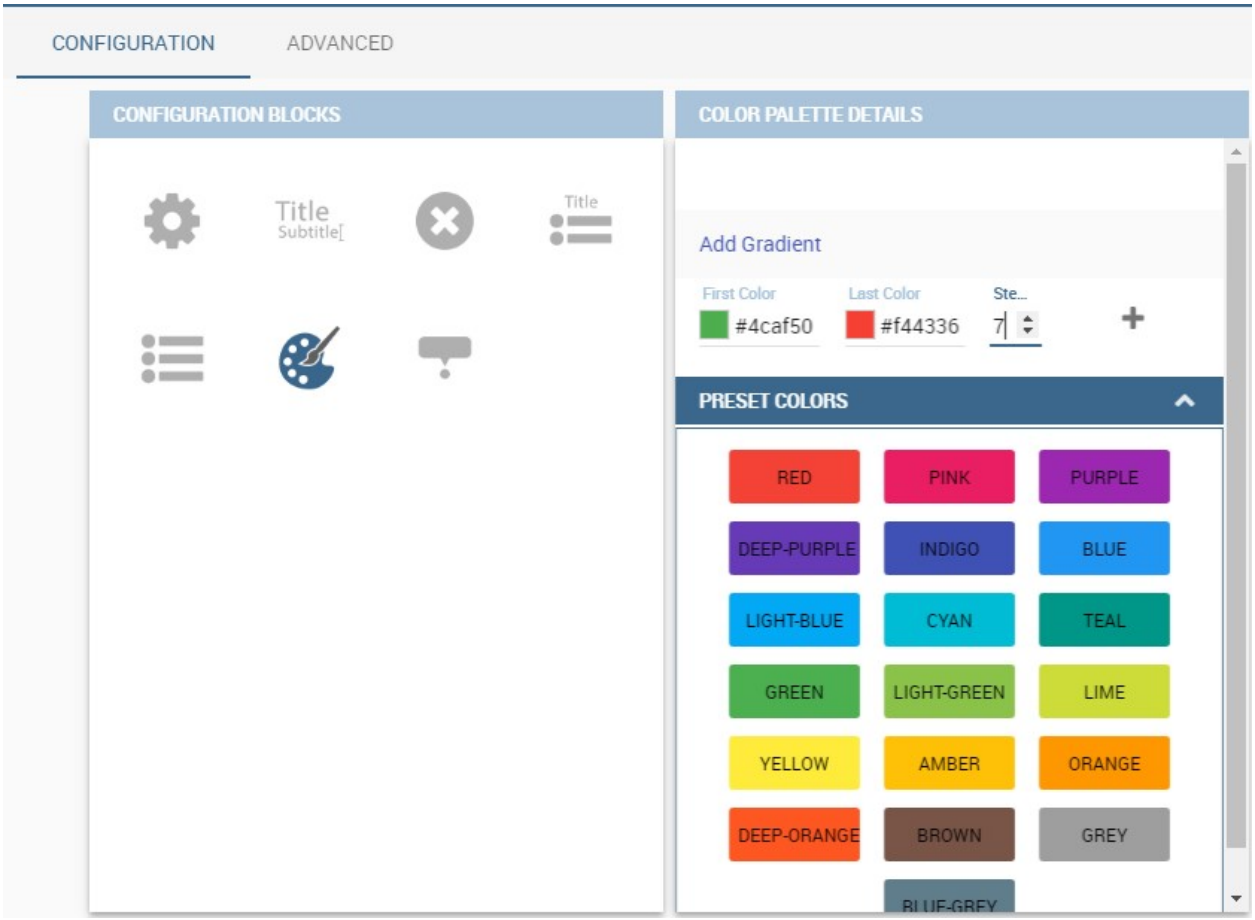


Fig. 5.93: Add gradient panel.

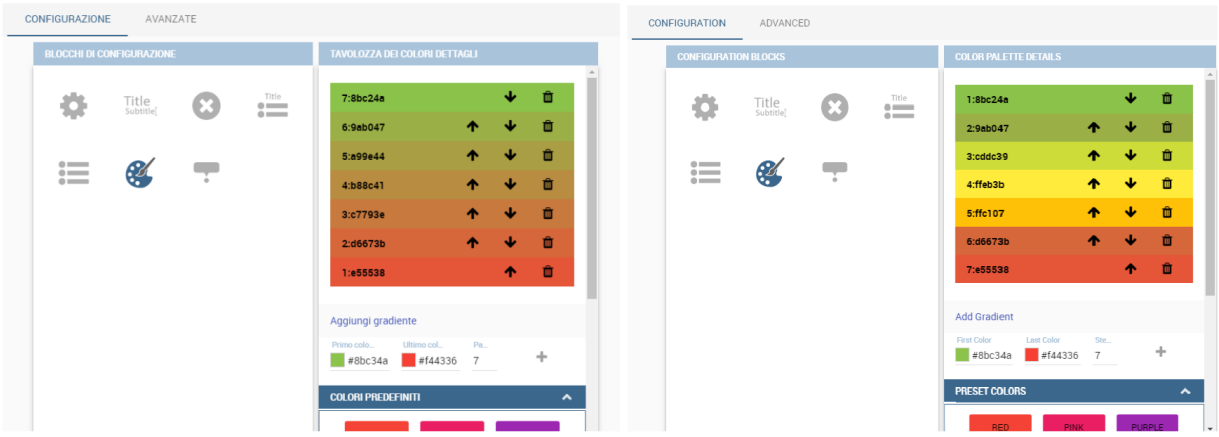


Fig. 5.94: Custom color scale.

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 direct 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:

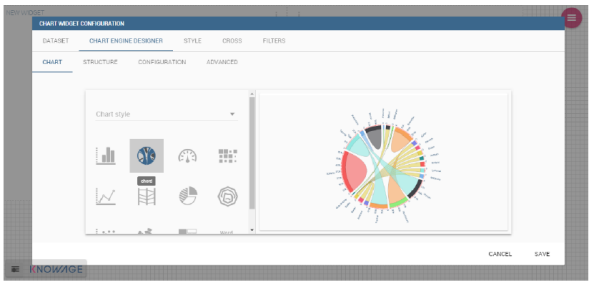
Table 5.7: Example of dataset for the chord chart.

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.



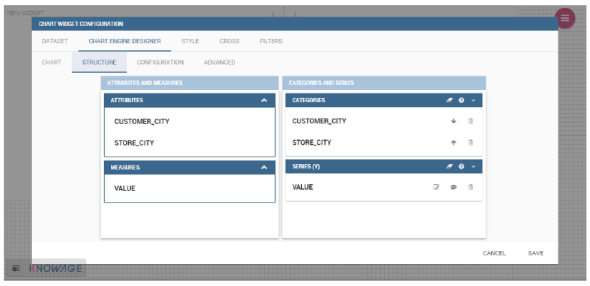


Fig. 5.95: Chord configuration.

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.

5.5.3 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:

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

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.

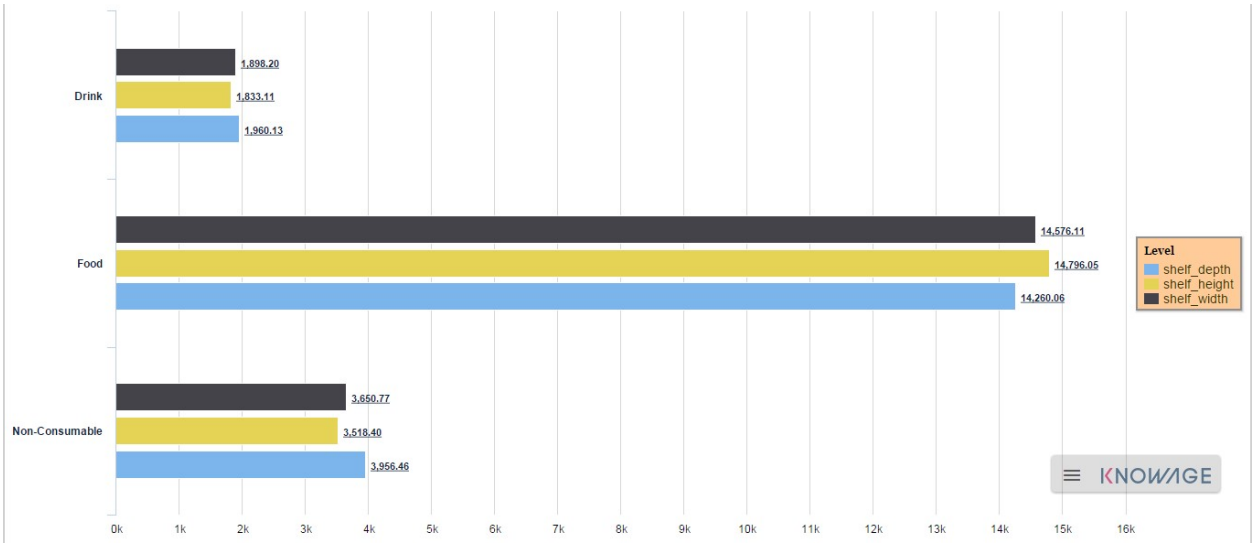


Fig. 5.96: Drillable Bar Chart

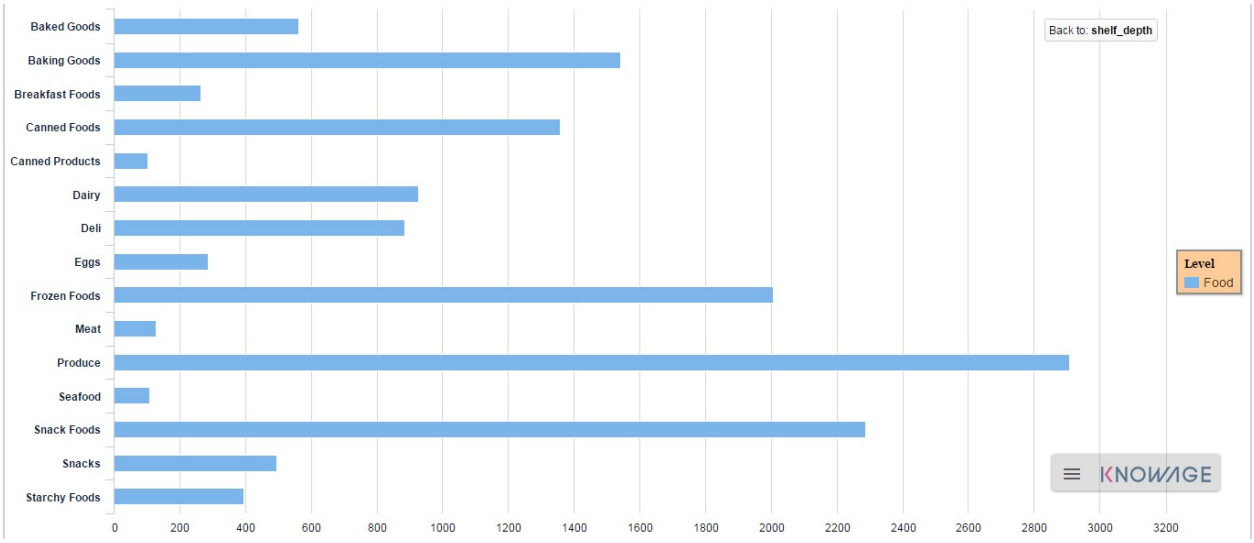


Fig. 5.97: Drillable Bar Chart: first drill

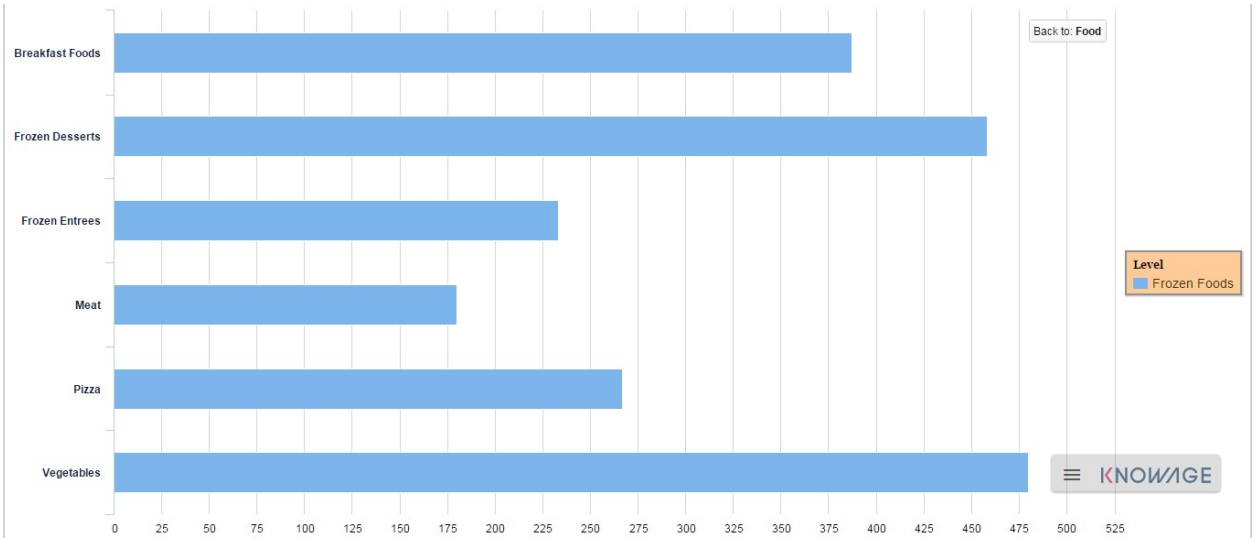


Fig. 5.98: Drillable Bar Chart: second drill

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

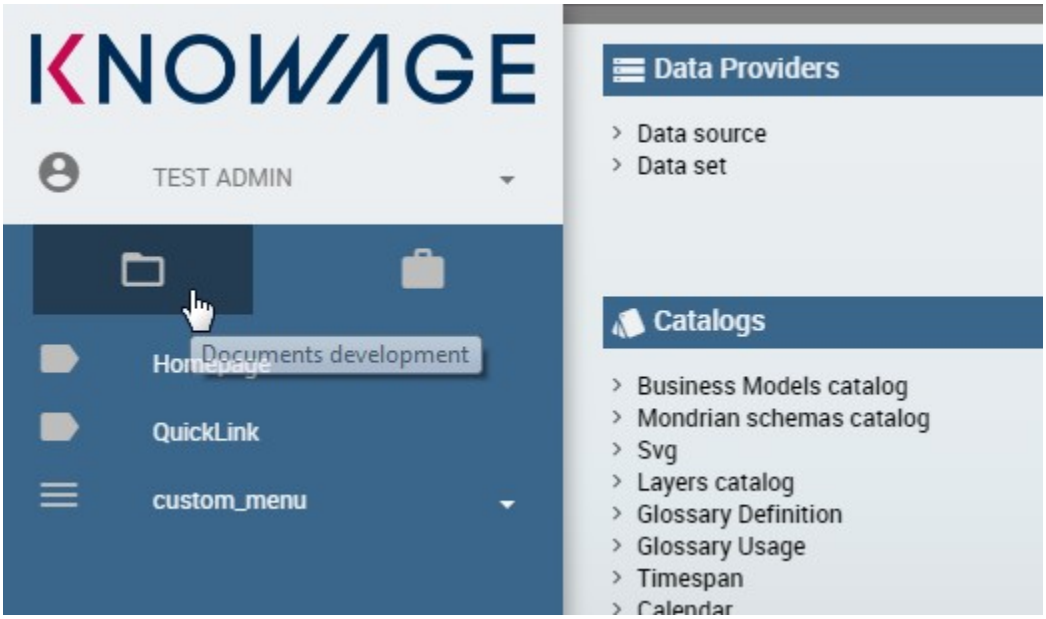


Fig. 5.99: Documents Development.

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.

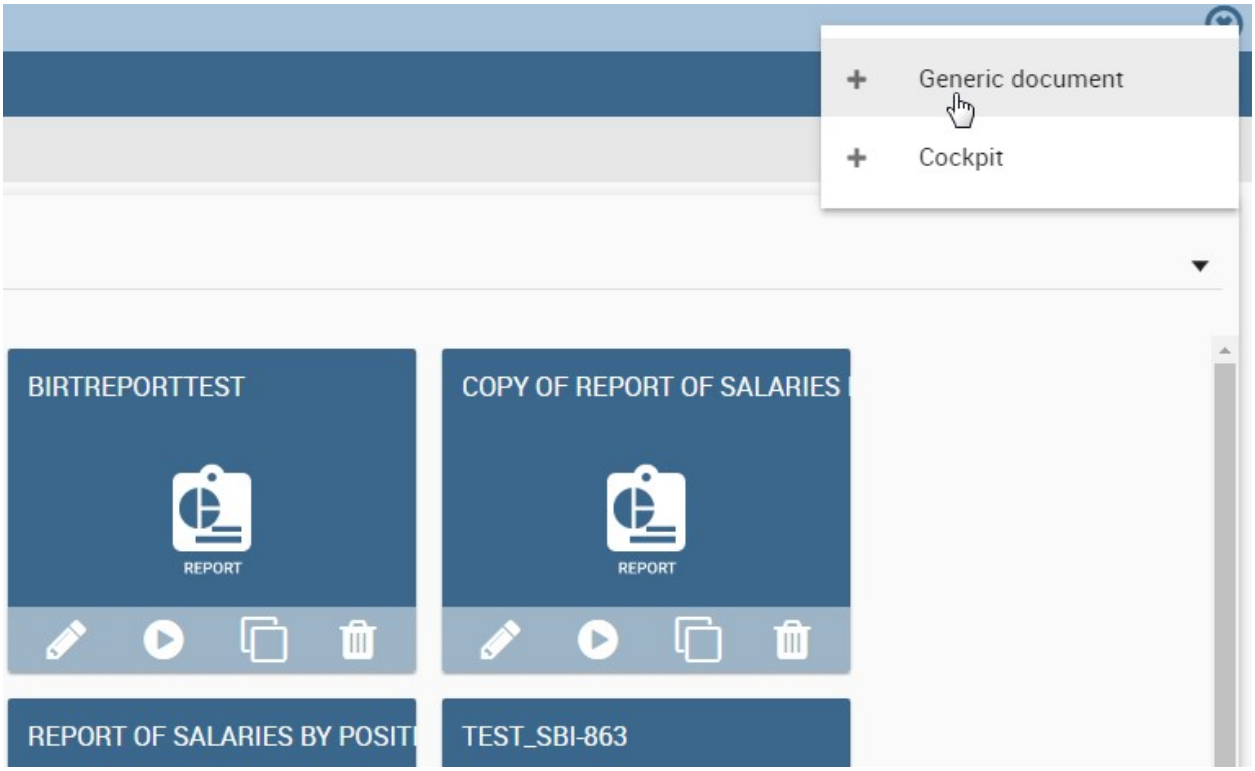


Fig. 5.100: Create a new document.

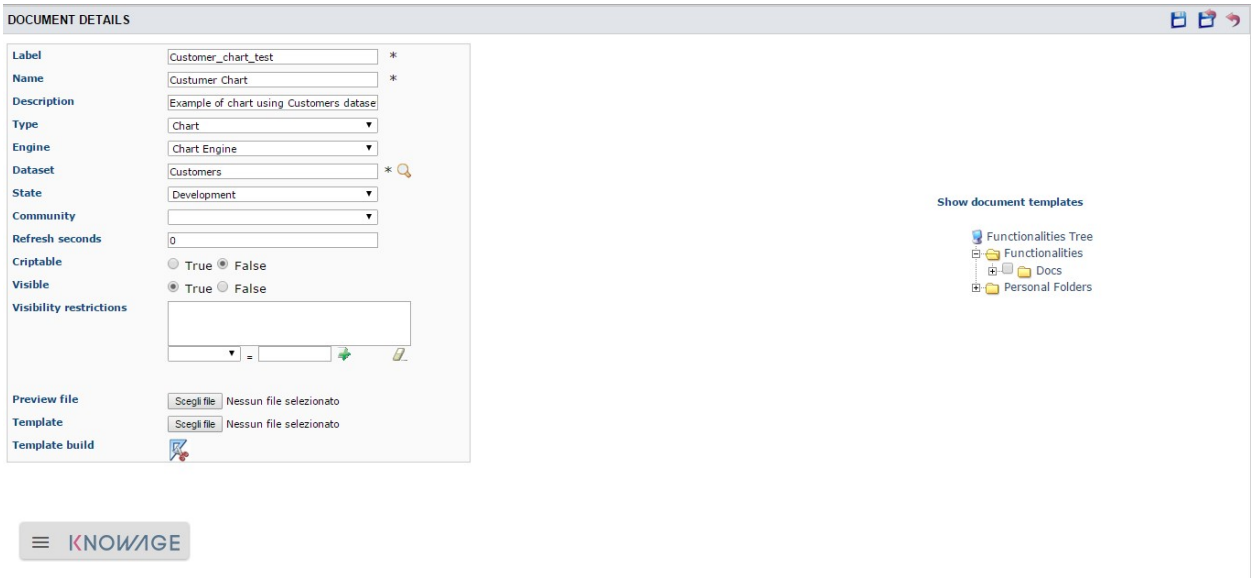


Fig. 5.101: Document Details.

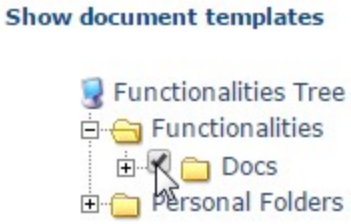


Fig. 5.102: Select the folder in which you want your chart to be saved.

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.

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.

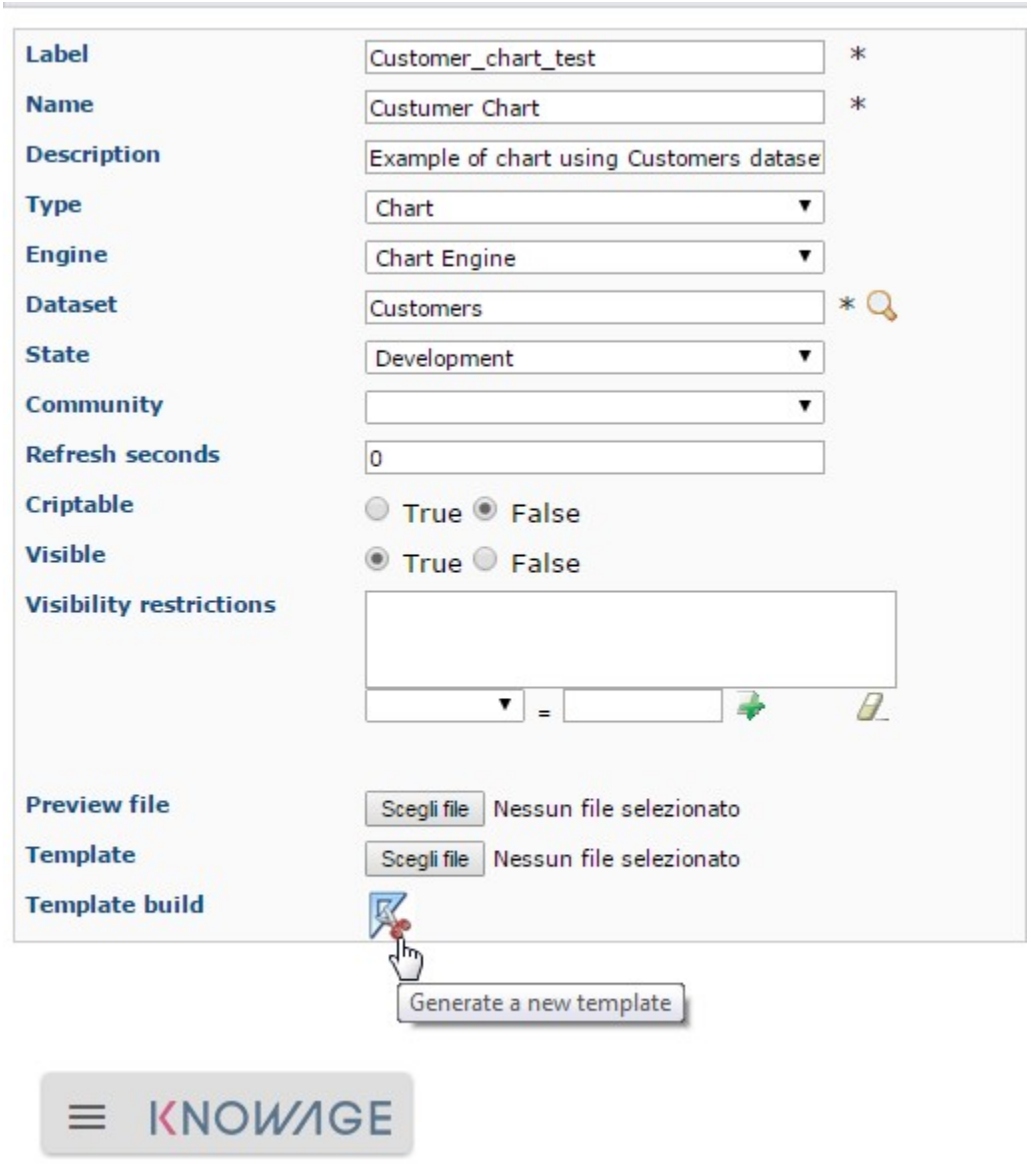


Fig. 5.103: Template build.

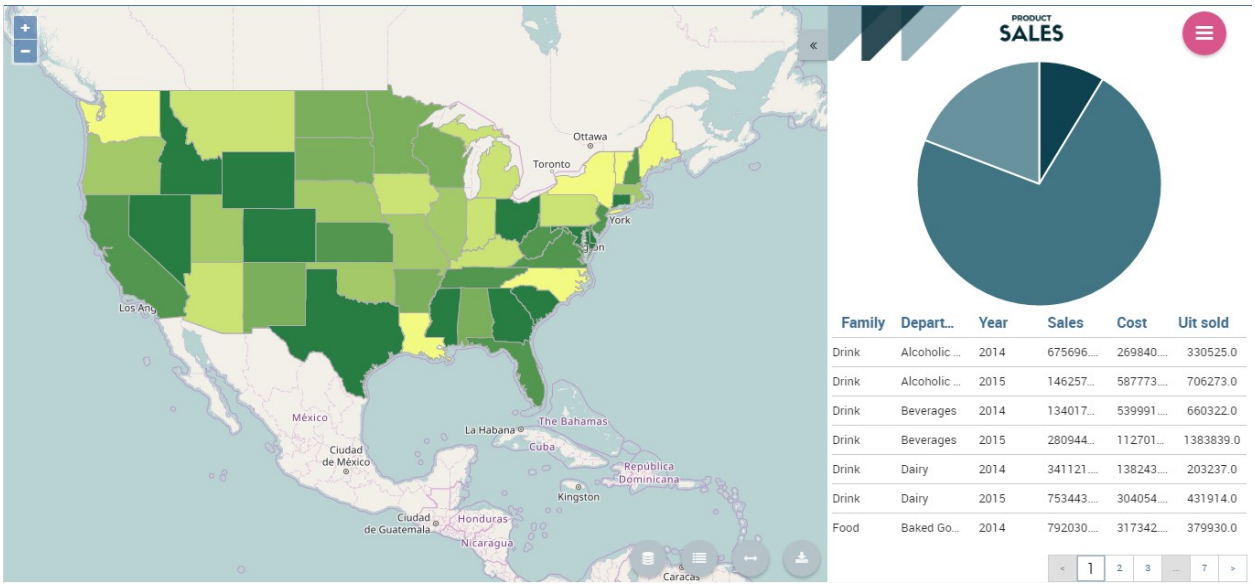


Fig. 5.104: Cockpit document example.

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.

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 possbile 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.

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 outup from the dataset. We put details off to the table widget subsection.

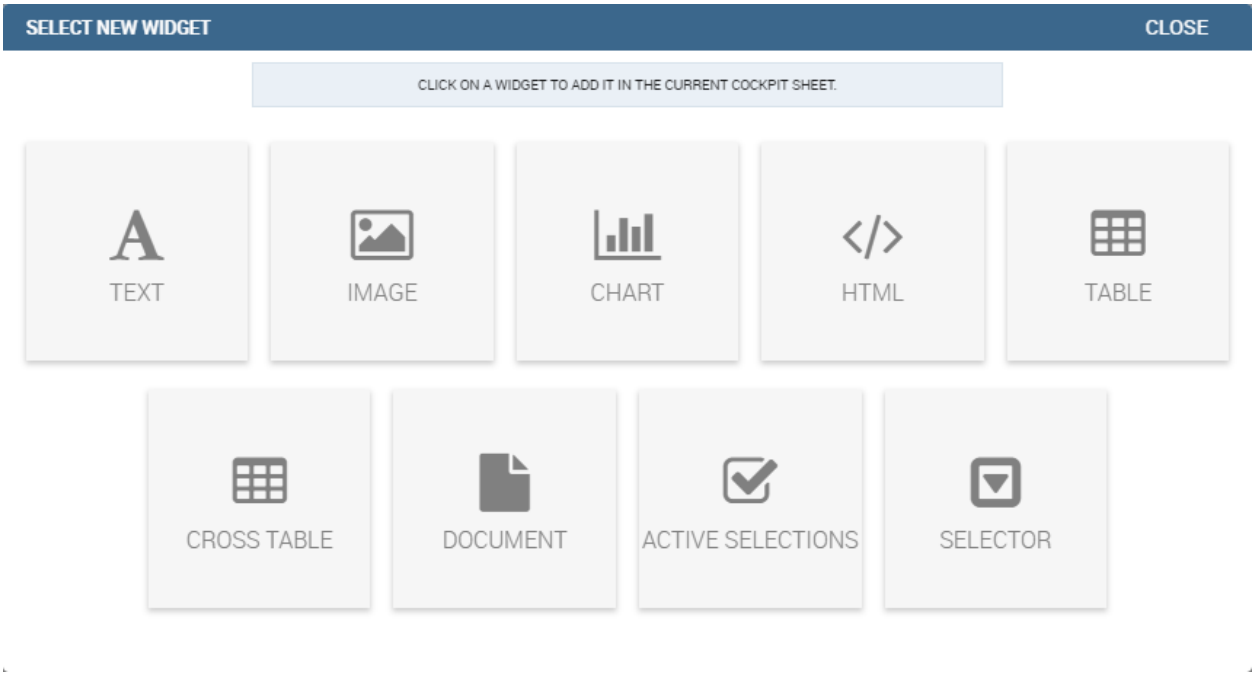


Fig. 5.105: Widget Type.

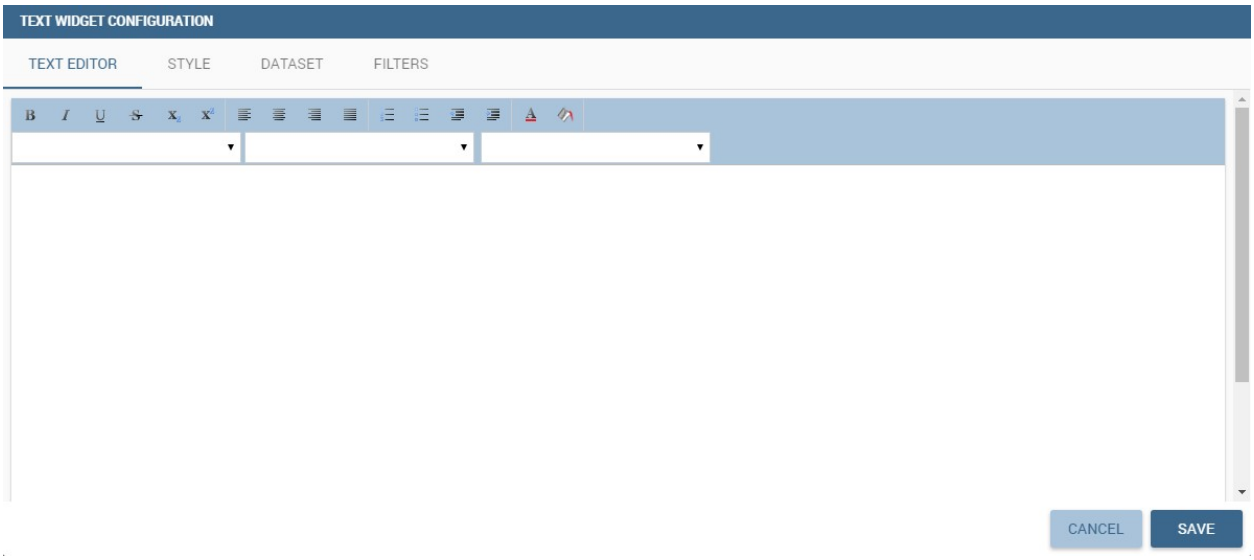


Fig. 5.106: Text editor of text widget configuration.

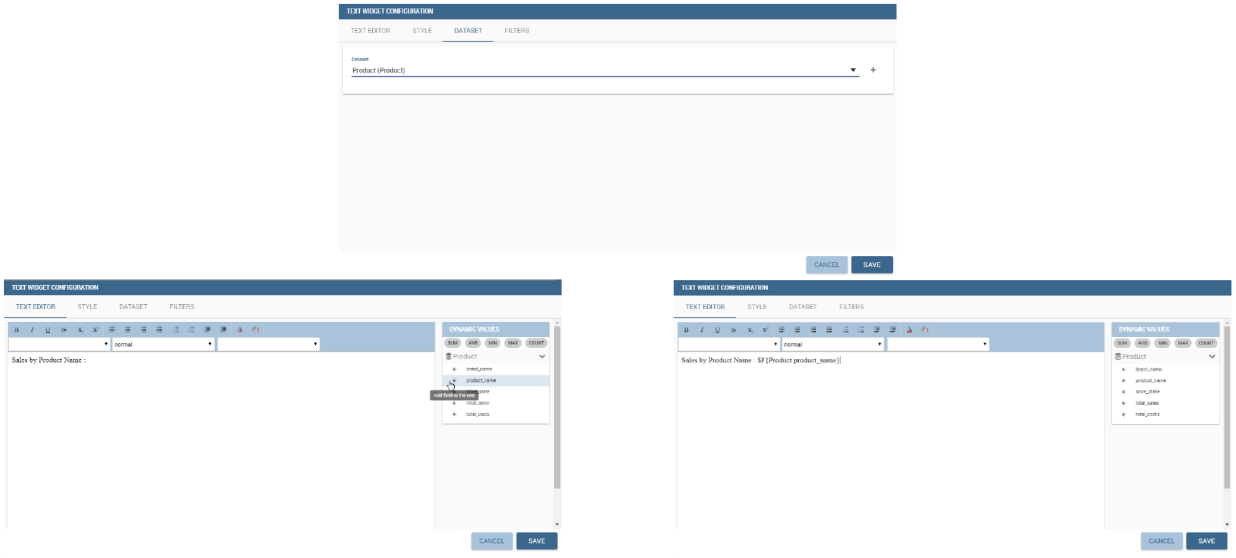


Fig. 5.107: Editing a dynamic text.

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.

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.

> **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 chart 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.

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.

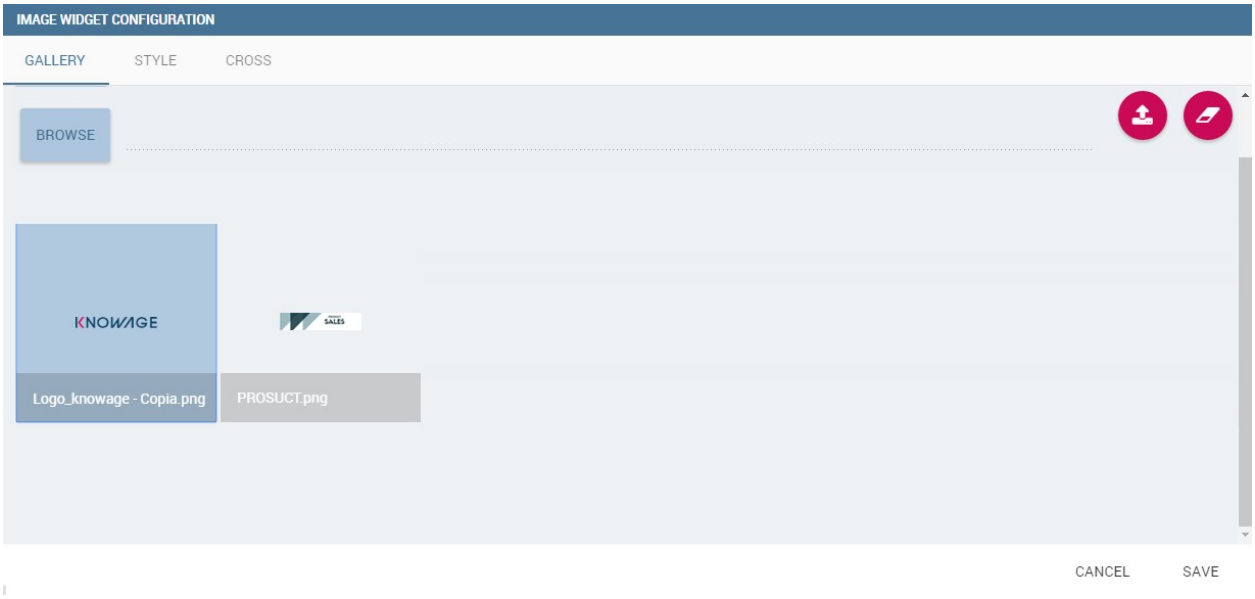


Fig. 5.108: Gallery tab of Image Widget Configuration.

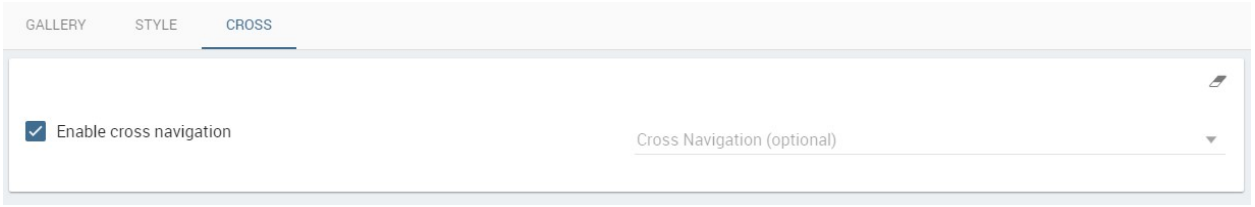


Fig. 5.109: Cross tab of Image Widget Configuration.

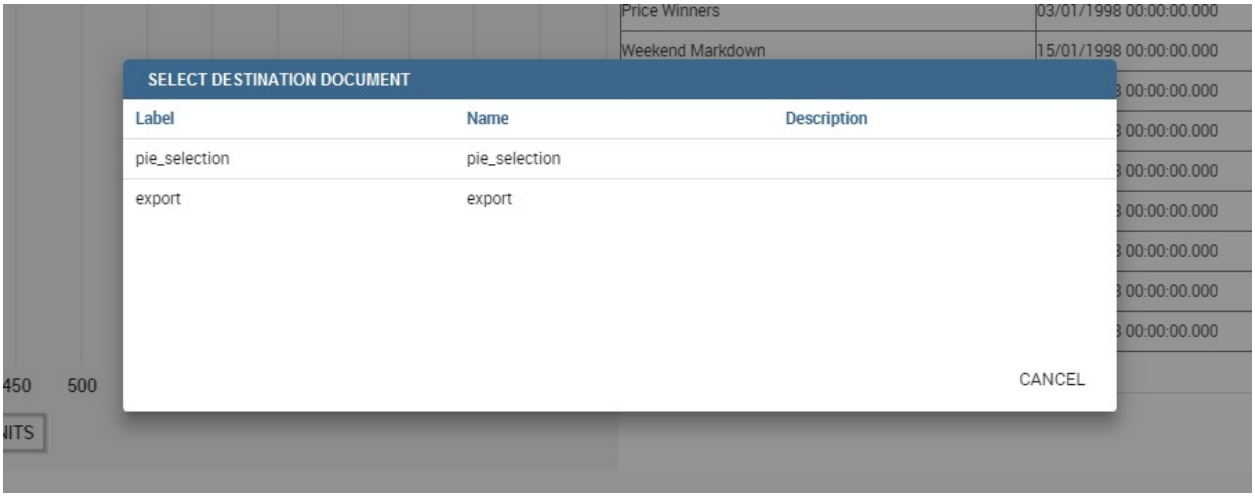


Fig. 5.110: Cross navigation multiple choices.

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 (previoulsy 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).

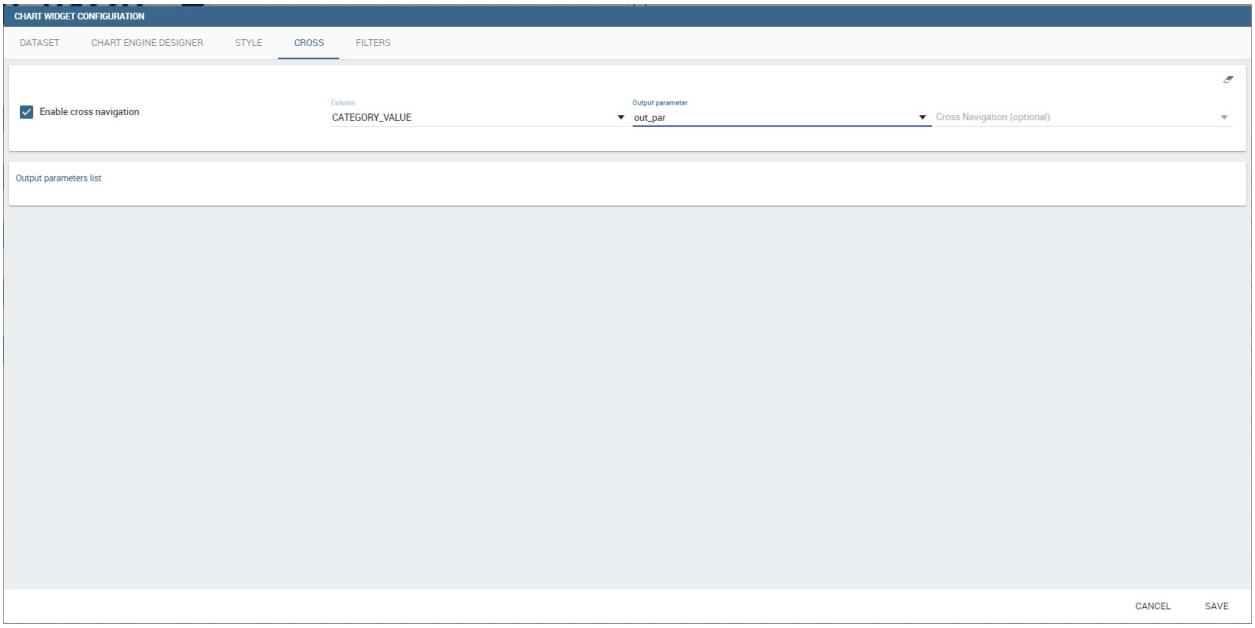


Fig. 5.111: Cross navigation for chart widget.

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.

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 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.

In fact, the column area is divided into two parts: on the left side you have columns ordering, on the right the user has the button to add a new column or a calcutated field. As soon as 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.

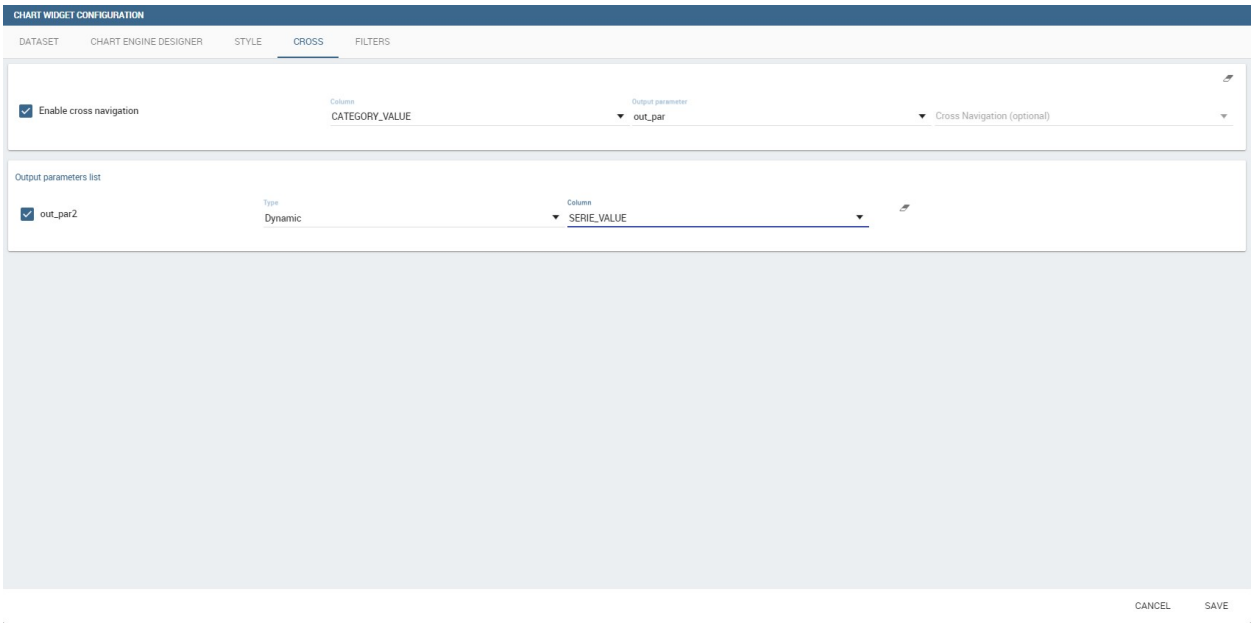


Fig. 5.112: Add all output parameters involved in the cross navigation.

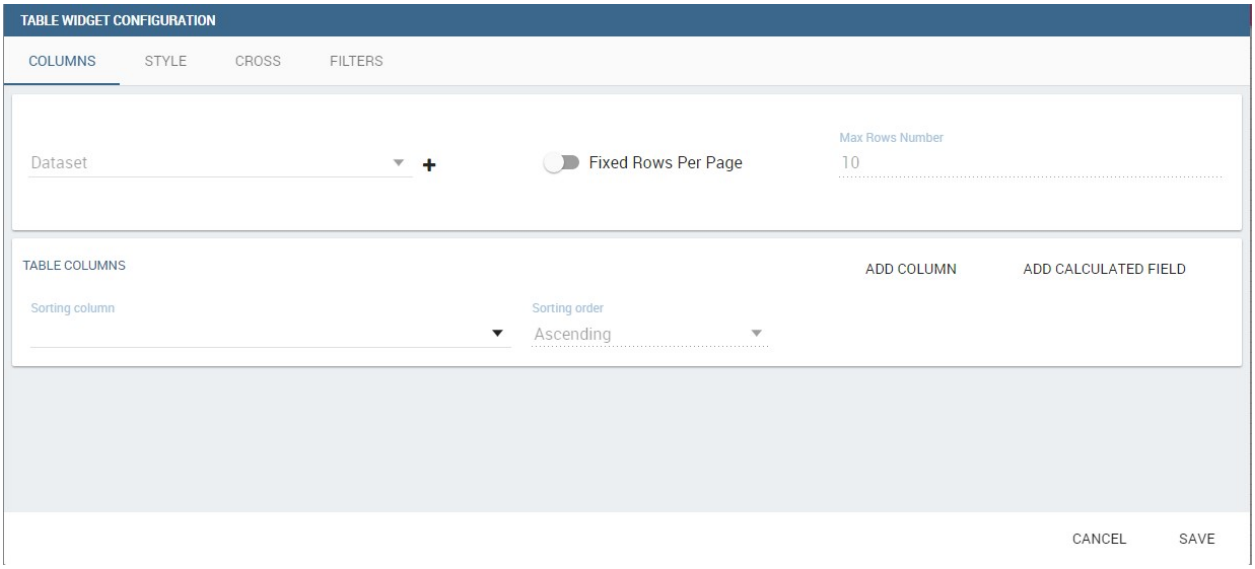


Fig. 5.113: Table configuration window.

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.

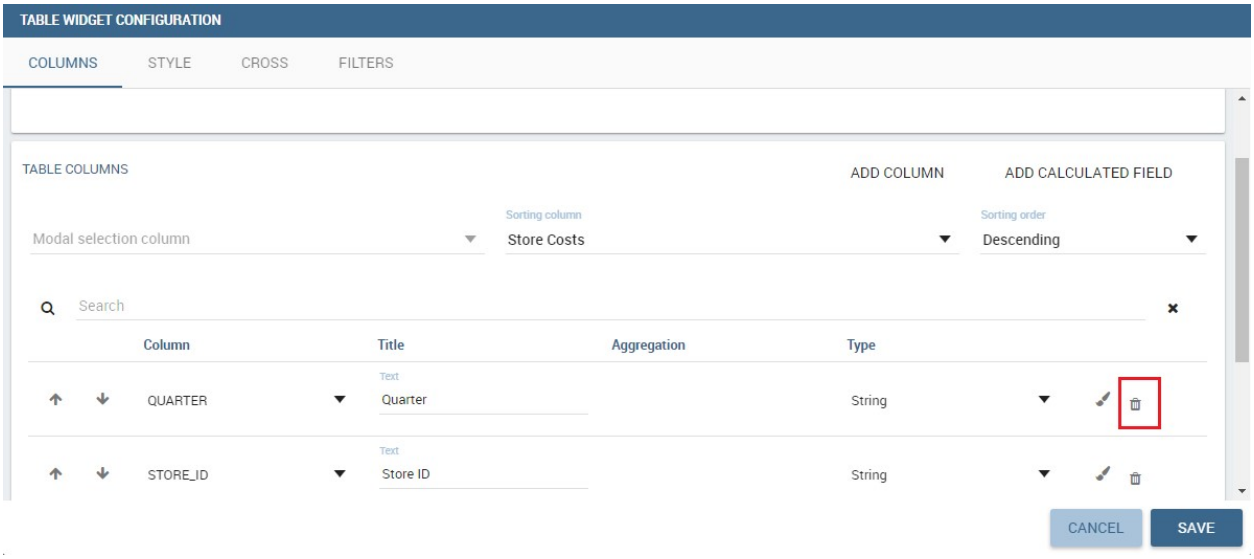


Fig. 5.114: Delete a column.

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.

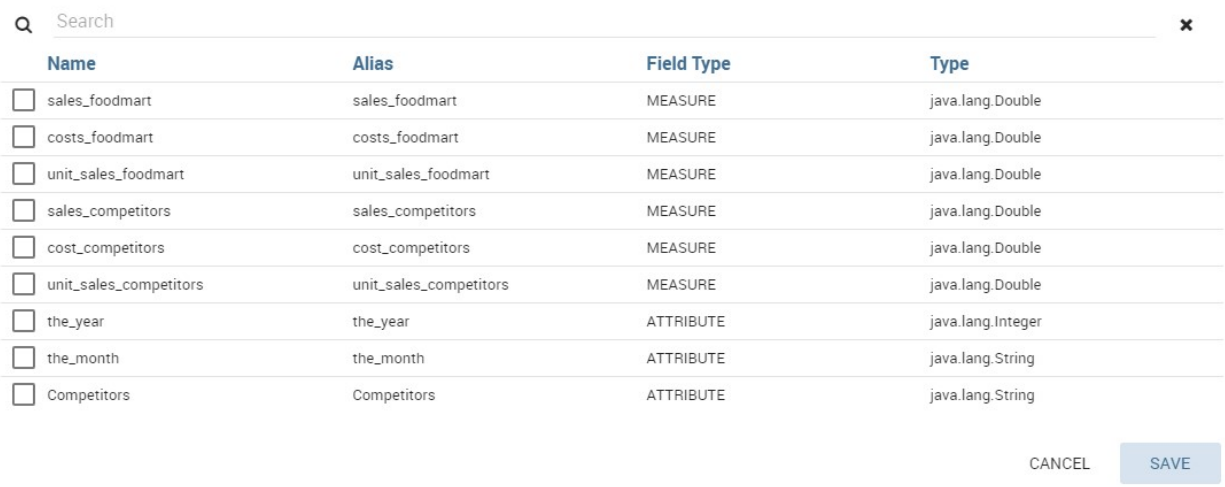


Fig. 5.115: Add a new column.

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

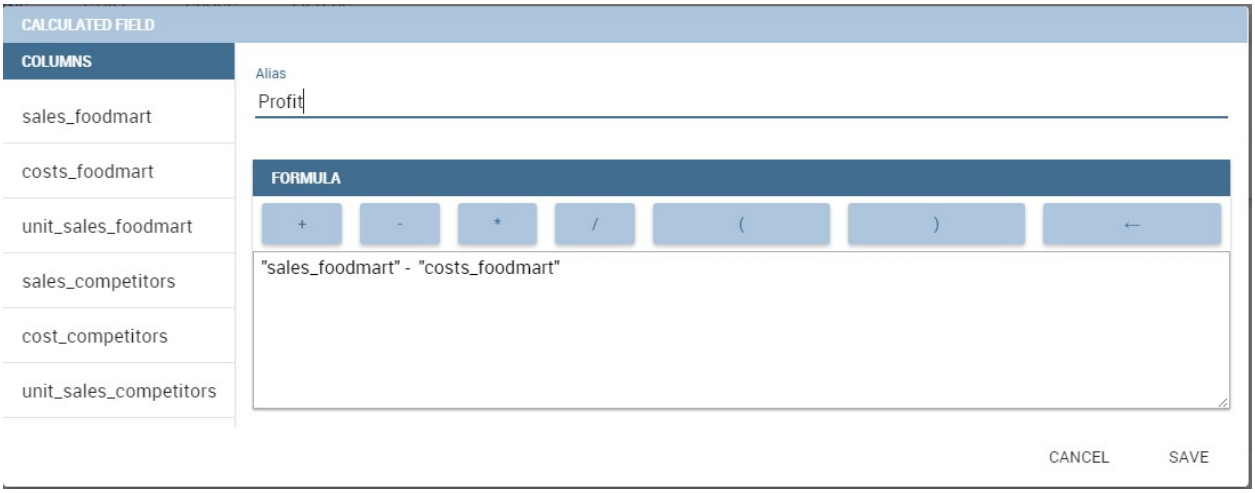


Fig. 5.116: Add a calculated field.

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

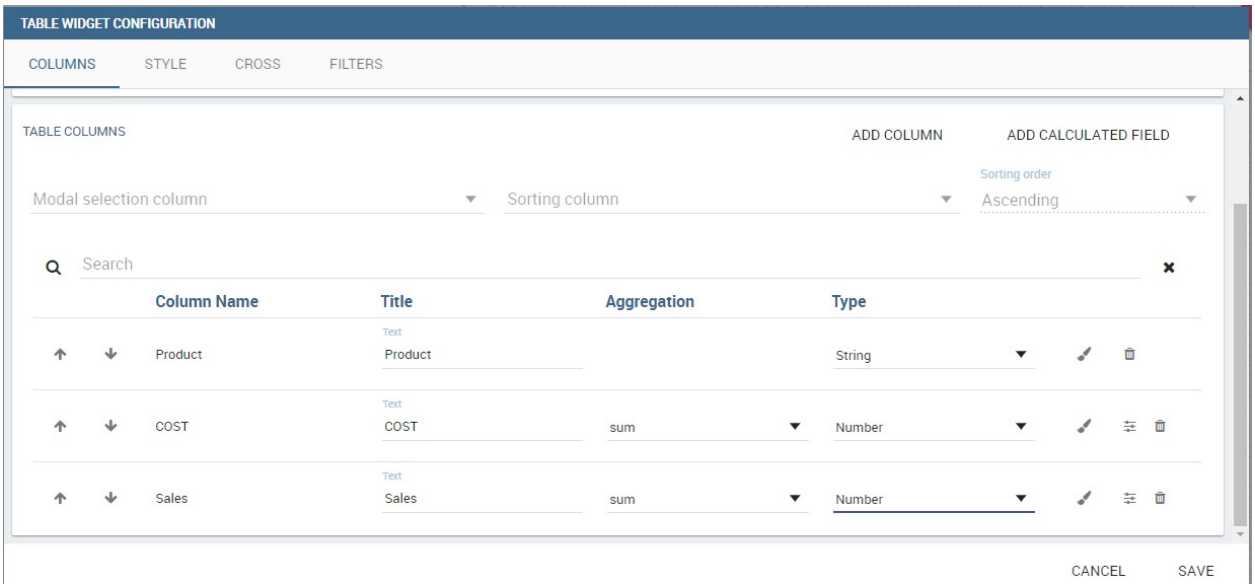


Fig. 5.117: Column settings.

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

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

- 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.
- In the **Rows** section you can set the table rows to be adapted in automatic or select a fixed height. You can also

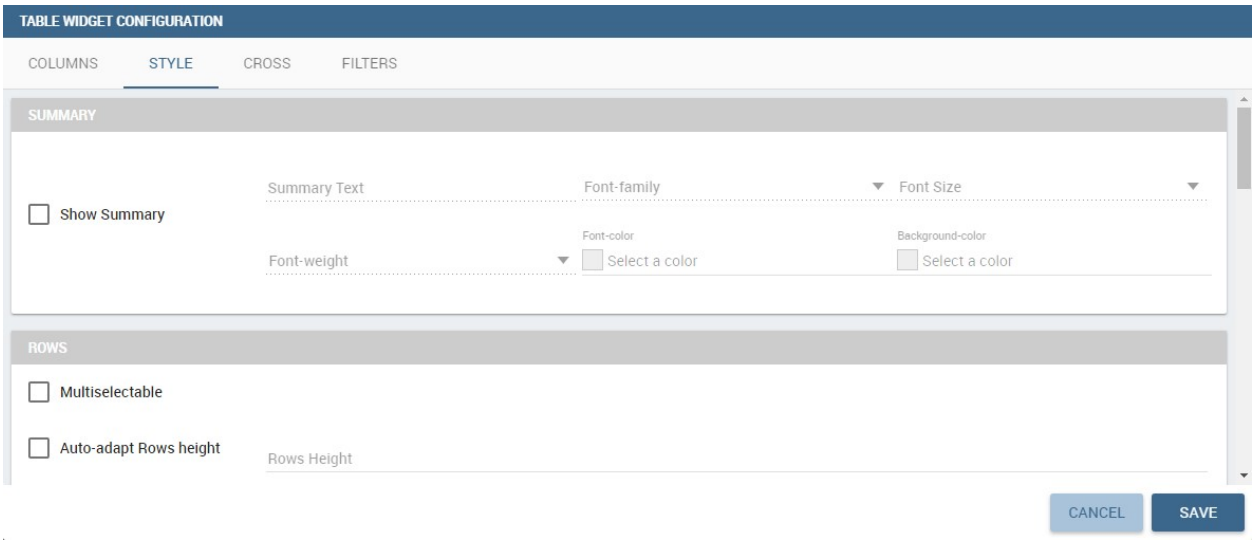


Fig. 5.118: Summary section of the Style tab.

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.

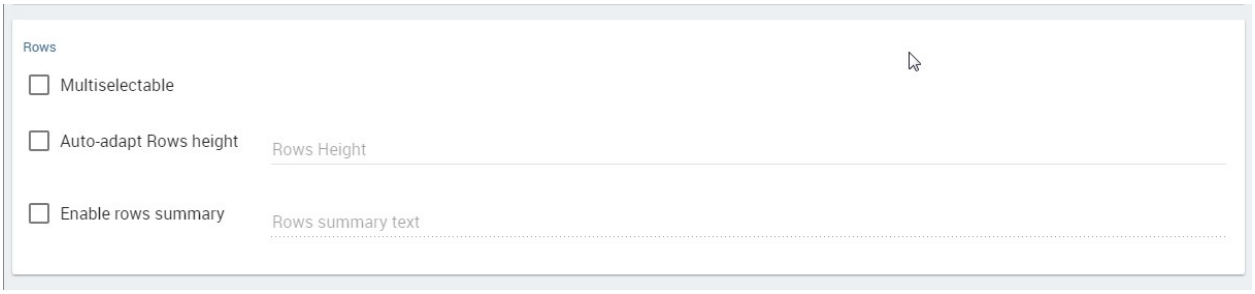


Fig. 5.119: Rows section of the Style tab.

- 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 **Shadows** section you can add the shadows in the widget. Refer to the following figure.
- In the **Background** color section you can set the background color of the widget. Refer to the Figure below.

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).

Fig. 5.120: Grid section of the Style tab.

Fig. 5.121: Header style section of the Style tab.

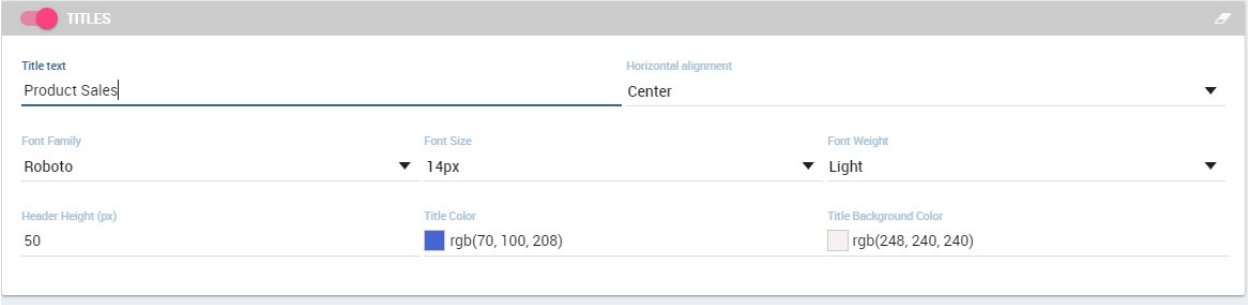


Fig. 5.122: Titles section of the Style tab.

Fig. 5.123: Borders section of the Style tab.

Fig. 5.124: Shadows section of the Style tab.

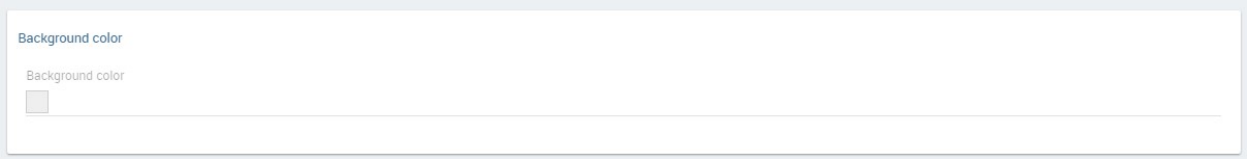


Fig. 5.125: Background color section of the Style tab.

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.

Referring to figure below, we sum up how to add a cross navigation to the cockpit with the following bullet list:

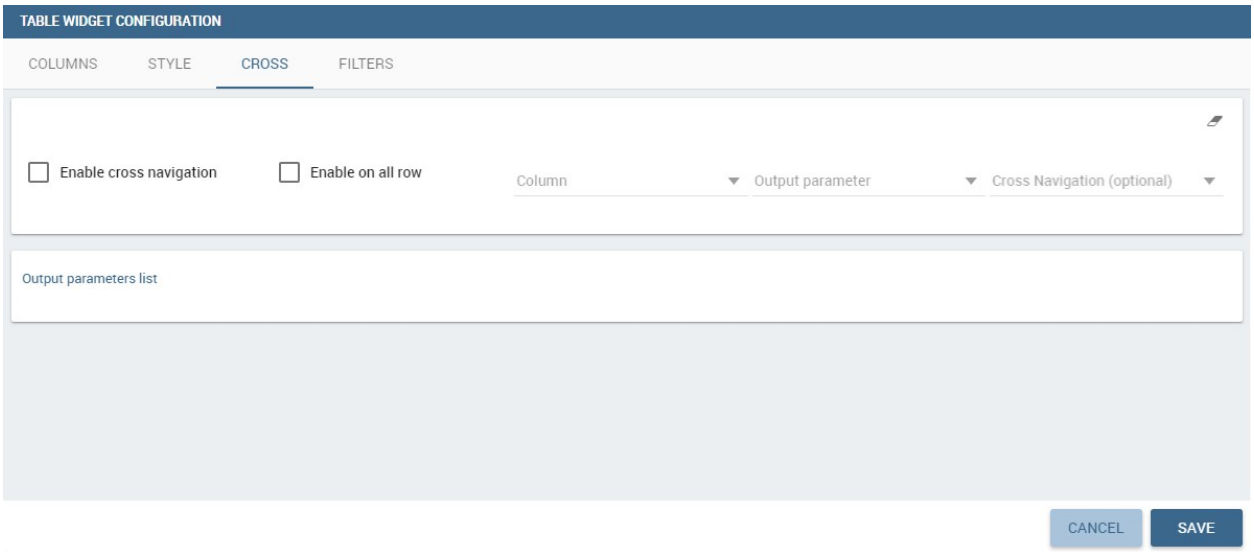


Fig. 5.126: Cross tab of the table widget configuration.

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

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.

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.

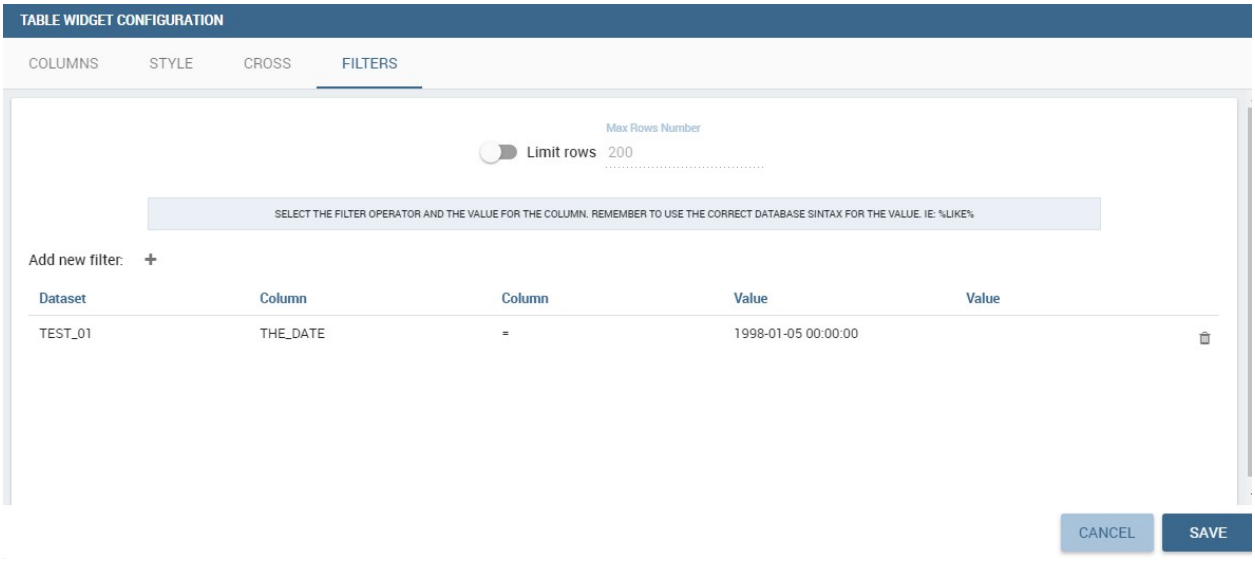


Fig. 5.127: Filters tab of the table widget configuration.

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.

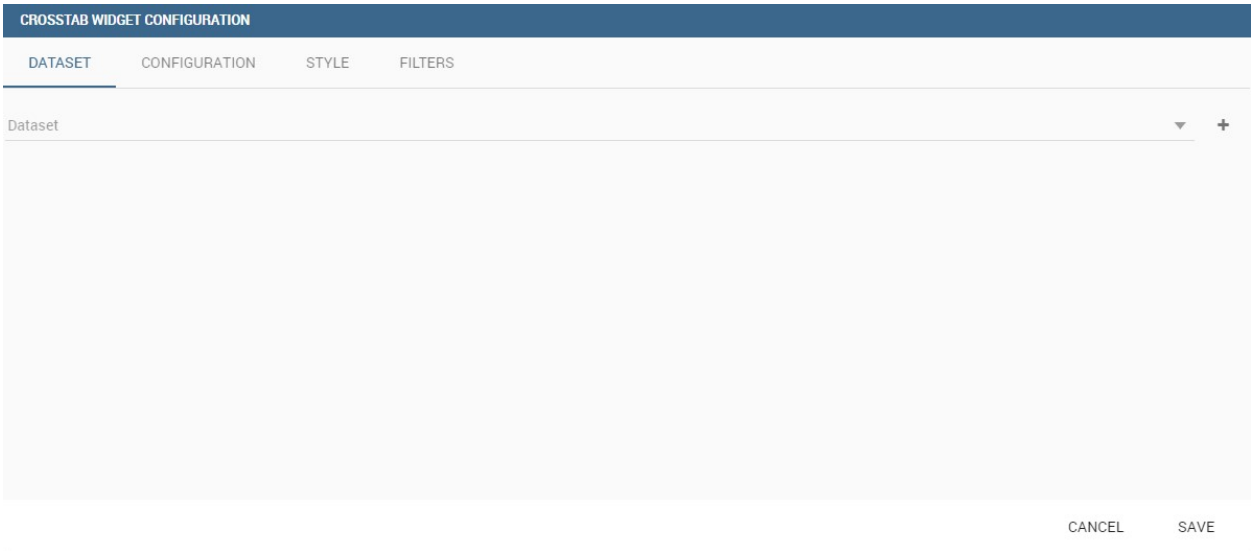


Fig. 5.128: Dataset section of the crosstab widget configuration.

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.

Here it can 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).

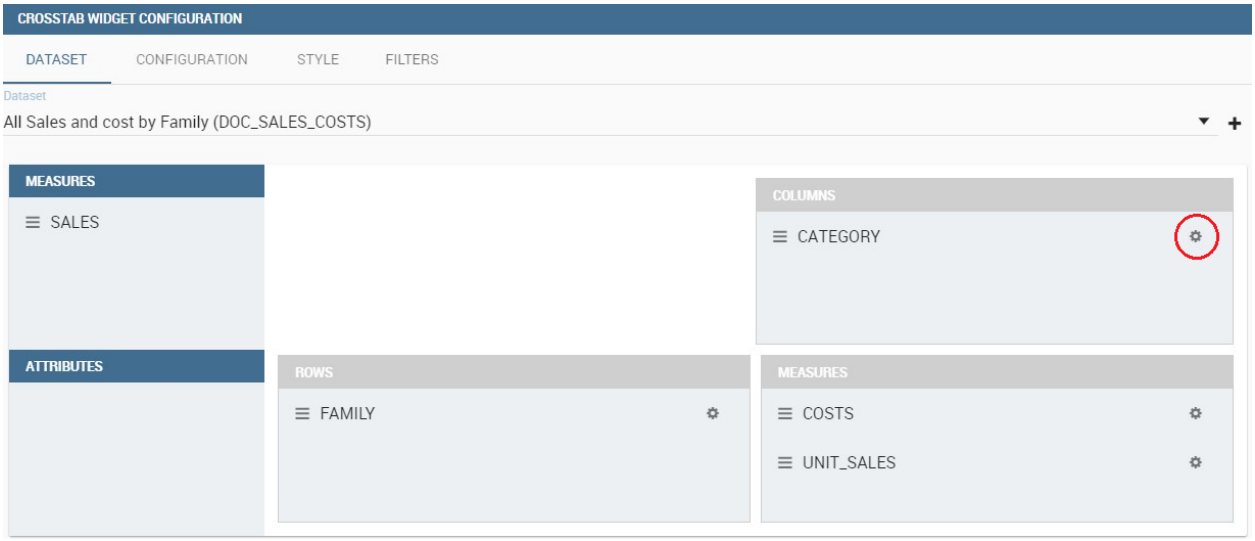


Fig. 5.129: Selecting columns, rows and measures of the crosstab.

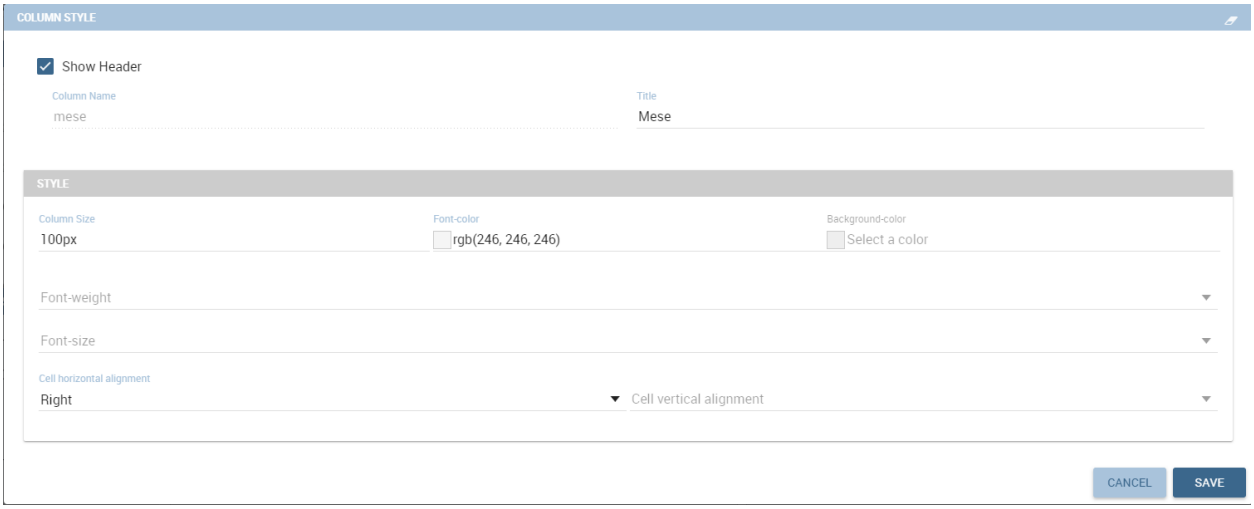


Fig. 5.130: Column style popup.

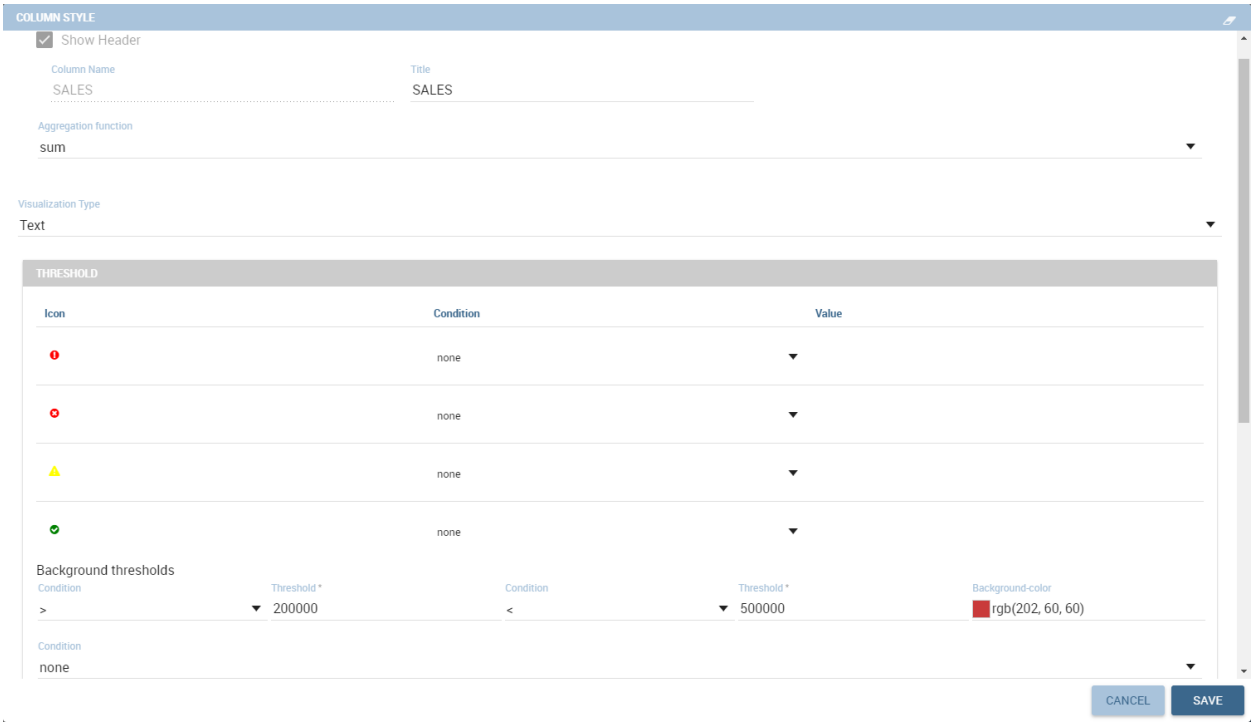


Fig. 5.131: Measure column style.

In case the selected column is of type measure, 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 Font General Options** where font and font size are set;
- **Crosstab Headers Font Options** where you can configure the header style settings as color, background, font, etc.
- **Measures Font Options** where you can configure several style options for measures, such as color, background, font size, etc.
- 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.

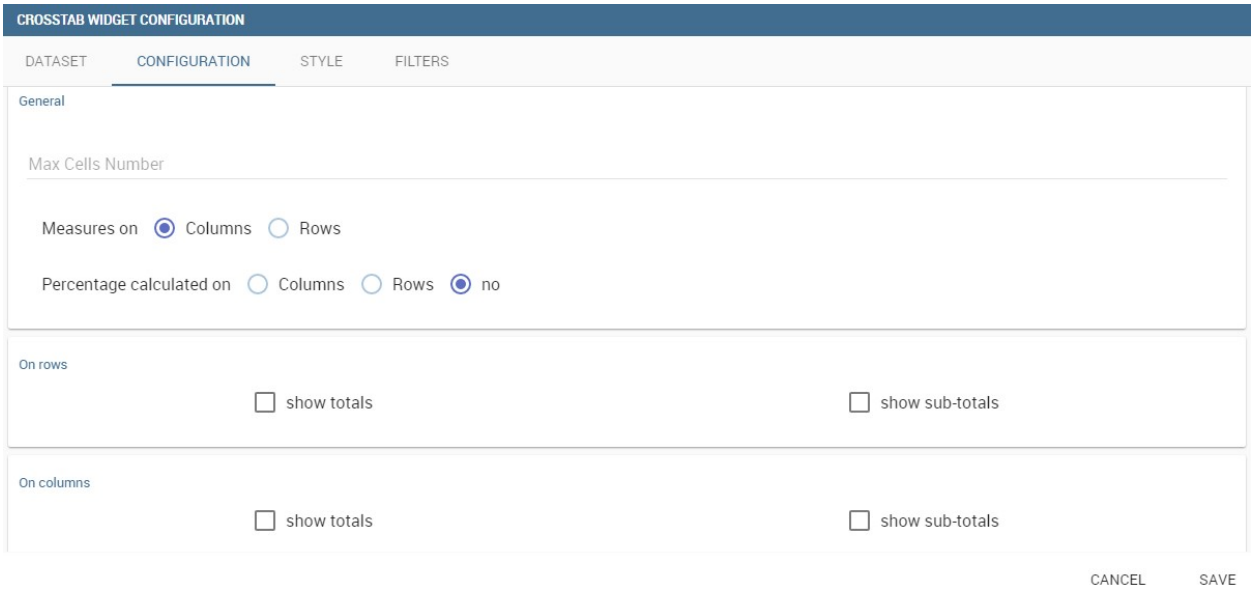


Fig. 5.132: Configuration tab interface.

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

Fig. 5.133: Computing totals and/or subtotals on rows.

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

Fig. 5.134: Computing totals and/or subtotals on columns.

Fig. 5.135: General style options for crosstab.

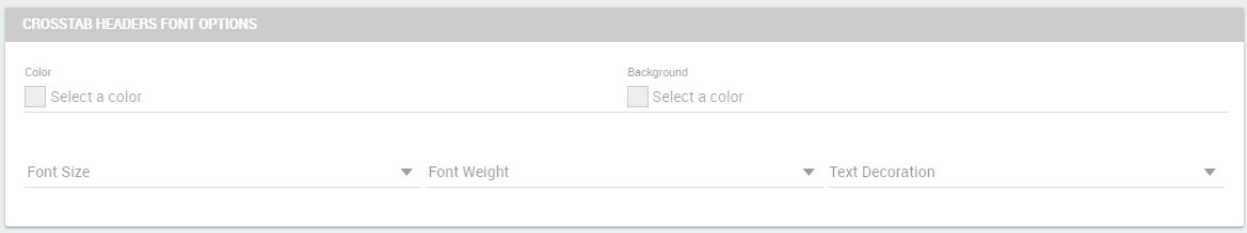


Fig. 5.136: Crosstab Headers Font Options for crosstab.

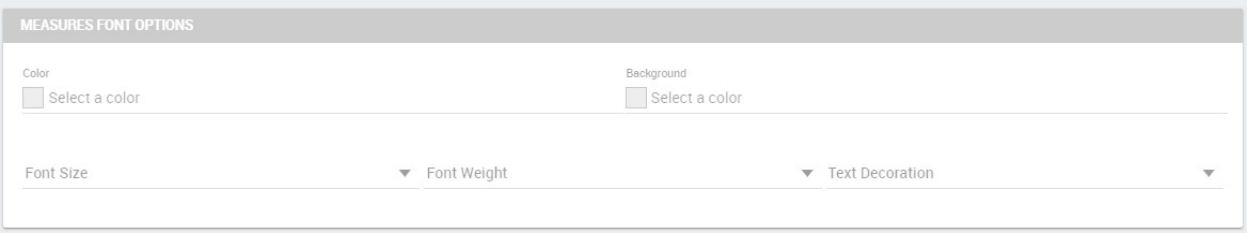


Fig. 5.137: Measures Font Options for crosstab.

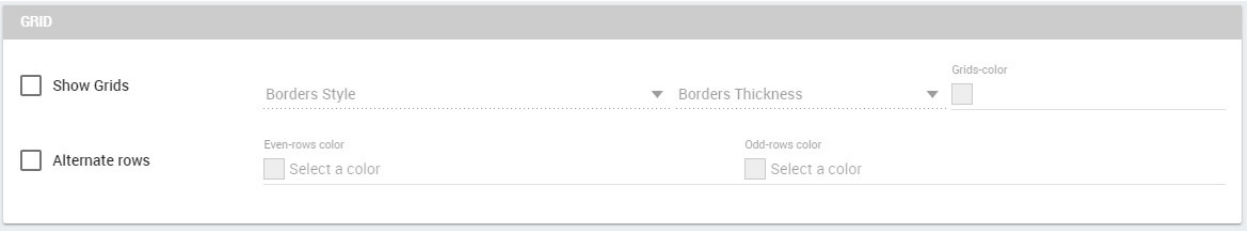


Fig. 5.138: Grid Options for crosstab.

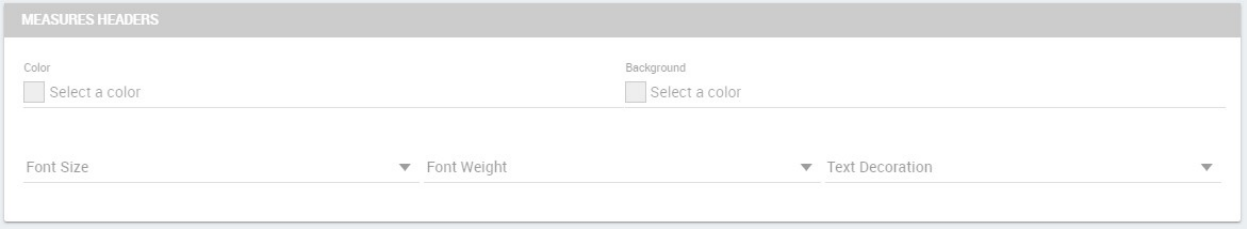


Fig. 5.139: Measures Headers Option for crosstab.

- In the **Total** section you can set color and background of totals (if any).

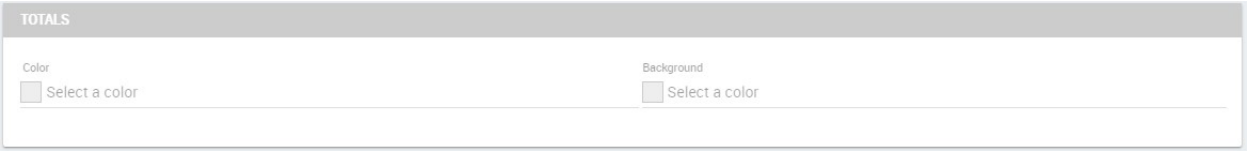


Fig. 5.140: Color settings for Totals.

- In the **Subtotal** section you can set color and background of subtotals (if any).

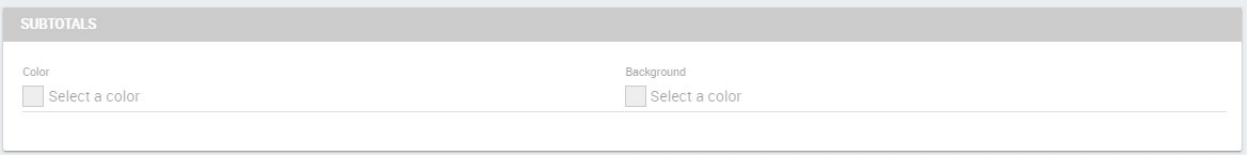


Fig. 5.141: Color settings for Subtotals.

- In the **Titles** section you can add titles to widget and customize them using different styles.

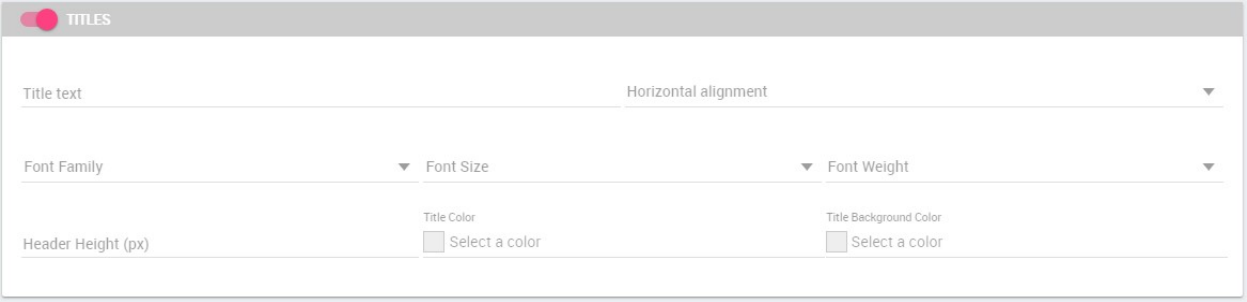


Fig. 5.142: Title settings.

- In the **Borders** section you can add borders to widgets and customize them using different styles.
- In the **Shadows** section you can add a shadow to widget layout and indicate its measure.
- In the **Background color** section you can color the widget background at convenience.

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.

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 souce 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.

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

Fig. 5.143: Border settings.

Fig. 5.144: Shadow settings.

Fig. 5.145: Background color.

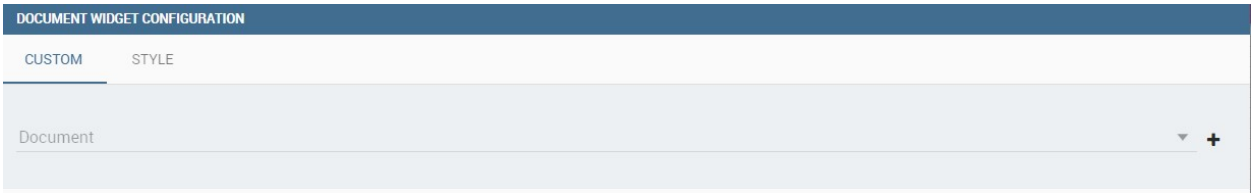


Fig. 5.146: Custom tab of the Document widget.

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 acccessible 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.

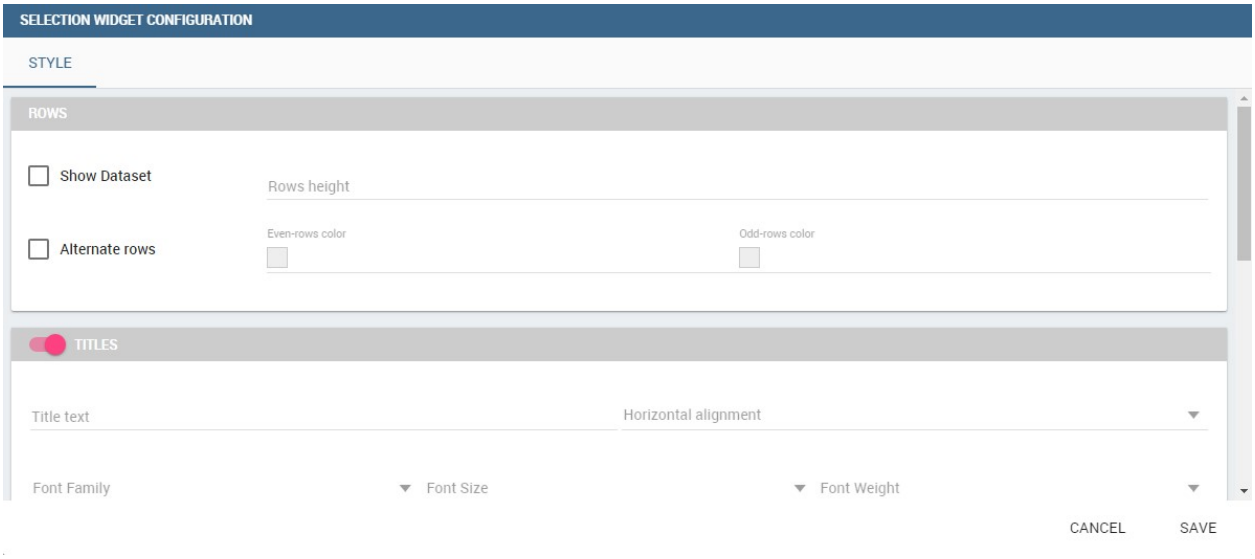


Fig. 5.147: Selection widget configuration.

The Selection Widget will display the elements selected by the user. Figure below shows an example.

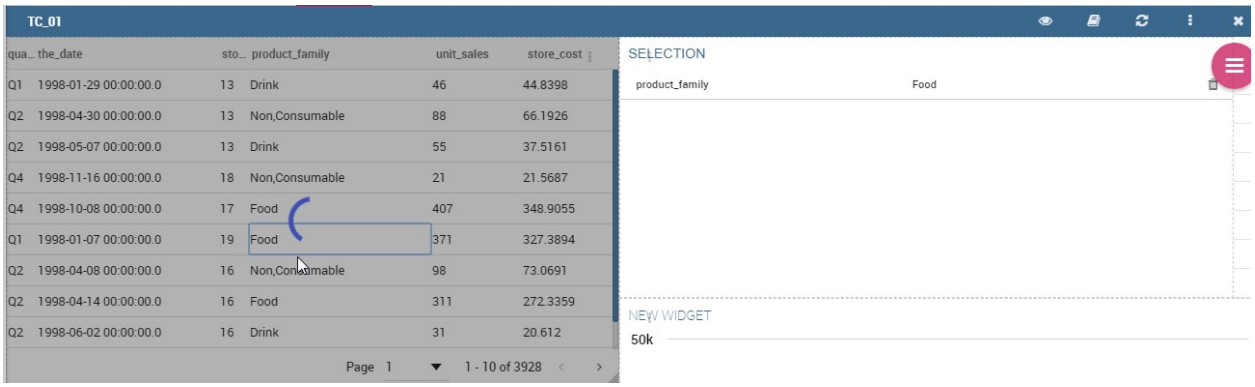


Fig. 5.148: Selection widget outlook.

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 tole 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.

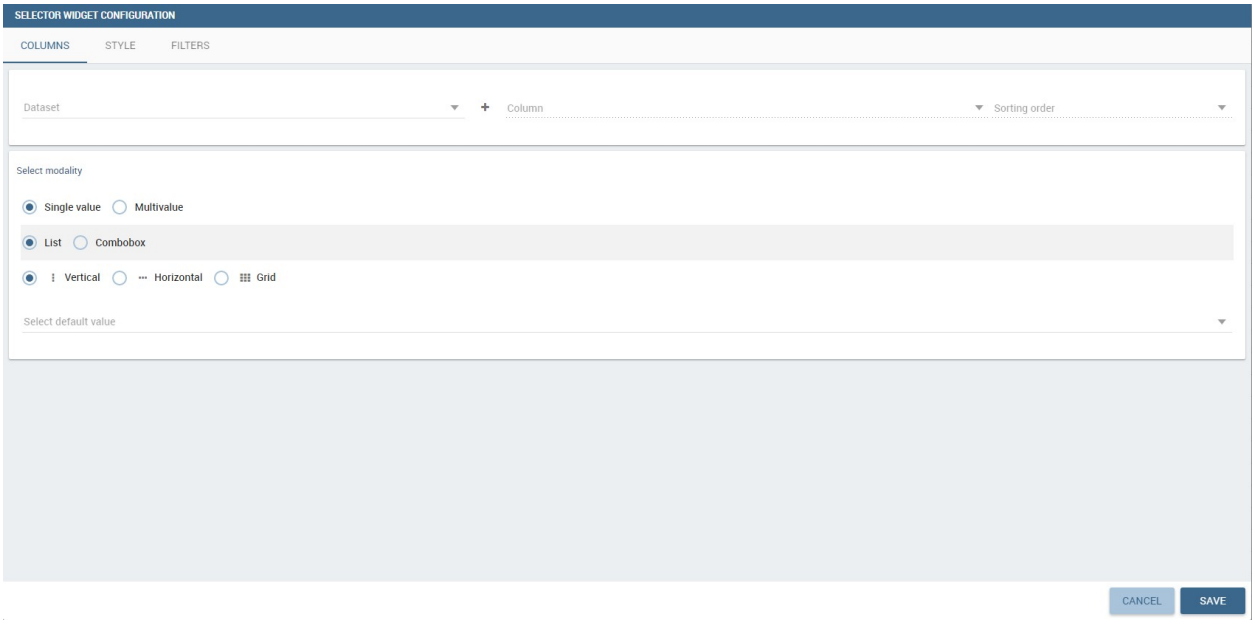


Fig. 5.149: Selector widget outlook.

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 Selector Widget choose among "vertical", "horizontal" or "grid". Finally, you can decide to add a dafault value, chosen from main column's first item, main column's last item or to simply assign a static value.

Move to the **Style** tab to set the widget style in terms of: titles, borders, shadows and background color. Figure below shows a customization example.

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.

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.

The Edit section of the widget is composed by tree tabs: the HTML editor, the style and the dataset. 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.

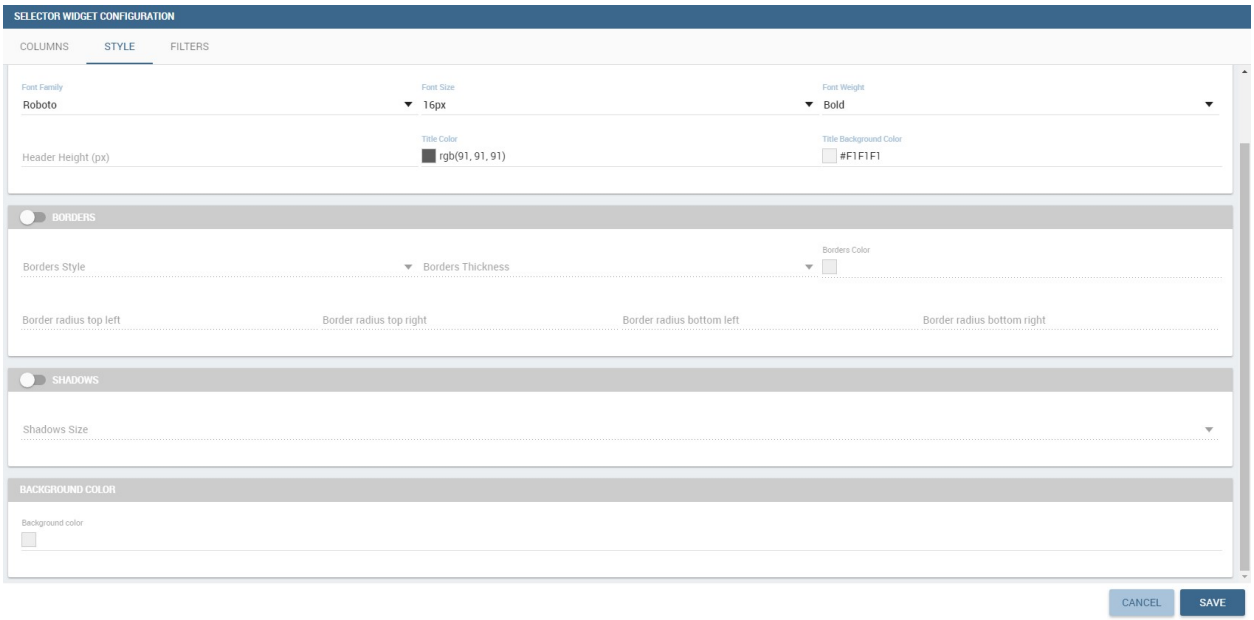


Fig. 5.150: Selector widget configuration.

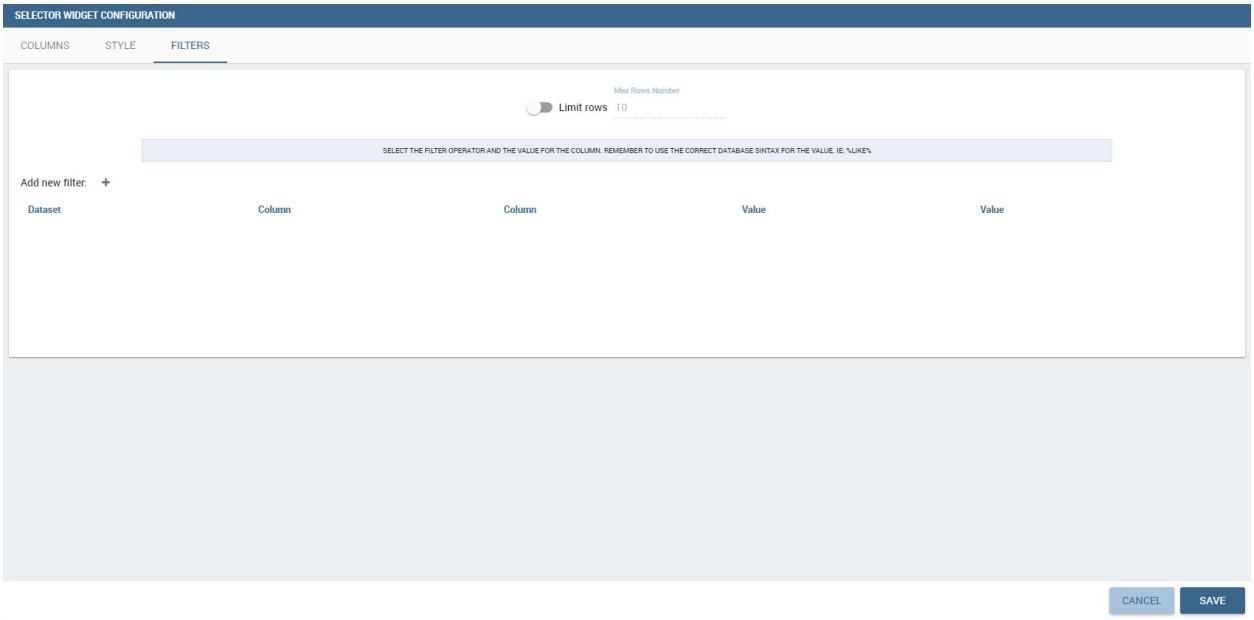


Fig. 5.151: Selector filters.

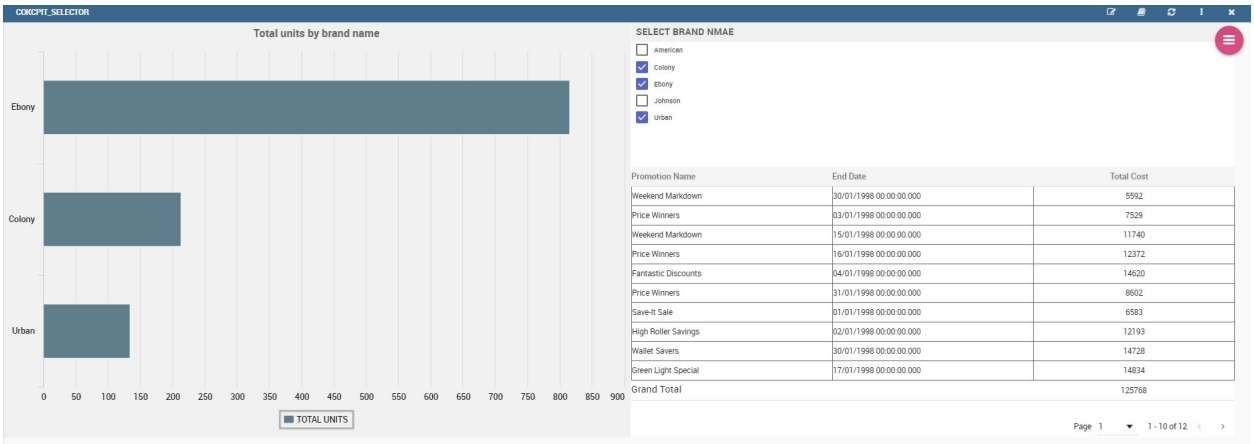


Fig. 5.152: Selector widget execution example.

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

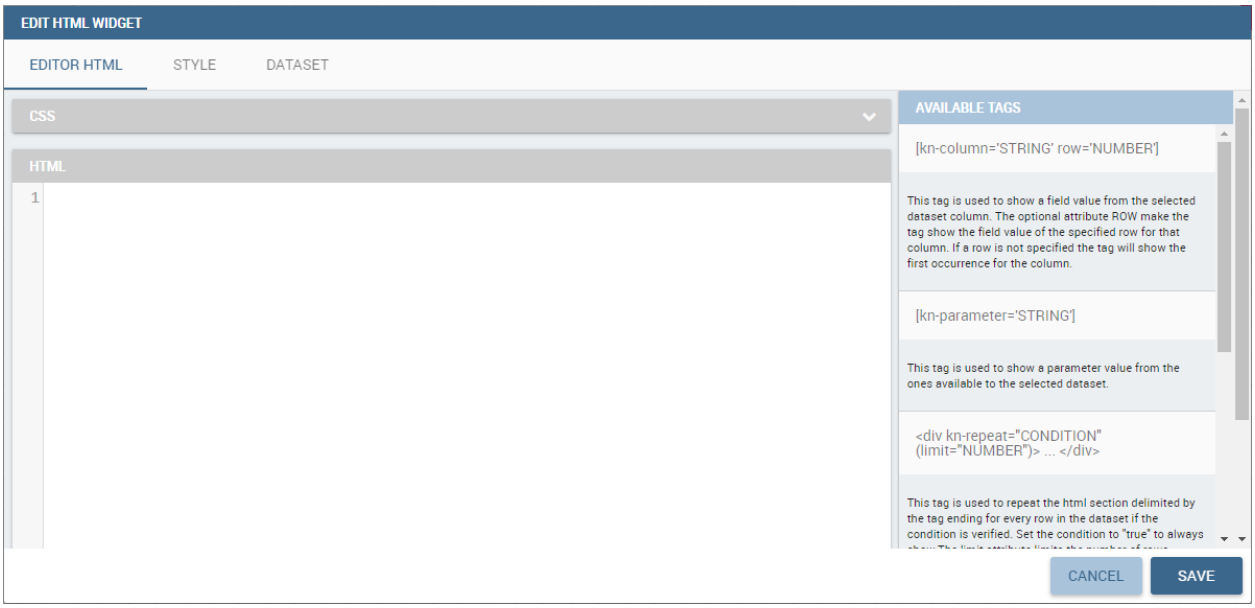


Fig. 5.153: HTML widget editor

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.

Available Tags

```
[kn-column='COLUMN-NAME' row='COLUMN-ROW-NUMBER' aggregation='COLUMN-AGGREGATION'
precision='COLUMN-DECIMALS']
```

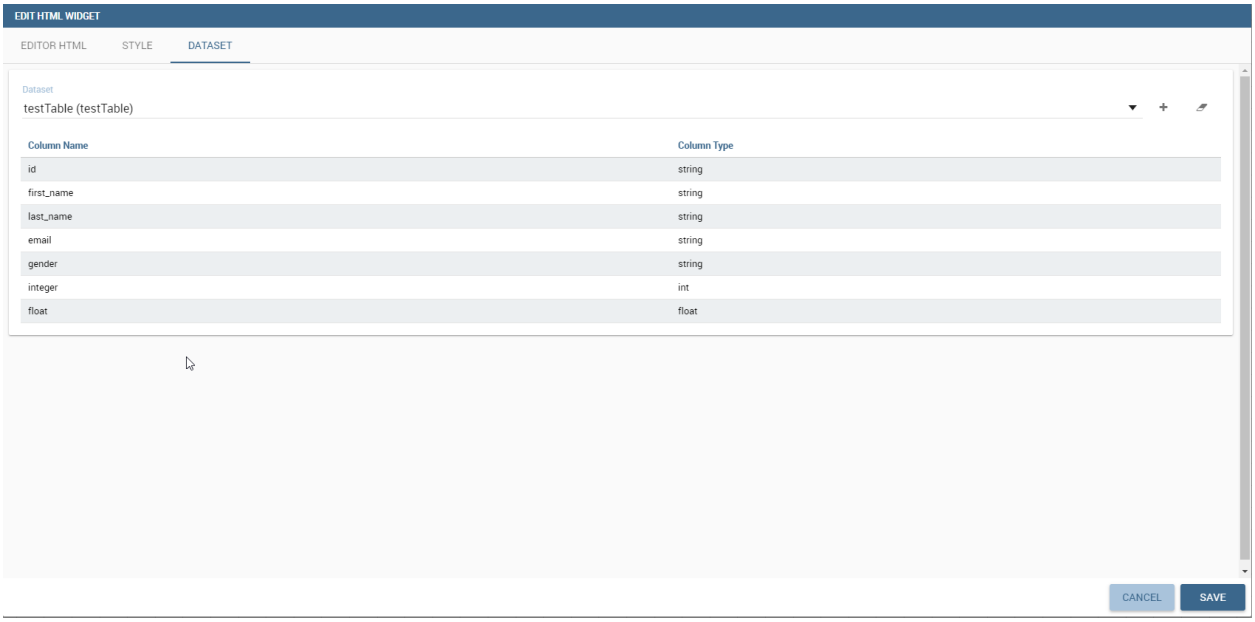


Fig. 5.154: Dataset selection

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.

Banned Tags

For Cross side scripting and security reasons some tags are removed on save by the security filter:

- `<button></button>`
- `<object></object>`
- `<script></script>`

If the tag is needed for some specific behaviour (ie. the button default hover), please replicate it with css using a different allowed tag.

Warning: Whitelist All external resources paths must be present inside a whitelist XML inside the resources folder of the server named `services-whitelist.xml`. The file should have the following structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<WHITELIST>
   <service baseurl="https://www.youtube.com" />
   <service relativepath="/knowage/themes/" />
</WHITELIST>
```

5.6.1.10 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.

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.

Chart widget are endowed with an additional option that allows the user to change the chart type, as you can see in Figure below.

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 chaned with a single-series

Fig. 5.155: Widget properties.

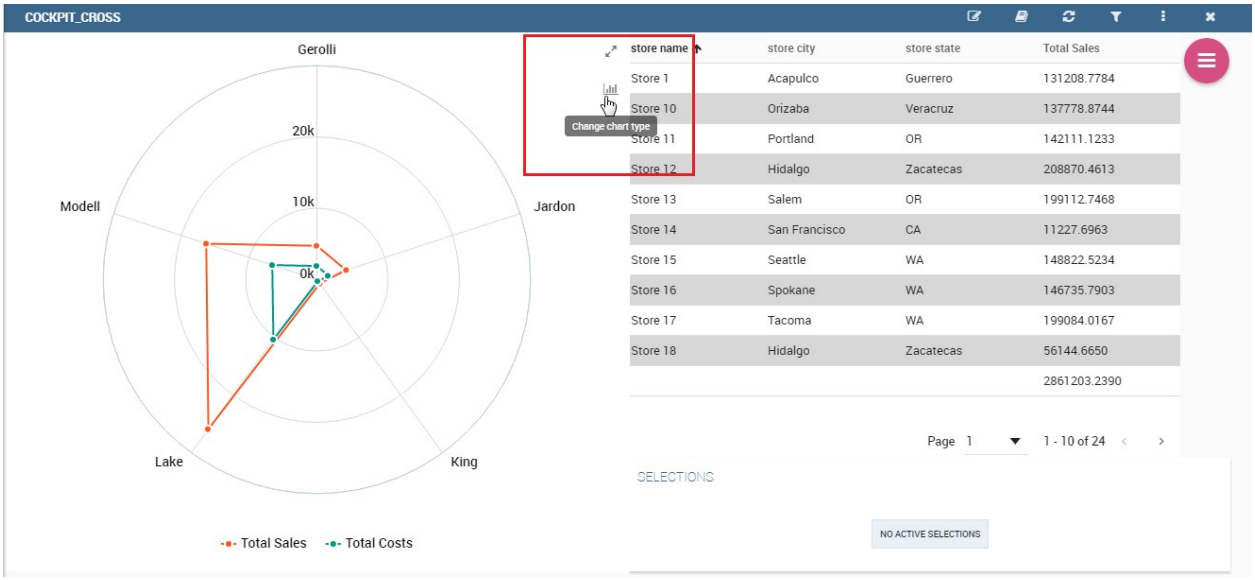


Fig. 5.156: Change chart type button.

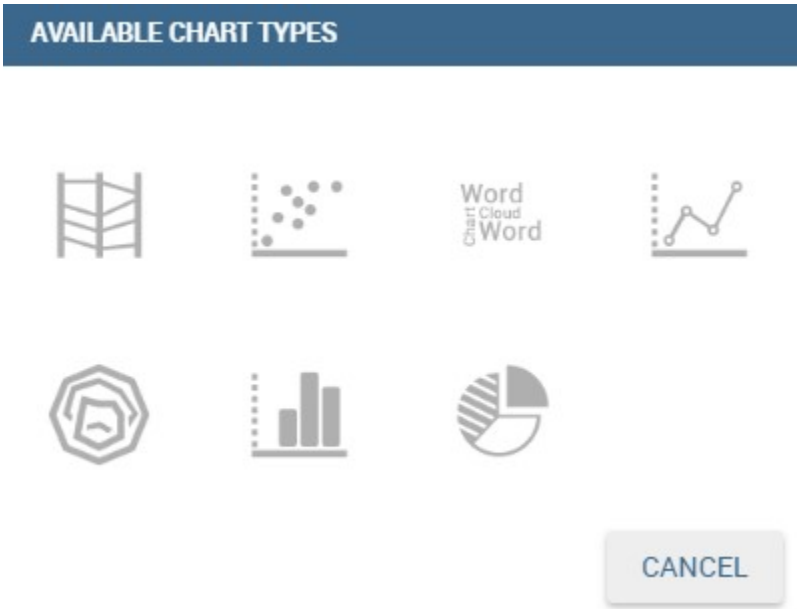


Fig. 5.157: Available chart types.

one. For instance the parallel chart works only when (at laest) 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.

Fig. 5.158: General configuration window.

Editing the fields of the first tab you can add or change the name and/or the description of your cockpit; moreover here

you can choose the sheet color and decide to enable the menu when the document runs in display mode. The second tab (Figure below allows to configure some style options of the cockpit, like borders, shadows, titles and background color.

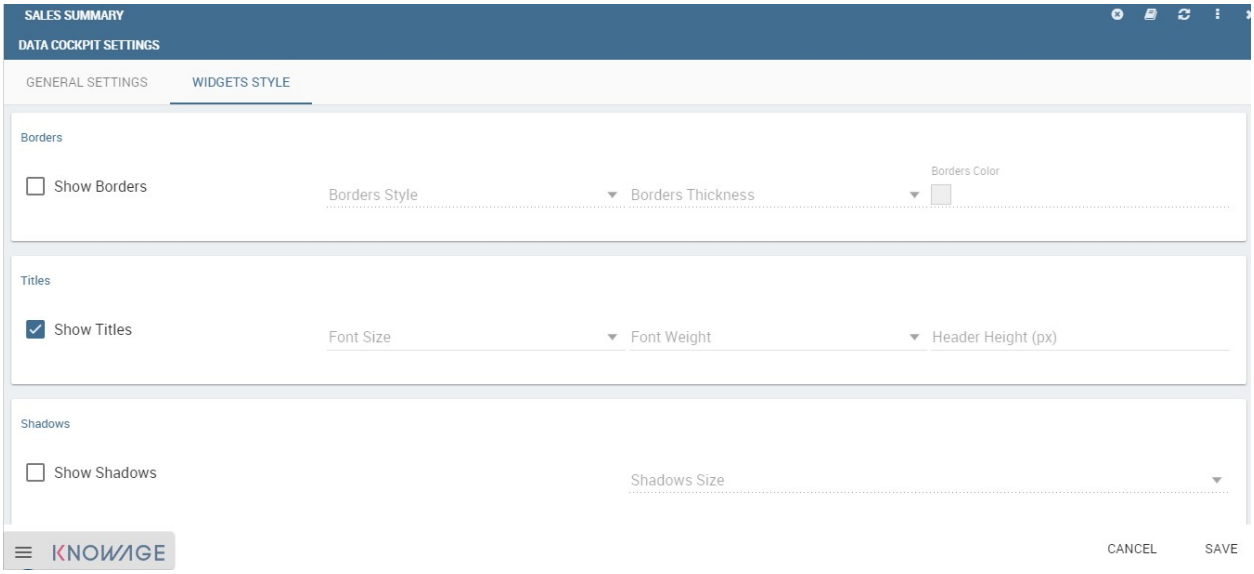


Fig. 5.159: Widget style tab.

5.6.3 Data configuration

This feature manages the data storage and usage. In fact, here there is the possiblity to save data in cache, create associations between datasets, schedule the (data) refresh frequency and so on. Referring to the figure below, the feature is implemented through several tabs: the **Source** tab, the **Associations** tab, 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 widgets. In the occurance of Associations parametric documents, parameter boxes are shown below. Note that it is mandatory to link them to analytical drivers (previuosly hooked to the document) or be assigned a fixed (default) value.

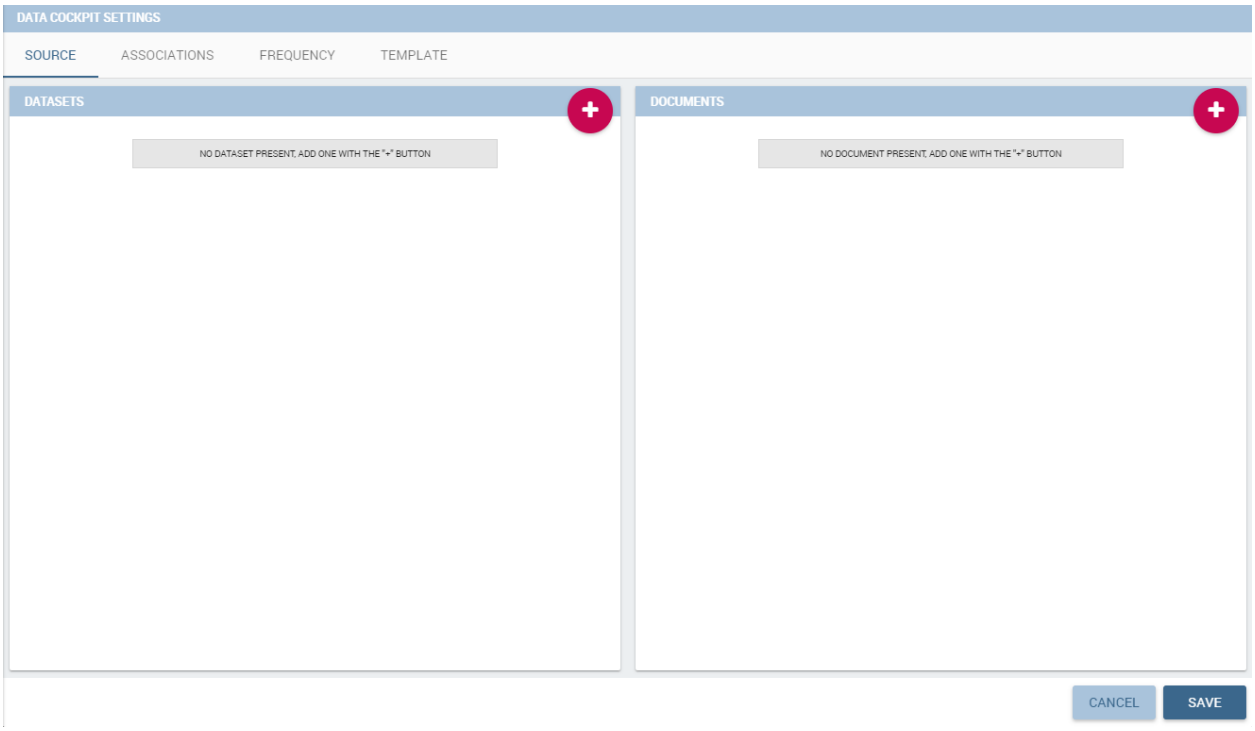


Fig. 5.160: Data configuration window.

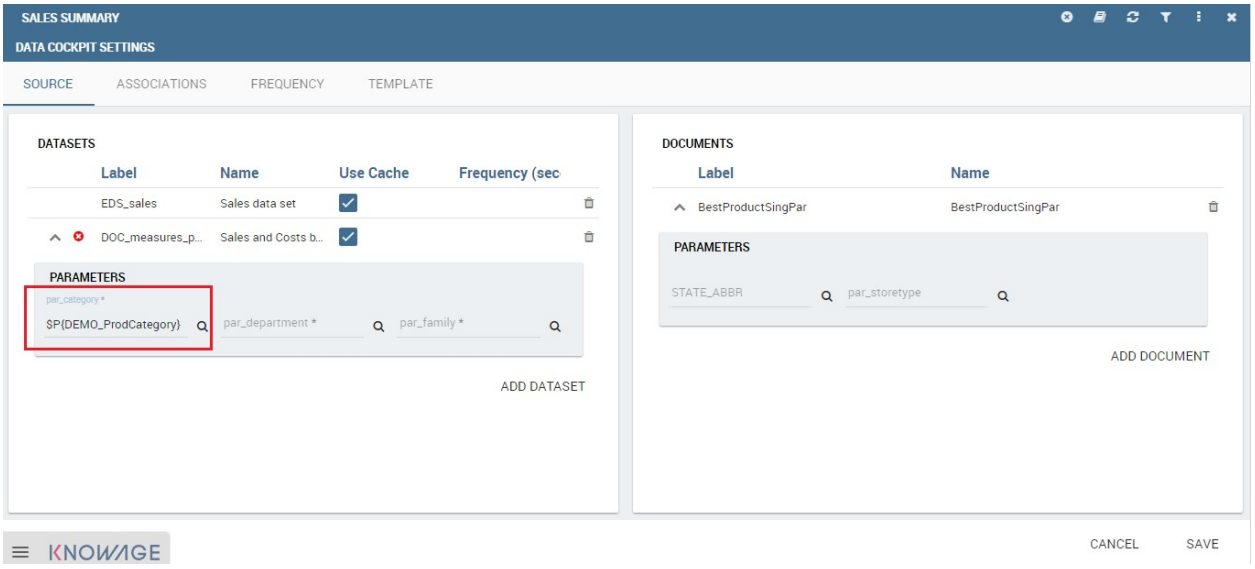


Fig. 5.161: Dataset management.

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.

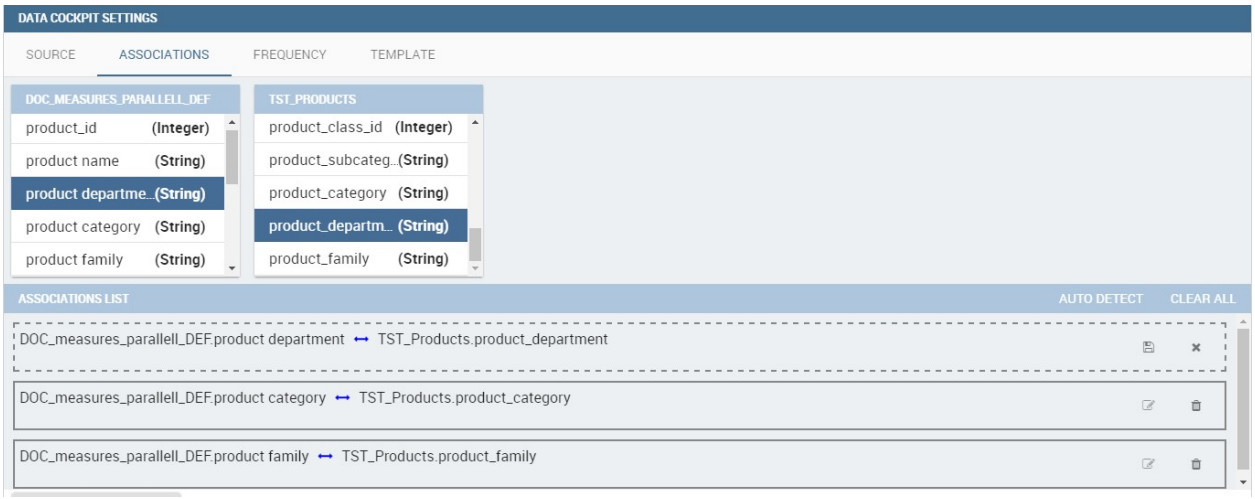


Fig. 5.162: Associations between dataset columns.

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

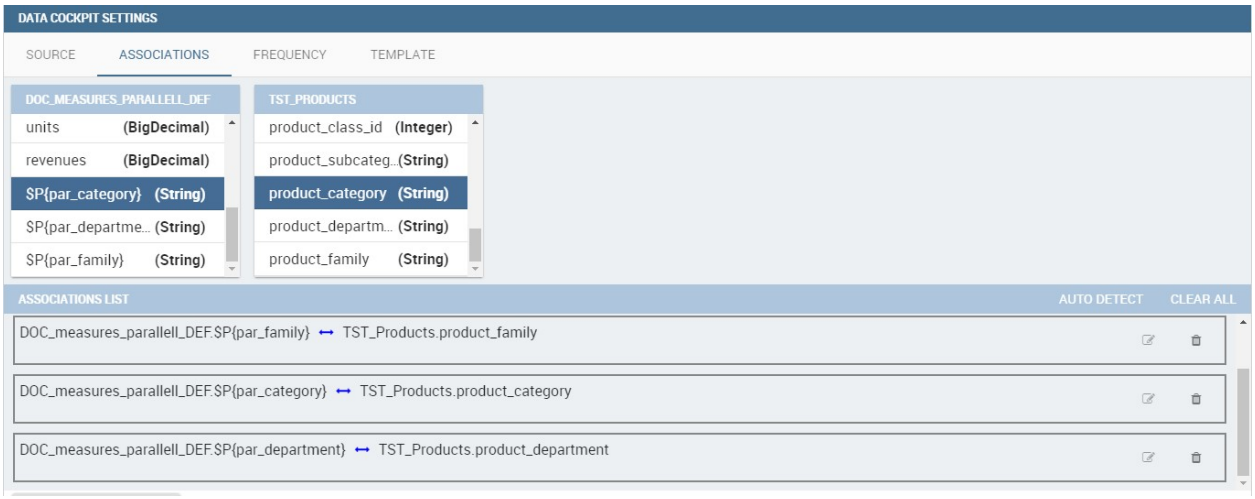


Fig. 5.163: Associations between dataset column and dataset parameter.

Another example is supplied in Figure below. Here the association is perfomed between a dataset Frequency 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.

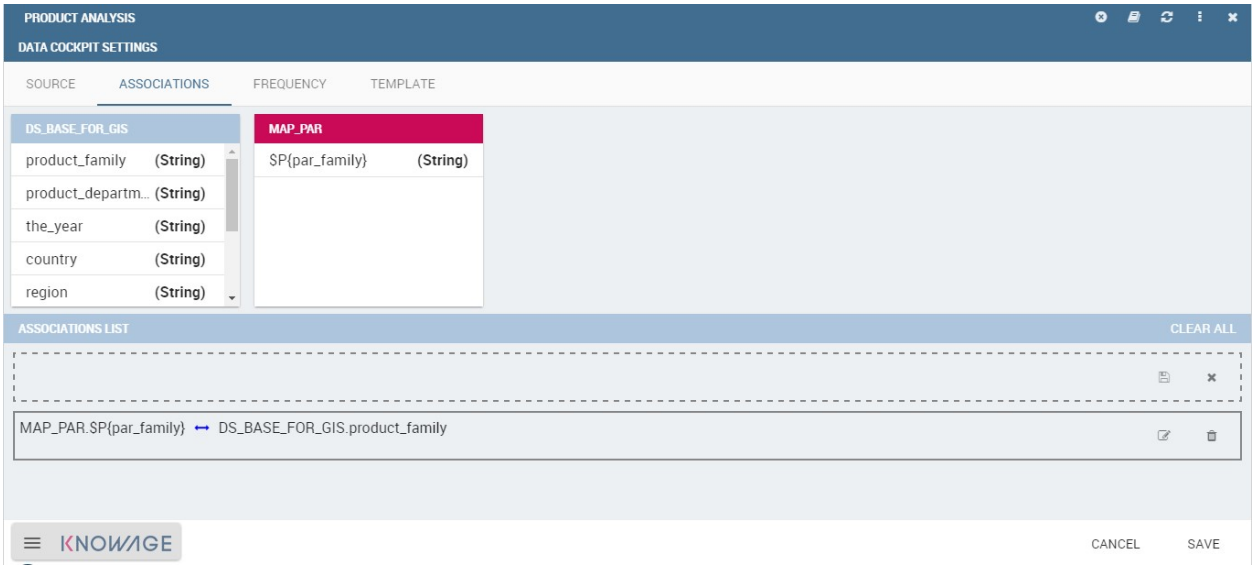


Fig. 5.164: Associations between dataset column and document parameter.

5.6.3.3 Frequency

The Frequency tab defines a schedulation 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.

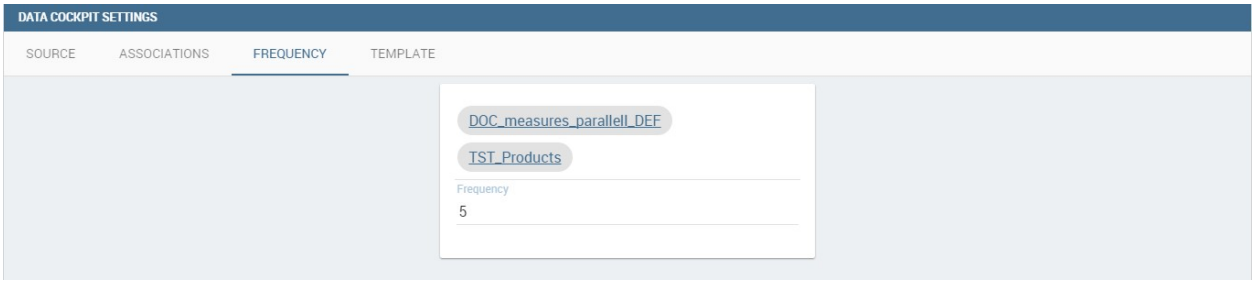


Fig. 5.165: Frequency settings example.

5.6.3.4 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 throught 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

Fig. 5.166: Template example.

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.

Fig. 5.167: Selection window.

5.6.5 Clear cache

The **Clear cache** button lets you reallign 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 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.

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.

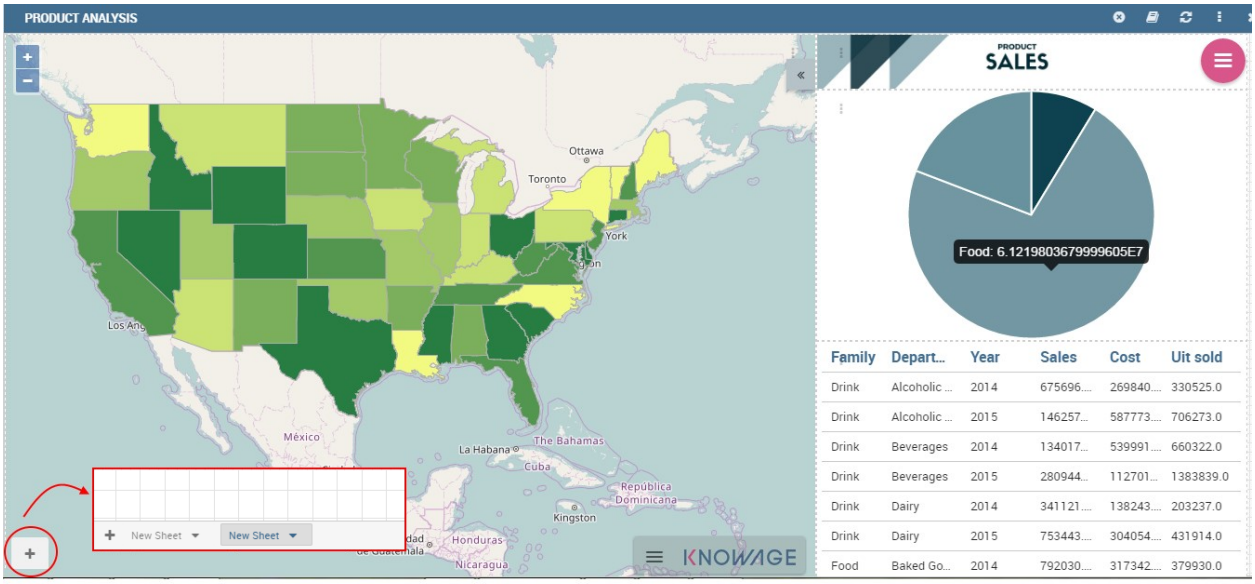


Fig. 5.168: Multisheet cockpit example.

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 analize. 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.

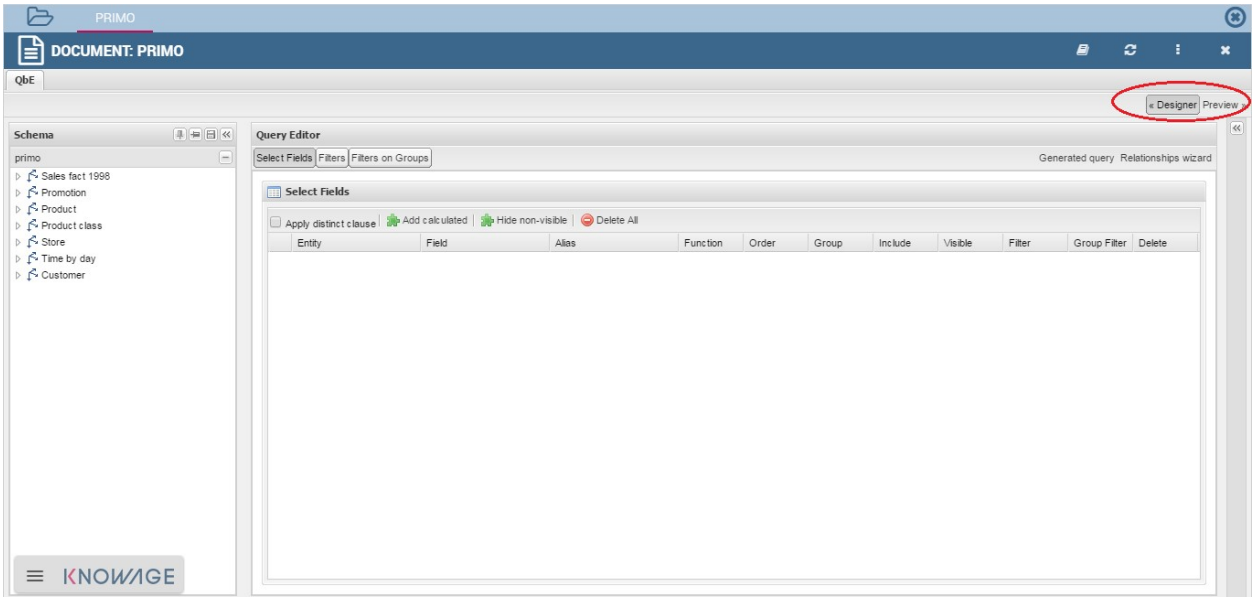


Fig. 5.169: QbE editor.

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.

5.7.1.1.1 Datamart Schema

Starting from the left side, the first Panel shows the searchable logical schema and the list of entities that can be queried to generate the query. Entities and relationships are represented in a tree structure, with user-defined names. Fields can be dragged from here and dropped onto the editor area.

In the top right corner of the panel you can find a small toolbar to configure the panel (e.g., expand, reduce) and to save changes made to the model (i.e. **Calculated Field** or **Range**), as shown below.

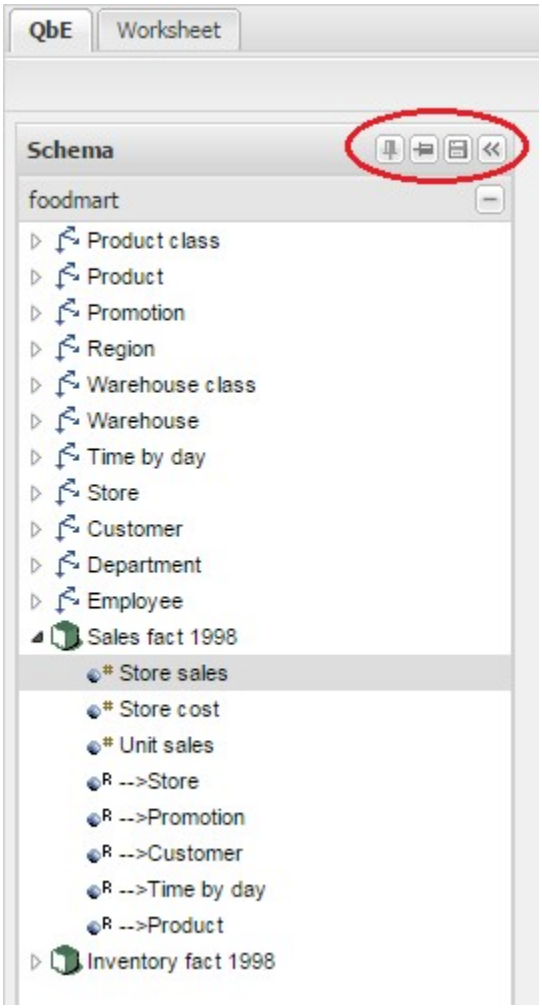


Fig. 5.170: Datamart schema toolbar.

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

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

- **measure**: it refers to fields associated with numeric data (e.g. number of sold items);
- **attribute**: it refers to fields that can be associated to a category (e.g. product category);
- **relation**: it refers to relationships or connections between two entities (e.g. relationship between the product sales fact and the product dimension).

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

- **Add calculated field**: to add a field that can be obtained via simple expressions combining existing fields and operators. Clicking on the contextual menu item, the wizard opens. Here you can combine fields with arithmetic and date functions. When you create a calculated field, you can add it to the model by clicking the **Save** button located in the top right corner of the panel. In addition, they can be used in queries. Calculated fields may also be managed by expert users via advanced functionalities, which will be described at the end of this section.
- **Edit field**: to rename a field.
- **Add/Edit Range**: to add or manage a range of values of the selected attribute (details are provided below).
- **Remove calculated field**: to remove a calculated field that was added before.

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

5.7.1.1.2 Calculated fields management

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

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

To build a calculated field, you shall define:

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

An example is provided below.

There are two types of calculated fields that you can add to the QbE query: *standard* and *expert*. The *standard* ones are SQL expressions that are injected into the query. With the *expert* calculated fields (you should mark the **Expert user** box in the calculated fields wizard) you can build Groovy scripts, show images, add links. This second type of calculated field is computed after the query has been executed.

5.7.1.1.3 Range management

It frequently happens that attributes of entities in a model have several different values. However, for the purpose of analysing data, it is often more useful to group those values into categories.

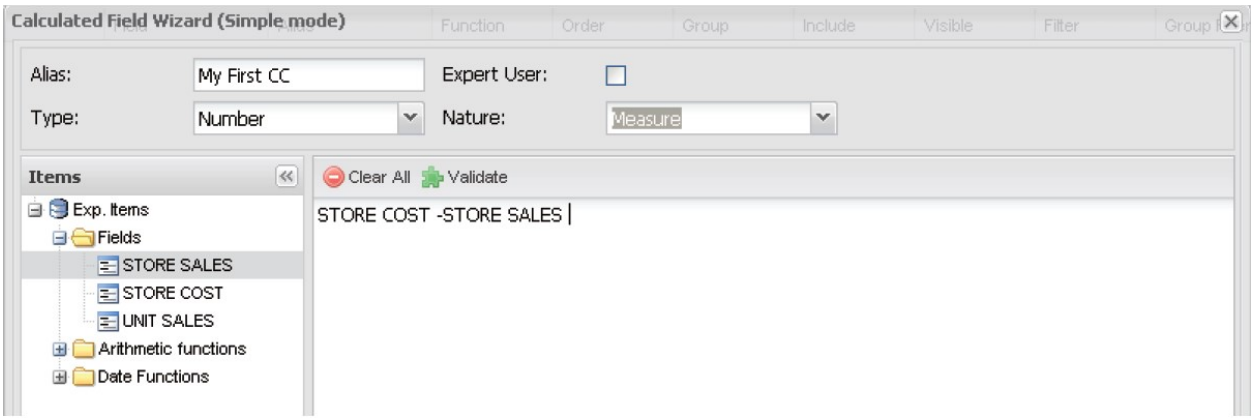


Fig. 5.171: Calculated field wizard.

For example, let's consider the customers' age: often analysts do not aim to know the exact age of customers, but rather if they belong to a certain age range, e.g., young, adult and elderly. For this and similar cases, the **QbE Engine** is able to define and manage ranges in queries. To create a new range for an attribute you can:

- right click a field and click on **Add Range** in the contextual menu;
- right click on an entity (a cube or a dimension), click on **Add Range** , then in the wizard choose a field or define a calculated expression, give a name to it and click on **Next**.

Both operations open the band creation wizard. Here click on **Add Band** to add a new instance and set the corresponding values and labels, as shown below.

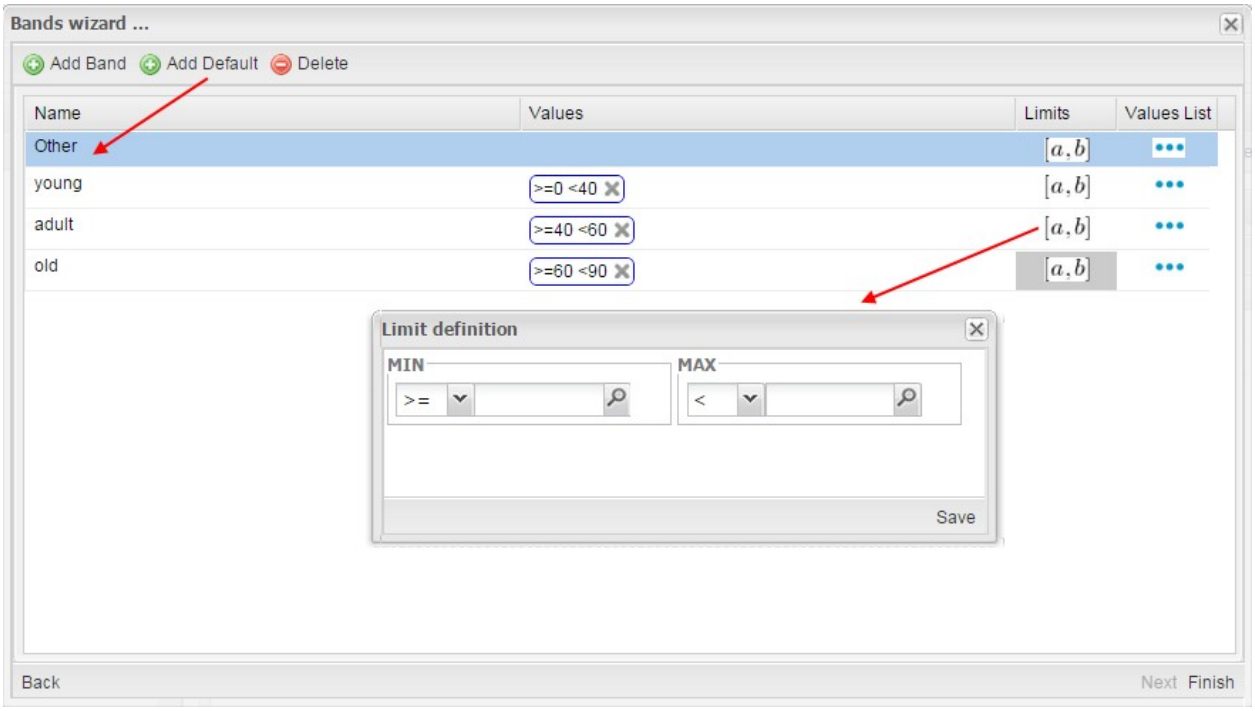


Fig. 5.172: Range instance creation.

You can set your band values by clicking on blue points under the Values List column to add them one by one, or by clicking on *[a,b]* under the **Limits** column to give only the end points of the interval. Then you can name your band

by double clicking on the related field under the Name. Repeat the procedure to generate all the bands you need.

Finally, you can click on **Add Default**: this creates a new category called **Others**, which groups all values not belonging to already defined range intervals. At this point, click on **Finish**. The range appears as a node in the schema panel on the left. If you want to edit the range, click on **Edit Range**.

5.7.1.1.4 Query Editor

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

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

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

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

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

5.7.1.1.5 Select Fields

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

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

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

With the other columns it is possible to:

- **Alias**: define aliases for fields: those aliases are shown as column headers in the result table;
- **Function**: in case of aggregation, define the aggregation function (e.g., **SUM**, **AVERAGE**, ...) on the non-grouped items;
- **Order**: define a sorting criteria: double click on the **Order** column to set the ordering criteria;
- **Group**: in case of aggregations, define the attribute that you want to group on (if you know SQL syntax, these attributes are the ones you should place in the GROUP BY clause);
- **Include**: indicate the column(s) to be included in the result (please notice that non-included attributes will not be returned by the query, but can be used in it, e.g. to apply grouping criteria);
- **Visible**: indicate whether a column shall be visible in the result (hidden attributes are used and returned by the generated query, but are not shown in the result table);
- **Filter**: add a filter criteria: clicking on this filter icon redirects you to the **Filters** tab;
- **Group Filter**: add a filter on groups: clicking on this filter icon redirects you to the **Filters on Groups** tab;

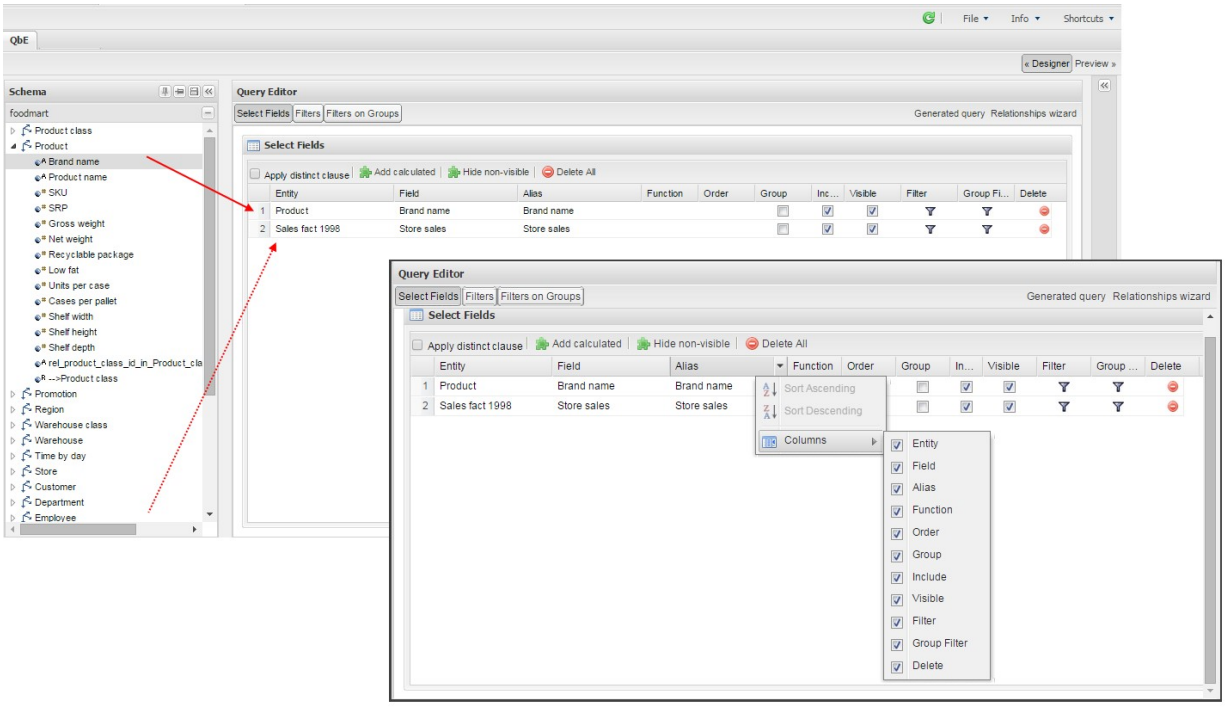


Fig. 5.173: Select fields interface.

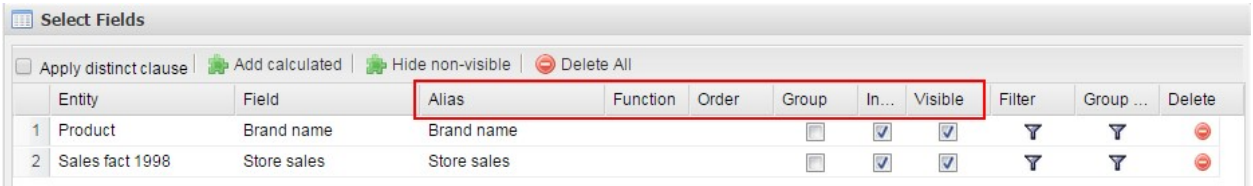


Fig. 5.174: **Select Fields** panel options.

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

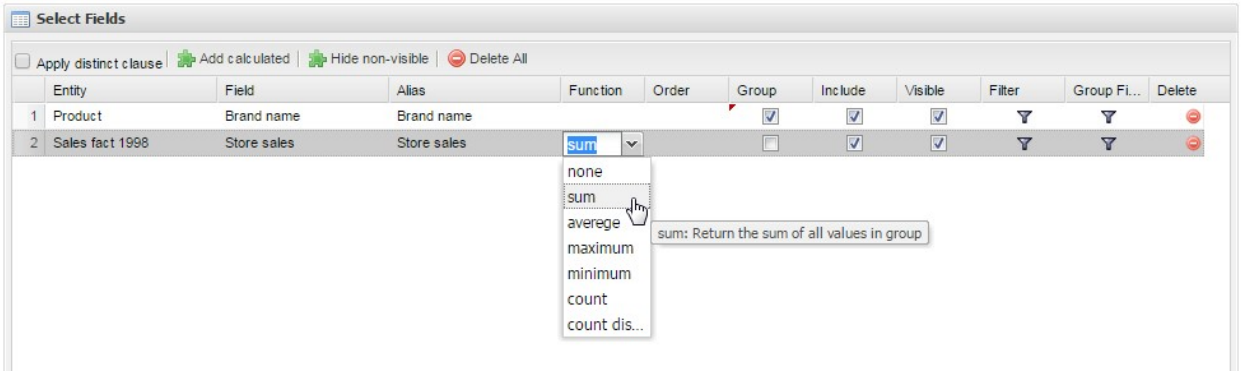


Fig. 5.175: Aggregation functions.

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 select sub-section has a toolbar with additional functionalities summarized in Table below.

Table 5.9: Select fields toolbar options

Button	Description
Apply distinct clause	Remove duplicated rows from results, if any
Hide non visible	Hide fields set as non visible in query results
Add calculated	Add a calculated field to the query
Delete all	Remove all rows from select area

5.7.1.1.6 Filters

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

There are three ways to create a filter:

Delete all Remove all rows from the select area

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

- click the **New** button in the **Filters** panel.

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

Filters are expressions of type:

Left operand + Operator + Right operand.

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

- the **Filter Name** column contains the (editable) name of the filter while the Filter Description column contains an editable description;
- the **Left operand, Operator, Right operand** columns allow you to define filters according to the syntax defined above. Double clicking in the Right operand column, a lookup function is activated to facilitate selection of values;
- the **LeftOperandType** and **RightOperandType** columns define the types of operands;
- the **Is for Prompt** column should be checked in order to insert dinamically the value for the parameters at execution time;
- the **Boolean Connector** column shall be used to control the evaluation order of the different filters conditions;

Not all available features of the editor panel are visible by default. To customize the editor appearance, double click on the arrow located on each column header and select **Columns**.

Here you can decide which columns you want to appear in the editor.

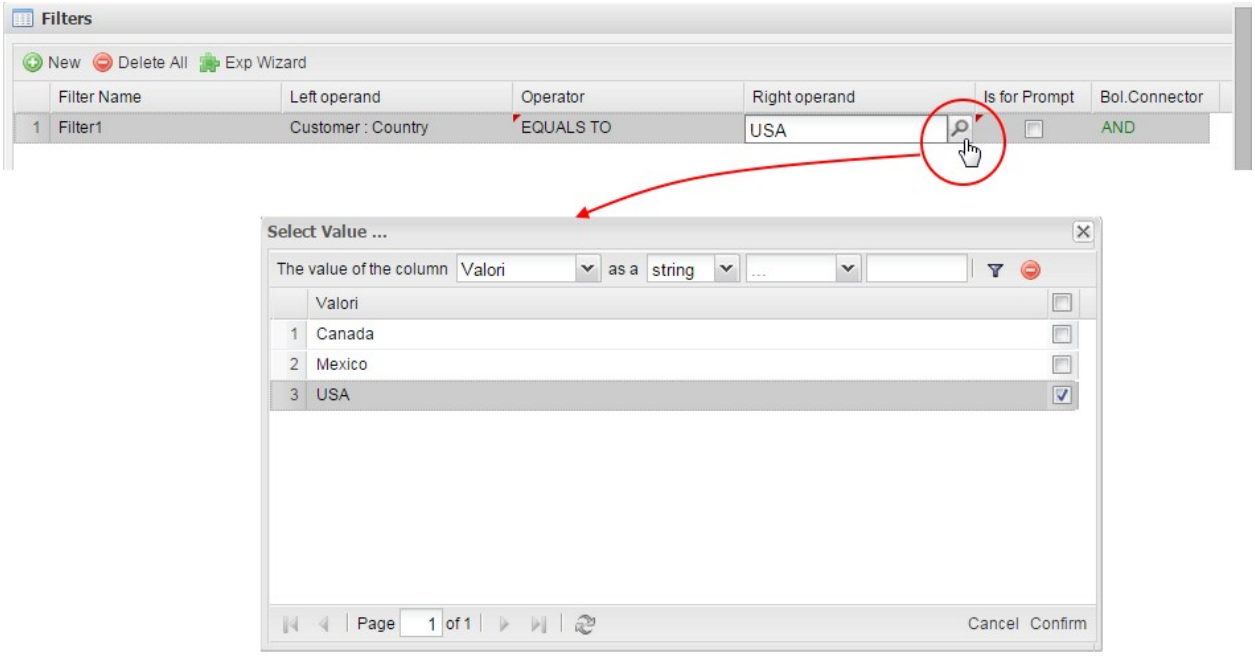


Fig. 5.176: Filter lookup for right operand selection.

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

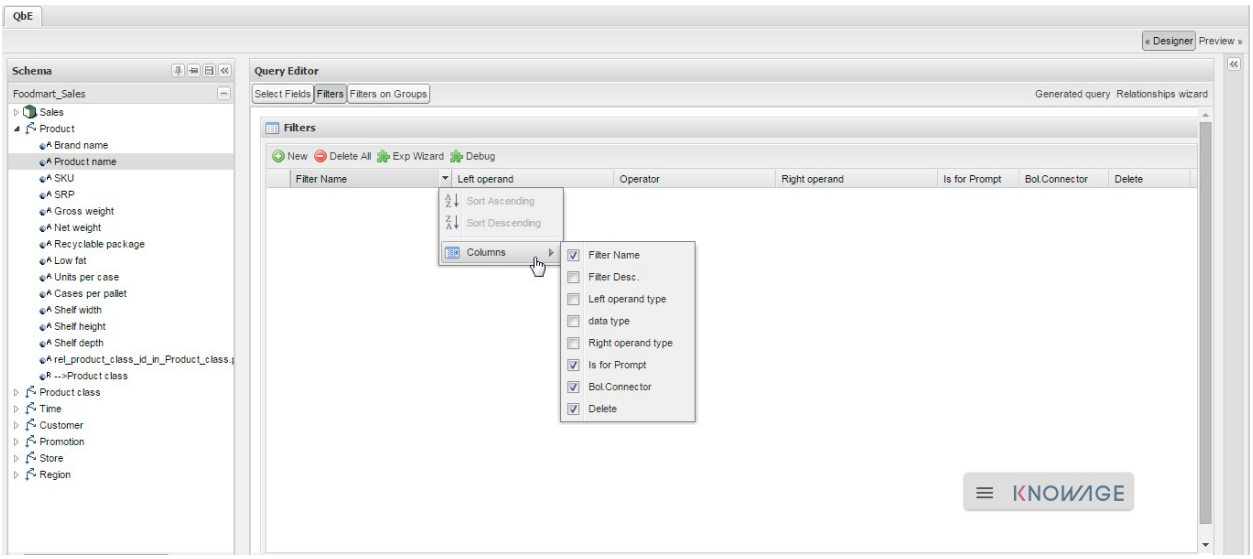


Fig. 5.177: Filter editor customization.

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

5.7.1.1.7 Filters on Groups

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

Filters on groups are expressions of type:

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

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

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

5.7.1.1.8 Query Preview

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

the definition of the query or switch directly to the **Worksheet** designer to start building your graphical representation of the extracted data.

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

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

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

Choose your scheduling option and save the dataset. Now the table where your data are stored will be persisted according to the settings provided.

- **Metadata** It recaps the metadata associated to the fields involved in your query.

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

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 possesed 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 distiguish 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 **Add calculated** 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 via drag and drop (or by a simple click on the field).

In addition note that the **Items** panel provides all the applicable functions sorted by categories:

- arithmetic functions,
- aggregation functions,
- date functions,

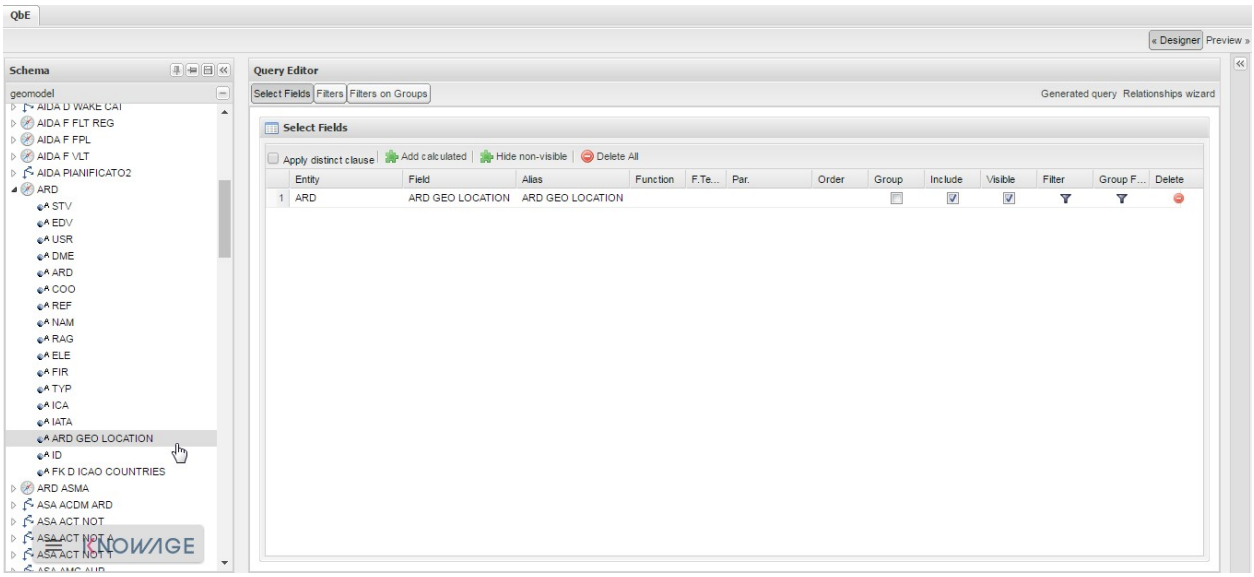


Fig. 5.178: QbE spatial dimensions.

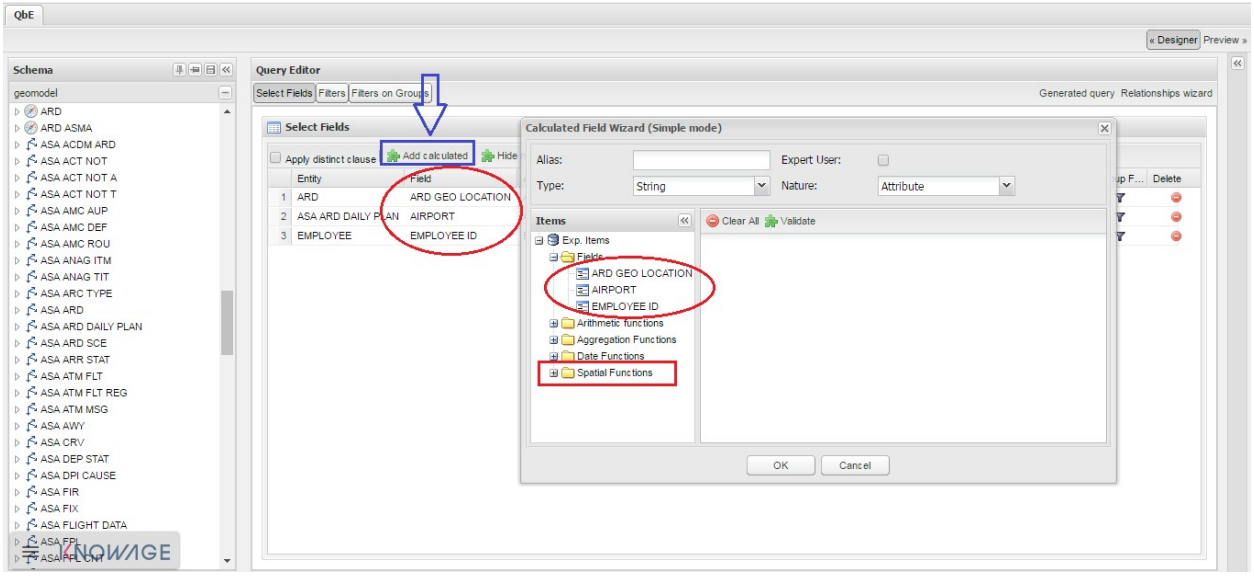


Fig. 5.179: Calculated field wizard with spatial filters.

- spatial functions.

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.

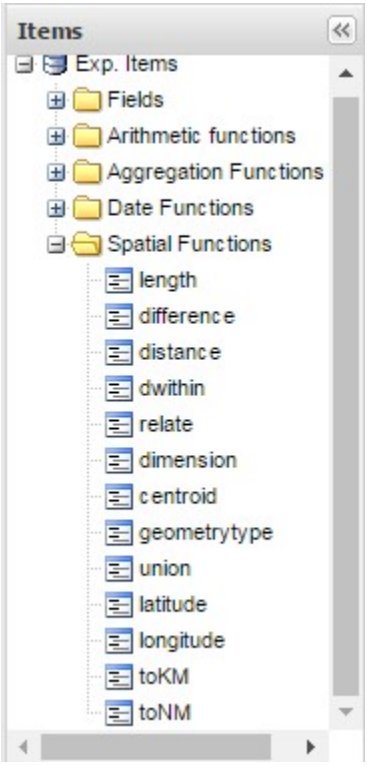


Fig. 5.180: Spatial function list.

Table 5.11: Link to Oracle spatial functions.

Function Name	Oracle Function	Link to Oracle web pages
distance	SDO_GEOM.SDO_DISTANCE	https://docs.oracle. com/cd/B19306_01/appd ev. 102/b14255/sdo_objgeo m.htm#i857957
dwithin	SDO_WITHIN_DISTANCE	https://docs.oracle. com/cd/B19306_01/appd ev. 102/b14255/sdo_operat .htm#i77653
dimension	GET_DIMS	https://docs.oracle. com/cd/B10501_01/appd ev. 920/a96630/sdo_meth.h tm#BABDEBJA
difference	SDO_GEOM.SDO_DIFFERENCE	https://docs.oracle. com/cd/B19306_01/appd ev. 102/b14255/sdo_objgeo m.htm#i857512
centroid	SDO_GEOM.SDO_CENTROID	https://docs.oracle. com/cd/B19306_01/appd ev. 102/b14255/sdo_objgeo m.htm#i860848
geometry-type	GET_GTYPE	https://docs.oracle. com/cd/B10501_01/appd ev. 920/a96630/sdo_meth.h tm#i866821
union	SDO_GEOM.SDO_UNION	https://docs.oracle. com/cd/B19306_01/appd ev. 102/b14255/sdo_objgeo m.htm#i857624
length	SDO_GEOM.SDO_LENGTH	https://docs.oracle. com/cd/B19306_01/appd ev. 102/b14255/sdo_objgeo m.htm#i856257
relate	SDO_GEOM.RELATE	https://docs.oracle. com/cd/B19306_01/appd ev. m.htm#BGHCDIDG

To apply one function click on the function name and the "Operands selection window" wizard opens. Figure below shows an example for the funtion "Distance". Fill in all boxes since all fields are mandatory.

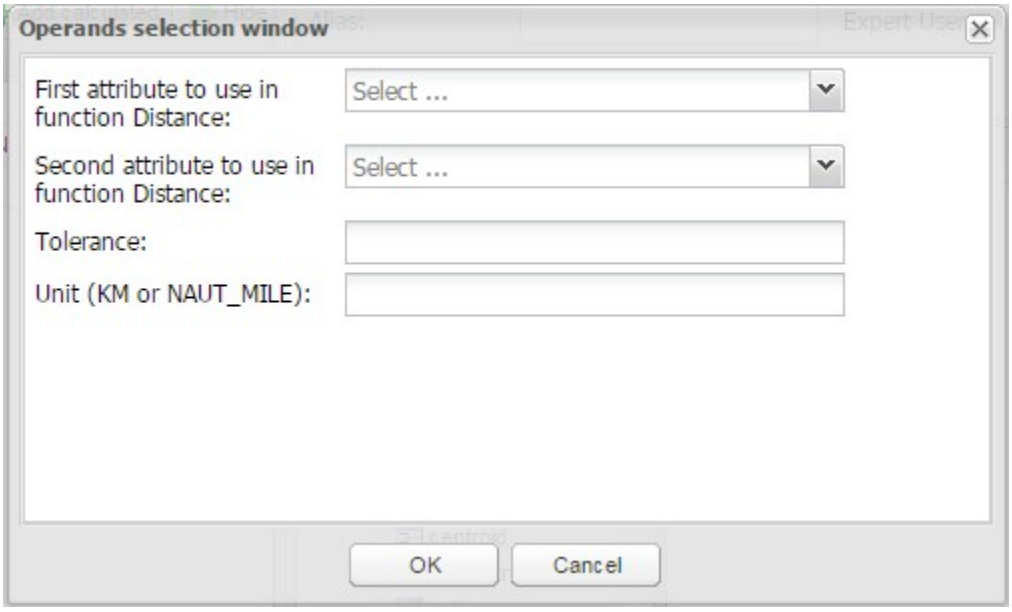


Fig. 5.181: Operands selection window.

Finally you can use spatial function to add a calculated field, as shown below.

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.

See the table below:

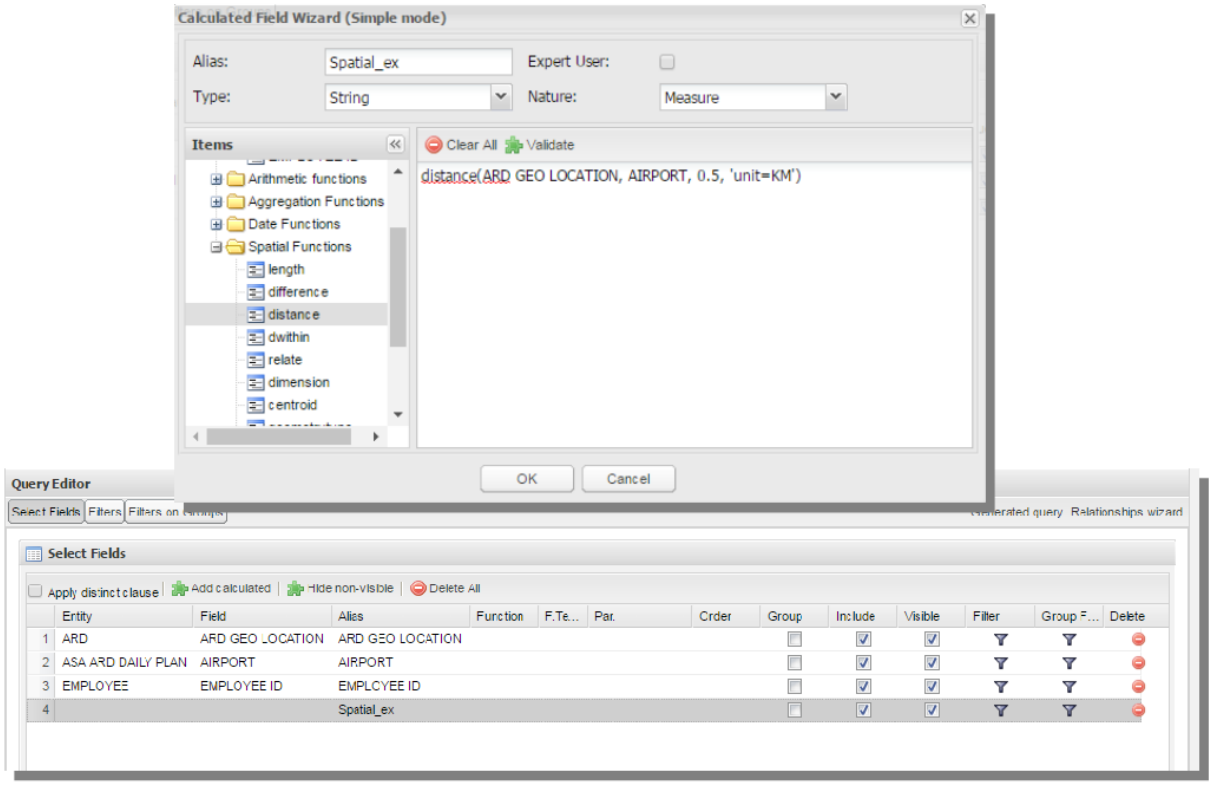


Fig. 5.182: Example of added calculated field using a spatial function.

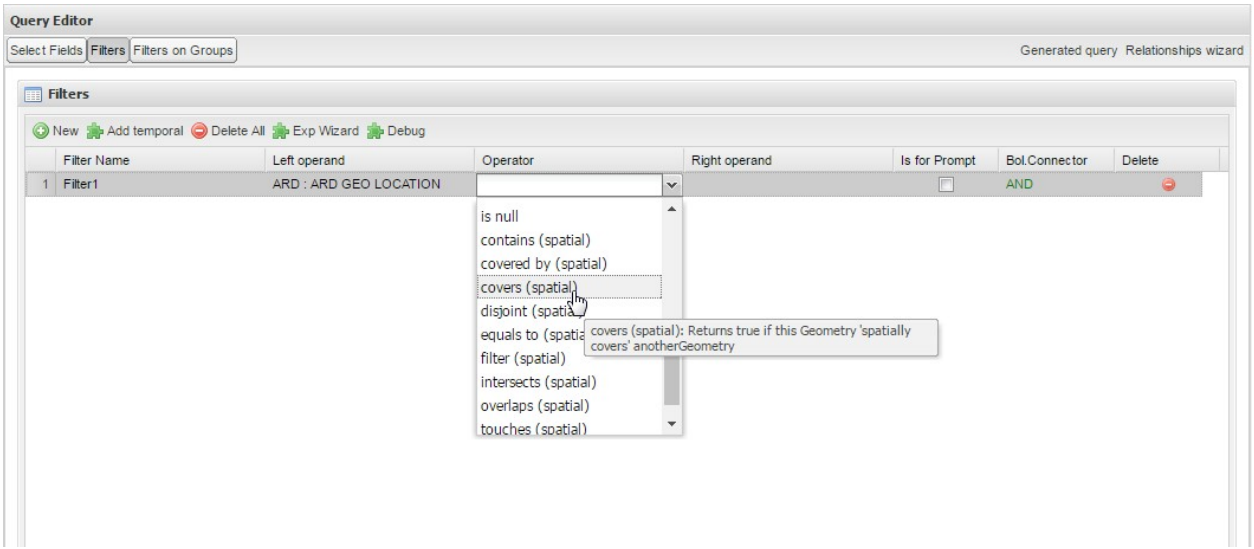


Fig. 5.183: Spatial filters.

Table 5.12: Link to Oracle filter functions.

Function Name	Oracle Function	Link to Oracle web pages
touches	SDO_TOUCH	https://docs.oracle. com/cd/B19306_01/appd ev. 102/b14255/sdo_operat .htm#BGEHHIGF
filter	SDO_FILTER	https://docs.oracle. com/cd/B19306_01/appd ev. 102/b14255/sdo_operat .htm#BJAFBCFC
contains	SDO_CONTAINS	https://docs.oracle. com/cd/B19306_01/appd ev. 102/b14255/sdo_operat .htm#BGEHCFDH
covered by	SDO_COVEREDBY	https://docs.oracle. com/cd/B19306_01/appd ev. 102/b14255/sdo_operat .htm#BGEHEAEJ
inside	SDO_INSIDE	https://docs.oracle. com/cd/B19306_01/appd ev. 102/b14255/sdo_operat .htm#BGEFABDH
covers	SDO_COVERS	https://docs.oracle. com/cd/B19306_01/appd ev. 102/b14255/sdo_operat .htm#BGEGIJFB
overlaps	SDO_OVERLAPS	https://docs.oracle. com/cd/B19306_01/appd ev. 102/b14255/sdo_operat .htm#BGEDACIF
equals to	SDO_EQUAL	https://docs.oracle. com/cd/B19306_01/appd ev. 102/b14255/sdo_operat .htm#BGEBCEJE
intersects	SDO_ANYINTERACT	https://docs.oracle. com/cd/B19306_01/appd ev. 102/b14255/sdo_operat .htm#BGEJHDGD

5.7.1.2.2 Temporal dimension

The Qbe engine on Knowage Server is endowed with some temporal functionalities that allow the final user to easily perfom queries based on time.

We highlight that the new features are available only if the model has at least one temporal dimension. The latter must be defined while creating the model using Knowage Meta.

Warning: Define first the temporal dimension on Knowage Meta

To have a temporal dimension that can be used in the Qbe interface an expert user must enable it first on the model using Knowage Meta. Use the **property view** to set/change the type of the dimension as shown in the following figure. Refer to *Meta Web* chapter to learn how to use Knowage Meta.

The temporal dimension can have one or more hierarchies. Only one of these can stay active and that is the one used by the query code. Figure below shows that a temporal dimension can have one or more hierarchies. In the case of more hierarchies the user can see which is the one set by default just exploring the dimension: the bold highlighted hierarchy is the primary. On the other hand the user can change the default choice by right-clicking on the target dimension hierarchy and selecting "*Set as Default Hierarchy*".

Furthermore there is the possibility to set a "time" dimension as Figure below displays.

The user can use the elements of each dimension as attributes in the "Select" instance. Note that if one drags and drops of element to be used as a filter also its parent nodes will be brought too. The following figure exhibits one example. Remember to assign a value to each parent node before you run the query.

Moreover, selecting the filters tab, you can use specific filters clicking on the button "Add Temporal" as shown in figure below (Left). The action opens the pop up displayed in next figure (Right).

In the list of available elements is made up of:

- filters defined by the admin through the TimeSpan GUI;

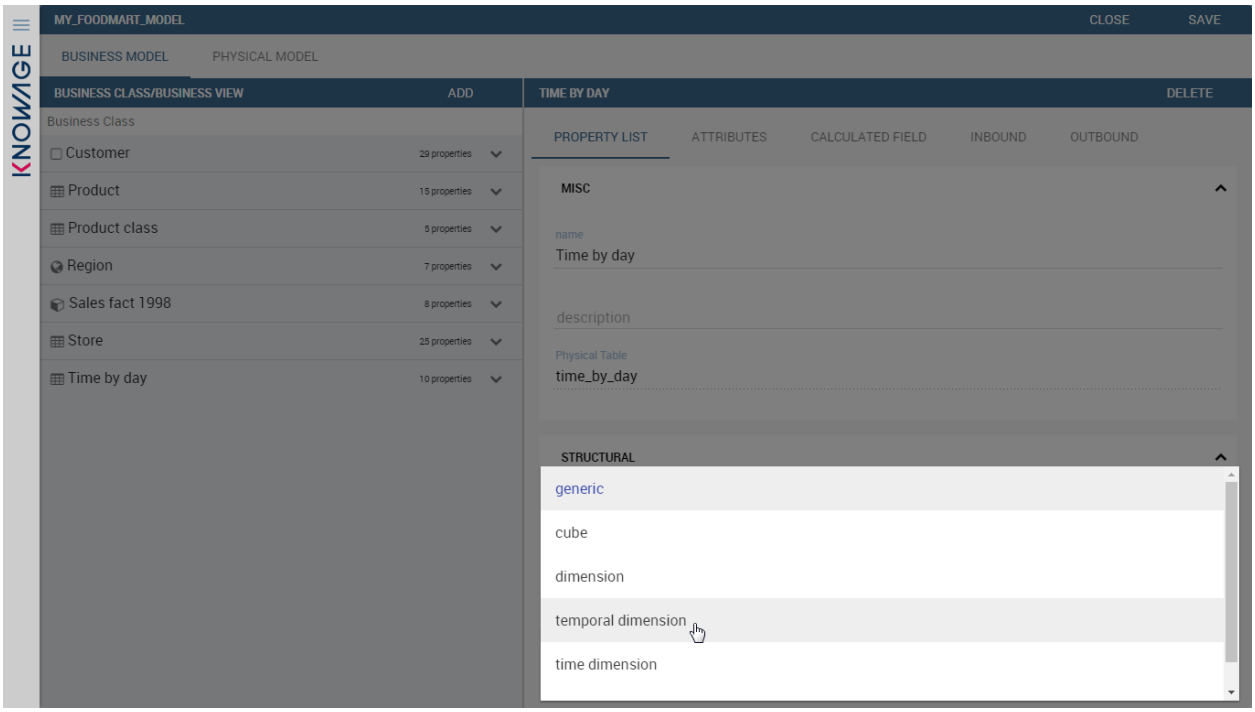


Fig. 5.184: Temporal dimension definition on Meta.

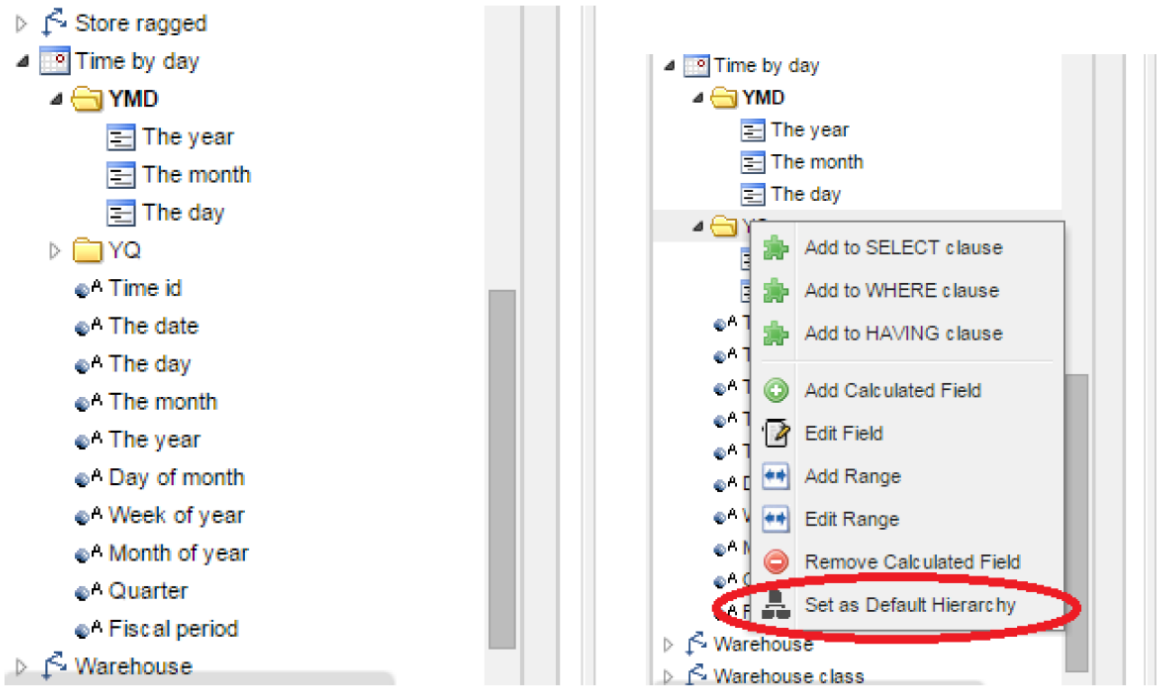


Fig. 5.185: Temporal hierarchy visualization (Left). Changing hierarchies (Right).

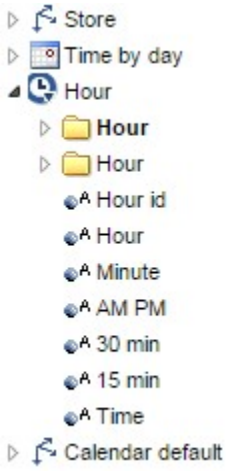


Fig. 5.186: Time dimension.

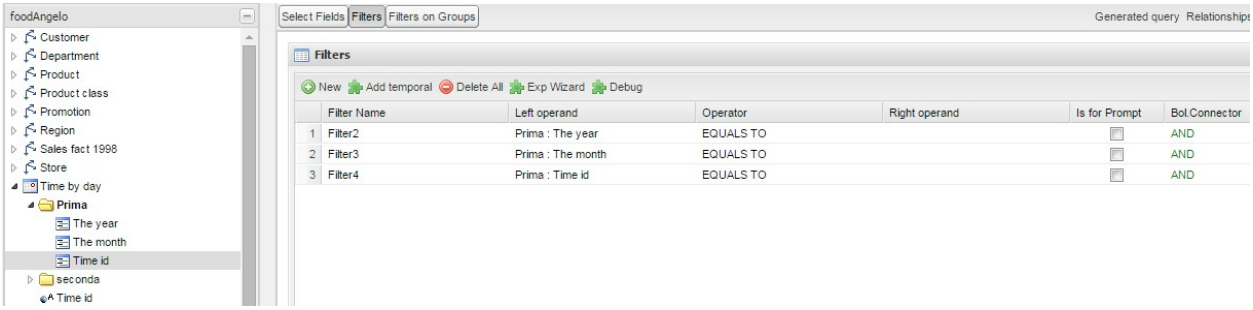


Fig. 5.187: Filter on an element means to filter also on its parent nodes.

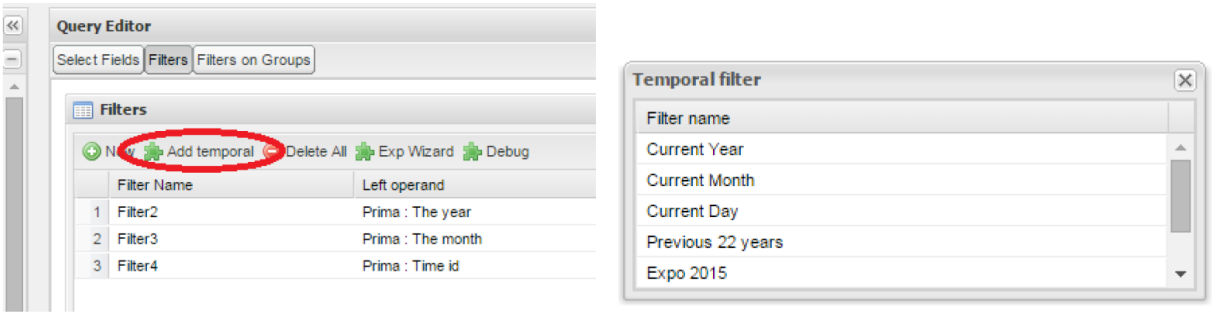


Fig. 5.188: Add temporal filters (Left). List of available elements (Right).

- system filters manageable through a table;
- the element “Current year”;
- the element “Current month”;
- the element “Current day”;
- the element “Last Period” for which you must indicate the number of years.

Inside the section “Select” you can use the temporal operators directly on attributes.

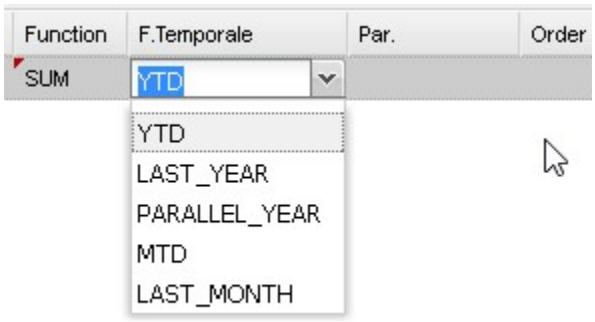


Fig. 5.189: Apply operators directly on attributes.

For each function there is the possibility to assing a value to a parameter that indicates how long the function will act. We now describe the working principles of temporal functions.

5.7.1.2.2.1 The PARALLEL_YEAR function.

This function allows to manage and study measures on parallel periods. For example if one wants to analize the product sales of the current year and, at the same time, those of the previuos year. The following are some possible use cases:

- no temporal filter is set and the temporal functions are applied directly on measures. In this case the current year is taken as default value. When the functions are applied on measures the user must apply them on ALL measures in order to have a coherent result.

1. In the case the user wants the sum of a measure relative to current year, he/she must drag and drop the measure in the “select fields” panel and launch the temporal function PARALLEL_YEAR passing 0 as value. See Figure below as example.

Fig. 5.190: PARALLEL_YEAR example: sum of a measure referred to a specific time year.

2. In the case the user wants to compare the sales of 2016 with those of the previous year. He/she has to drag twice the measure inside the “select fields” panel and indicate the temporal function “PARALLEL_YEAR” using 0 and 1 as value parameters.

- Suppose now that the analysis requires to compare the unit sold from January to March of the current year with that of the same time interval of the previous one. In this instance the user must set the temporal filter which will be the point of reference.

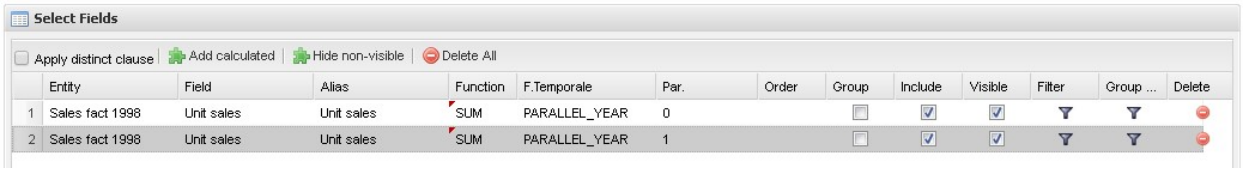


Fig. 5.191: PARALLEL_YEAR example: comparing data with different time interval.

Fig. 5.192: PARALLEL_YEAR example: setting the temporal filter.

Remember that the temporal filter uses the "IN" operator.

- In the case one wants to compare the sales per month of the current year with the ones of the parallel year, the user should add the month field in the select clause (picking it up from the used temporal hierarchy) and group by it.

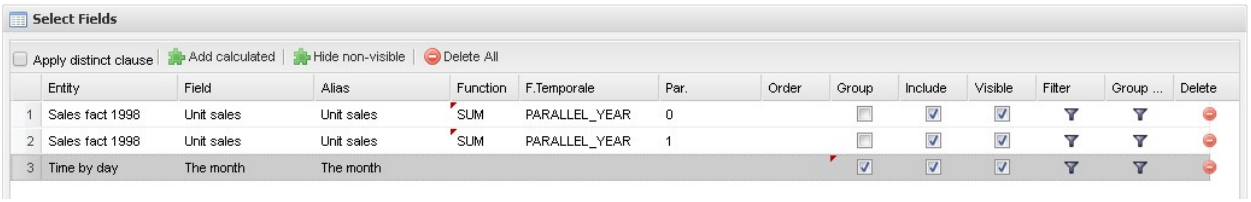


Fig. 5.193: Comparing results with those of the parallel year.

An example of data visualization is given in the two figures below.

5.7.1.2.2.2 The LAST_YEAR function

This function allows the user to sum a measure referring to last period data. If the temporal filter isn't set, the engine takes the current year by default, otherwise the chosen one.

- In our example in the two figures below the period is the year. Here we compare last-year sold products to the sum of those sold in last two years.
- Referring to figures below give an example of how to define a time reference, for instance 2015. In this case I pass 2015 to the filter.
- In case the user wants to inspect the evolution of sales per month of the current year comparing them with those of last year plus the current. It is sufficient to add the month in the "selected fields" area and the measure "unit sold" where the LAST_YEAR function is set on 0 or 1. The two following figures show an example.

Note that the operator allows to visualize the sum of sales upon 2 years per month. In other words, LAST_YEAR(1) set to the month level starts the progression from the aggregated value of 2015 to which it adds the sales of 2016.

	Unit sales_PARALLEL_YEAR_0	Unit sales_PARALLEL_YEAR_1	The month
1	23,156.50	46,313.00	January
2	22,215.50	44,431.00	February
3	23,167.00	46,334.00	March
4	22,524.50	45,049.00	April
5	22,542.50	45,085.00	May
6	0.00	45,611.00	June
7	0.00	46,671.00	July
8	0.00	44,777.00	August
9	0.00	47,964.00	September
10	0.00	43,945.00	October
11	0.00	53,807.00	November

Fig. 5.194: Comparing results with those of two parallel years.

Unit sales_PARALLEL_YEAR_0	Unit sales_PARALLEL_YEAR_1	Unit sales_PARALLEL_YEAR_2	The month
23,156.50	46,313.00	66,161.28	January
22,215.50	44,431.00	63,472.71	February
23,167.00	46,334.00	66,191.28	March
0.00	45,049.00	64,355.57	April
0.00	45,085.00	64,406.99	May
0.00	45,611.00	65,158.42	June
0.00	46,671.00	66,672.71	July
0.00	44,777.00	63,966.99	August
0.00	47,964.00	68,519.84	September
0.00	43,945.00	62,778.43	October
0.00	53,807.00	76,866.97	November

Fig. 5.195: Comparing results with those of three parallel years.

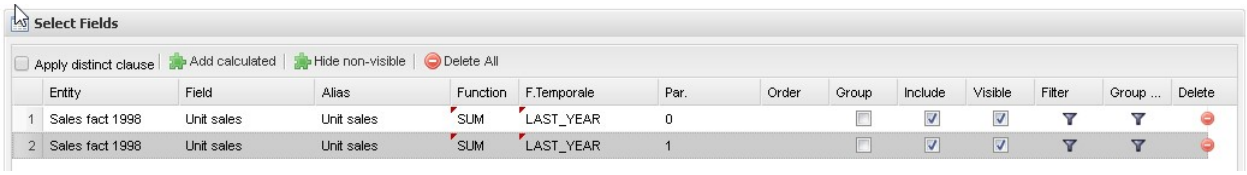


Fig. 5.196: LAST_YEAR function.

	Unit sales_LAST_YEAR_0	Unit sales_LAST_YEAR_1
1	113,606.00	623,593.00

Fig. 5.197: Comparing LAST_YEAR results.

Fig. 5.198: LAST_YEAR function example: changing the reference year.

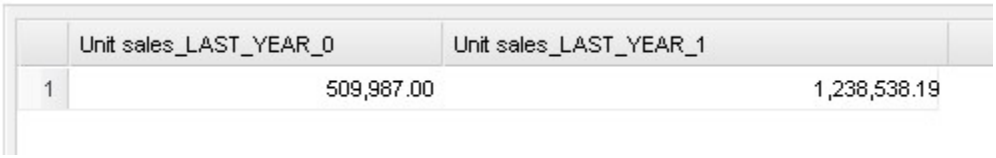


Fig. 5.199: LAST_YEAR function example: output of changing the reference year.

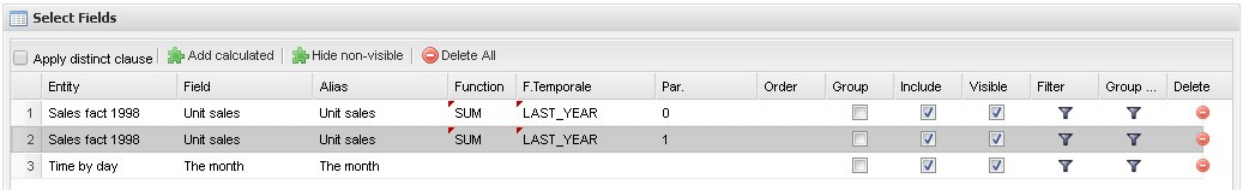


Fig. 5.200: LAST_YEAR function example: last-year sold products compared to the last-twoyear ones.

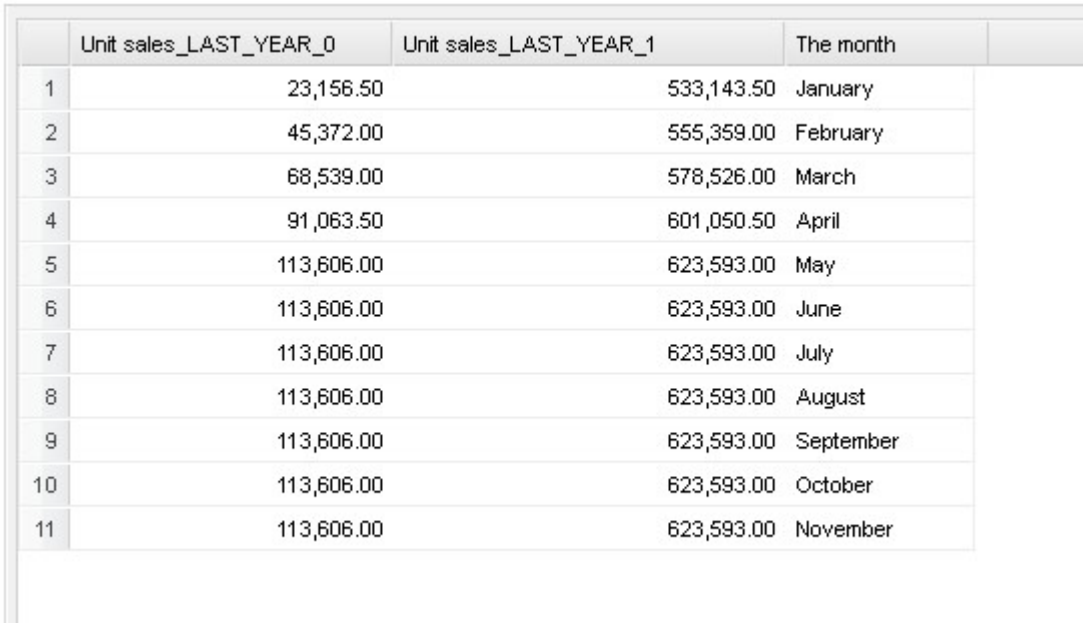


Fig. 5.201: LAST_YEAR function example: output of last-year sold products compared to the last-two-year ones.

5.7.1.2.2.3 The LAST_MONTH function

This operator is very similar to the previous one. In this case the reference time period is the month. Remember that if the user does not specify the name of the referenced month the system will take the current one by default.

- The user wants to count the sales of last three months.

Fig. 5.202: LAST_MONTH function example: setting time reference.

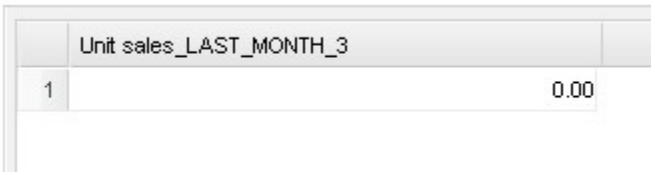


Fig. 5.203: LAST_MONTH function example: setting time reference.

- Figure below shows how to aggregate data up to last three months per each month of the current year. Remember to add the month in the section "selected fields". Therefore, inserting the month in the select clause the user obtains a projection on current year of sales of last 3 months per each month. Note that data are related to the current year, namely there is no shift to the passed one. Pay attention to the fact that if one month is missing the system does not notice it and return a sum relative to a bigger time period.

Fig. 5.204: LAST_MONTH function example: sum up to last 3 months.

- The same query can be performed cosidering a specific year. In the following figures year 2015 has been selected.
- If the user wants to compare sales per month to those of the previuos month summed to the current one. Results in the following figures reflect this selection.

5.7.1.2.2.4 The YTD function

This operator aggregate the measure of the first day of the year up to the execution date (currentDay). If the user sets temporal filters the YTD function must refer to the filter. The chosen day will be used as reference by the function. For example, if the user sets "15/03/2016" as reference day, the function sums starting from the first of January up to the 15th of March (2016). Observe that if the filter is monthly the engine will take the last day of the month, while if it is yearly the engine will take the whole year. If the user inserts a temporal element as aggregation function the measure must be aggregated progressively.

- Below shows the case in which the user wants to count the sales from the beginning of the year up to now.

	Unit sales_LAST_MONTH_3	The month
1	23,156.50	January
2	45,372.00	February
3	68,539.00	March
4	91,063.50	April
5	90,449.50	May

Fig. 5.205: LAST_MONTH function example: sum up to last 3 months output.

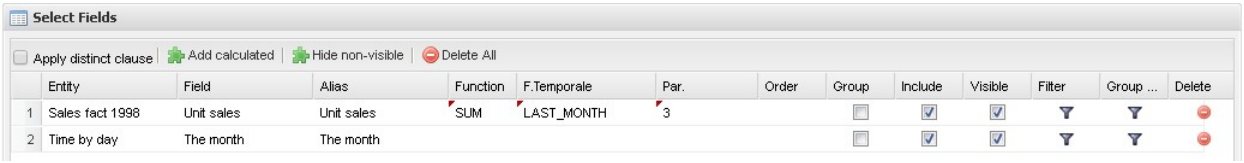


Fig. 5.206: LAST_MONTH function example: sum up to last 3 months where year is 2015.

	The month	Unit sales_LAST_MONTH_3
1	January	46,313.00
2	February	90,744.00
3	March	137,078.00
4	April	182,127.00
5	May	180,899.00
6	June	182,079.00
7	July	182,416.00
8	August	182,144.00
9	September	185,023.00
10	October	183,357.00
11	November	190,493.00

Fig. 5.207: LAST_MONTH function example: output when one sums up to last 3 months output where year is 2015.

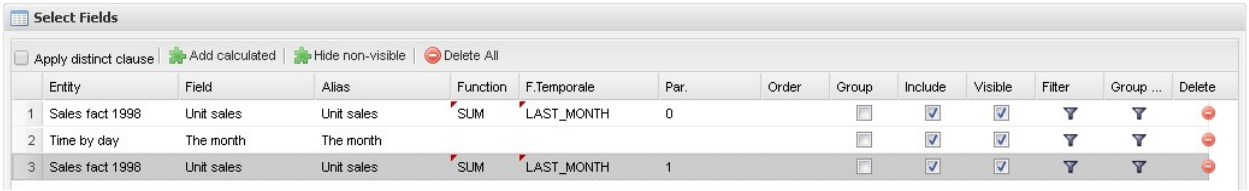


Fig. 5.208: LAST_MONTH function example: sales per month aganst the sum of current and previous month sales.

	The month	Unit sales_LAST_MONTH_0	Unit sales_LAST_MONTH_1
1	January	23,156.50	23,156.50
2	February	22,215.50	45,372.00
3	March	23,167.00	45,382.50
4	April	22,524.50	45,691.50
5	May	22,542.50	45,067.00

Fig. 5.209: LAST_MONTH function example: results of sales per month aganst the sum of current and previous month sales.

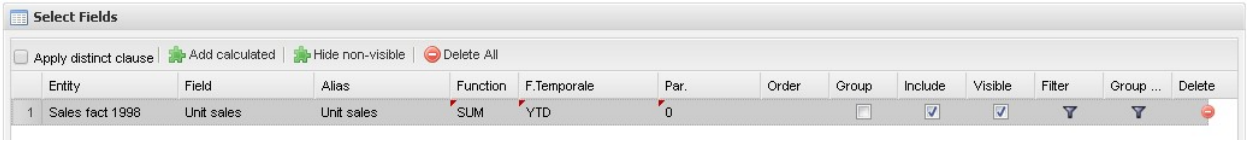


Fig. 5.210: YTD function example: to count the sales from the beginning of the year up to now.

	Unit sales_YTD_0
1	113,606.00

Fig. 5.211: YTD function example: number of sales from the beginning of the year up to now.

- Below shows the case in which the user wants to count the sales from the beginning of the year up to the end of March.

Fig. 5.212: YTD function example: to count the sales from the beginning of the year up to the end of March.

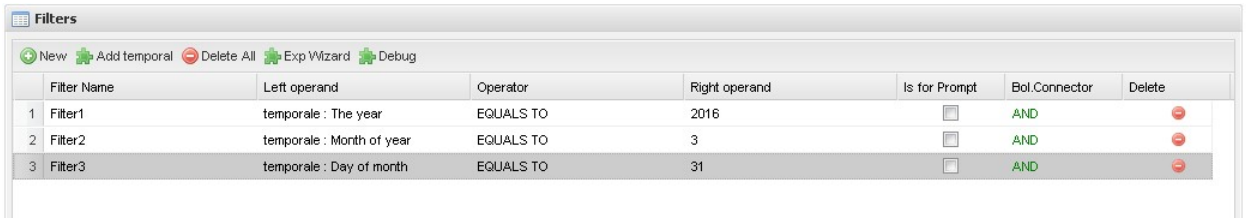


Fig. 5.213: YTD function example: to count the sales from the beginning of the year up to the end of March.

- The following figure refers to the case where the user wishes to sum 2015 sales considering the day in which the query is executed but of the previuos year.

Fig. 5.214: YTD function example: sum 2015 sales considering the day in which the query is executed but of the previuos year.

- The following figure refers instead to the case where the user wishes to sum 2015 sales of first 3 months of 2015.
- In the following figures the user is comparing the unit sold from the beginning of the year with those of the previuos year. The engine considers the day of query execution as end of the time period.
- Figures below shows the instance when the user wants to see the sum of unit sold each month after having added the month field in the select clause.

5.7.1.2.2.5 The MTD function

The MTD function follows the same logic as the YTD function but using the month.

- Figure below shows the case in which the user wants to check the unit sold during the current month.
- The following figures the user wants to check the aggregated sales of last 7 months, current (relative to the execution time) month included.
- Another case is shown in figures below where sales are aggregated on current month plus the previuos one, relative to the current year (referring to the query execution time).
- Figures below shows a user that is summing the sales of current month
- The following figures shows as a user can compare sales of aggregated months (up to the current) to the current one.
- Below shows a case very similar to the previous one. In this case the next month is added to the sum.

Fig. 5.215: YTD function example: sales summed up to the first 3 months of 2015.

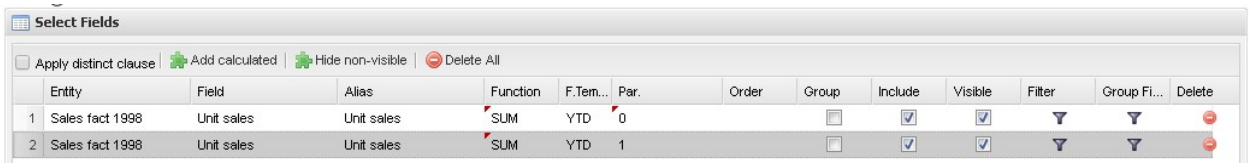


Fig. 5.216: YTD function example: comparing the unit sold from the beginning of the year with those of the previuos year.

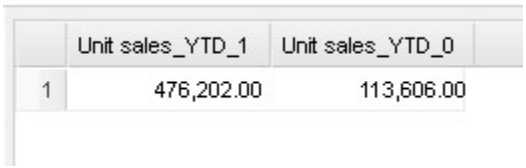


Fig. 5.217: YTD function example: output when comparing the unit sold from the beginning of the year with those of the previuos year.

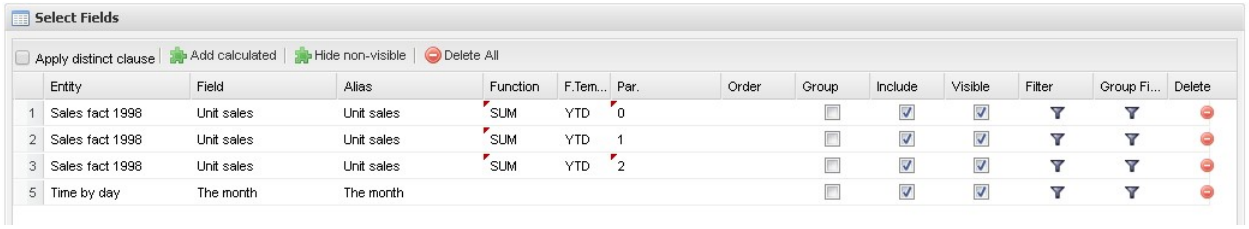


Fig. 5.218: YTD function example: sum of unit sold each month after having added the month field in the select clause.

	Unit sales_YTD_1	Unit sales_YTD_0	The month	Unit sales_YTD_2
1	46,313.00	23,156.50	January	66,161.28
2	90,744.00	45,372.00	February	129,633.99
3	137,078.00	68,539.00	March	195,825.27
4	182,127.00	91,063.50	April	260,180.84
5	227,212.00	113,606.00	May	324,587.83
6	272,823.00	113,606.00	June	389,746.26
7	319,494.00	113,606.00	July	456,418.96
8	364,271.00	113,606.00	August	520,385.95
9	412,235.00	113,606.00	September	588,905.79
10	456,180.00	113,606.00	October	651,684.23
11	509,987.00	113,606.00	November	728,551.19

Fig. 5.219: YTD function example: output when one sums unit sold each month after having added the month field in the select clause.

Fig. 5.220: MTD function example: check the unit sold during the current month.

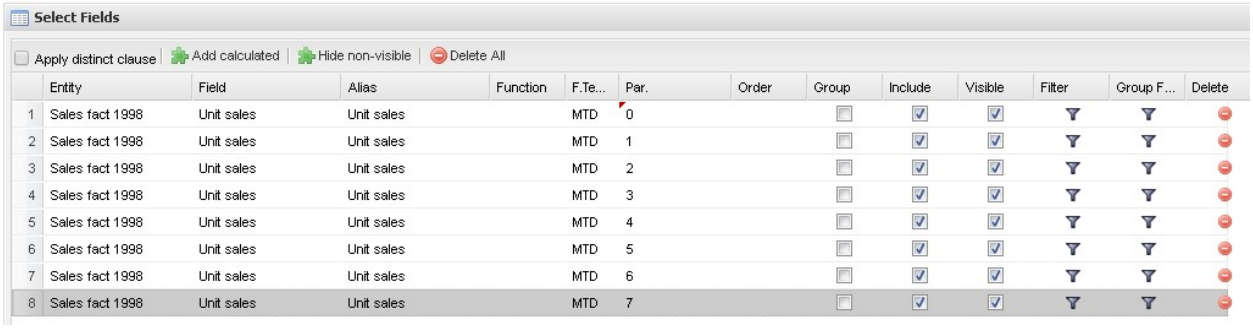


Fig. 5.221: MTD function example: aggregated sales of last 7 months.

	Unit sales_MTD_0	Unit sales_MTD_1	Unit sales_MTD_2	Unit sales_MTD_3	Unit sales_MTD_4	Unit sales_MTD_5	Unit sales_MTD_6	Unit sales_MTD_7
1	0.00	0.00	0.00	0.00	0.00	22,542.50	45,067.00	68,234.00

Fig. 5.222: MTD function example: output of the aggregated sales of last 7 months.

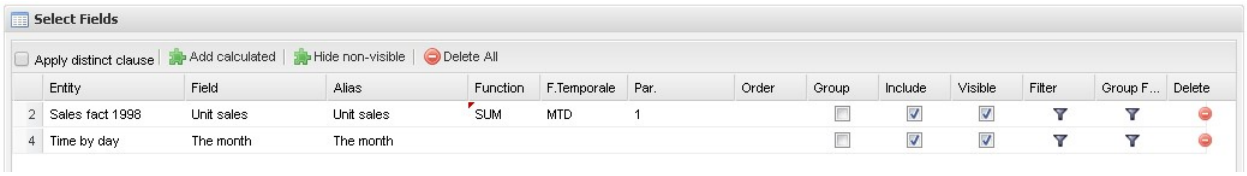


Fig. 5.223: MTD function example: sales are aggregated on current month plus the previuos one, relative to the current year.

	The month	Unit sales_MTD_1
1	January	23,156.50
2	February	45,372.00
3	March	45,382.50
4	April	45,691.50
5	May	45,067.00

Fig. 5.224: MTD function example: output when sales are aggregated on current month plus the previuos one, relative to the current year (referring to the query execution time) for the present year.

Fig. 5.225: MTD function example:sales of current month for the present year.

	The month	Unit sales_MTD_0
1	January	23,156.50
2	February	22,215.50
3	March	23,167.00
4	April	22,524.50
5	May	22,542.50

Fig. 5.226: MTD function example: output of the sales of current month for the present year.

Fig. 5.227: MTD function example:sales of current month for the present year.

	The month	Unit sales_MTD_0	Unit sales_MTD_1
1	January	23,156.50	23,156.50
2	February	22,215.50	45,372.00
3	March	23,167.00	45,382.50
4	April	22,524.50	45,691.50
5	May	22,542.50	45,067.00

Fig. 5.228: MTD function example: output of the sales of current month for the present year.

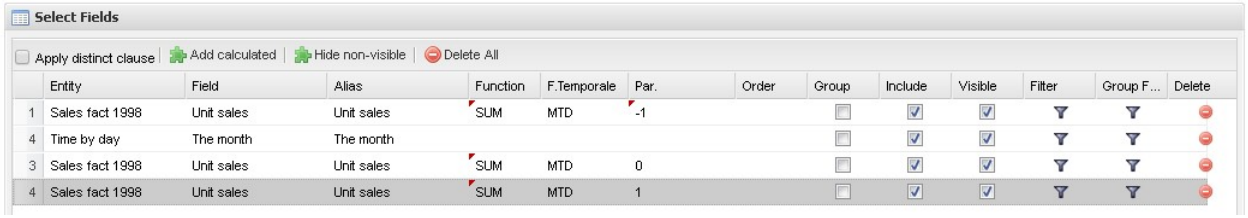


Fig. 5.229: MTD function example: sum of sales of months up to now plus next month.

	The month	Unit sales_MTD_0	Unit sales_MTD_1	Unit sales_MTD_-1
1	January	23,156.50	23,156.50	45,372.00
2	February	22,215.50	45,372.00	45,382.50
3	March	23,167.00	45,382.50	45,691.50
4	April	22,524.50	45,691.50	45,067.00
5	May	22,542.50	45,067.00	22,542.50

Fig. 5.230: MTD function example: output when one sums sales of months up to now plus next month.

- The following figures shows a case very similar to the previous one. In this case the reference year is specified through a filtering condition.

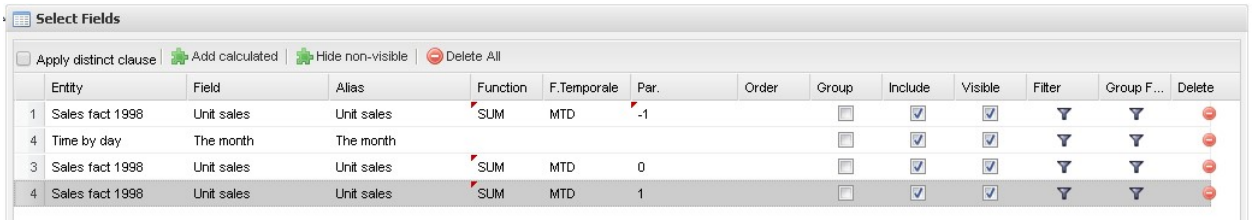


Fig. 5.231: MTD function example: sum of sales of months up to now plus next month for a different year.

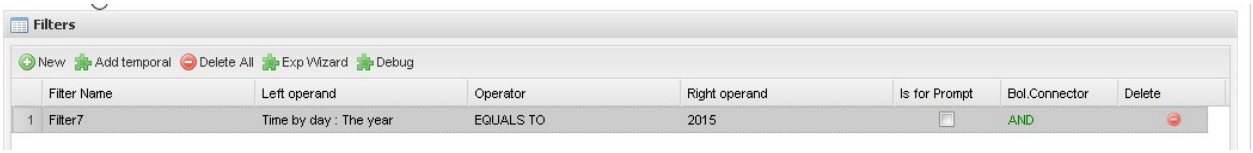


Fig. 5.232: MTD function example: output when one sums sales of months up to now plus next month for a different year.

5.7.1.2.2.6 Catalogues

A hidden panel is activated once you click on the arrow on the right side of the QbE editor, right under the **Preview** button. This panel contains two elements:

- the catalogue of queries (at the top);
- the list of analytical drivers linked to the QbE document (bottom).

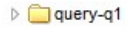
The catalogue of queries is the list of all queries defined in the QbE document, while the lower panel lists all analytical drivers linked to the QbE document.

5.7.1.2.2.7 Queries catalogue and subqueries

Several queries can be built over the same QbE datamart. The catalogue lists all saved queries on the current datamart. The base query that we are creating in the query editor appears with a default name (query-q1): to rename it, simply double click on the query item in the catalogue tree.

To create a new query, click the icon . The query appears in the catalogue at the same level as the base query. Using the query editor you can create the query and save it.

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 . The query appears in the catalogue as a child node of the base query.

Once defined the main query and the filter that contains the subquery, go to the **Query Catalogue** panel and click on 

. The query appears in the catalogue as a child node of the base query.

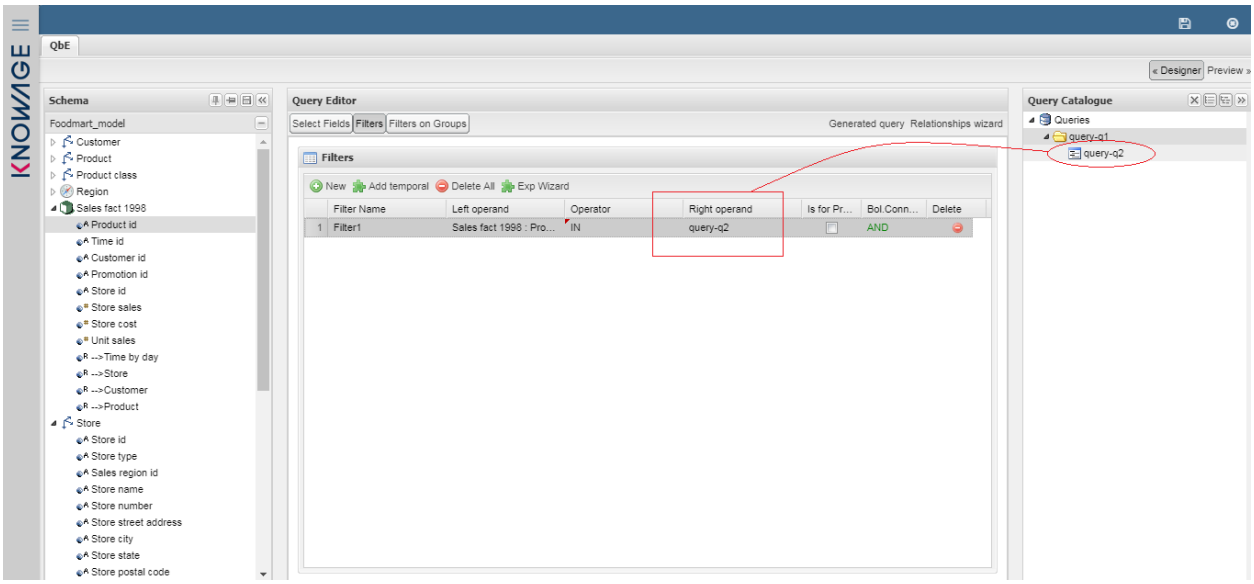


Fig. 5.233: QbE query: use of a subquery in a filter.

To use the sub-query inside the main query, simply drag and drop it into the columns corresponding to the left or right operand of the filter and set the type of operand (**IN** or **NOT IN**). Now the subquery is used to provide values within the filter, in a similar way to SQL subqueries.

5.7.1.2.3 Multiple relationships

The QbE includes a specific feature to thoroughly manage relationships among entities: users can create join paths from one table to another to be used in case of ambiguity. Let's see in detail how it works through an example.

Using the schema and data model represented in figure above, suppose you have a model with the following relationships:

- **Store** - **Region**;
- **Customer** - **Region**;
- **Sales Fact** - **Store**;
- **Sales Fact** - **Customer**.

Ambiguity arises when attributes coming from the various tables are dragged and dropped into the query that is build in the QbE, as in Figure below. In this case, in order to identify the items sold by region, you may have one of the following join relationships:

- **Sales Fact - Customer - Region**,
- **Sales Fact - Store - Region**,

By clicking on the **Relationship Wizard** button in the top right corner of the query editor a pop-up window appears, where users can define the path as shown below.

The images of figure above show the double relationship between **Sales Fact** and **Region**, specifically:

- relationship between **Sales Fact** and **Customer**;
- relationship between **Sales Fact** and **Store**.

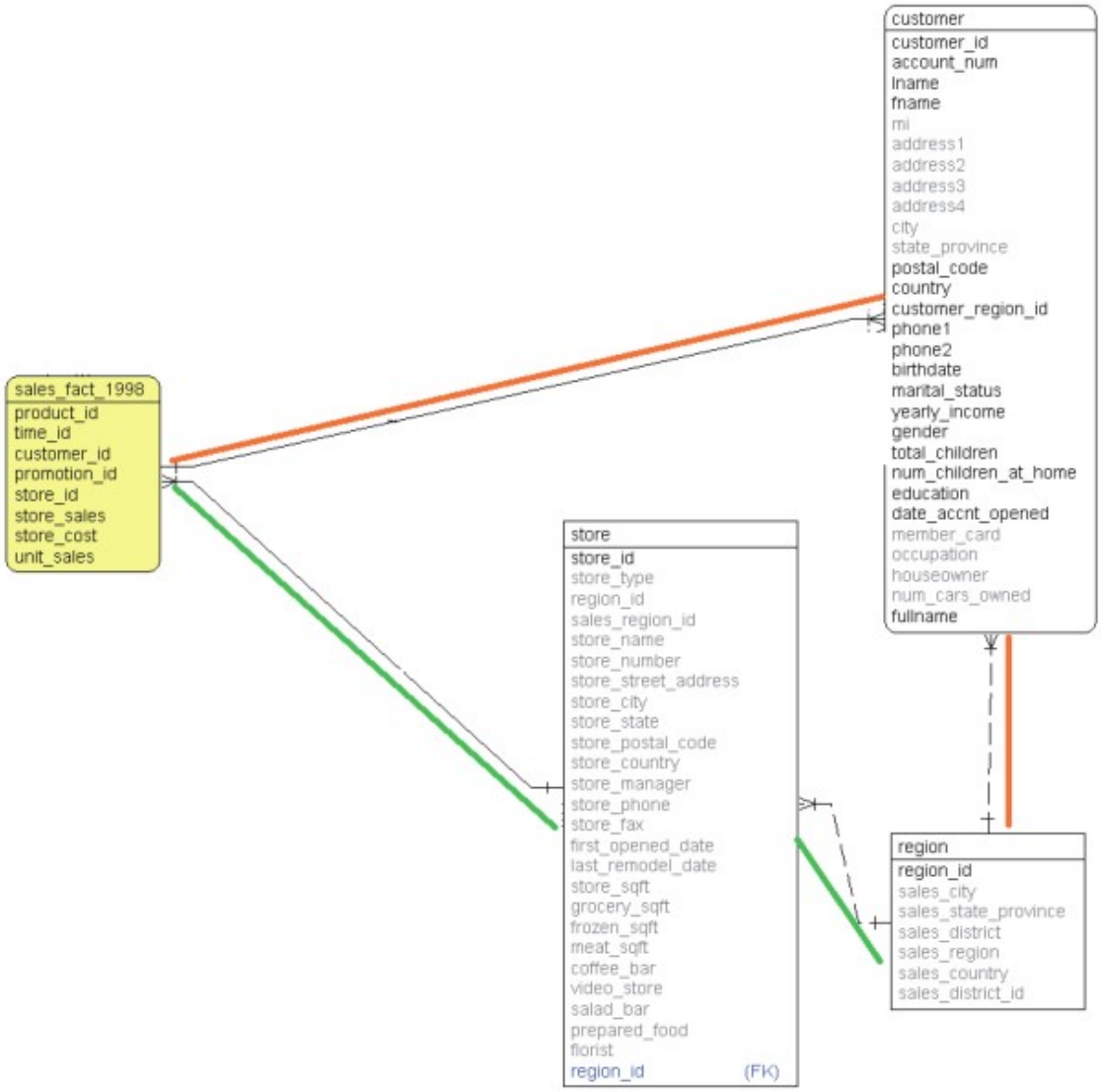


Fig. 5.234: Relationships ambiguity - Schema.

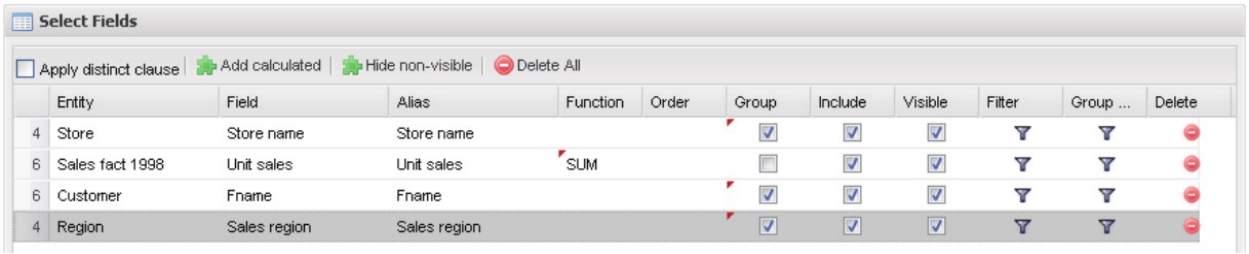


Fig. 5.235: Relationships ambiguity - Query definition.

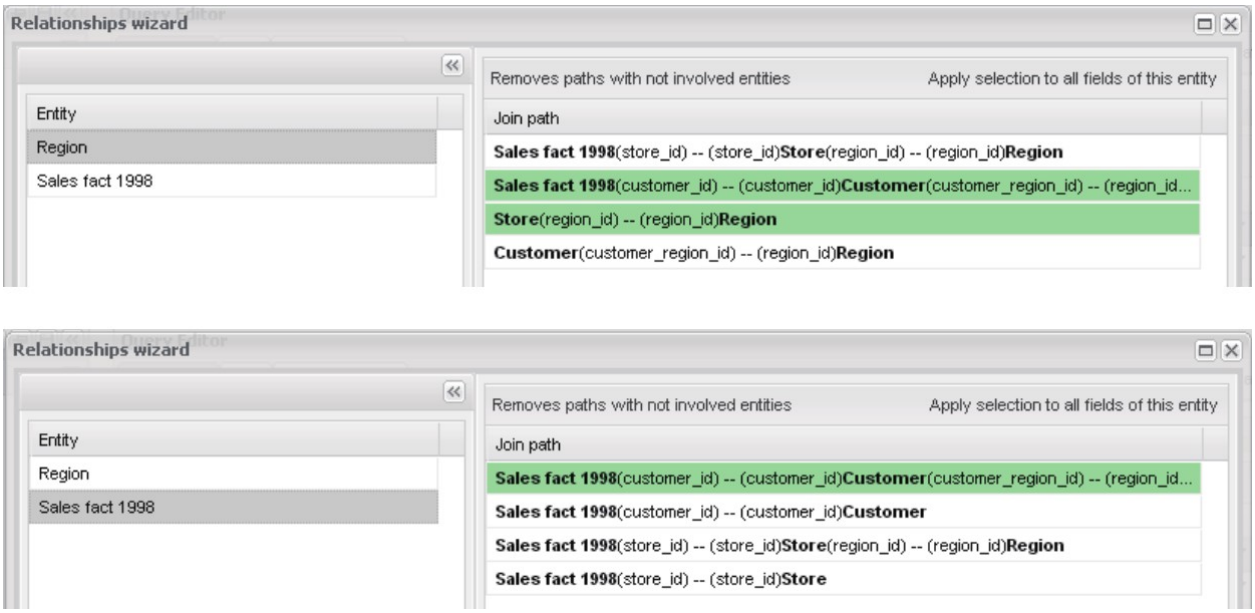


Fig. 5.236: Relationship wizard.

At this point, you can modify the relationship so as to eliminate ambiguity: for instance, if you wish to view the region related to a specific customer, first select the **Region** entity in the **Entity** panel on the left and double click the correct path in the panel on the right (the correct path and only the correct path has to be green-colored to be correctly selected).

Remember to repeat this operation for all the entities listed in the **Entity** panel: now select the **Sales Fact** table and the correct path. If a wrong path is selected (green background), double click on the corresponding row to de-select it. The new configuration is shown below.

Once you are done, you can check the SQL code generated by the QbE query by clicking the Generated Query button. The relationship between Customer and Region is highlighted in bold, as shown below.

5.7.1.2.4 Aliases and relationships

If the data model includes various relationships between two tables, the QbE allows users to manage them using aliases.

To describe this feature, it is worth using an example. Suppose there is a double relationship between **Promotion** and **Time by Day** entities (see the figure below)).

The two relationships concern the start date and end date of the promotion. As shown in the figure above, this information can be retrieved from the QbE graphical interface. The **Promotion** entity includes two relationships (see points 1 and 2 in the figure), whose tooltip returns information on how the relationship is structured (see point 3).

If you wish to see the list of promotions with a specific start date and end date, it is necessary to drag and drop the **Name** of the promotion (from the **Promotion** entity) and the **The Date** field (from the **Time by Day** entity) two times by changing the alias, as well as the name of the column to be visualized in the results of the query (see points 4 and 5).

By executing the query, you will see that in the absence of specific indications, the system selects two relationships (path) at random.

As mentioned in the previous paragraph, by opening the relationships wizard, users can see the list of entities relating to various paths, as well as the list of paths involving various entities. If you wish to use both relationships (end date

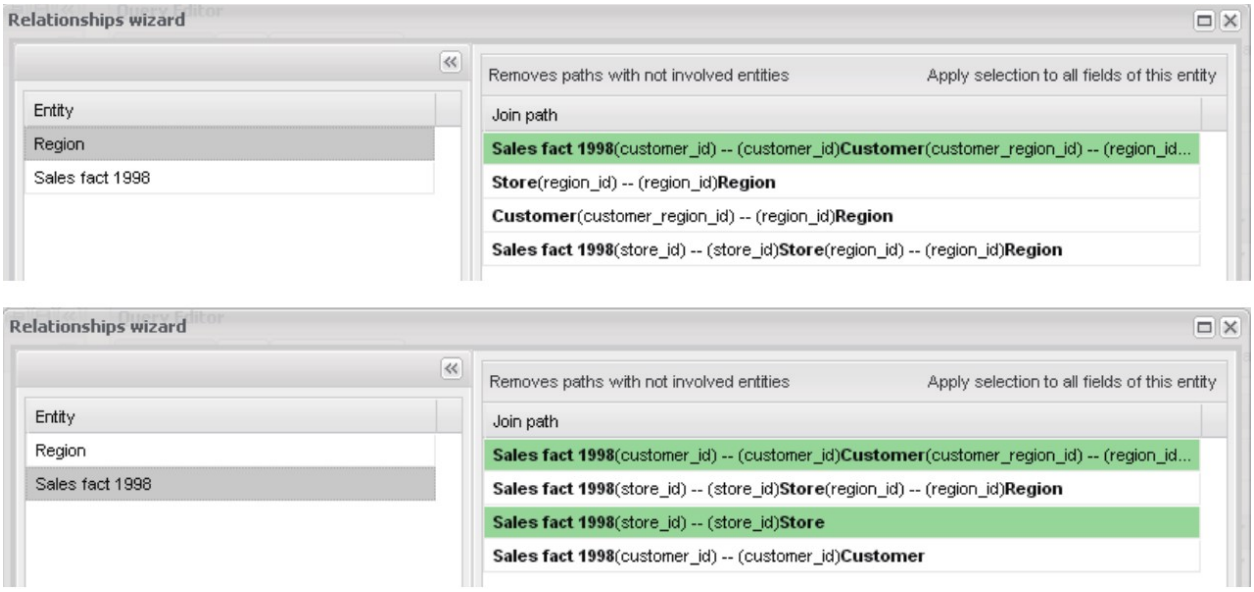


Fig. 5.237: Relationship Wizard - Choosing paths.

```
select
    customer0_.`fname` as `Fname`,
    store1_.`store_name` as `Store_name`,
    region2_.`sales_region` as `Sales_region`,
    sum(sales_fact3_.`unit_sales`) as `Unit_sales`
from
    `foodmart_key`.`customer` customer0_ cross
join
    `foodmart_key`.`store` store1_ cross
join
    `foodmart_key`.`region` region2_ cross
join
    `foodmart_key`.`sales_fact_1998` sales_fact3_
where
    sales_fact3_.`customer_id`=customer0_.`customer_id`
    and customer0_.`customer_region_id`=region2_.`region_id`
    and sales_fact3_.`store_id`=store1_.`store_id`
group by
    customer0_.`fname` ,
    store1_.`store_name` ,
    region2_.`sales_region`
```

Fig. 5.238: Generated query.

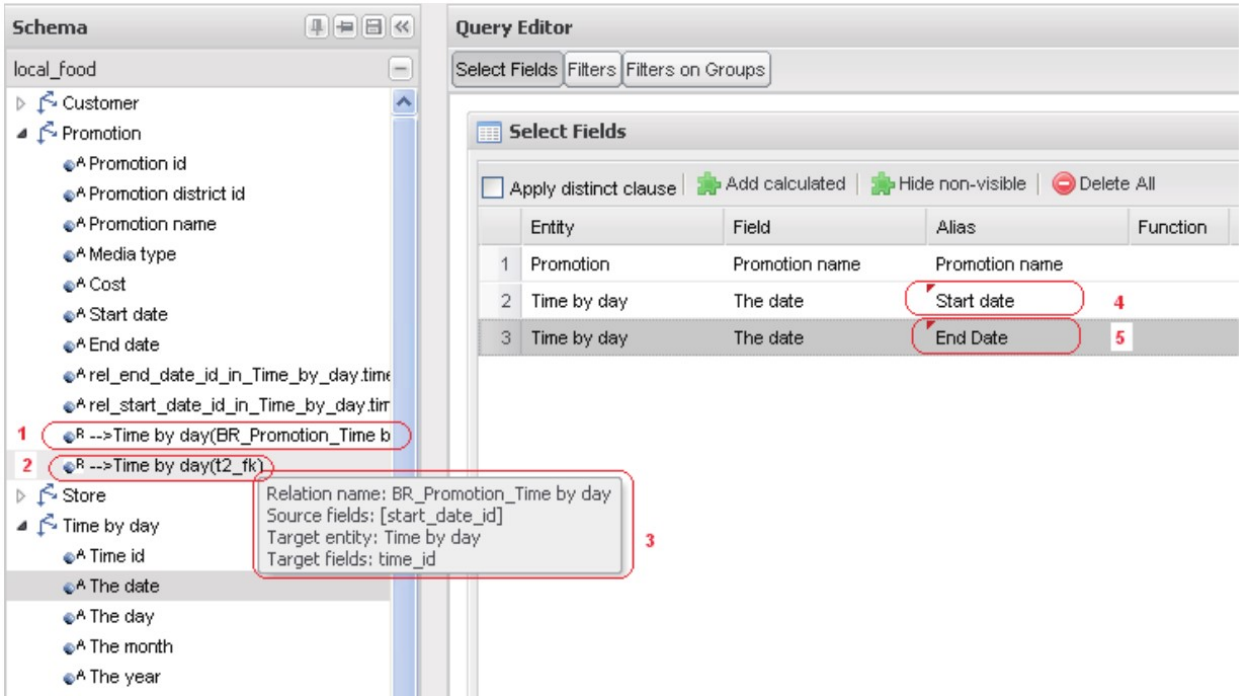


Fig. 5.239: Double relationships.

and start date), select both as shown below. The tooltip shows the complete path using an intuitive tree layout.

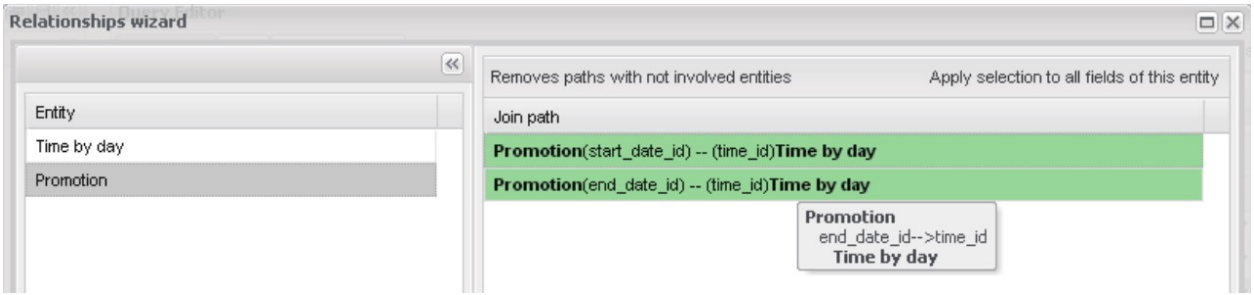


Fig. 5.240: Relationship wizard - Double relationships (I).

Once the relationships are selected in both entities, click on **Apply**.

The window shown below will appear.

It includes three sections:

1. List of aliases: the first column on the left contains the different entity fields;
2. List of fields associated to the entities: here you can set the associations between aliases and entity fields;
3. List of fields: the first column on the right contains the aliases that you previously defined in the query, and corresponds to the columns that you expect to be shown in the resulting table.

To distinguish the fields during the execution of the query, it is necessary to identify all the fields involved in the query (included in the third section List of fields) with the aliases of the entities that contain them (included in the first section List of aliases).

In this case, select the **Time by day (rel BR_Promotion_..)** entity in the first column, then drag and drop the **End**

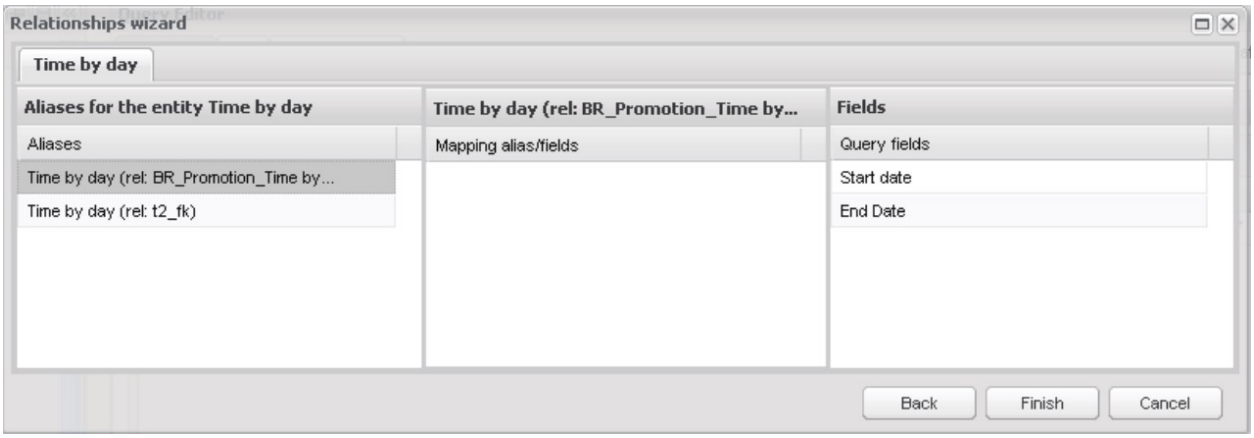


Fig. 5.241: Relationship wizard - Double relationships (II).

Date field from the third column to the one in the middle. Repeat the same with the **Time by day (rel: t2_fk)** entity and the **Start date** field.

The results are shown below. To check whether the association was correctly set, you can refer to the relationship specified in the tooltip.

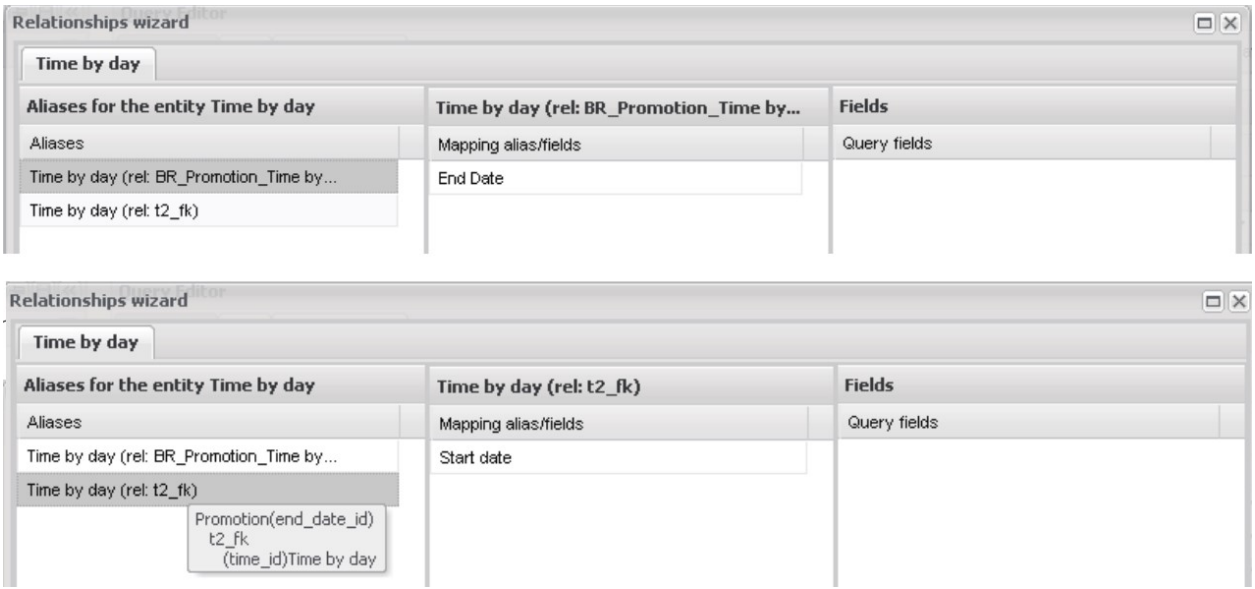


Fig. 5.242: Relationship wizard - Double relationships (III).

Click **Finish** and check the SQL code clicking **Generated Query**. Figure below shows the desired result.

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.

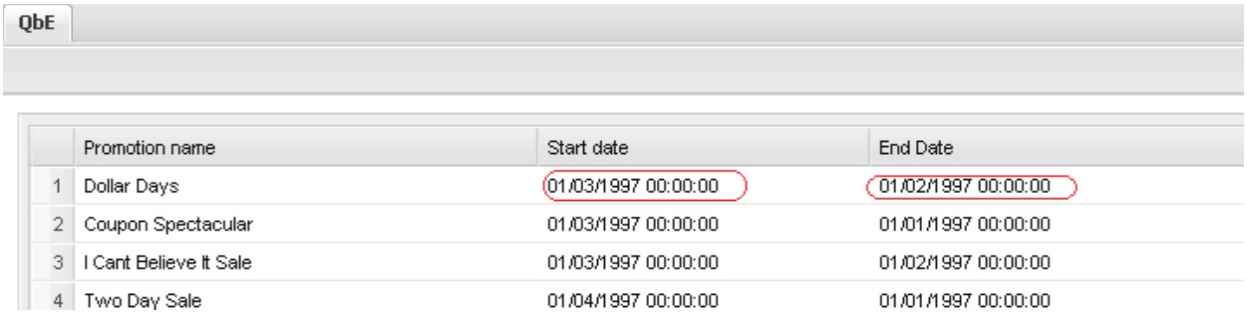


Fig. 5.243: Double relationship preview.

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

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.

5.8.1.1 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.

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

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

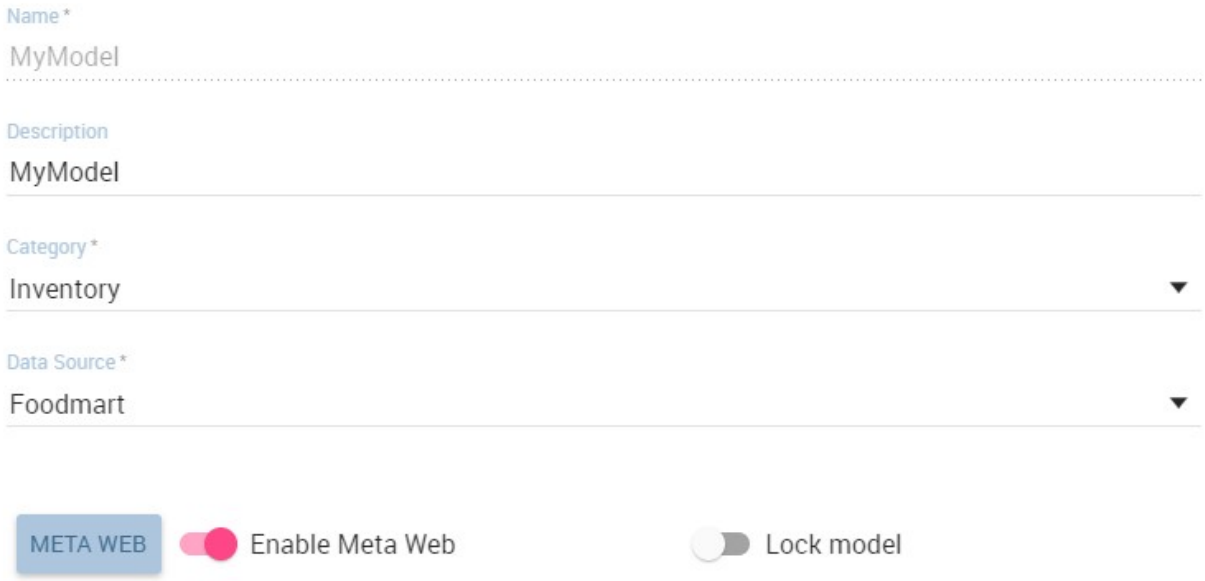


Fig. 5.244: Setting the metamodel basic information.

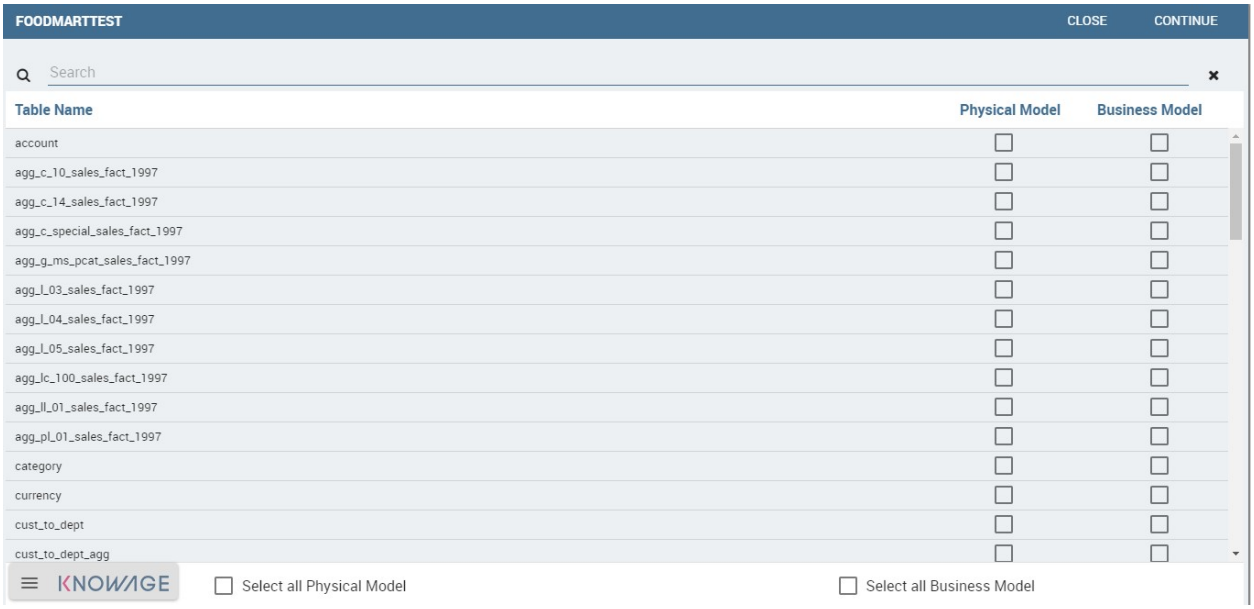


Fig. 5.245: Metaweb interface.

If you choose to include a table only in the physical model is always pssible 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.

5.8.1.2 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).

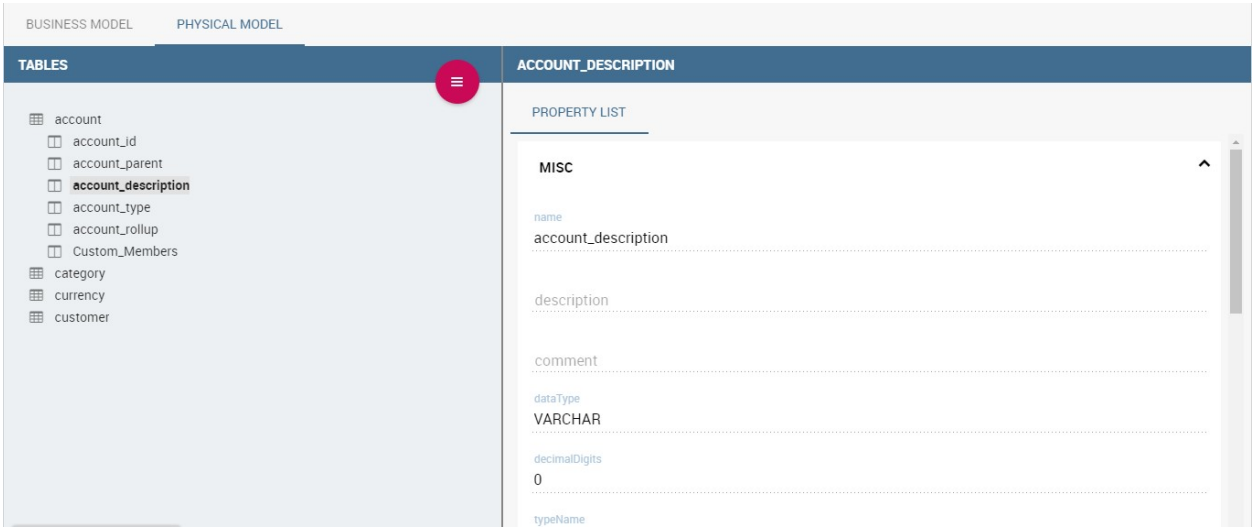


Fig. 5.246: Physical Model Tab.

The "hambuger-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.

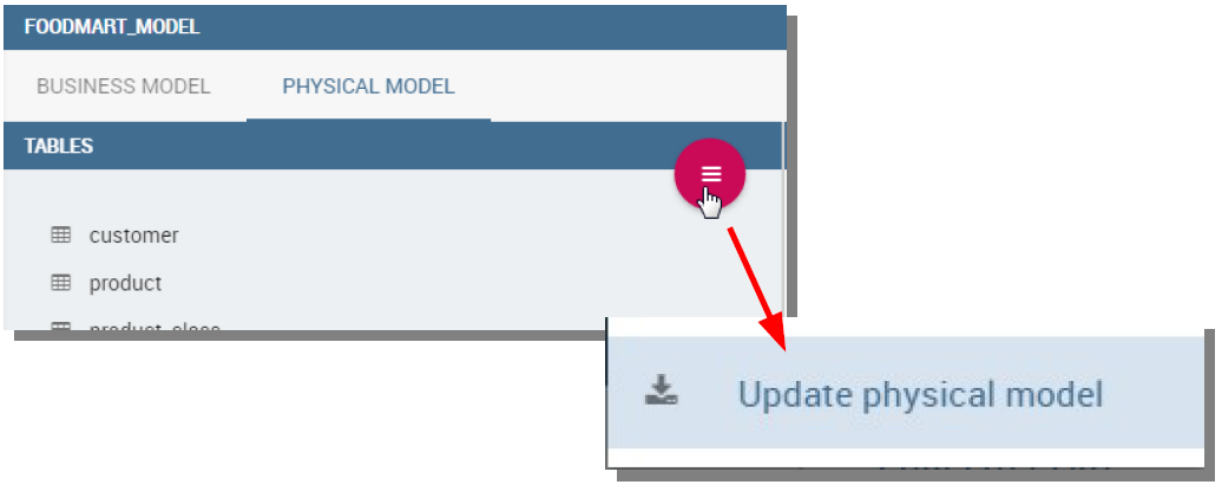


Fig. 5.247: Update the physical model.

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.

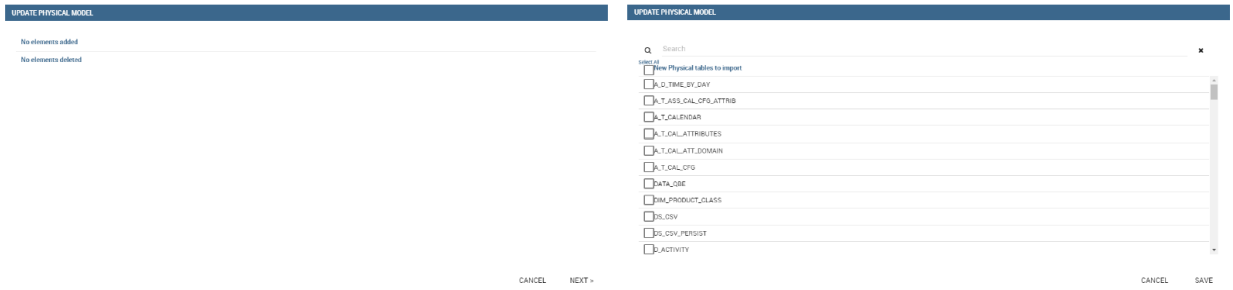


Fig. 5.248: Update the physical model.

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.

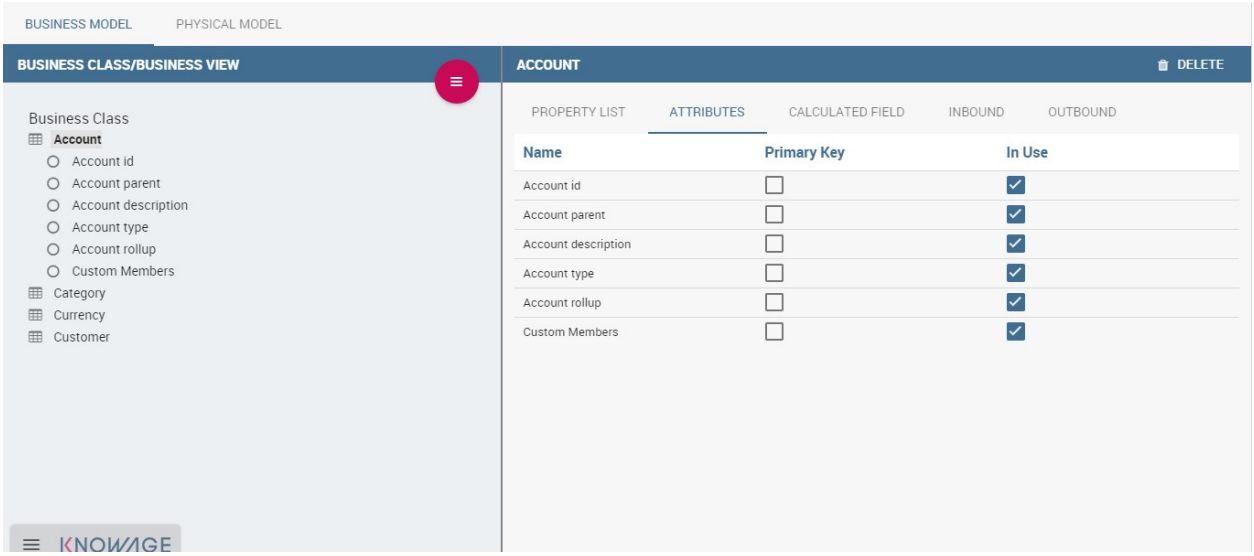


Fig. 5.249: Physical Model Tab.

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).

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 spatia 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 istance, it is not possible

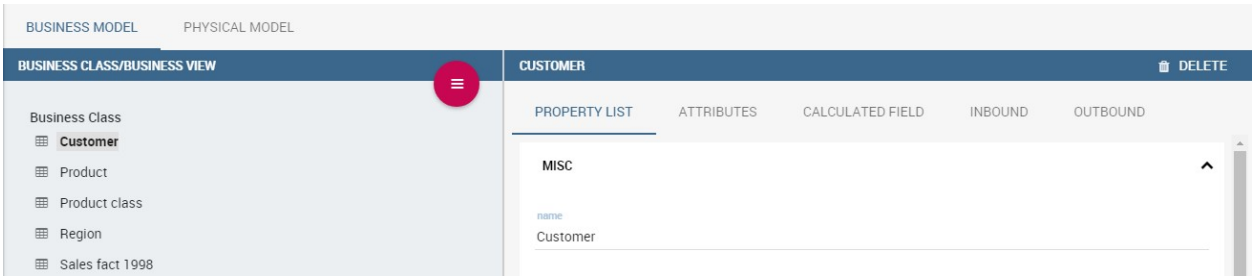


Fig. 5.250: Exploring Business Class properties.

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.

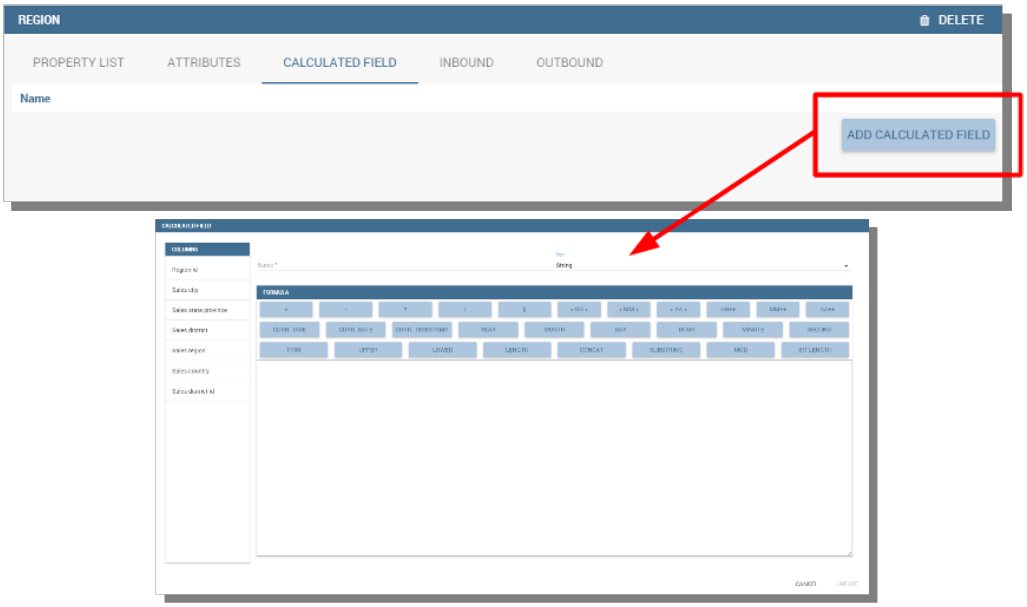


Fig. 5.251: Add calculated fields.

The **Inbound** and **Outbound** tabs are described in the following section.

5.8.1.3 Create a new relationship

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

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

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.

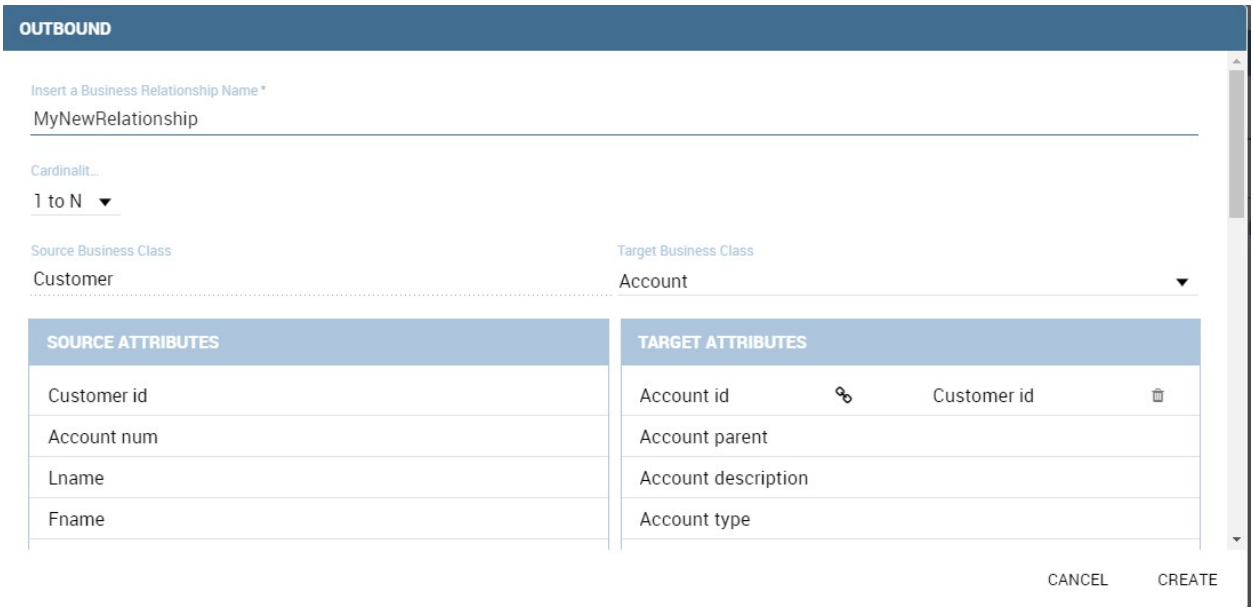


Fig. 5.252: Setting the outbound relationship.

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.

5.8.1.4 Create a new business class

In the “Business Model” tab, the sandwitch 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:

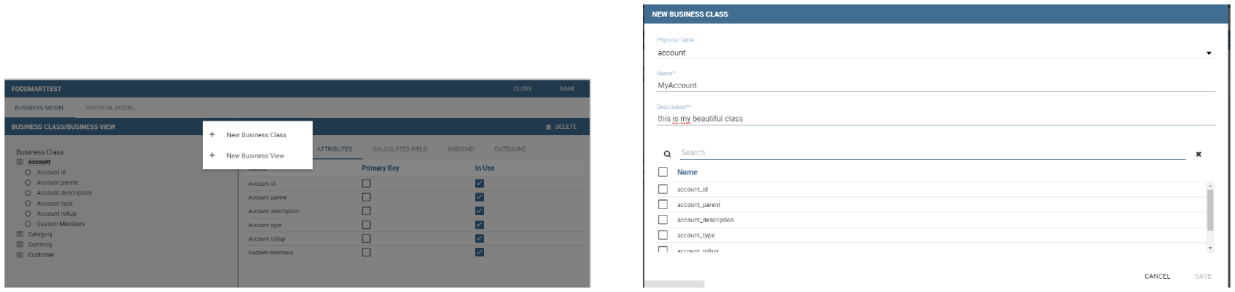


Fig. 5.253: Create a new business class.

- 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.

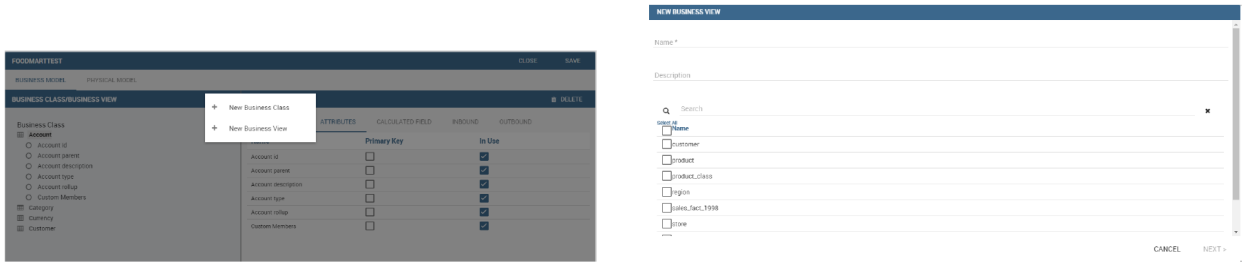


Fig. 5.254: Create a new business view.

Then, moving to the next step, the user must join tables through specific columns, typically the tables' foreign keys. Figure below shows an example.

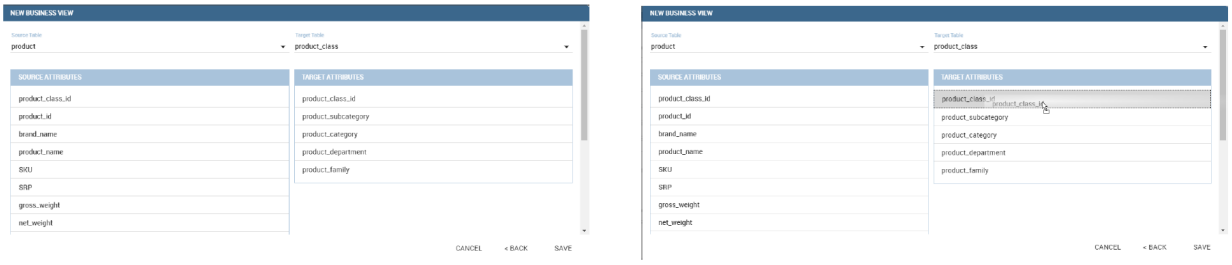


Fig. 5.255: Create a new business view.

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.

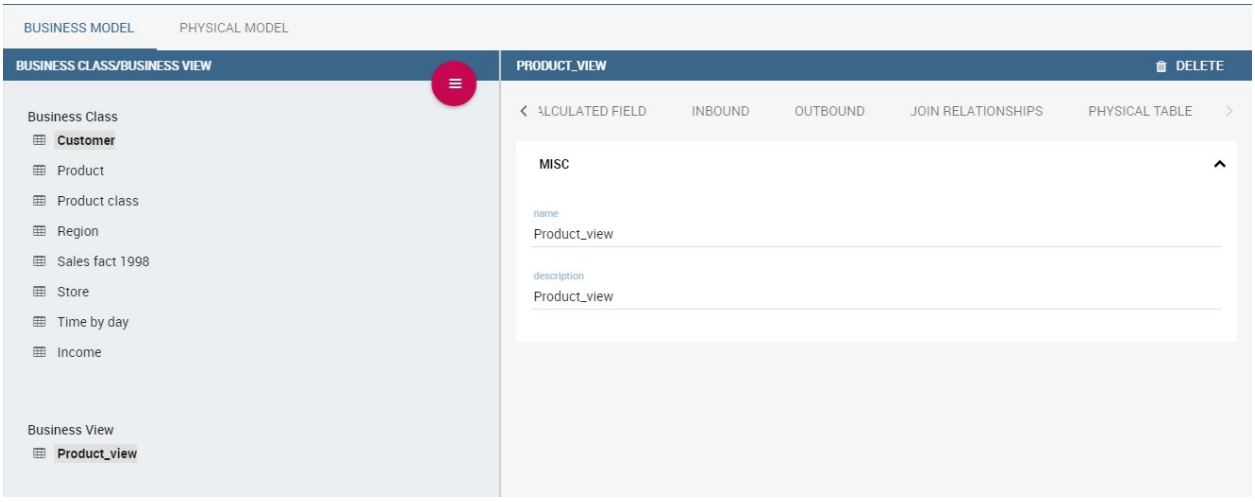


Fig. 5.256: Additional property tabs for business view.

5.8.1.5 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.

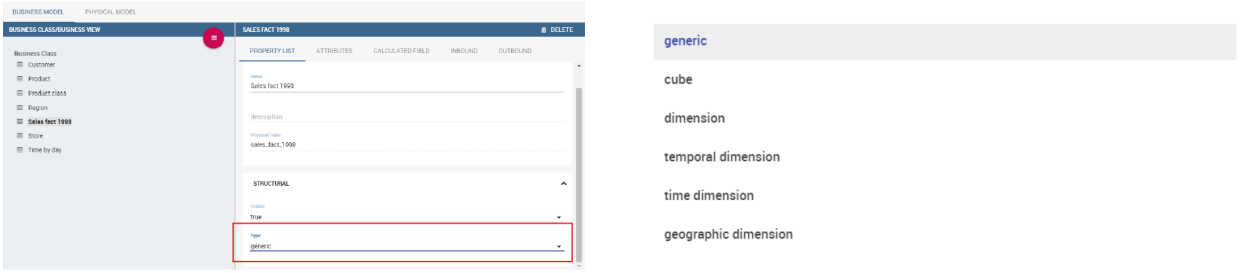


Fig. 5.257: Table property list.

We highlight that for temporal and time dimension type, the user must define at least one hierarchy for the related dimension. the next figure shows that it is possible to set a hierarchy clicking on the button avaiable at the end of the comobobox line. Then, clicking on the "Add" button, the user can configure a new hierarchy. Remember to choose the **Level type** using the dedicated combobox. Note that it is possibile to shift levels using the up and down arrows avaiable at the end of each level row. Furthermore, the user can add a "Has all" node the hierarchy: just enable the "has-all" box and type the all member name.

According to the chosen type, the user will be able to use the specific functions introduced in Section **XXX**.

5.8.1.6 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:

- **Visibility** over the field,
- **Type**, among measure, attribute, calendar, temporal_id, the_date and hour_id,

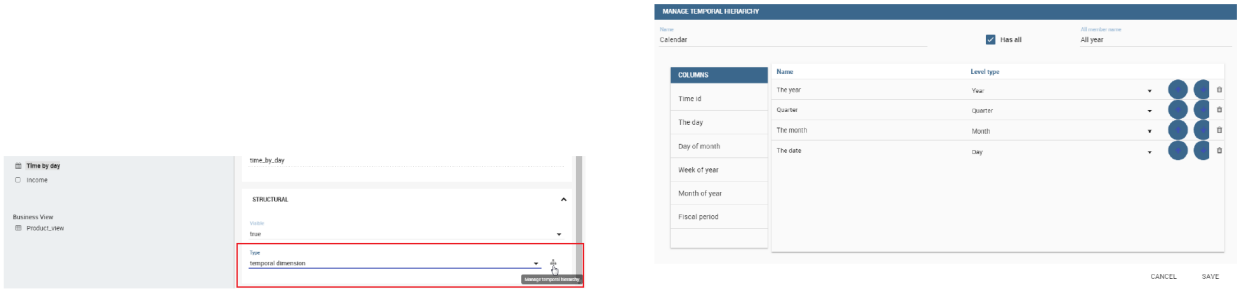


Fig. 5.258: Temporal hierarchy definition.

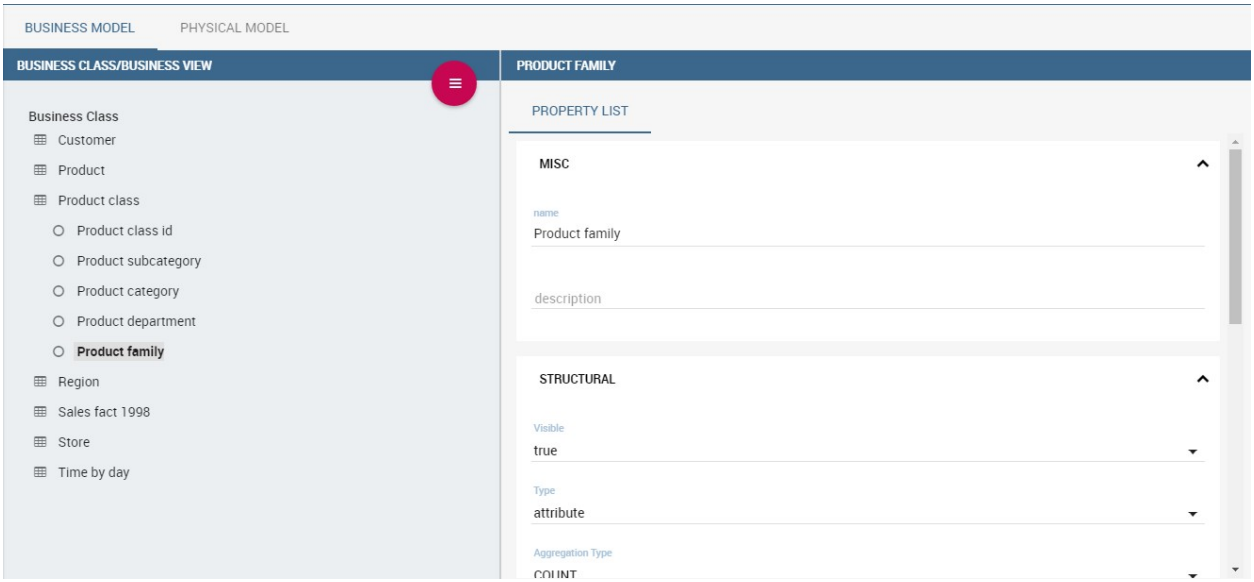


Fig. 5.259: Column property list.

- **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 visbility permission to specific roles.

In the **Physical** area, recalls the physical table and field name from which the field have been take.

5.8.1.7 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:

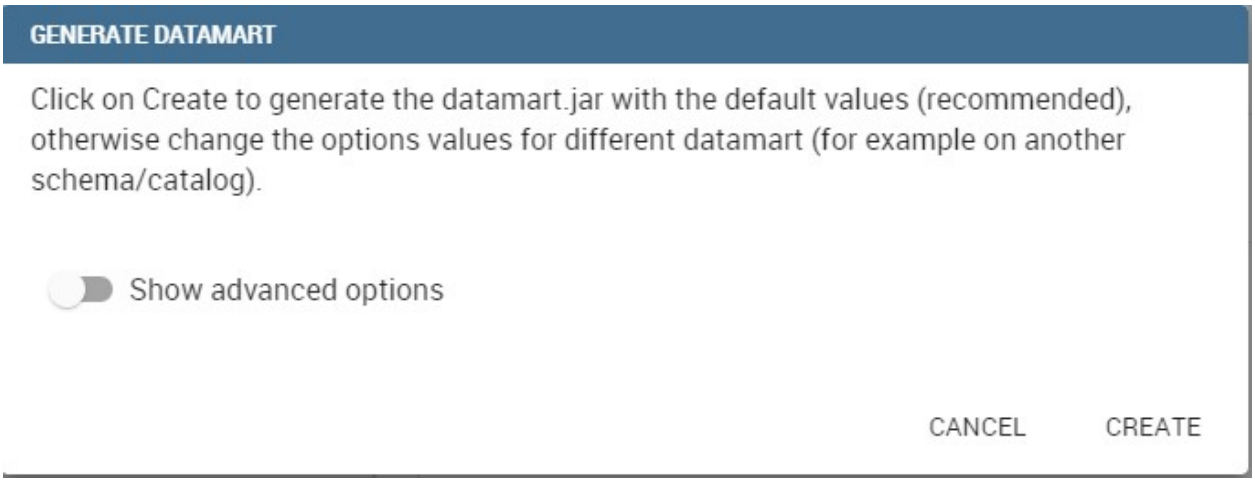


Fig. 5.260: Generate datamart dialog.

If you just press "Create" the generation of the datamart begins otherwise clicking on the switch "Show Advacend options" (see fegure 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 buid 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 debagging process.

When the datamart is generated it will be possible to query the metamodel accessing it in the Workspace interface.

5.8.1.8 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). The **Export CWM** allows the user to export metadata

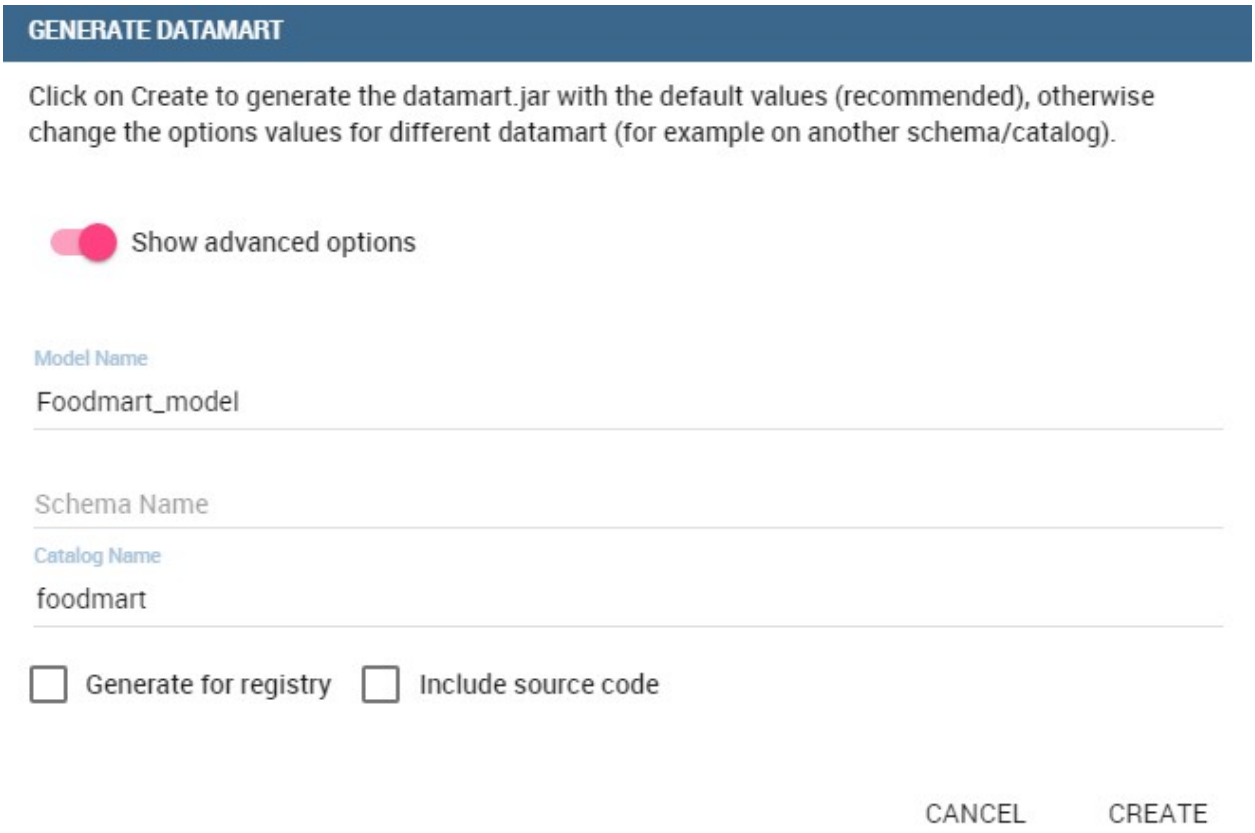


Fig. 5.261: Generate datamart dialog: advanced options.

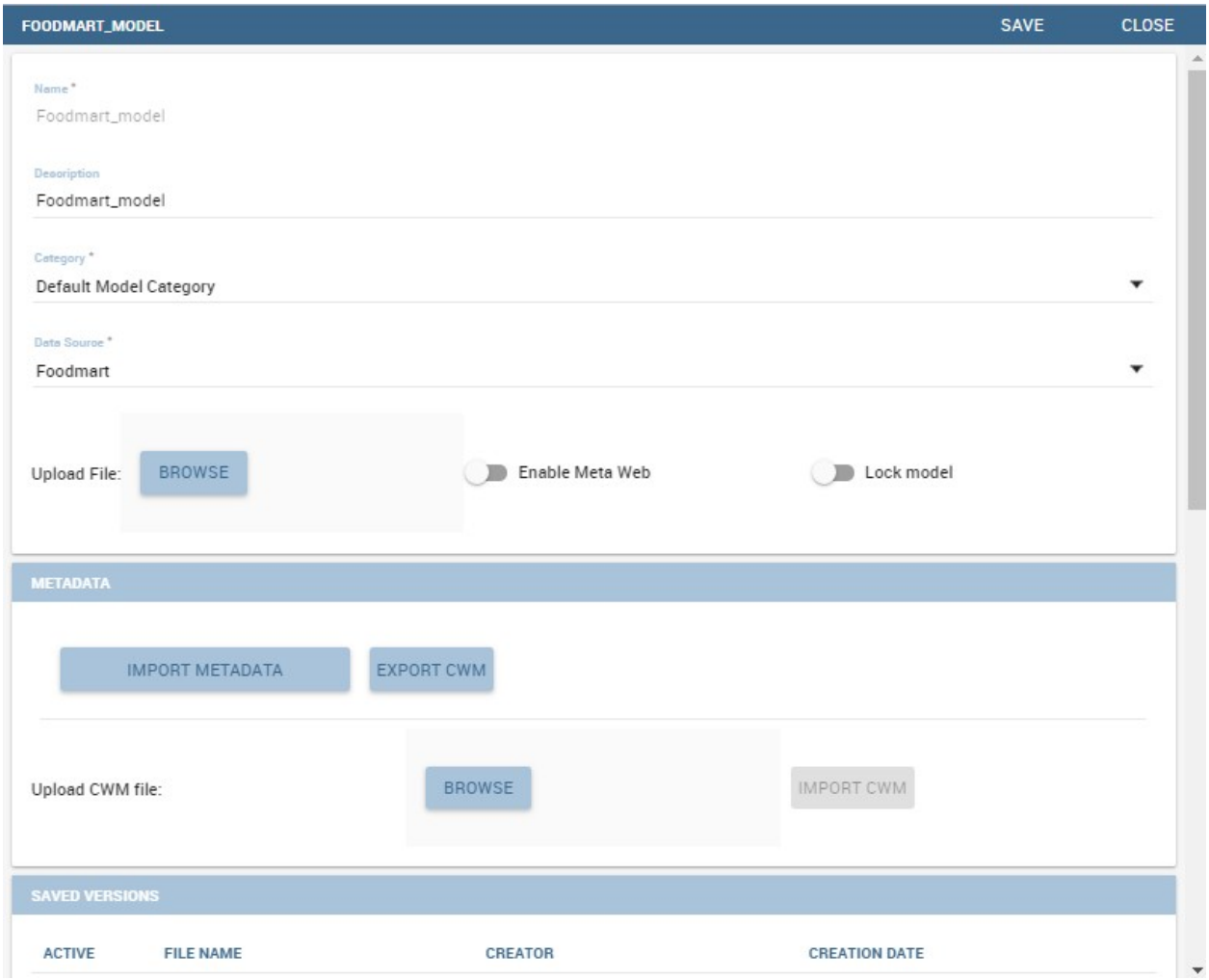


Fig. 5.262: Additional functions for business model.

information in the CWM format. Vice versa the **Import CWM** allows the user to import a CWM file containing metadata infomration. The user must browse the CWM file into personal folder and then click on the "Import CWM" button to correctly upload it. Remember to save to validate the process.

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 belowshows an example.

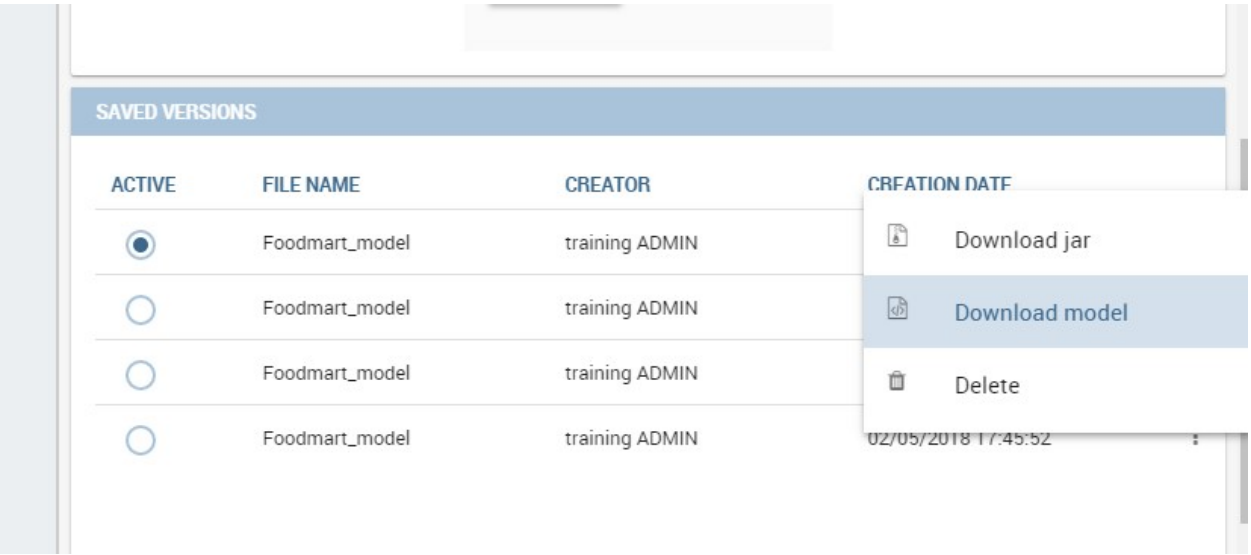


Fig. 5.263: Saved version functionalities.

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 table. Just double-click on a cell you may wish to rearrange and type a string or a numeric value accordingly. Some examples are highlighted below.

Moreover, you can add new rows from scratch selecting the "Plus" icon on the left of the functionality bar (see next figure) available at the top of the window. Insert in each cell the corresponding value: string or number.

Vice versa, you can delete one or more rows using the "Minus" icon of the functionality bar (Fig. 5.266) available at the top of the window.

It is important to click on the Save button to store the adjustments done in the datamart.

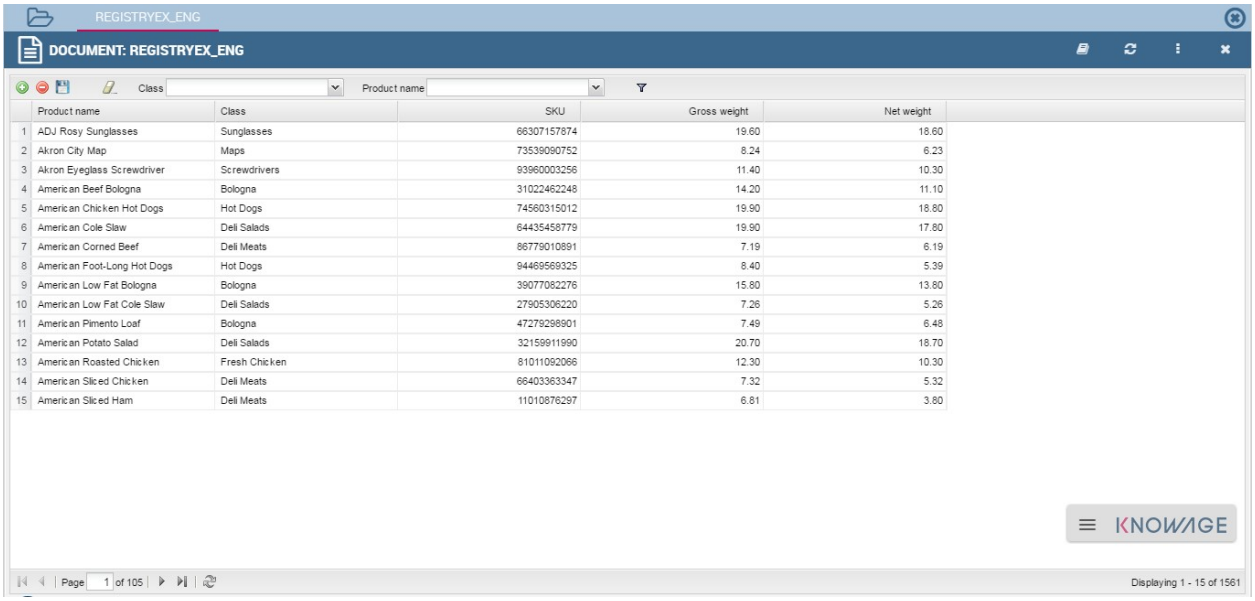


Fig. 5.264: Example of Registry document.

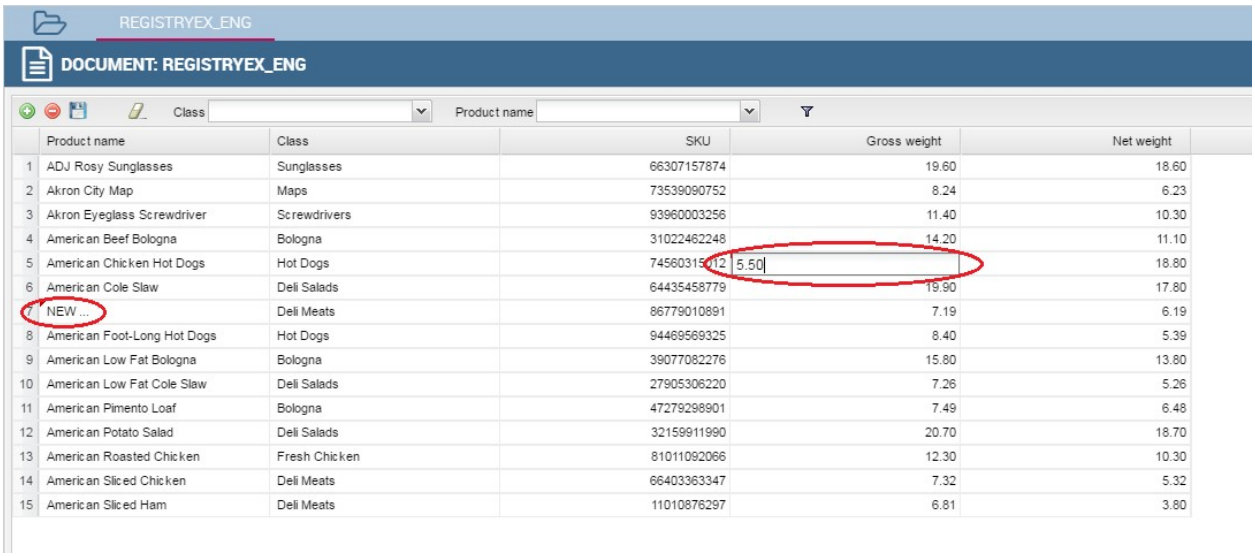
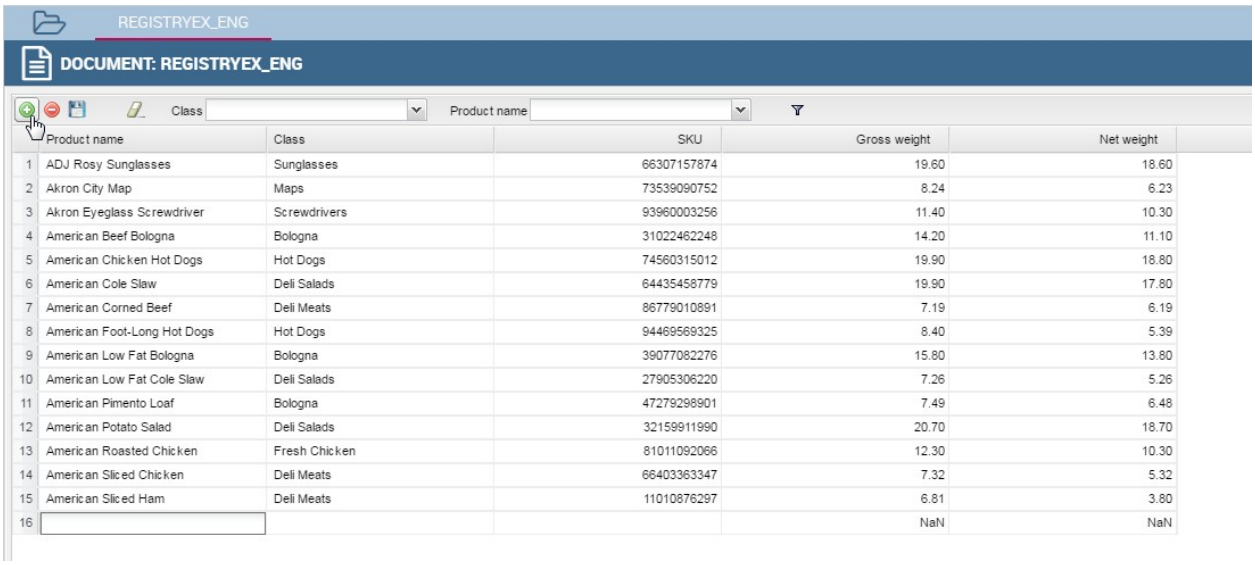


Fig. 5.265: Editing table cells.

Fig. 5.266: Functionality bar.

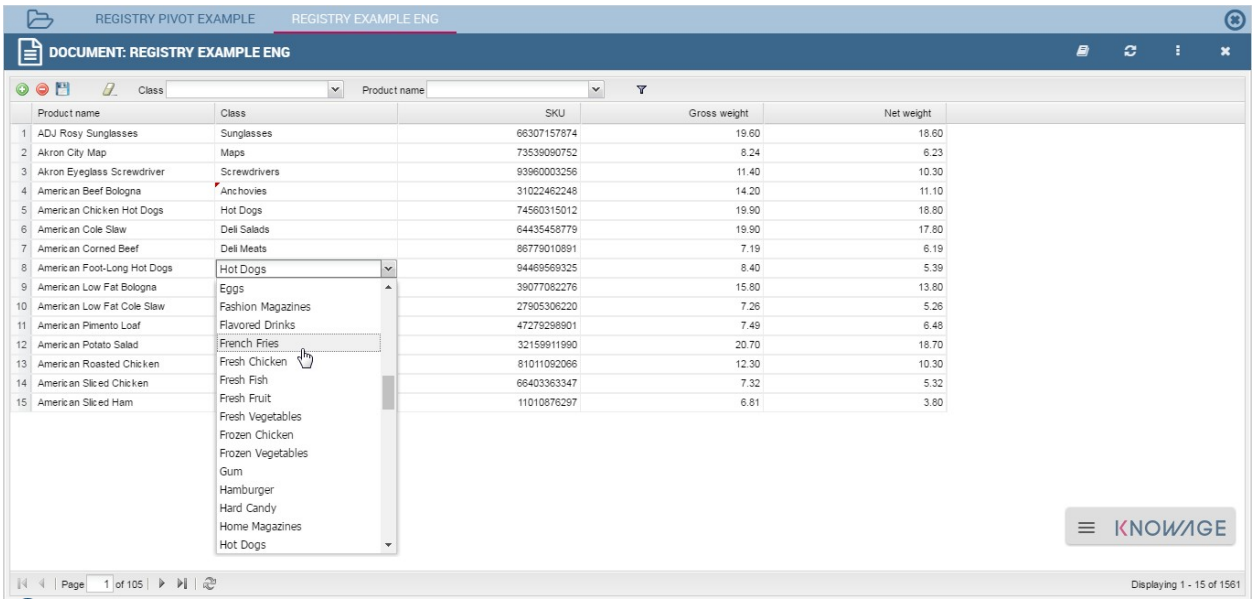


	Product name	Class	SKU	Gross weight	Net weight
1	ADJ Rosy Sunglasses	Sunglasses	66307157874	19.60	18.60
2	Akron City Map	Maps	73539090752	8.24	6.23
3	Akron Eyeglass Screwdriver	Screwdrivers	93960003256	11.40	10.30
4	American Beef Bologna	Bologna	31022462248	14.20	11.10
5	American Chicken Hot Dogs	Hot Dogs	74560315012	19.90	18.80
6	American Cole Slaw	Deli Salads	64435458779	19.90	17.80
7	American Corned Beef	Deli Meats	86779010891	7.19	6.19
8	American Foot-Long Hot Dogs	Hot Dogs	94469569325	8.40	5.39
9	American Low Fat Bologna	Bologna	39077082276	15.80	13.80
10	American Low Fat Cole Slaw	Deli Salads	27905306220	7.26	5.26
11	American Pimento Loaf	Bologna	47279298901	7.49	6.48
12	American Potato Salad	Deli Salads	32159911990	20.70	18.70
13	American Roasted Chicken	Fresh Chicken	81011092066	12.30	10.30
14	American Sliced Chicken	Deli Meats	66403363347	7.32	5.32
15	American Sliced Ham	Deli Meats	11010876297	6.81	3.80
16				NaN	NaN

Fig. 5.267: Adding a new row.

Furthermore you can use filters, if implemented, of the functionality bar. Pick one field out of the combobox. 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.



	Product name	Class	SKU	Gross weight	Net weight
1	ADJ Rosy Sunglasses	Sunglasses	66307157874	19.60	18.60
2	Akron City Map	Maps	73539090752	8.24	6.23
3	Akron Eyeglass Screwdriver	Screwdrivers	93960003256	11.40	10.30
4	American Beef Bologna	Anchovies	31022462248	14.20	11.10
5	American Chicken Hot Dogs	Hot Dogs	74560315012	19.90	18.80
6	American Cole Slaw	Deli Salads	64435458779	19.90	17.80
7	American Corned Beef	Deli Meats	86779010891	7.19	6.19
8	American Foot-Long Hot Dogs	Hot Dogs	94469569325	8.40	5.39
9	American Low Fat Bologna		39077082276	15.80	13.80
10	American Low Fat Cole Slaw		27905306220	7.26	5.26
11	American Pimento Loaf		47279298901	7.49	6.48
12	American Potato Salad		32159911990	20.70	18.70
13	American Roasted Chicken		81011092066	12.30	10.30
14	American Sliced Chicken		66403363347	7.32	5.32
15	American Sliced Ham		11010876297	6.81	3.80

Fig. 5.268: 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.

Fig. 5.269: 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 5.14: Example (a) of template for Registry.

```
<?xml version="1.0" encoding="windows-1250"?>
<QBE>
  <DATAMART name="RegFoodmartModel" />
     <REGISTRY>
     <ENTITY name="it.eng.spagobi.meta.Product">
         <FILTERS>
           <FILTER title="Class" field="product_subcategory" presentation="COMBO" />
           <FILTER title= "Product name" field="product_name" presentation="COMBO" />
         </FILTERS>

```

(continues on next page)

(continued from previous page)

```
        <COLUMNS>
           <COLUMN field="product_id" unsigned="true" visible="false" editable="false" format="####" />
           <COLUMN field="product_name" title="Product name" size="200"
                   editor= "MANUAL" sorter="ASC"/>
           <COLUMN field="product_subcategory" title="Class" size="200"
                   subEntity="rel_product_class_id_in_product_class" foreignKey="rel_product_class_id_in_
↪product_class" />
           <COLUMN field="SKU" title="SKU" size="200" editor="MANUAL" />
           <COLUMN field="gross_weight" title="Gross weight" size="200" editor="MANUAL" />
           <COLUMN field="net_weight" title="Net weight" size="200" editor="MANUAL" />
        </COLUMNS>

        <CONFIGURATIONS>
          <CONFIGURATION name="enableButtons" value="true"/>
          <CONFIGURATION name="isPkAutoLoad" value="true"/>
       </CONFIGURATIONS>
    </ENTITY>
  </REGISTRY>
</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 specifing 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 specifing:
  - **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 correlatd to other registry's column is possible specifiy 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 5.15: Pointing at a numerical column.

```
<COLUMNS>
  <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 5.16: Example (b) of template code for Registry.

```
<QBE>
   <DATAMART name="foodmart" />
   <REGISTRY pagination = "false" summaryColor="#00AAAA">
      <ENTITY name="it.eng.spagobi.meta.Store">

         <FILTERS>
            <FILTER title="Store Type" field="store_type" presentation="COMBO" />
         </FILTERS>

         <COLUMNS>
              <COLUMN field="store_id" visible="false" editable ="false" />
              <COLUMN field="store_country" title="store country" visible="true"
                      type="merge" editable ="false" sorter ="ASC" summaryFunction="sum" />
              <COLUMN field="store_state" title="store state" visible="true"
                      type=" merge" editable ="false" sorter ="ASC" />
              <COLUMN field="store_city" title="store city" visible="true"
                      type="merge" editable ="false" sorter ="ASC" />
              <COLUMN field="store_type" title="store type" type="merge" sorter="ASC" />
              <COLUMN field="store_number" title="Number" size="150"
                      editable="true" format="########" color="#f9f9f8" type="measure"/>
         </COLUMNS>

         <CONFIGURATIONS>
            <CONFIGURATION name="enableButtons" value="false"/>
         </CONFIGURATIONS>
      </ENTITY>
   </REGISTRY>
</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 BIRT reporting

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, acronim 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.

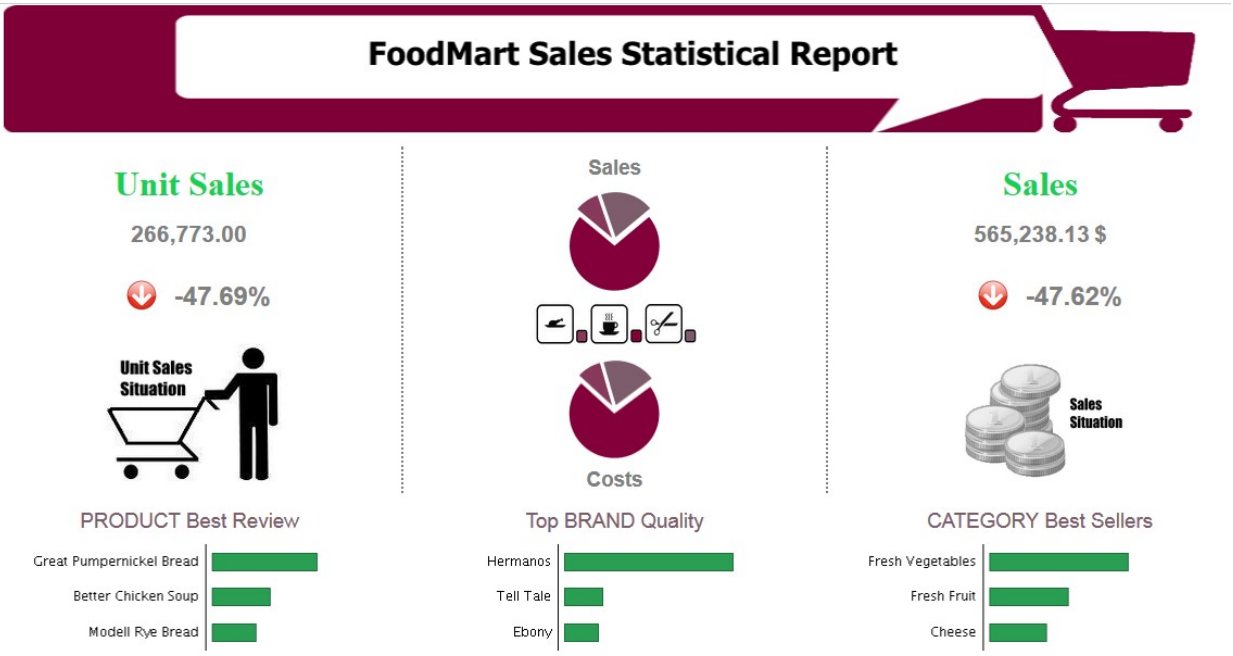


Fig. 5.270: Example of a BIRT report.

Knowage Studio contains BIRT designer while Knowage Server contains the runtime engine. A BIRT template consists of an XML text file with the .rptdesign extension which is automatically generated by the Eclipse platform embedded into Knowage Studio. It can be manually modified by very expert users even though it is not a recommended practice. In this manual we will go into details on how to develop a BIRT report, from its implementation inside the Knowage Studio to its visualization inside the Knowage Server. All standard BIRT functionalities are available in the Studio but for a full overview of BIRT reporting tools and a detailed developer guide, the reader can also refer to the official documentation at http://www.eclipse.org/birt

5.10.1 Introduction to Knowage Studio

Knowage Studio is the development environment based on Eclipse. It allows the developer to design and modify some analytical documents, for instance reports and dashboards. This module supports the developer while designing documents as well as during the installation and testing processes, directly on Knowage Server. The interaction between these two components is possible thanks to Knowage SDK module. The users can display the list of analytical documents which are available on the server and download them, in order to modify them on their own computer.

It allows to build a Knowage project within Eclipse, through which users can create new analytical documents. It also integrates the tools needed to create Jasper and BIRT reports.

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

- it has a JDK 1.7.x already installed;
- it has the JAVA_HOME variable already set;
- it has a certified operative system (Windows, Linux, Unix are generally accepted). The list of certified environments mainly depends on the ones supported by Eclipse. Please, refer to the Knowage Studio release notes to find out the Eclipse version included in each released package.

If so, the second step is to download the suitable package from your personal area in Knowage website.

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

The selected workspace works as a local repository for all the developed objects (reports, charts, cockpits, etc.). However documents are neither visible nor reusable by other users until they are published on Knowage Server.

Warning: Connection between Server and Studio

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

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

These points will be described in the following.

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

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

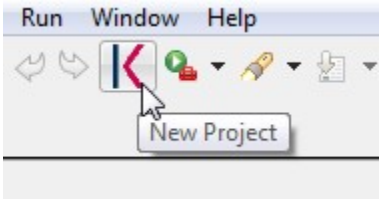


Fig. 5.271: New Knowage project icon.

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

- **Business Analysis:** it contains all developed documents (reports, charts, cockpits, etc.) that can be uploaded into or downloaded from Knowage Server. This folder can be structured with subfolders, to freely organize ones own documents;
- **Private folders:** they contain the users personal or project documentation. This folder can be freely managed by the user;
- **Resources:** they contain all technical resources used in the project (i.e., the reference to a Knowage Server).

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

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

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

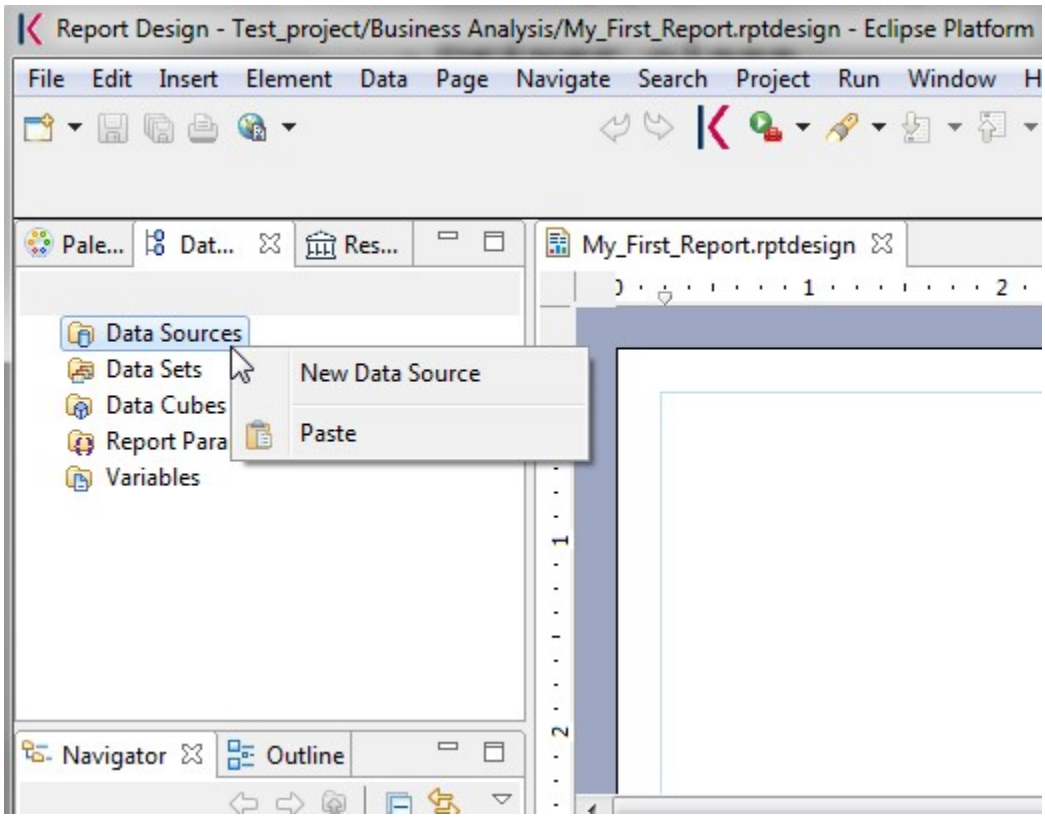


Fig. 5.272: New Data Source connection.

A form will ask you for:

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

Warning: Connection to Knowage Server

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

At this point, Knowage Studio is ready to work!

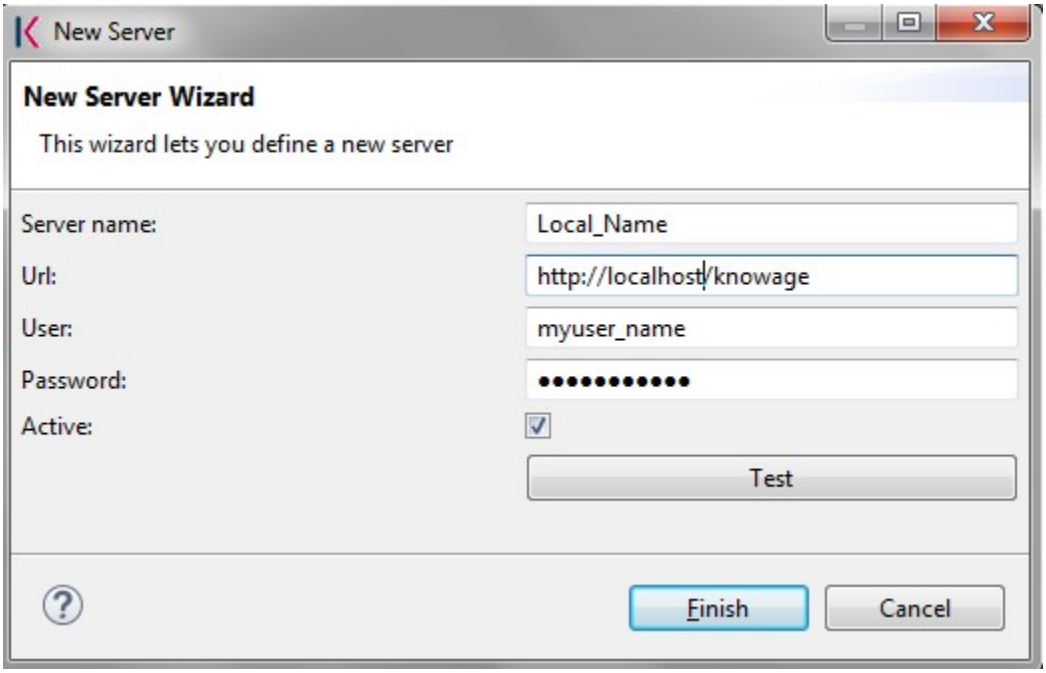


Fig. 5.273: Server configuration wizard.

5.10.1.1 Metadata definition

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

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

Knowage Studio supports BI developers steering the implementation of the template for each analytical document through an easy graphical interface and simple wizards. Each document type has its own designer and manages the relation with data sources and data sets. Furthermore it enhances technical users with all the needed functionalities to design, develop, test, deploy and maintain Knowage analytical documents. As said above, each document is mainly associated to a template describing its layout and a data set defining how data will fill it. Knowage Studio assists the developer in writing these templates and/or data sets by means of a graphical user interface and of easy-to-use wizards.

Warning: Datasets created with the Business Model

These data sets are often based on specific business models created through Knowage Meta. By the way, we will concentrate on how to manage the implementation of a data set using the BIRT Report designer available in Knowage

We want to remark that an expert developer can work directly on the server, managing documents and data sets by hand, thanks to the web interface for administrators and developers. Usually, this procedure is faster when only small changes are required on already released documents, whereas the Studio is particularly useful when a developer works on new documents.

The target users of the Studio module are:

- BI developers, who define analytical documents and data sets to be released onto a remote Knowage Server
- administrators, who define or update analytical documents and data sets.

In other words, Knowage Studio covers the development processes of more technical documents. On the other hand, high-level documents are created directly through Knowage Server, where a power user can access graphical designers without need to use the Studio, which requires more technical skills to manage the installation and configuration process.

5.10.1.2 Data set definition

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

5.10.2 Developing a BIRT report

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

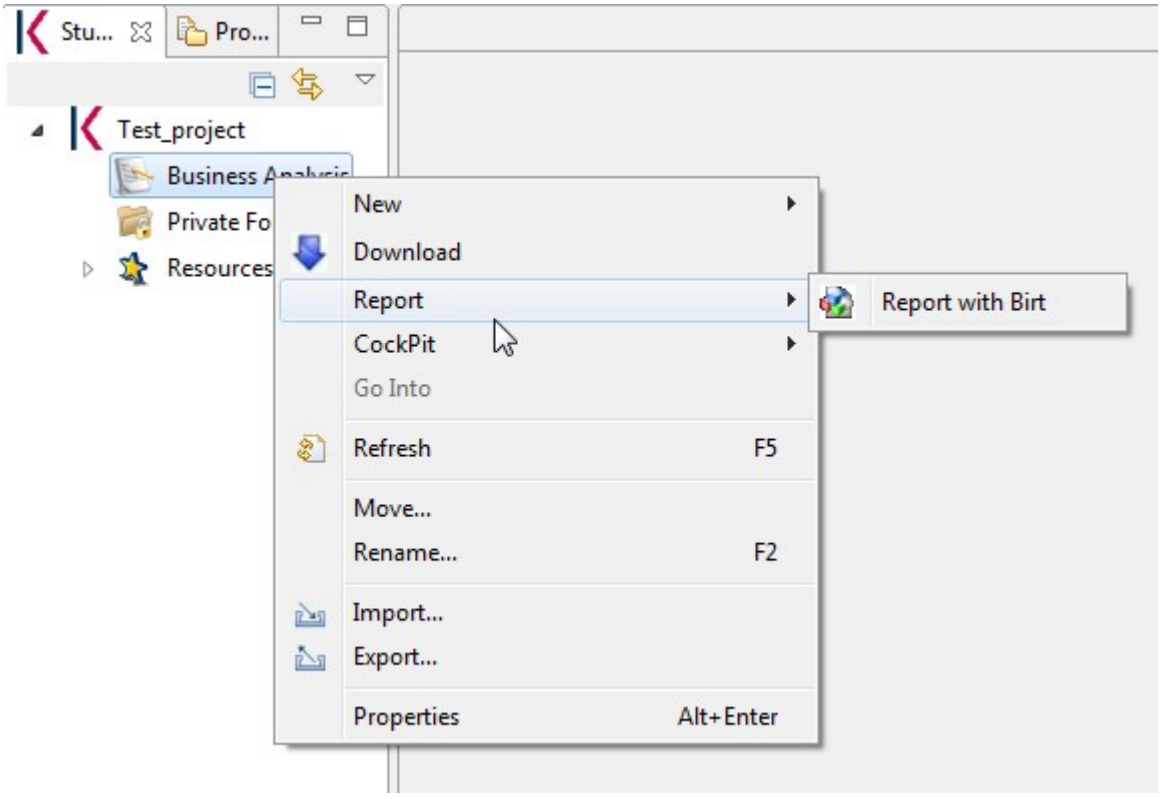


Fig. 5.274: New document creation.

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

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

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

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

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

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

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

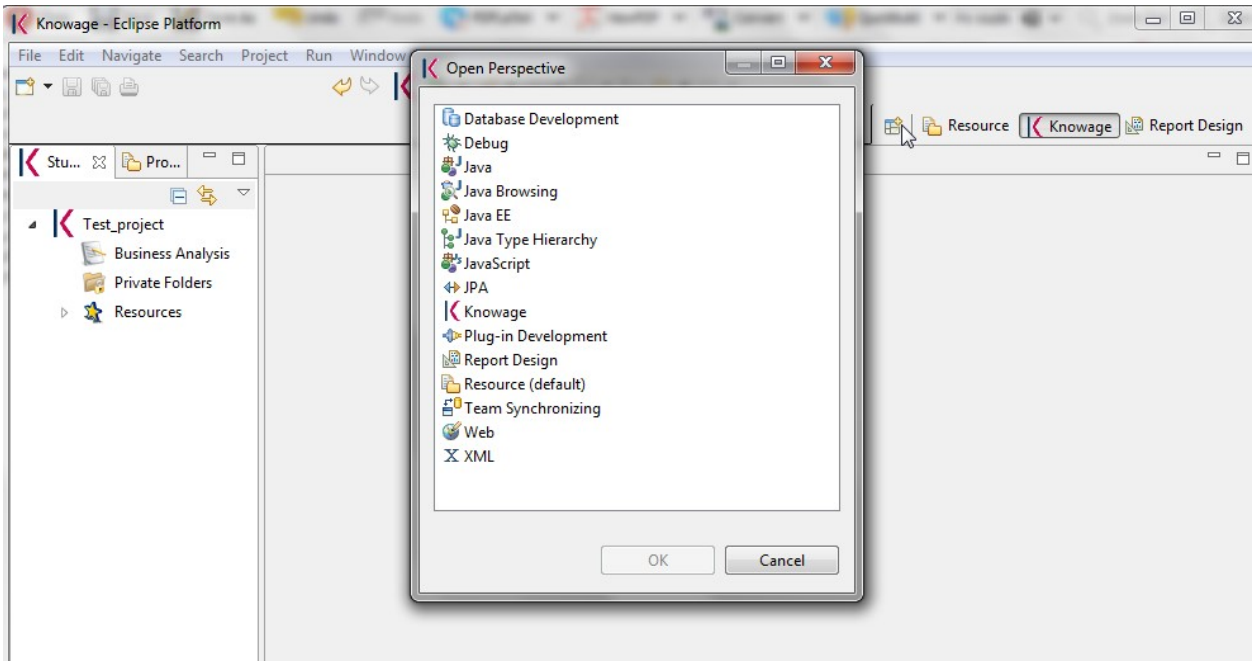


Fig. 5.275: Change perspective.

The next steps are the creation of a datasource and of a dataset. As previously described in the section Dataset Definition, Knowage Studio allows the development of analytical documents using either internal or external datasets. In this specific example, we will show how to create a report with an internal dataset. First of all, in case of an internal dataset, define a **JDBC Data Source**.

Right click on the **Data Source** item and select the corresponding data source. A pop up editor will open, prompting you the connection settings:

- **Driver class**
- **Database URL**

- **Username** and **password**

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

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

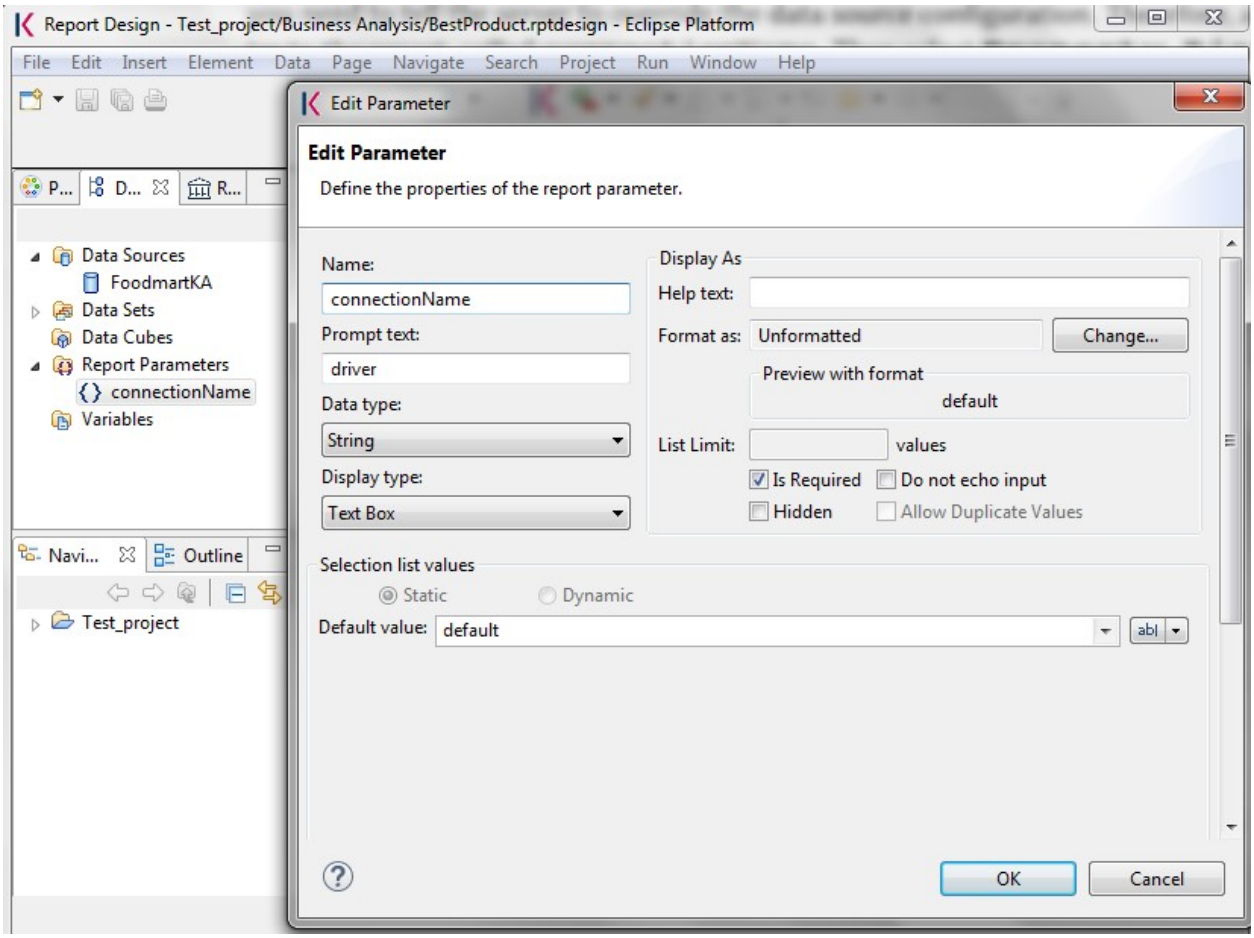


Fig. 5.276: Adding connectionName Parameter.

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

Warning: JNDI URL

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

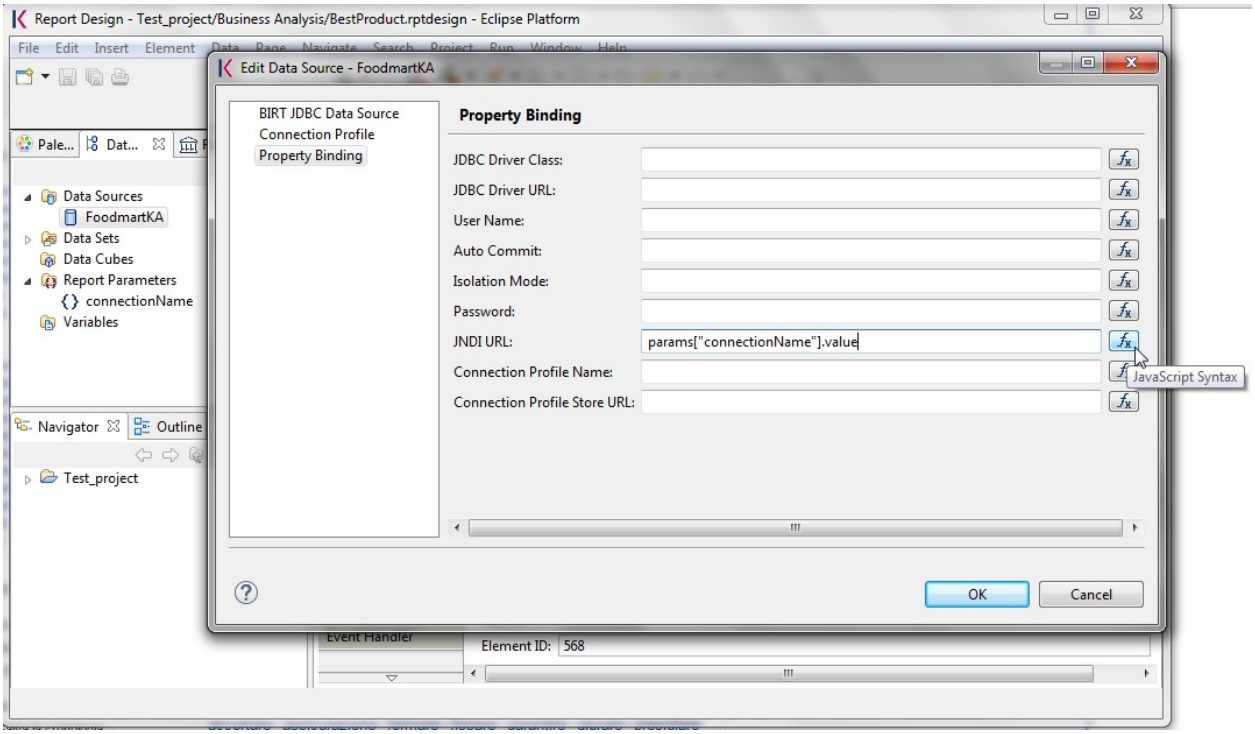


Fig. 5.277: Setting the connectionName parameter in the Data Source editor

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

Now you can define your dataset by writing the SQL query in the editor and testing the results (see Fig. 5.279). At any time, you can modify the dataset by clicking on it, which will re-open the query editor.

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

The most generic way, which applies to all graphical elements, consists in switching to the **Palette** menu on the left panel, keeping the designer in the central panel. Drag and drop the table into the editor area. Consider that this can be done with all other elements listed in the Palette. At this point, you can edit the table (as well as any other graphical element on the report) using the **Property Editor** tab below the editor area.

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

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

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

Note: Deploy on Knowage Server

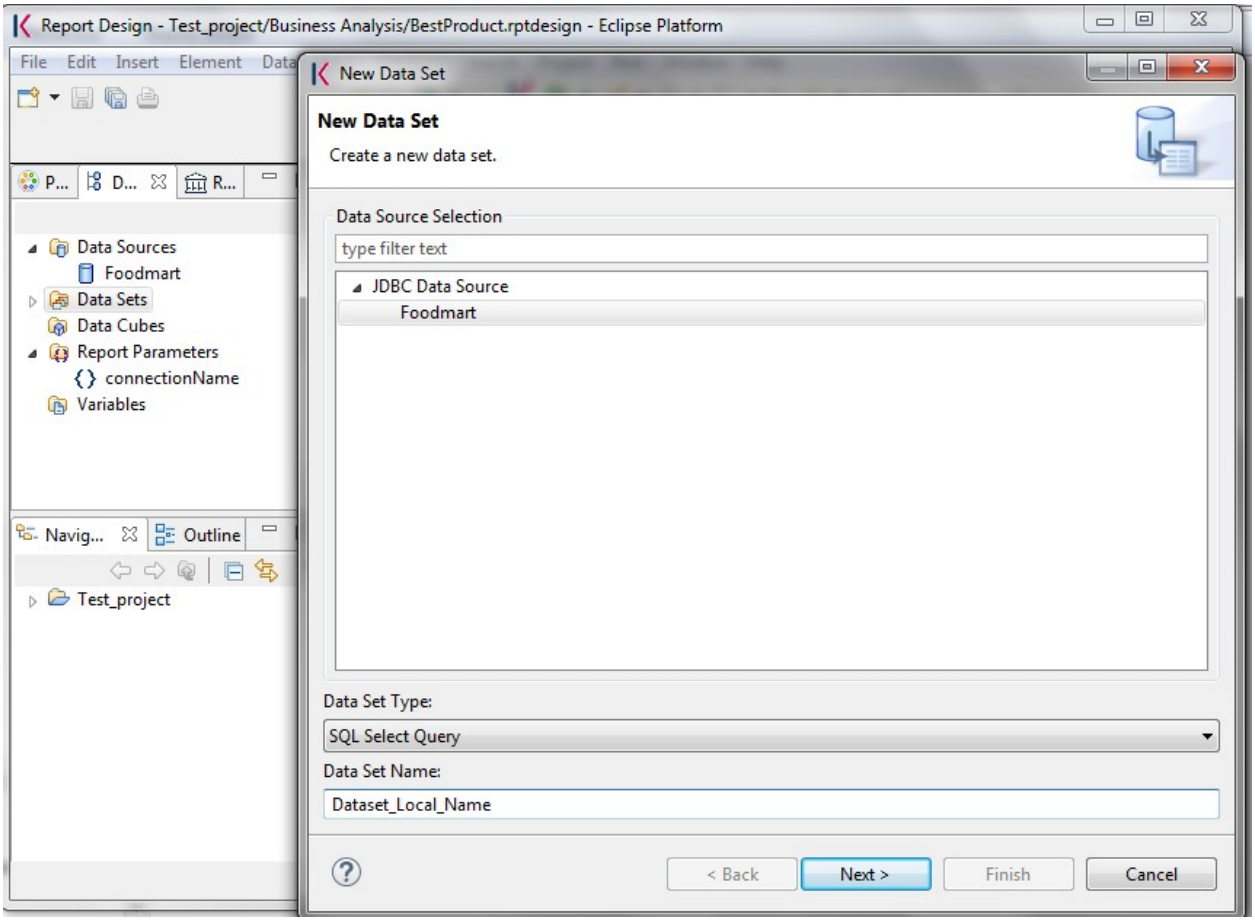


Fig. 5.278: Dataset definition.

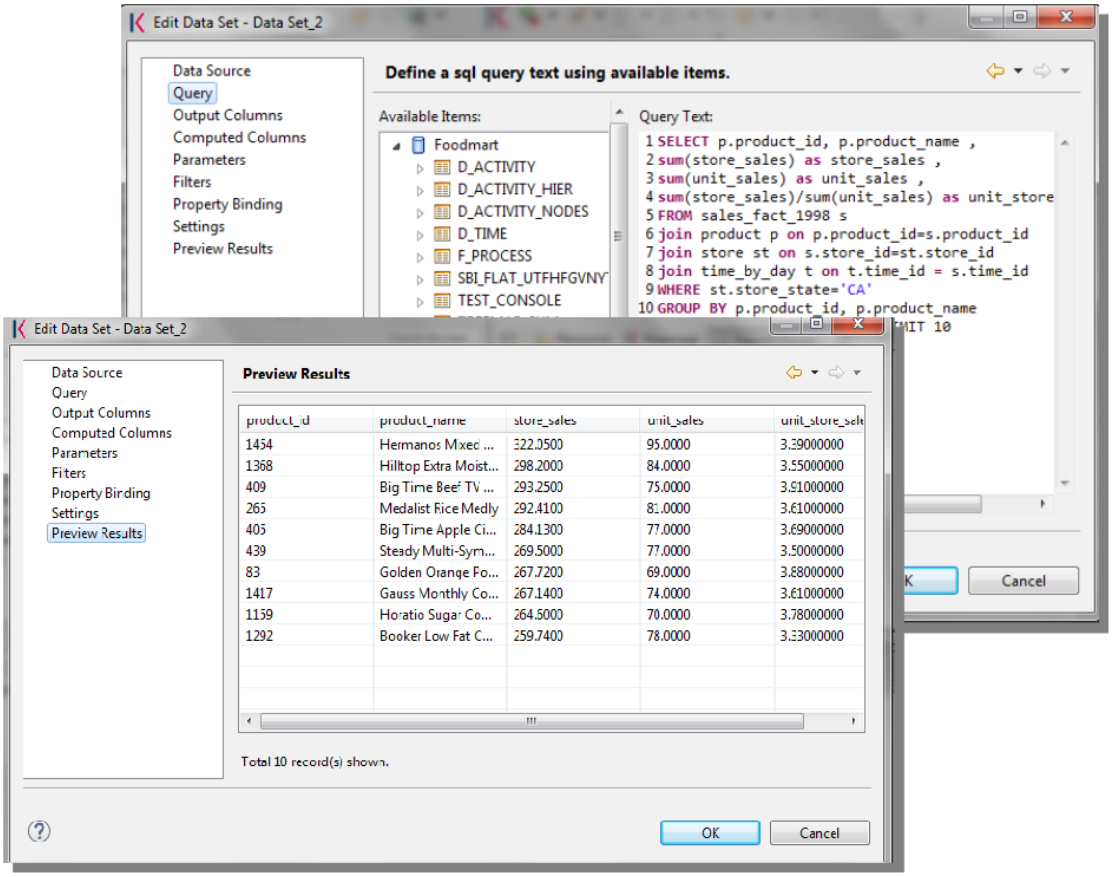


Fig. 5.279: Dataset editor, with preview.

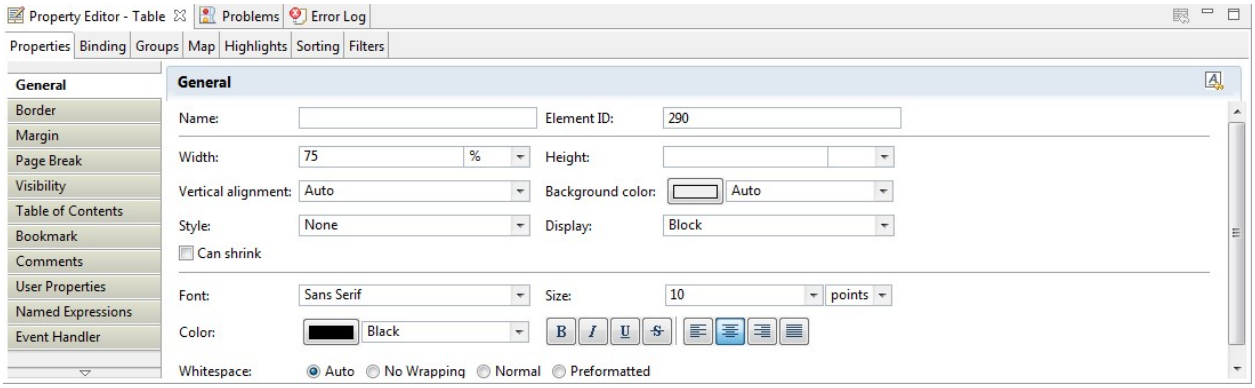


Fig. 5.280: BIRT Property Editor.

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

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

Note: BIRT Designer

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

5.10.2.1 Using an external Data Set

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

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

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

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

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

After filling in the configuration fields, test the connection and save it. The new data source will appear in the left tree menu. Instead of connecting to a database via a JDBC driver, connect to the server as the source of data. Obviously, the actual data source and dataset must have previously been defined on the Server.

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

Warning: Use of BM queries in report development.

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

5.10.2.2 Adding parameters to reports

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

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

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.

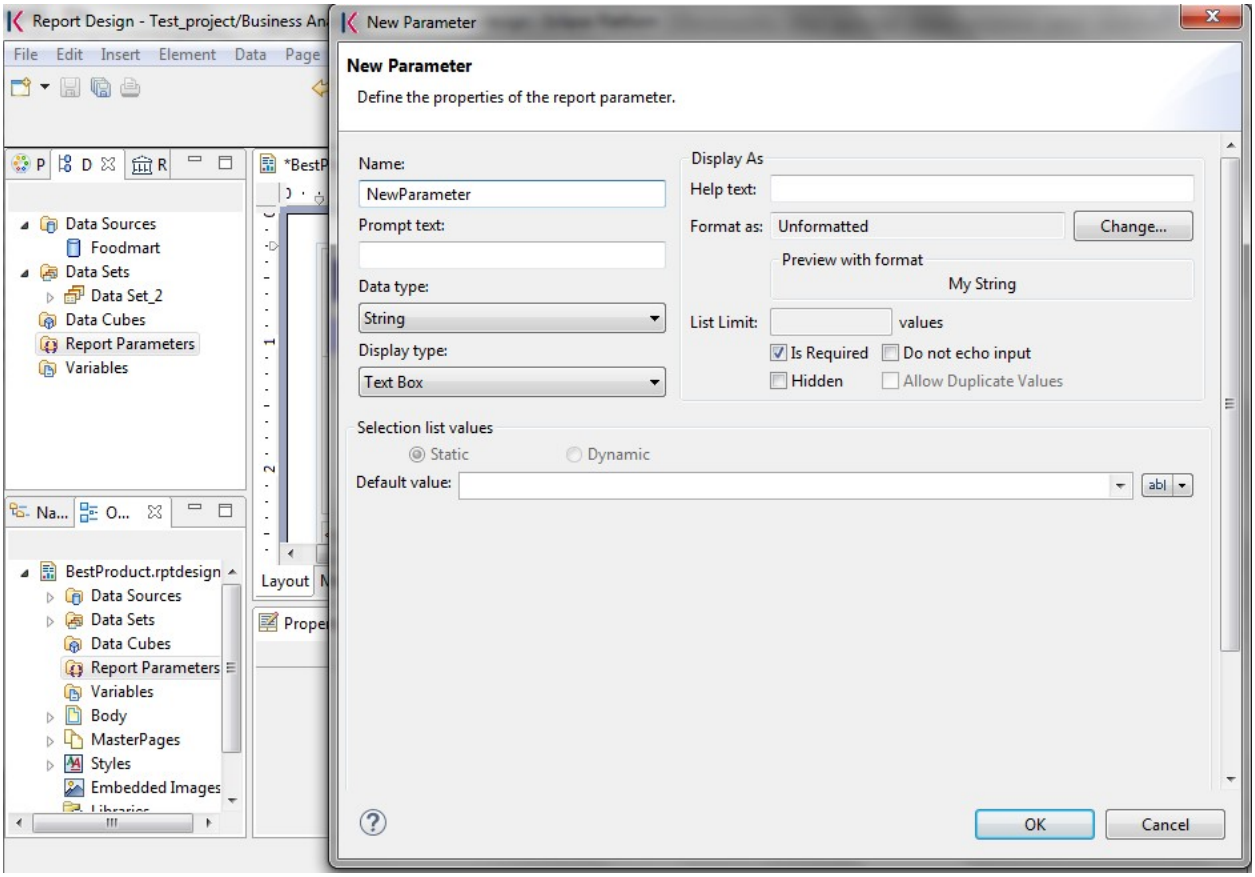


Fig. 5.281: Creation of a new parameter in a BIRT report.

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 Studio to Server and vice versa

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

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

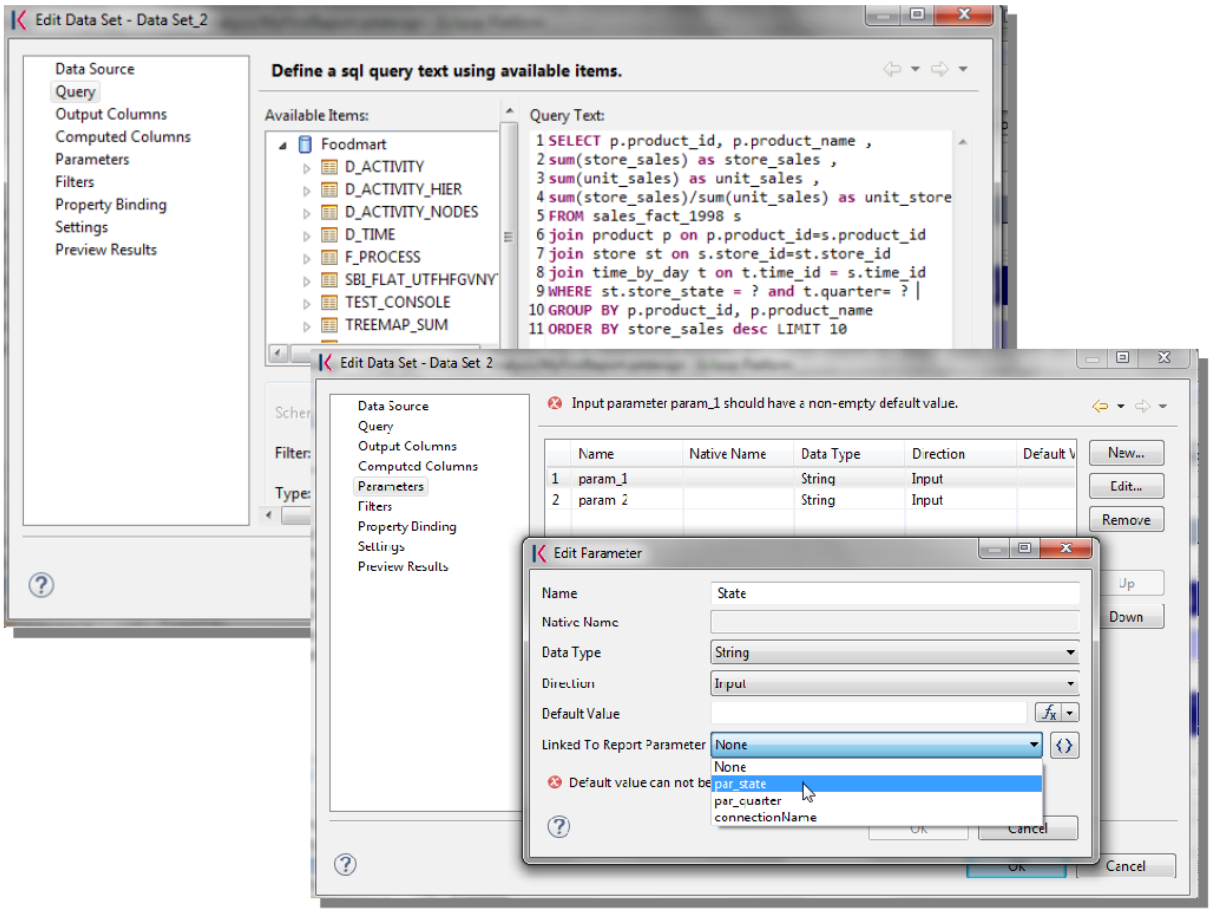


Fig. 5.282: Insert parameters into the dataset definition.

Listing 5.17: Parameters syntax

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

5.10.3 Download and deploy

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

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

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

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

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

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

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

Warning: Analytical documents

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

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

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

5.10.4 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 a chart, a console, a geo or any other analytical document.

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

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

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

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

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

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

- **Location:** write here the URI,
- **Target:** select Self,
- **Tool Tip.** write the text you wish to appear on the link, as showed in the following Figure 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 5.18: Cross Navigation syntax.

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

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

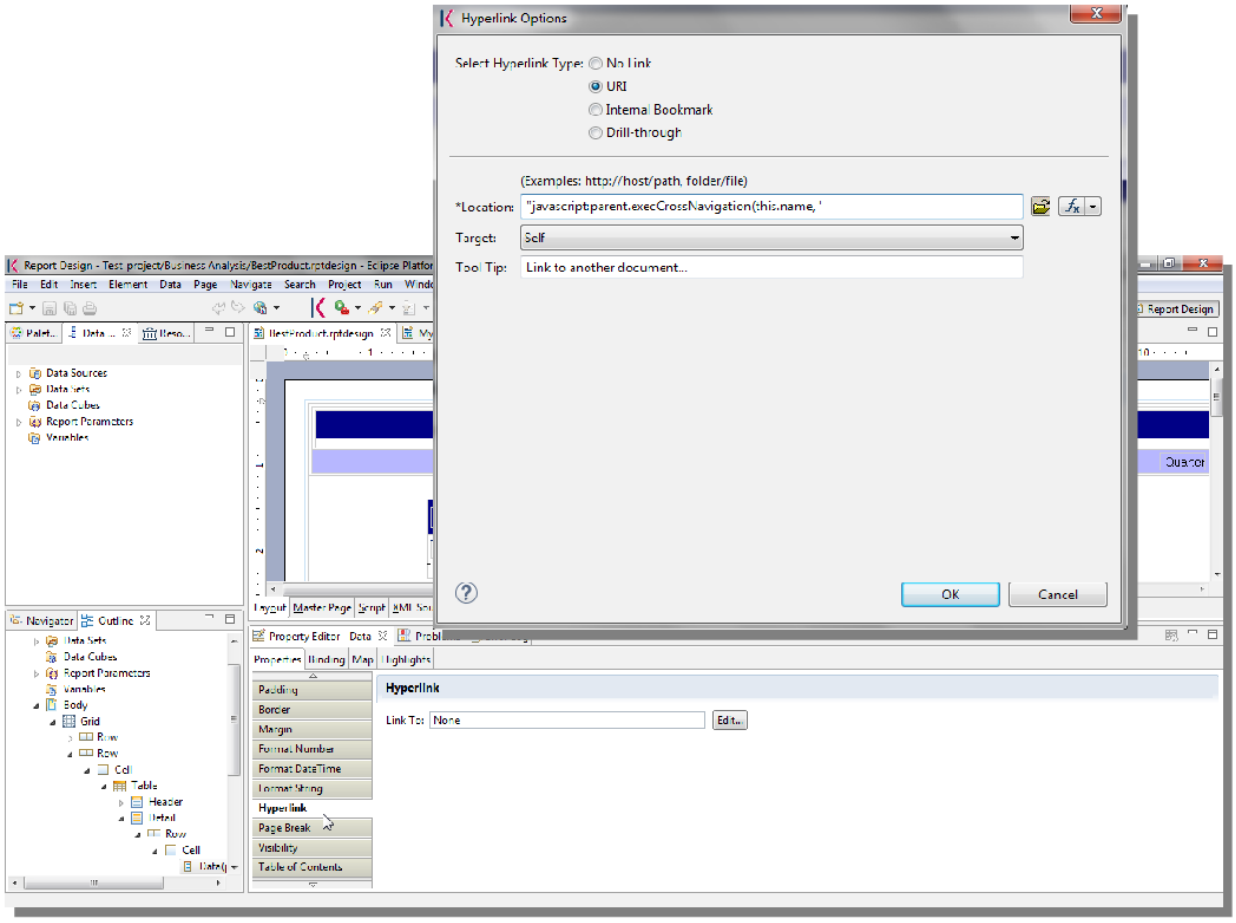


Fig. 5.283: Hyperlink editor.

the deployed document on the Server (see *Analytical Document* Chapter, *Cross Navigation* Section).

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.

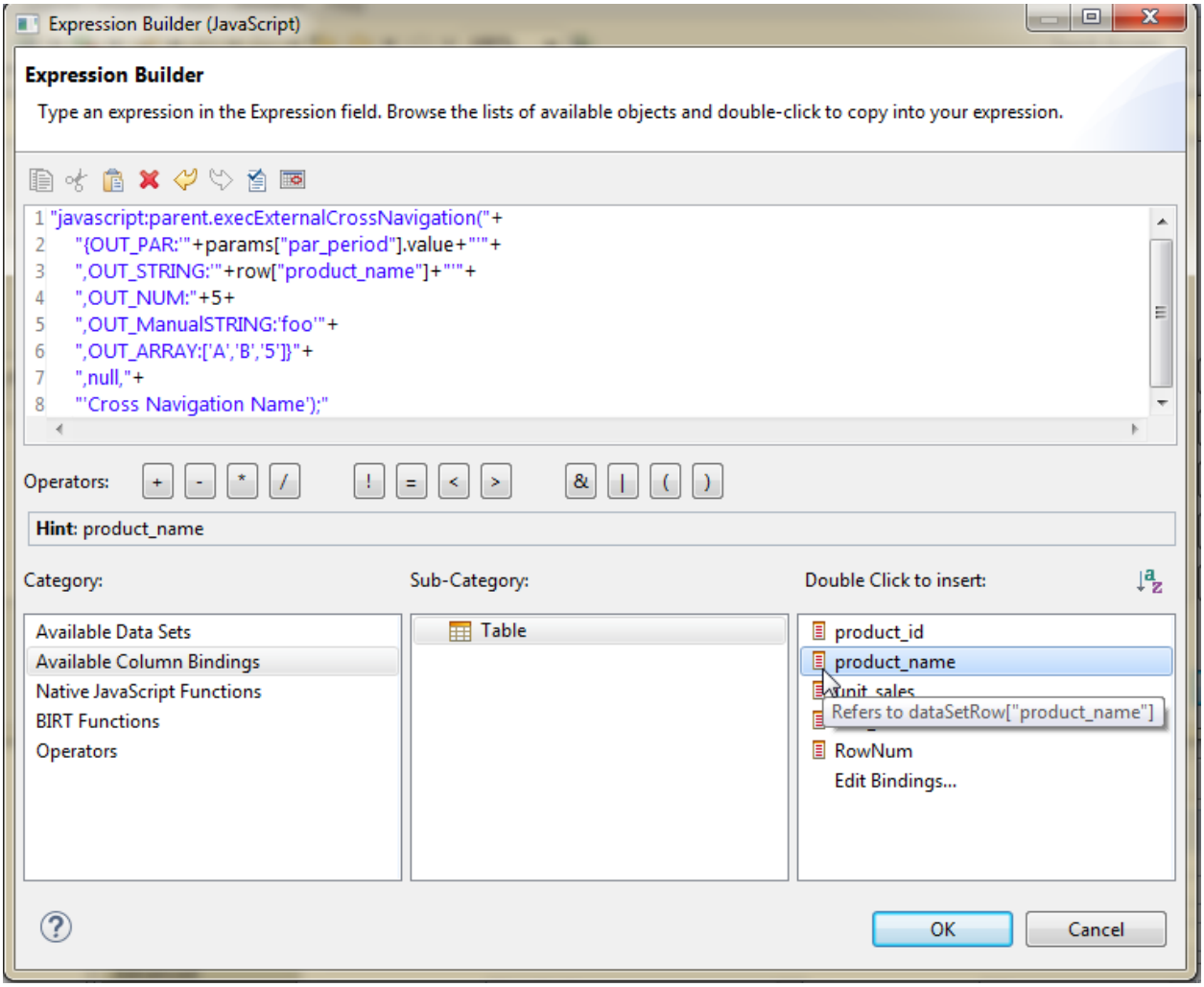


Fig. 5.284: Column bindings.

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 5.19: Cross Navigation syntax

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

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.

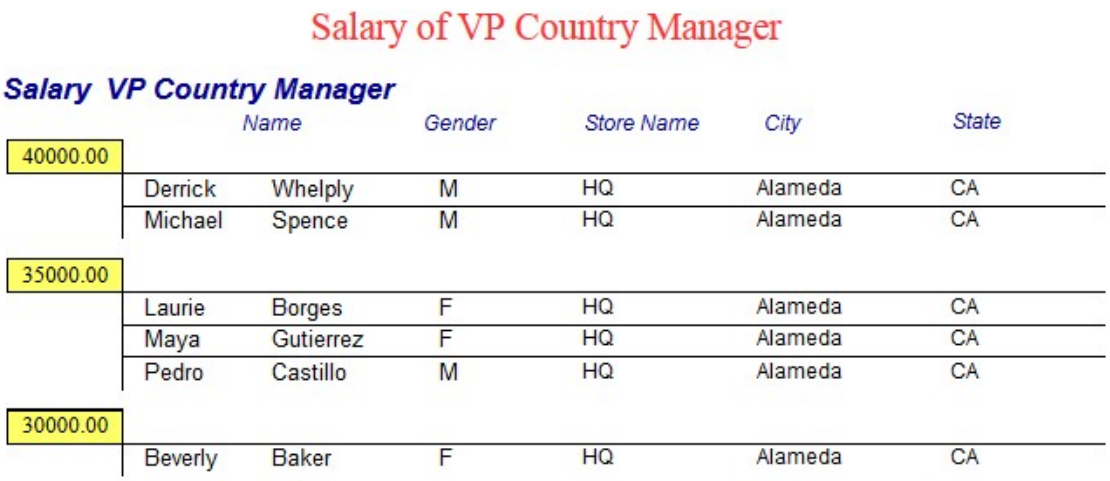


Fig. 5.285: Example of a Jasper report.

Plese note that this engine is available only in KnowageER.

5.11.1 Document defintion*

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.

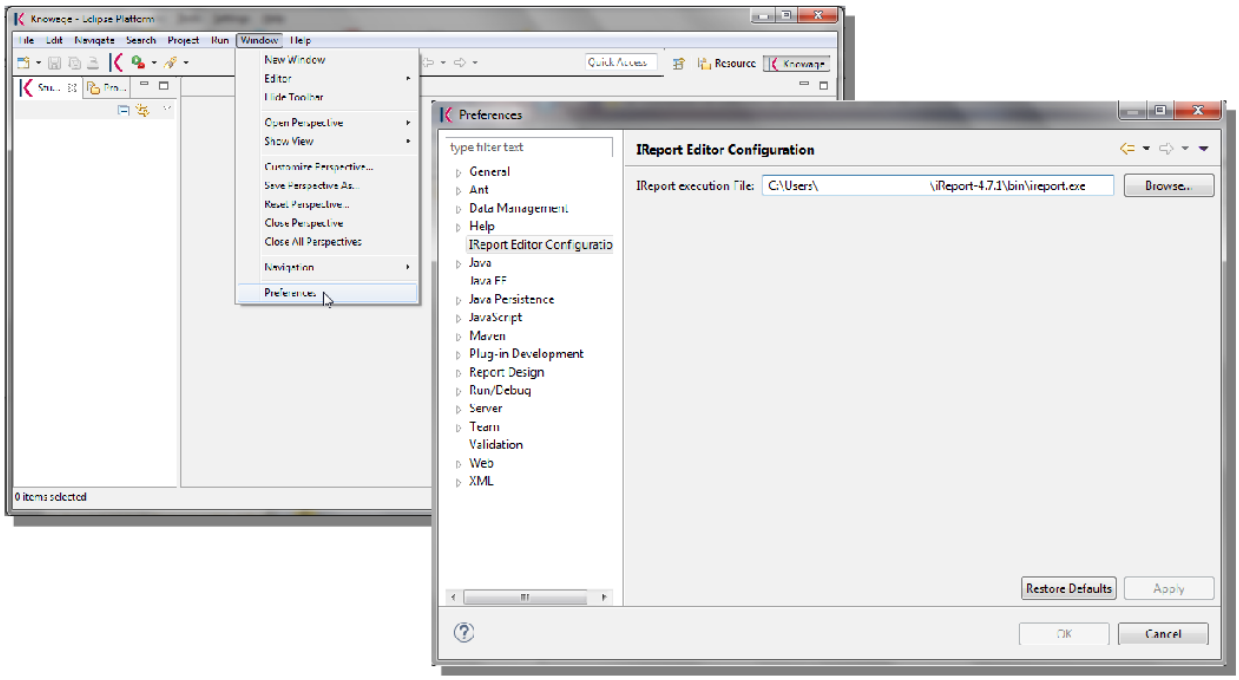


Fig. 5.286: iReport Configuration in Knowage Studio.

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.

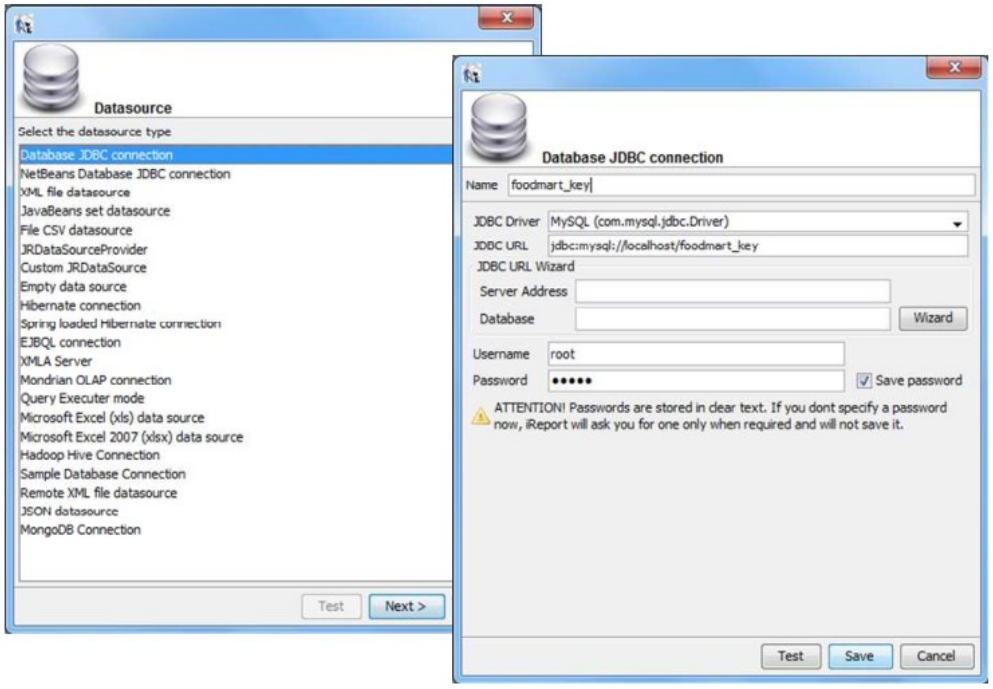


Fig. 5.287: Creation of a JDBC data source in a Jasper report.

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.

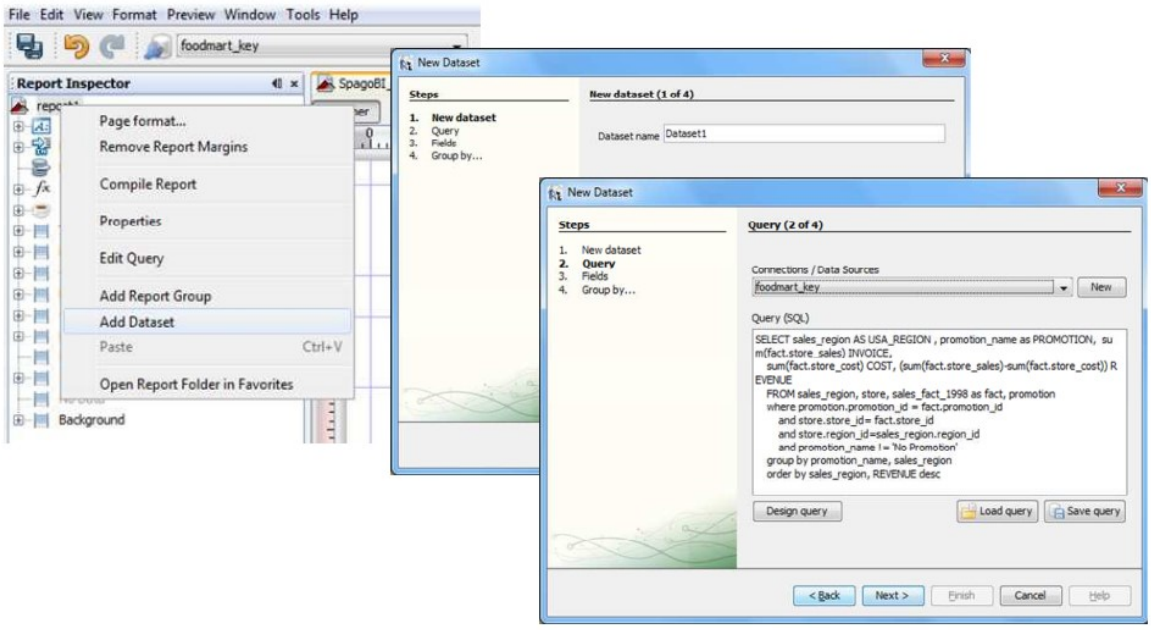


Fig. 5.288: Data Set definition.

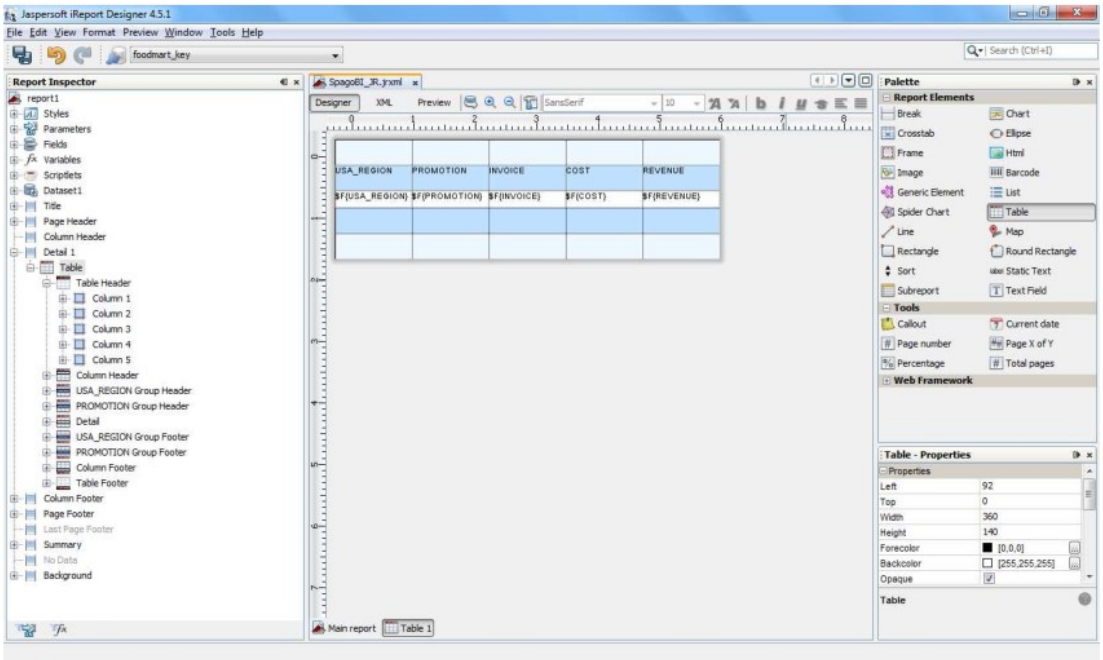


Fig. 5.289: iReport graphical editor.

Listing 5.20: 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.

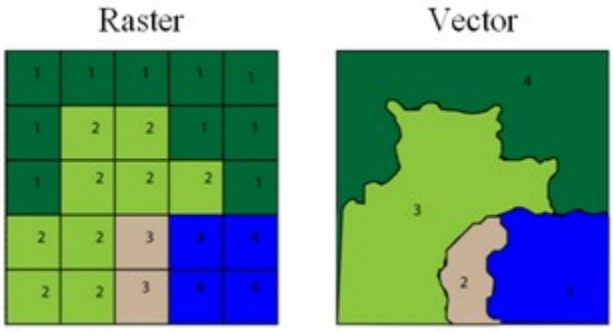


Fig. 5.290: A base layer in raster and vector format.

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 referencez coordinate system, according to their relative position.

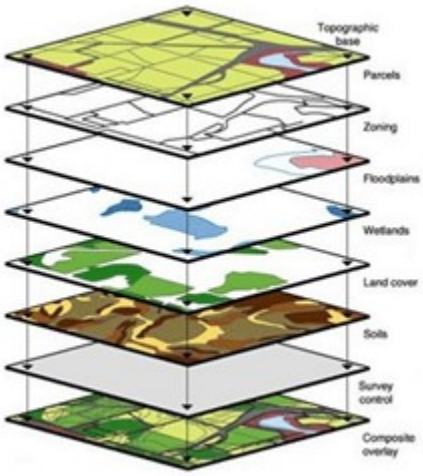


Fig. 5.291: Overlapping layer.

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.

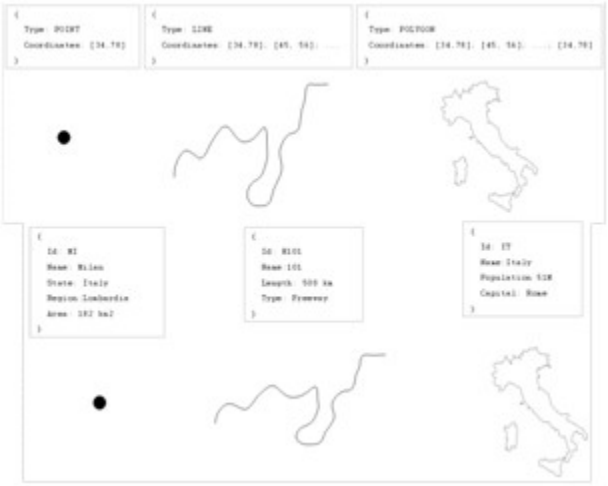


Fig. 5.292: Examples of feature.

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. 5.293: 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 Gesospatial 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 redistribution 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 belowwe 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 Inteligence options panel. Here you can choose the **Map Type**, the indicators to be displayed on the map and you can enter filters.

The **Map Type** available are:

- **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 Inteligence options panel.

Here click on **select form catalog**, choose the layers you want to add. Mark them in the bottom part of the Location Intelligence area in the Layer box and the selected layer are displayed. These steps are shown in figure below.

In our example we upload some waypoints, you can see the results obtained in next figure.

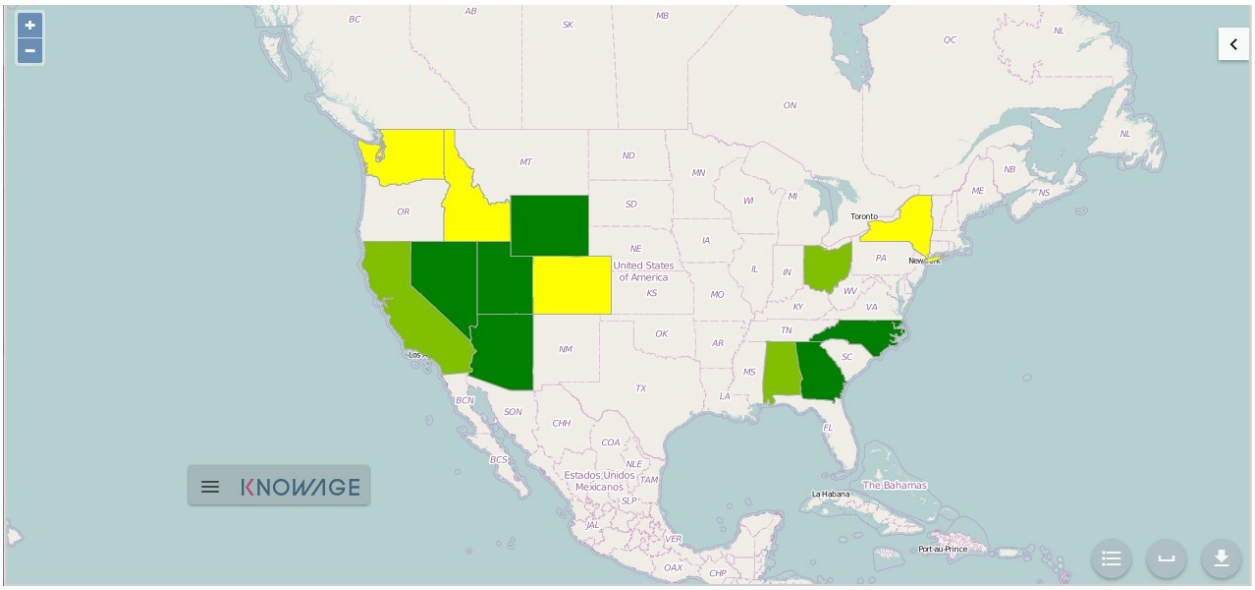


Fig. 5.294: Example of GIS document. USA sales per store

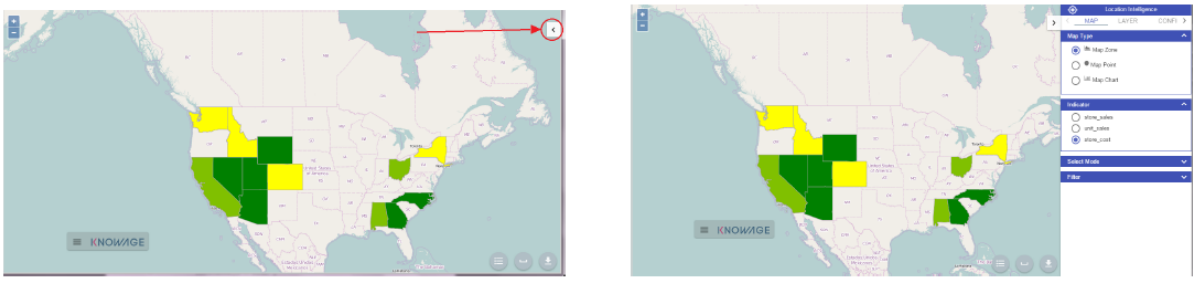


Fig. 5.295: Arrow button (left) Location Inteligence options panel (right) .

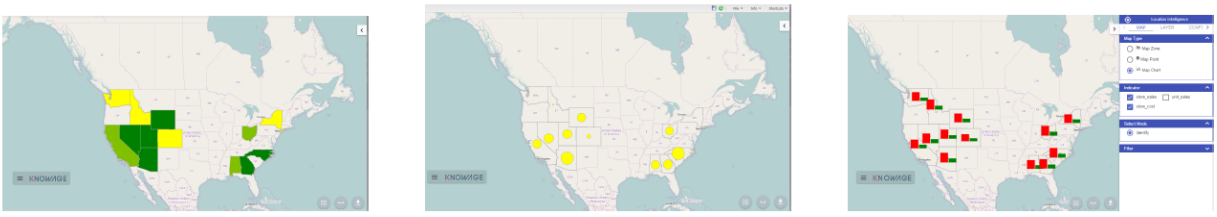


Fig. 5.296: Map Zone (left) Map Point (center) and Map Chart (right).

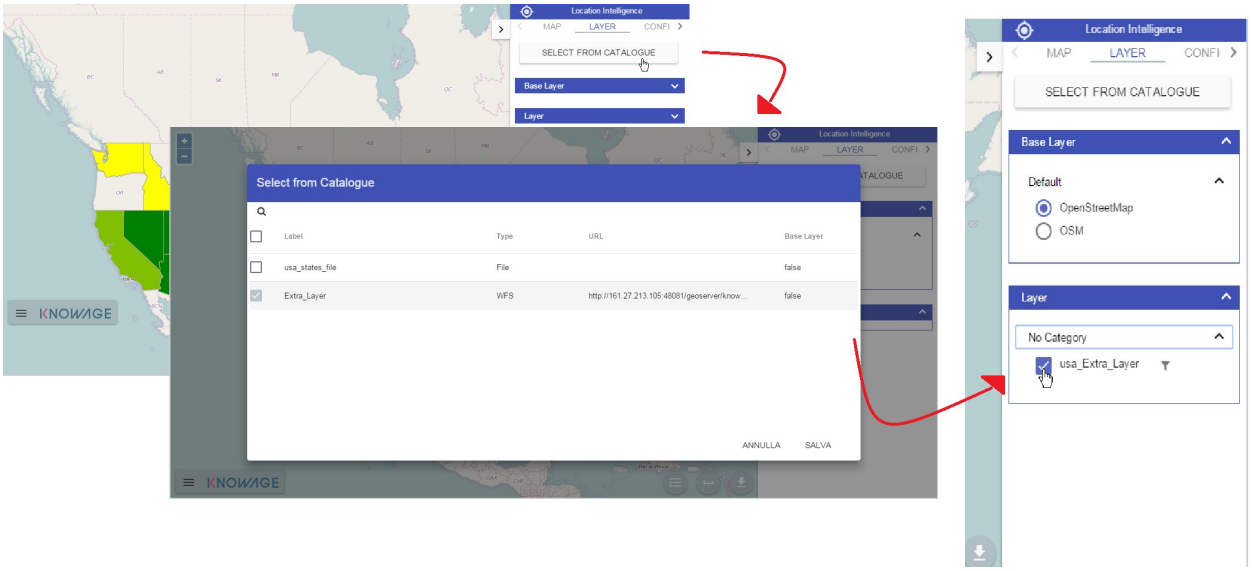


Fig. 5.297: Steps for layer adding

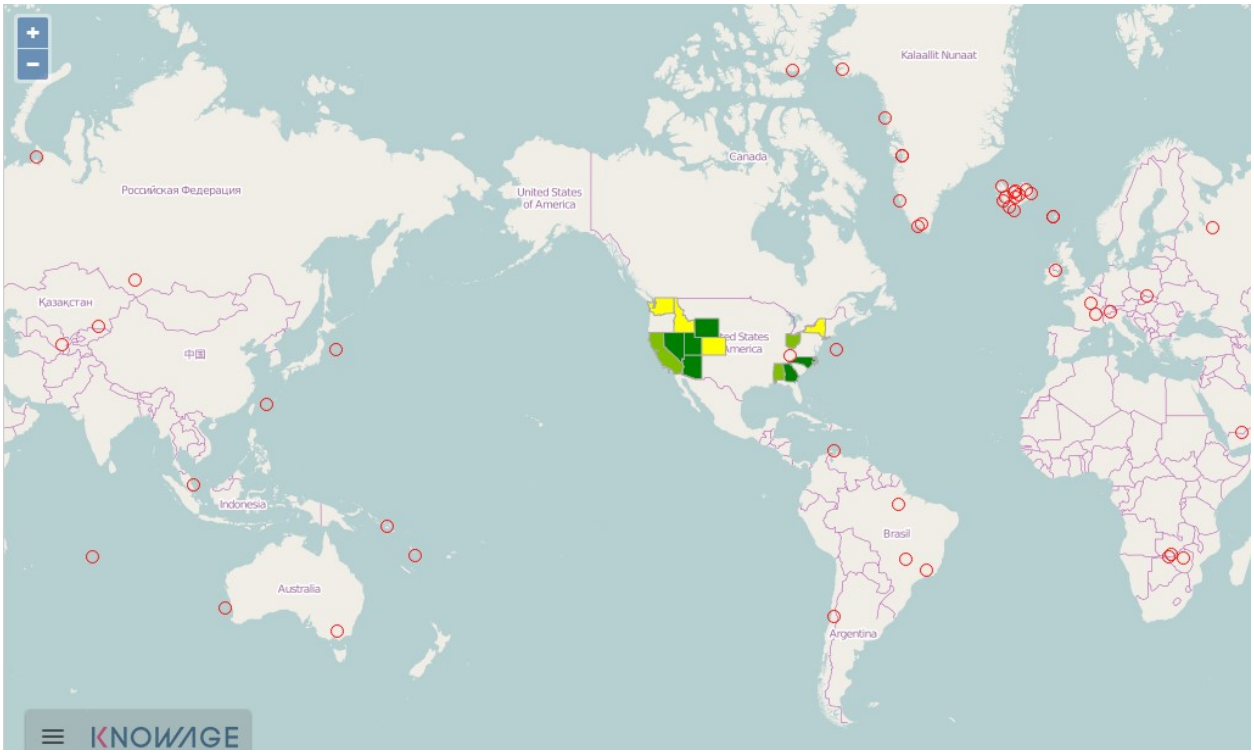


Fig. 5.298: Map with two layers

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

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

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

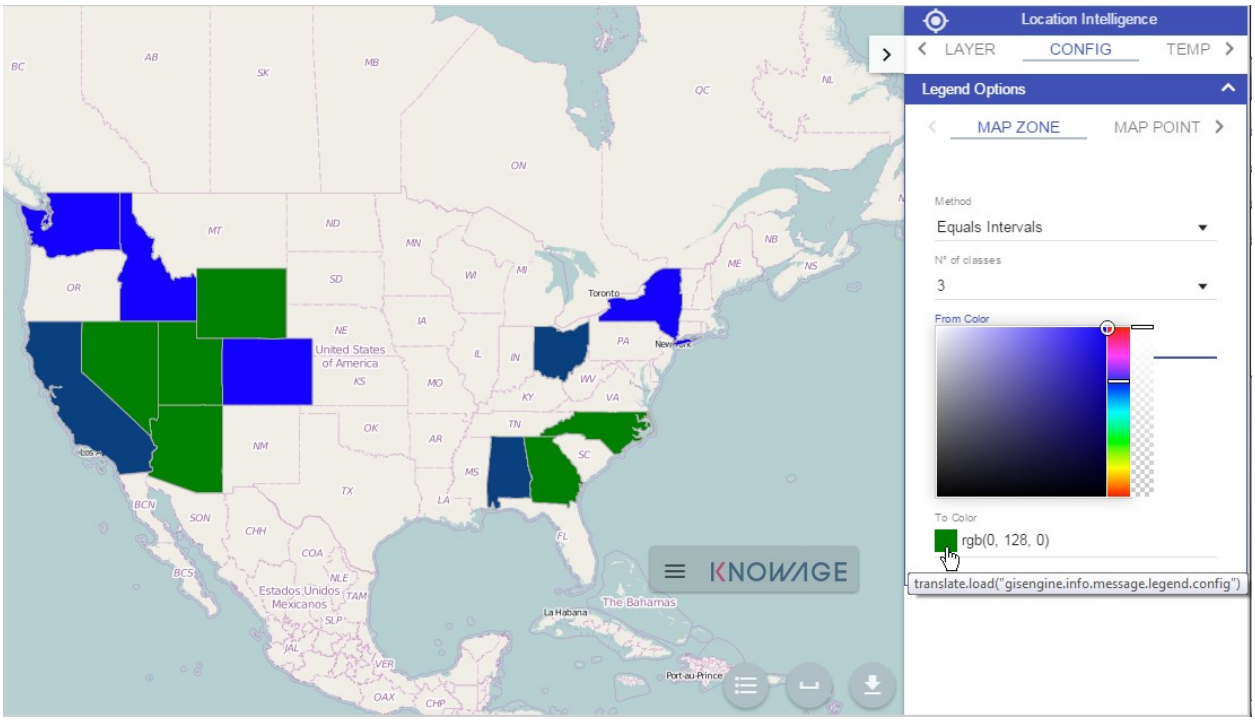


Fig. 5.299: Map Zone extra configurations

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 bottons can be used for: have a look at the legend, compute a measure of an area of the map and do the .pdf export of the map.

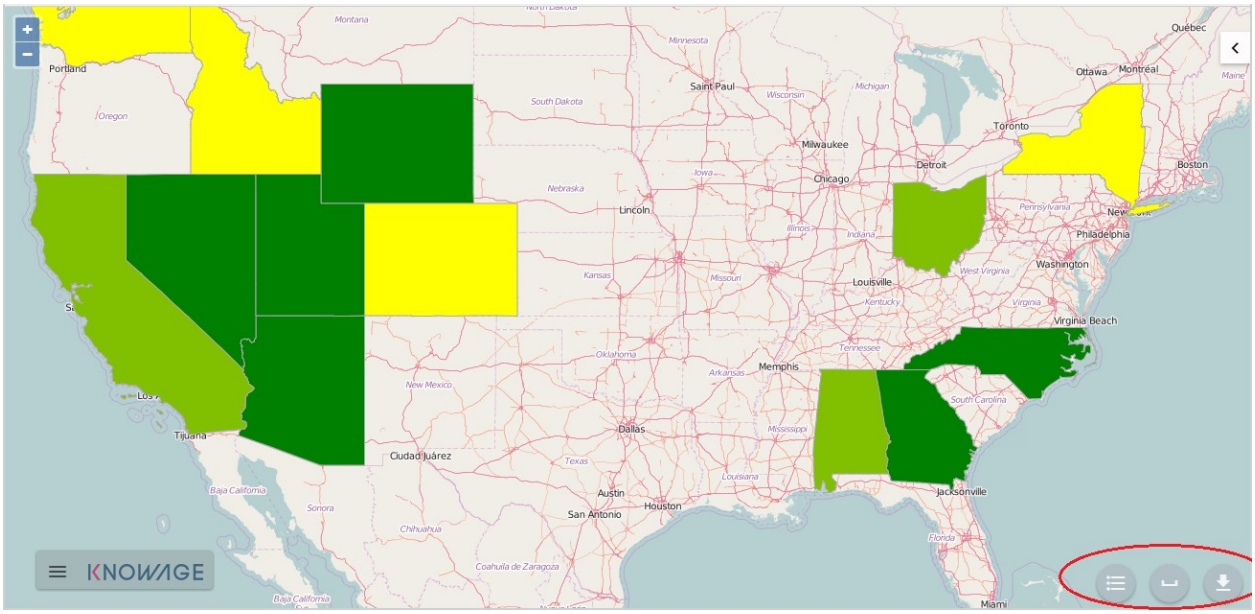


Fig. 5.300: From the left to the right: Legend, Measure and Export bottom.

5.12.3.1 Extra functionalities

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

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

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

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

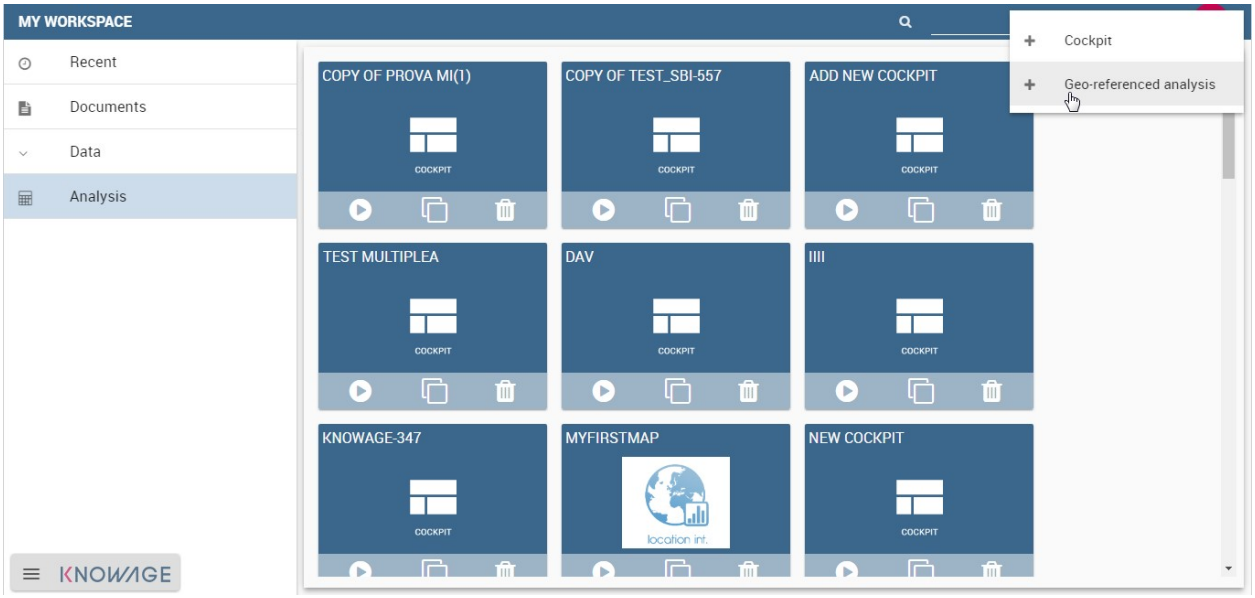


Fig. 5.301: Start a new Geo-referenced analysis.

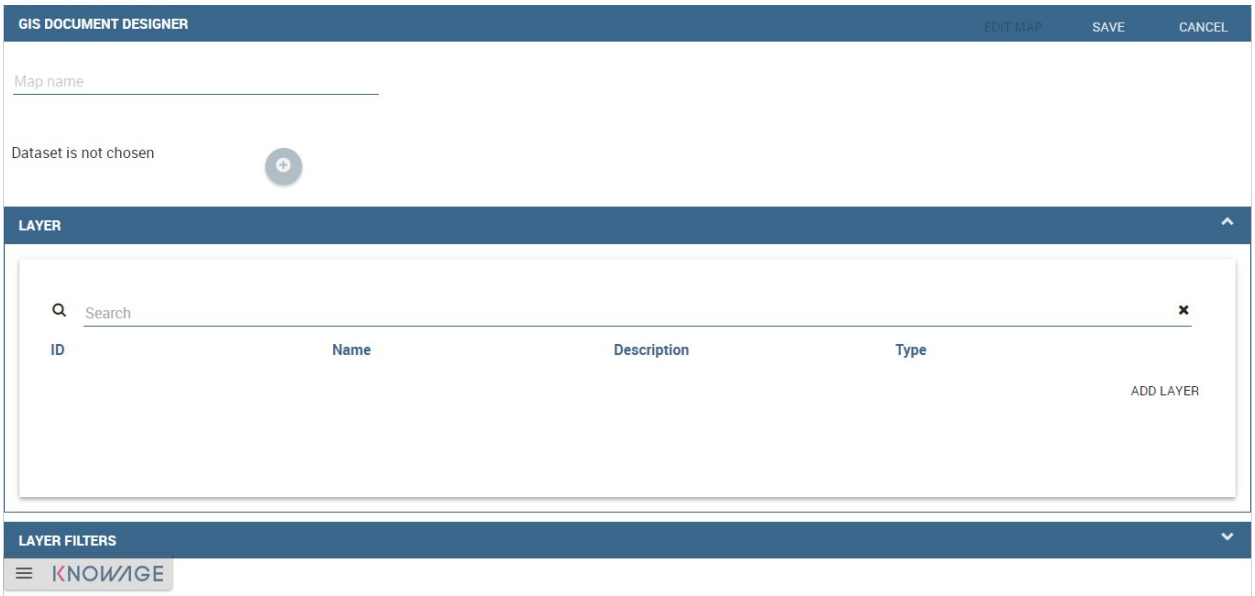


Fig. 5.302: GIS document designer window.

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.

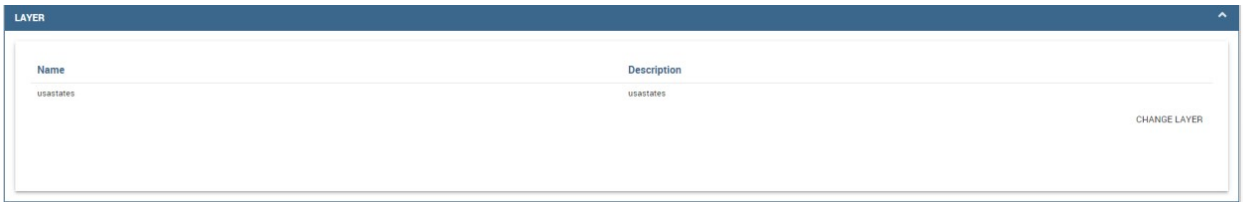


Fig. 5.303: Target layer definition.

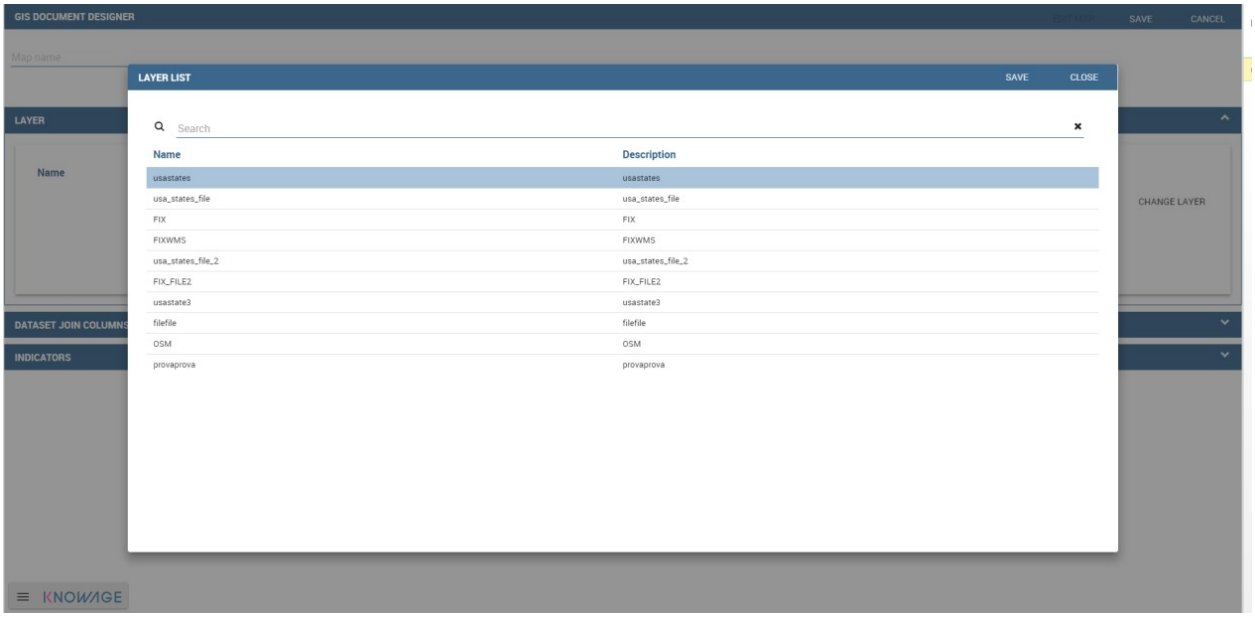


Fig. 5.304: List of available layer catalogs.

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. 5.305: Multiple selection of available layers.

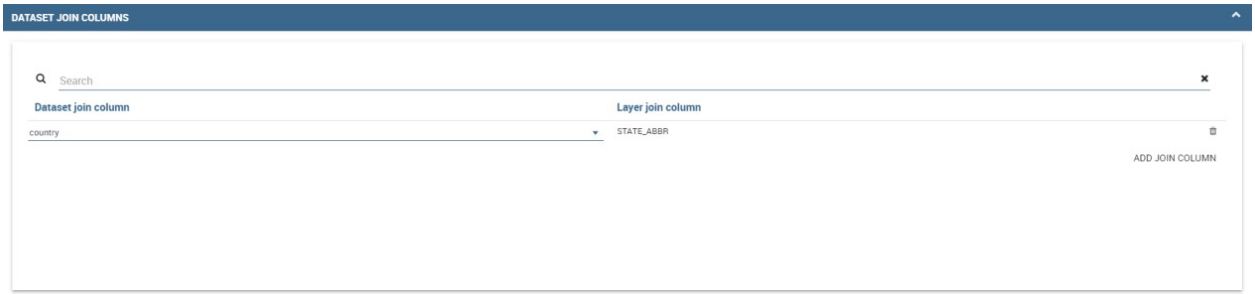


Fig. 5.306: Dataset join columns interface.

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 filed should be selected by picking one option from combo box that contains measure fields from selected dataset. Label should be inserted as free text by editing corresponding table column.

Fig. 5.307: Indicators interface.

5.12.5.4 Filters

Using the filtering dedicated area, as ahown 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.

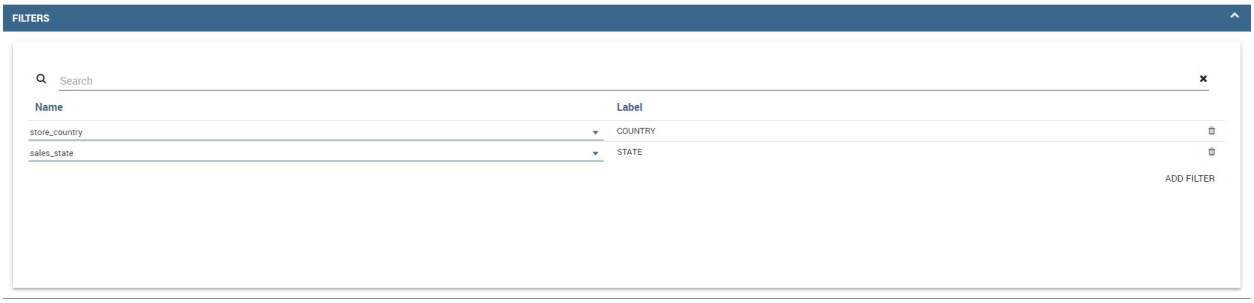


Fig. 5.308: Filters interface.

5.12.5.5 Map menu configuration

Through the **Map menu configuration** panel the user can desides 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.

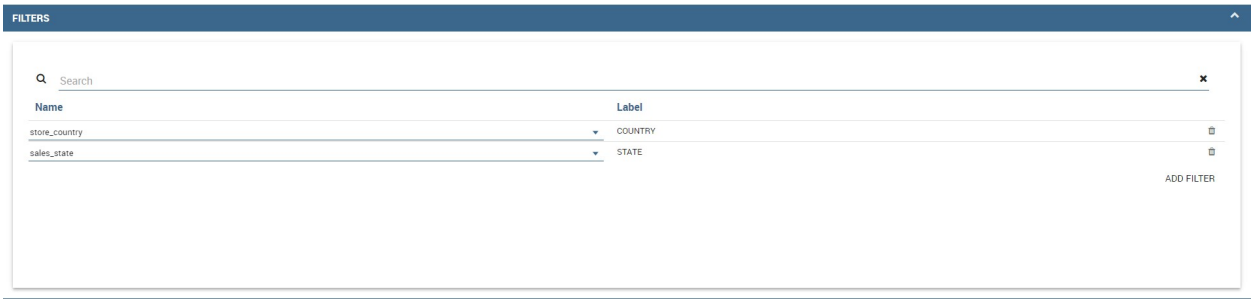


Fig. 5.309: Map menu configuration.

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. 5.310: Layer filters interface.

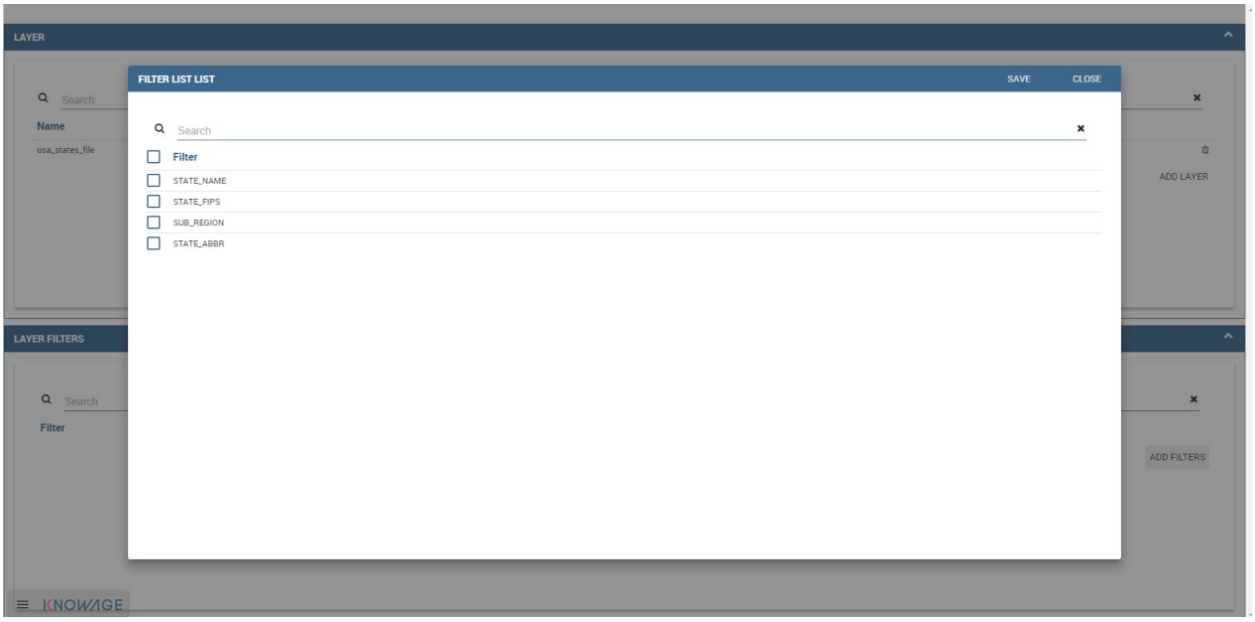


Fig. 5.311: List of available filters.

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 successfuly EDIT MAP button is enabled in the right part of the main toolbar.

Clicking the edit map button will open created map. An example is given 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;

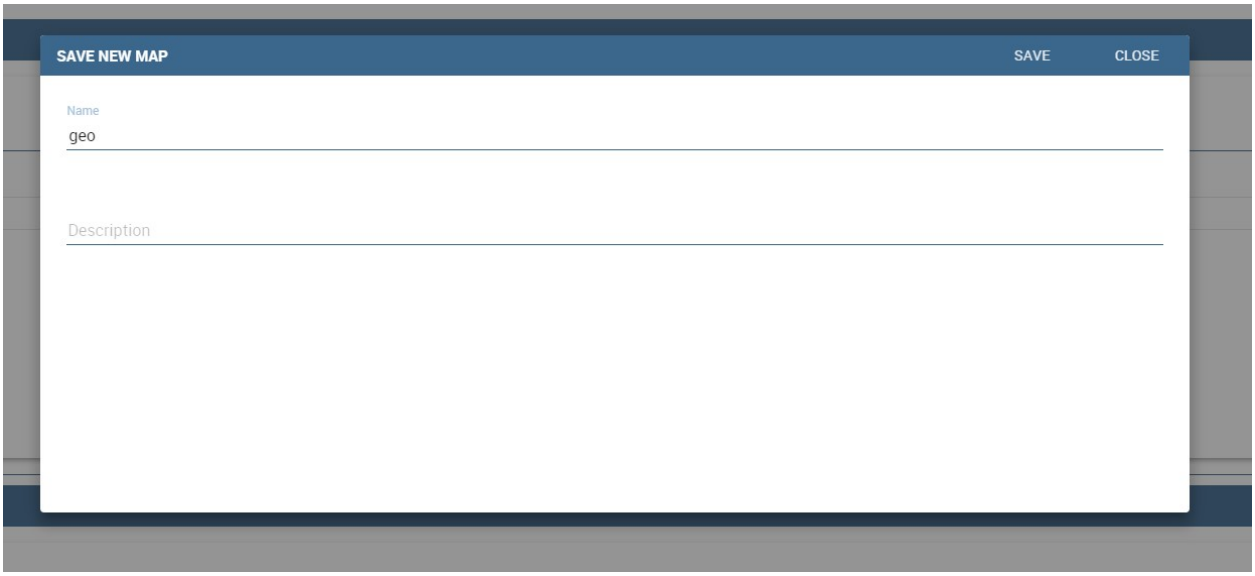


Fig. 5.312: interface for name and description of new geo document for end user.

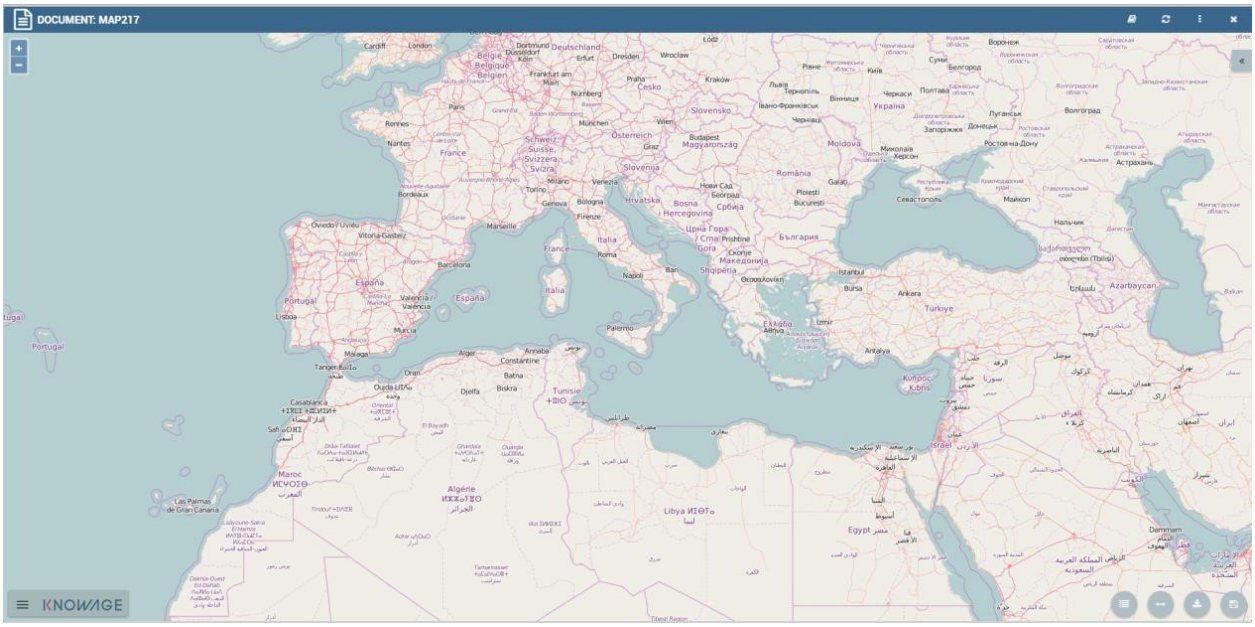


Fig. 5.313: Map in edit mode with save template available.

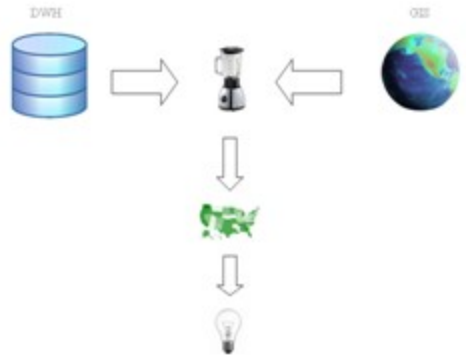


Fig. 5.314: Bridge integration architecture of the **GEOReport Engine**.

- 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 chose 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 previlegies on this layer. If none is marked, the layer is visibile 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.

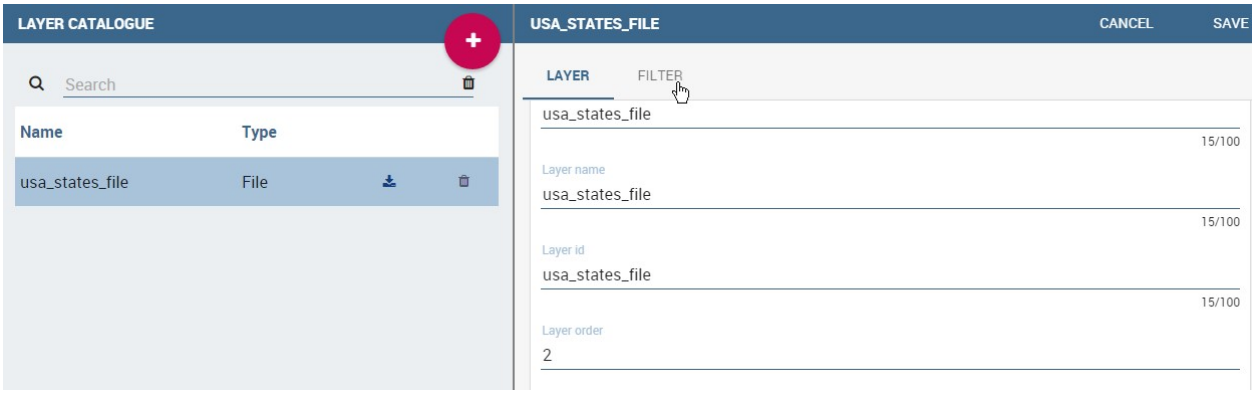


Fig. 5.315: Filter tab

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 5.21: GeojSON file except.

```
SELECT r.region_id as region_id
     , s.store_country
     , r.sales_state as sales_state
     , r.sales_region
     , s.store_city
     , sum(f.store_sales) + (CAST(RAND() \*60 AS UNSIGNED) + 1) store_sales
     , avg (f.unit_sales)+(CAST(RAND()\* 60 AS UNSIGNED) + 1) unit_sales
     , sum(f. store_cost) store_cost
FROM sales_fact_1998 f
   , store s
   , time_by_day t
   , sales_region r
WHERE s.store_id=f.store_id
AND f.time_id=t.time_id
AND s.region_id = r.region_id
AND STORE_COUNTRY = 'USA'
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 datail 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 possibile to navigate from the GIS document to a target document. There is still a little step to do to activate the cross navigation.

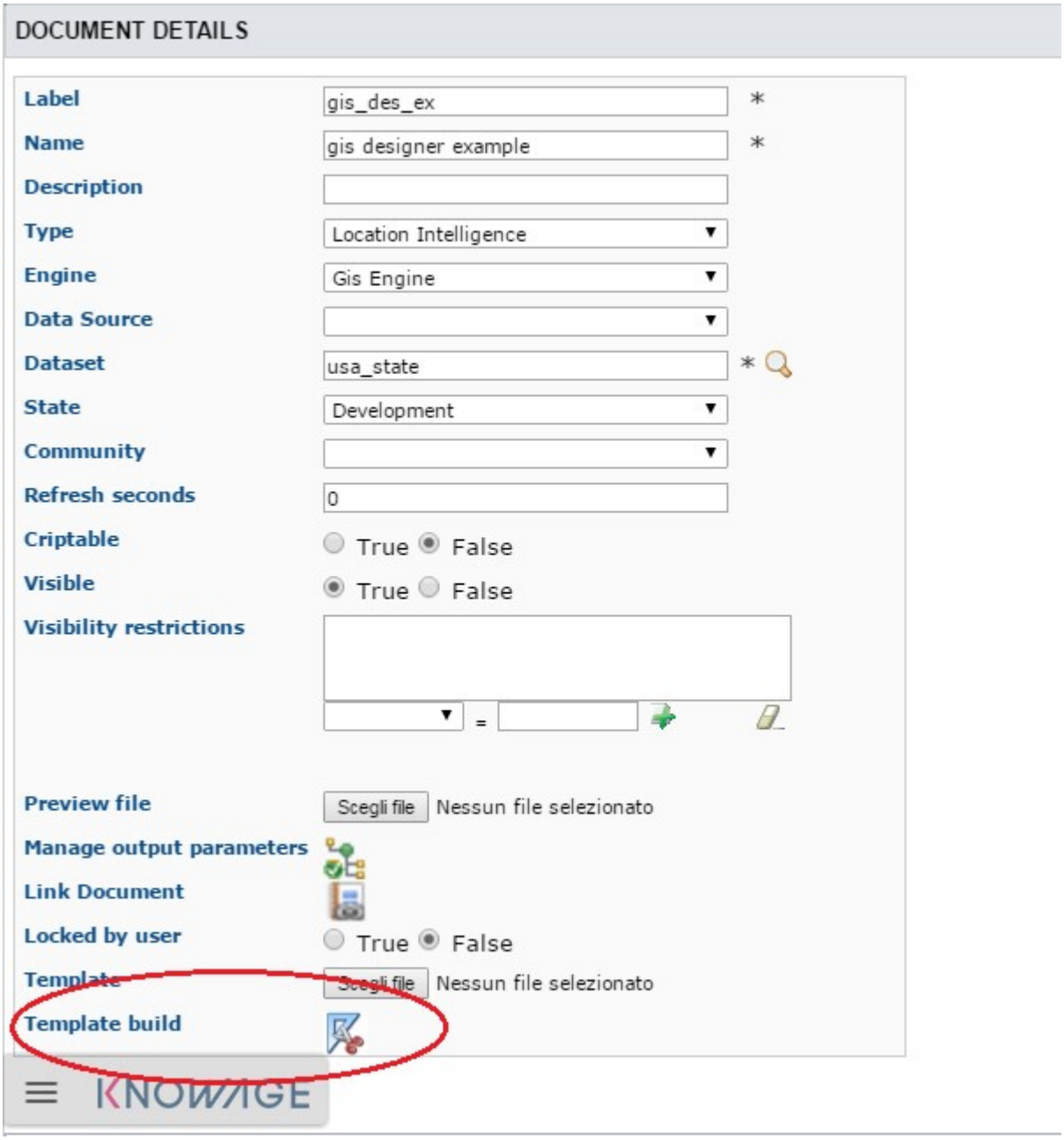


Fig. 5.316: Gis designer accessible from the template build.

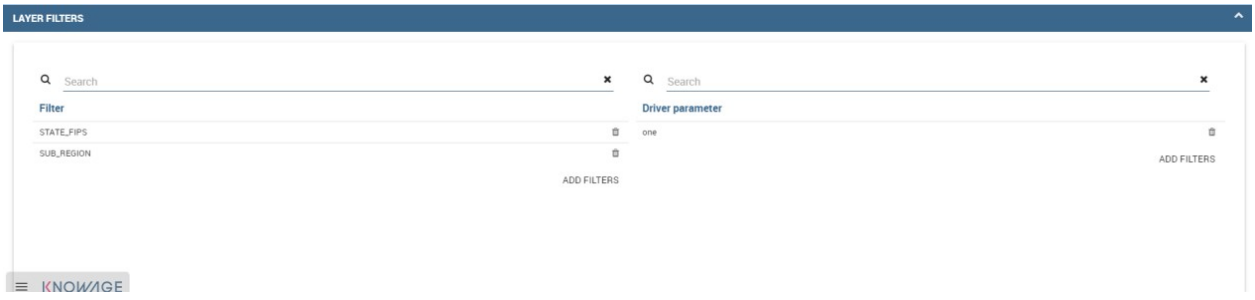


Fig. 5.317: Layer filters interface with analytical drivers.

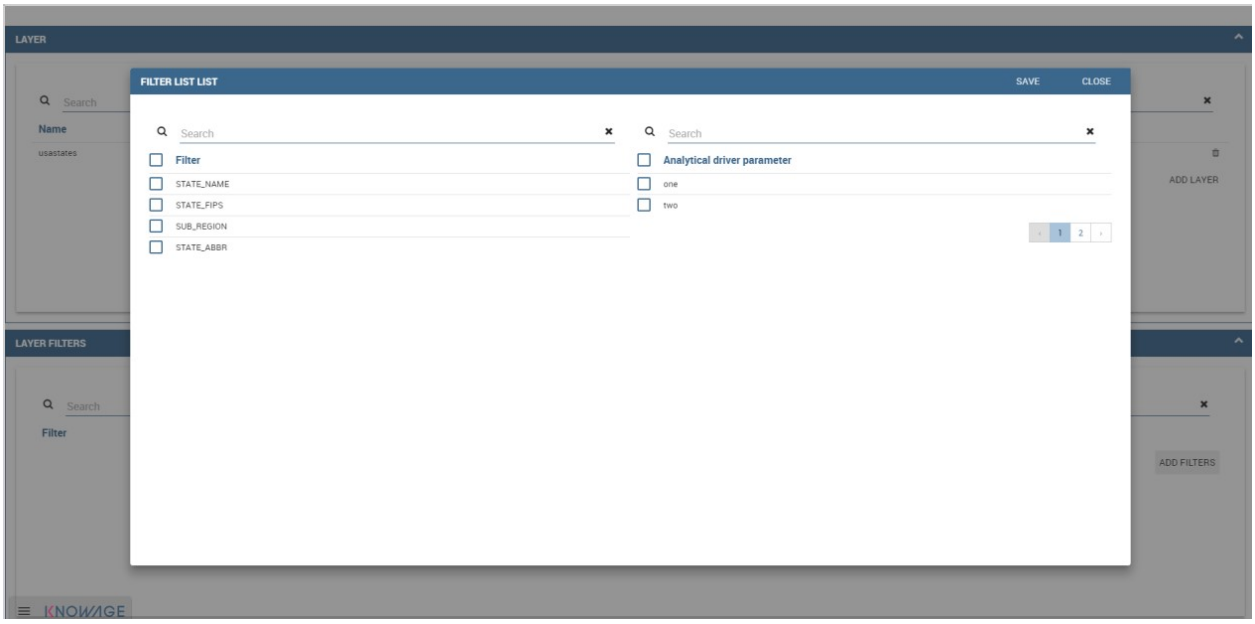


Fig. 5.318: List of available filters with list of analytical drivers.

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 throught an SVG the USA map. At first we can show data at the "Regions" level and then throught the click / drill - show the same or other information at the States "level". We give an example of map document produced with the SVG engine in 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.

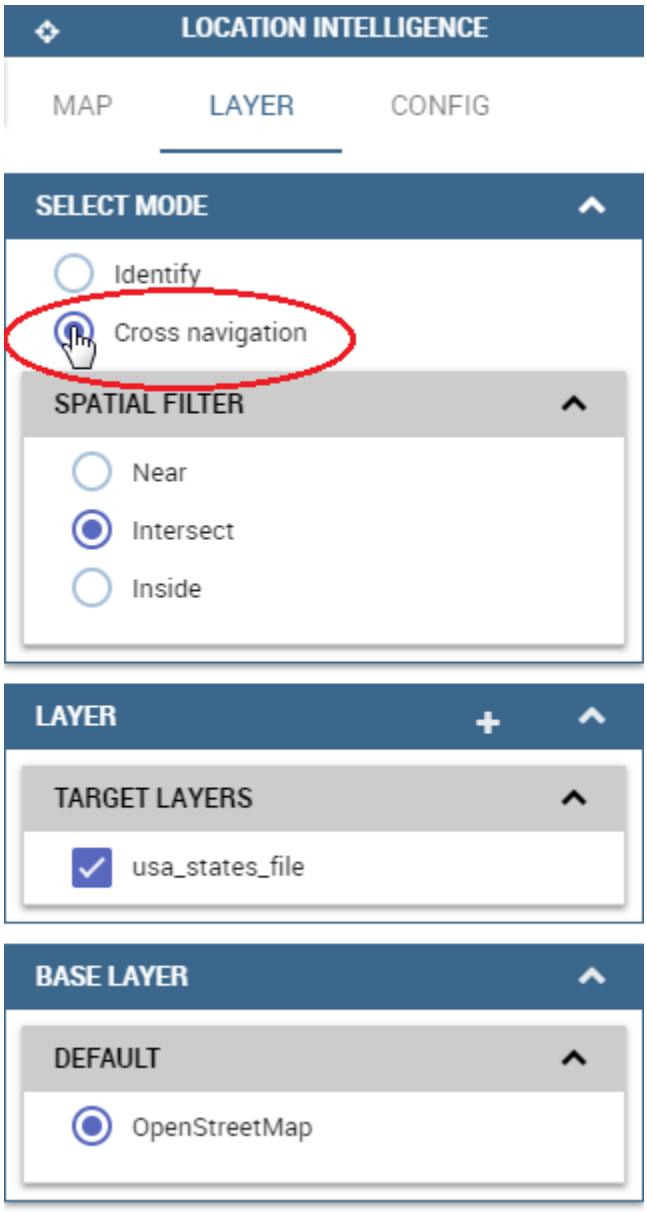


Fig. 5.319: Cross navigation option.

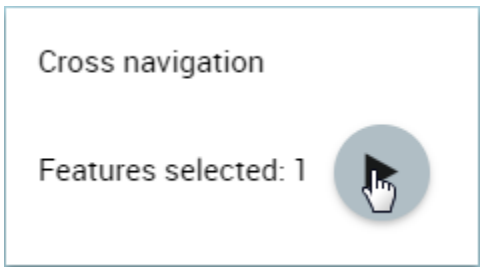


Fig. 5.320: Cross navigation popup.

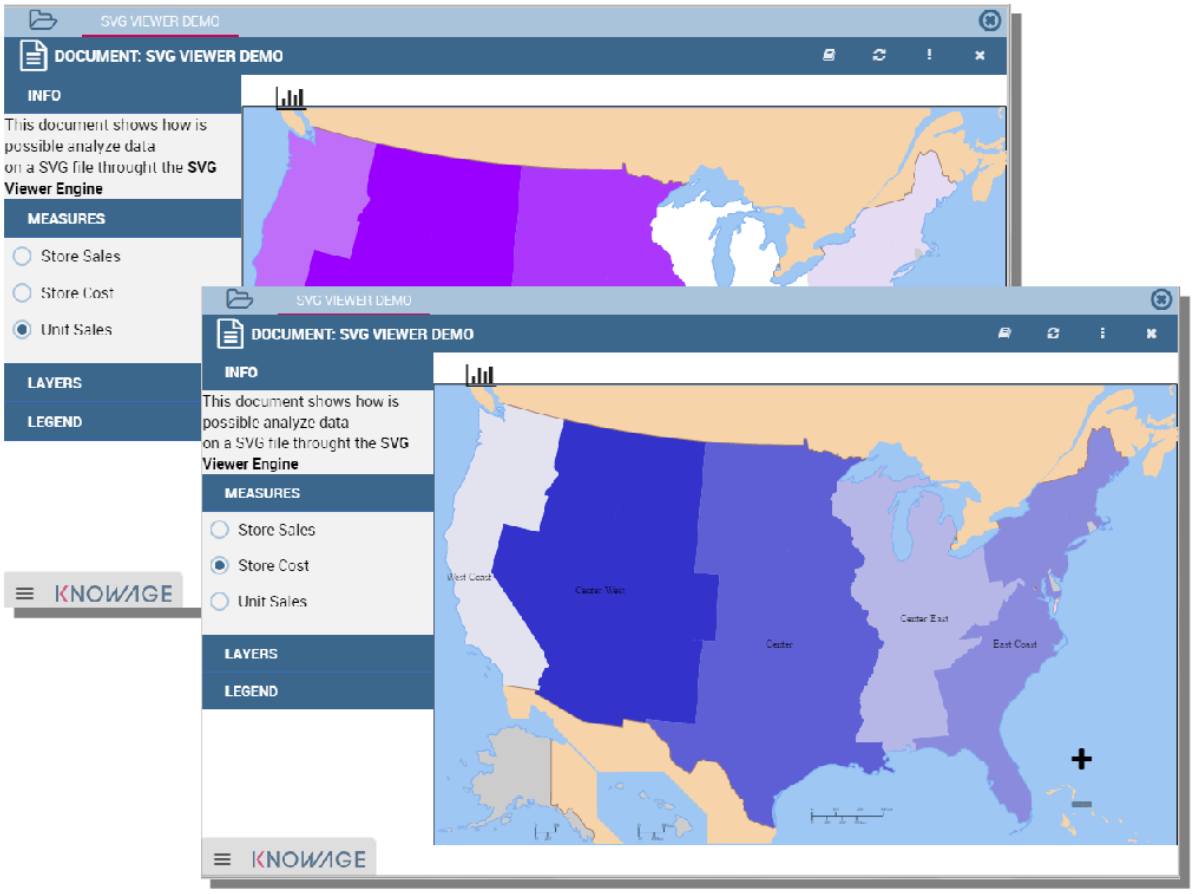


Fig. 5.321: SVG document visualization example.

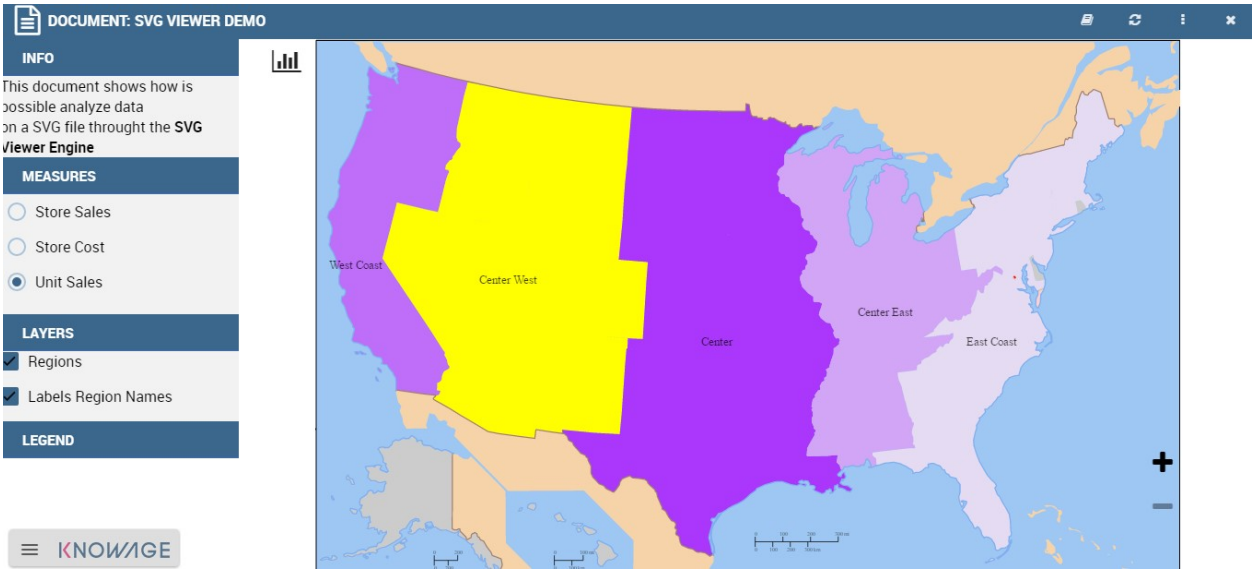


Fig. 5.322: SVG document example at the USA Regions level.

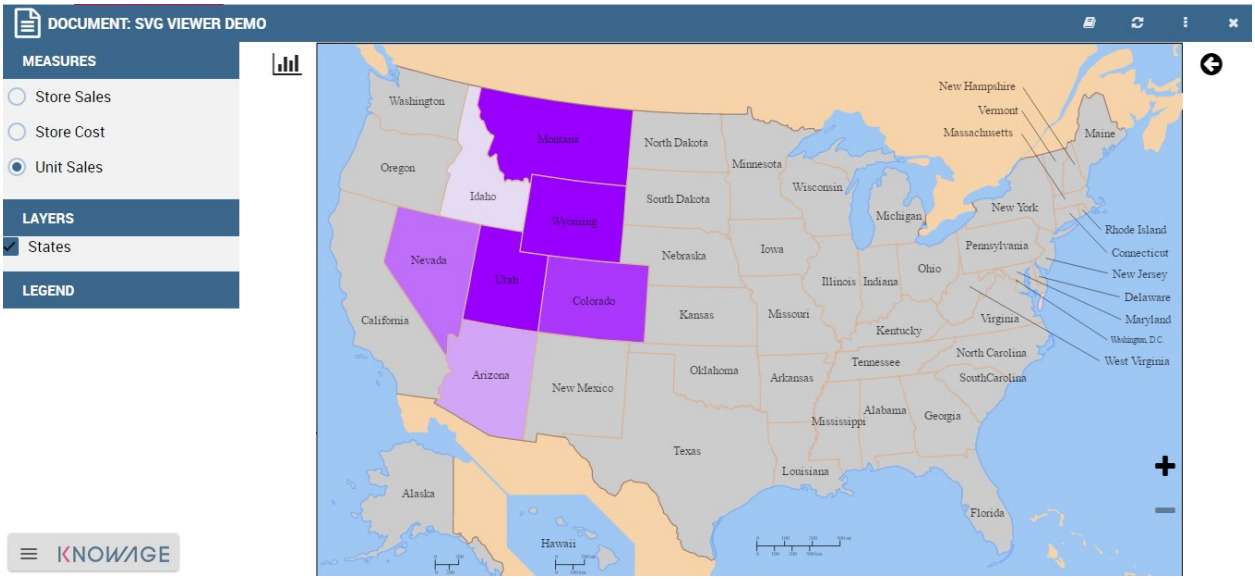


Fig. 5.323: SVG document example at the States level after the selection of the “Center West” Region.

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 throught specific hierarchies. In this context a hierarchy is a definition of three concepts:

- the hierarchy itself,

- the level,
- the member.

These three information are used from the system to recover the correct SVG into the catalogue.

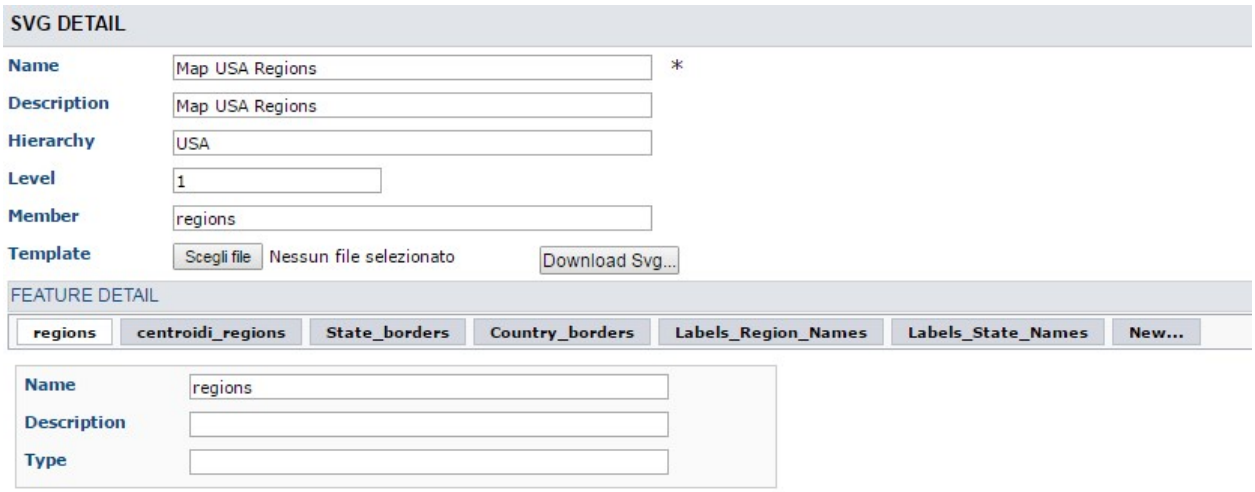


Fig. 5.324: Entering the hierarchy details.

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:

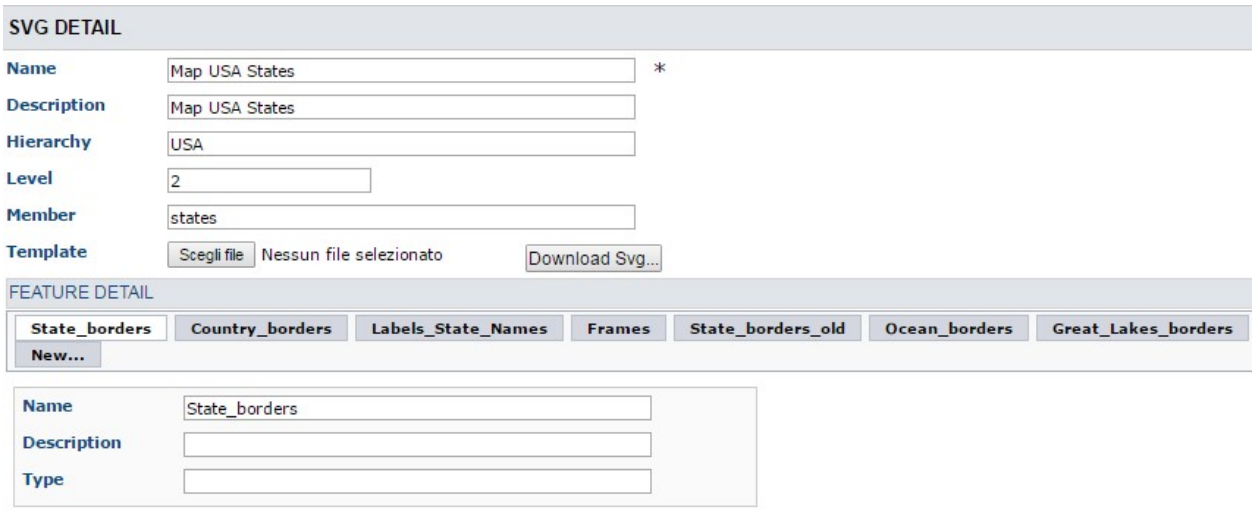


Fig. 5.325: Entering the hierarchy details.

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:

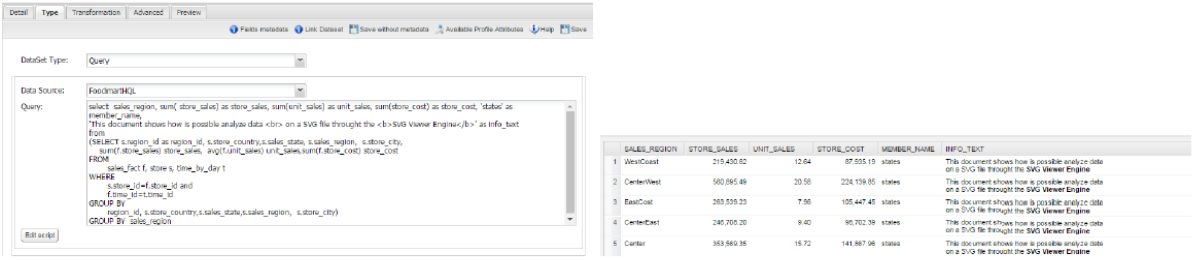


Fig. 5.326: Left. Dataset behind the SVG document. Right. Dataset preview.

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 5.22: Example of SVG code for template file.

```
<?xml version="1.0" encoding="UTF-8"?>
<MAP>
    <DATAMART_PROVIDER>
    <HIERARCHY name="USA">
    <MEMBER name ="regions" measure_dataset = "ds_regions" level="1" >
    <MEMBER name ="states" measure_dataset = "ds_states" level="2" >
    <HIERARCHY>
    <DATAMART_PROVIDER>
<MAP>
```

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 5.23: Example of SVG code for template file.

```
<MEMBER name ="regions" measure_dataset = "ds_regions" level="1" >
  <METADATA>
  <LAYERS>
  <MEASURES default_kpi="UNIT_SALES">
<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 5.24: Example of SVG code for template file.

```
    <METADATA>
      <COLUMN TYPE="geoid" column_id="sales_region" />
      <COLUMN TYPE="measure" column_id="store_sales" />
      <COLUMN TYPE="measure" column_id="store_costs" />
      <COLUMN TYPE="measure" column_id="unit_sales" />
      <COLUMN TYPE="drillid" column_id="member_name" />
      <COLUMN TYPE="info" column_id="info_text" />
```

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 actived or disactived directly by an action of the the final user. At least one layer must be defined.

Fig. 5.327: 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 possibile get its value dinamically throught an analytical driver by using the standard syntax $P<driver_url>
MEMBER	measure_dataset	Mandatory. The label of the dataset defined in Knowage Dataset configuration.
MEMBER	level	Mandatory. The number of the level. This value must match the level property into the catalogue for the hierarchy and the member specified.
COLUMN	TYPE	Mandatory. The type of the specific column. Possible values are: • **geoid**: mandatory. The engine uses this column to join the dataset records and the corresponding features in the svg. Also, it's the default value passed within the drill action to the svg of lower level (alternatively to the drillid property). • **measure**: mandatory. Defines the column like measure. All measures defined in this section will be shown into the detail panel (Measure section). • **drillid**: optional. Defines the alternative value to pass within the drill action to the next svg • **parentid**: optional. Defines the column that the system need to use for get correctly data linked to the parent value selected. • **crosstype**: optional. Defines the column that set the cross navigation type. Possible values are "cross" for external navigation or "drill" for internal navigation. If the single element returns null the link will be disabled • **visibility**: optional. Defines the column that throught

- **MEASURES** Measures are all the business values (KPI) that the user want to monitor throught this document type. Each measure defined in this section will be shown into the detail panel ("Measures" section) with a specific thematization and could be enabled or disabled directly by an action of the the final user. When the measure is active all its values are shown onto the SVG and each area has a specific tonality of the color in according to the threshold definition and its real value. All thresholds range are visualized into the "Legend" section of the detail panel as highlight in the following figure. Is possibile to choose the thematization logic that it could be as quantile, percentage, uniform or static. Next, we'll see both definitions (see Thresholds details).Remember, that at least one measure must be defined.

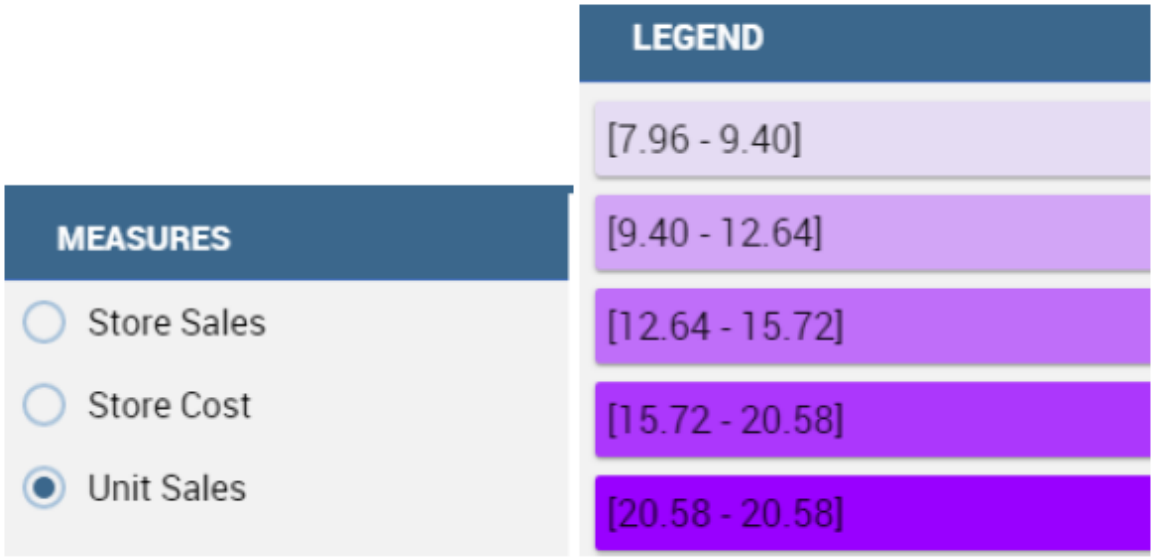


Fig. 5.328: Left. Measure panel. Right. Legend panel.

Listing 5.25: Code for setting the KPI into SVG document.

```
<MEASURES default_kpi="UNIT_SALES">
   <KPI column_id="STORE_SALES" description="Store Sales" >
     <TRESHOLDS type="quantile" lb_value="0" ub_value="none" >
        <PARAM name="GROUPS_NUMBER" value="5" />
     </TRESHOLDS>
     <COLOURS type="grad" outbound_colour="#FFFFFF" null_values_color="#CCCCCC" >
     <PARAM name="BASE_COLOR" value="#009900" />
     <!--<PARAM name="opacity" value="0.5" />--> </COLOURS>
   </KPI>
  <KPI column_id="STORE_COST" description="Store Cost" >
  <KPI column_id="UNIT_SALES" description="Unit Sales" >
<MEASURE>
```

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 definiton, 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 wont enable at the beginning, when we start the document execution. Its value must exist into the METADATA section as measuere type.
KPI	column_id	Mandatory. The column_id property 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: - **quantile**: it partitions the interval into N quintiles. • **perc**: it partitions the interval into subintervals whose extent represents a specific fraction of the overall interval extent. • **uniform**: it partitions the interval into N subintervals of a same extent. • **static**: it partitions the interval into smaller fixed-size subintervals, statically defined by the RANGE parameter
THRESHOLDS	lb_value	Mandatory. The lower value outside of which no value is considered.
THRESHOLDS	ub_value	Mandatory. The upper value outside of which no value is considered.
PARAM	name	Mandatory. Specify the parameter value necessaty to define correctly the temhatization. Its value depends by the threshold type. This attribite could be present more than once.
PARAM	value	Mandatory. It's the parameter name value.
PARAM	label	Optional. Specify the static labels for the legend when thresholds type is "static".
PARAM	value	Optional. It's the parameter label value.
COLOURS	type	Mandatory. Specify the logic type for defining colors range. It could be: • **static**: it assigns each subinterval a specific color that is statically defined. • grad: it assigns each subinterval a specific color that is dynamically calculated through a gradient function.

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 5.26: 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 outuput parameter as shown in figure below. It means that the identifer of the area selected is able to be passed. Other default output parameters are **Hierarchy**, **Member** and **Level**.

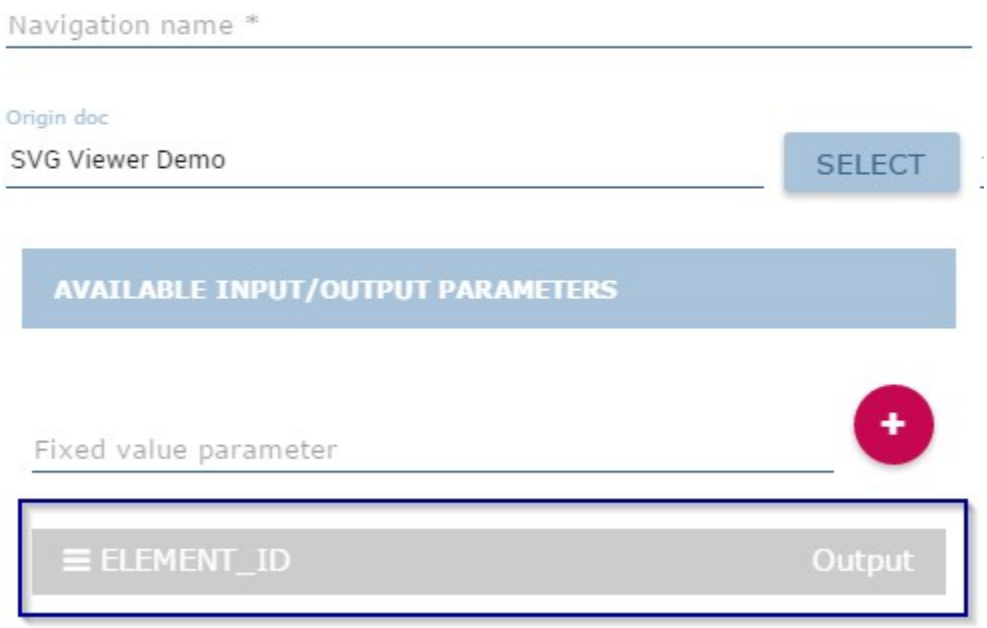


Fig. 5.329: Using the Cross Navigation definition to link to external documents.

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 andvanced algorithms andm 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 Pyhton scripts in an interactive way and visualise several outputs, including the powerful R graphics. Another importang 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 comunicate 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. 5.330: 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 lauched, 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 throught the template. Otherwise you can upload a file for Knowage datasets that cannot be changed from

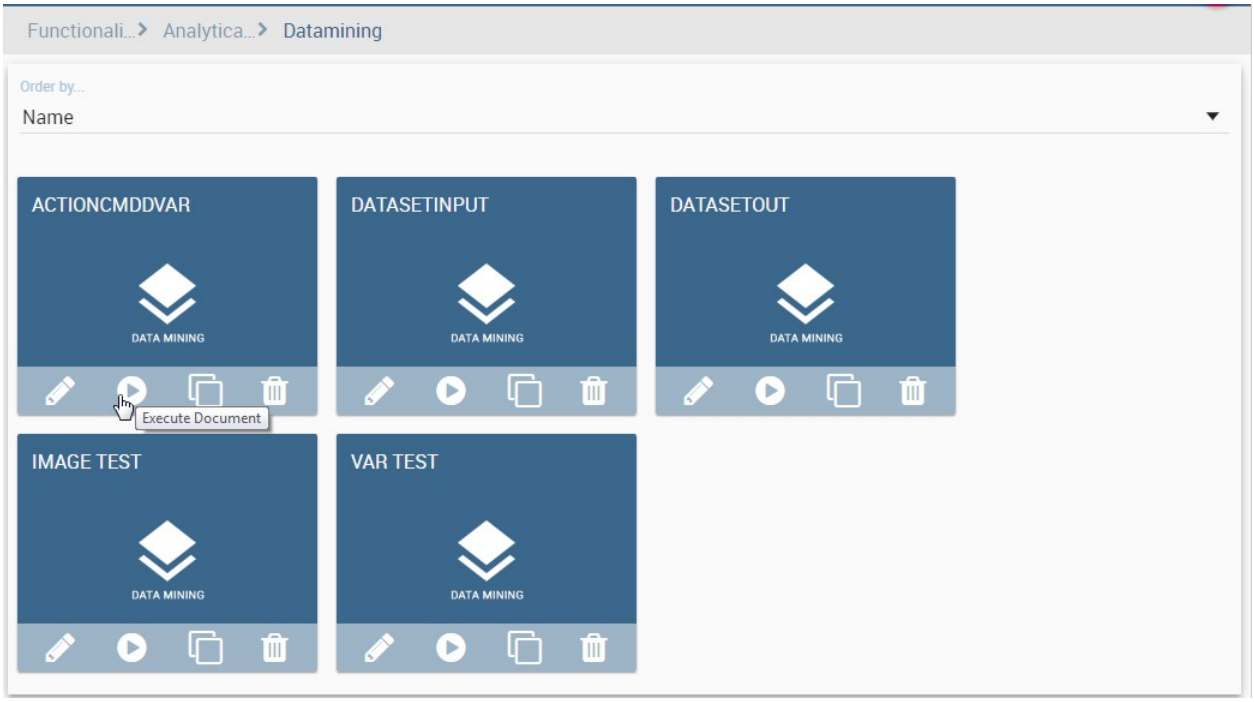


Fig. 5.331: Document browsing on server.

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

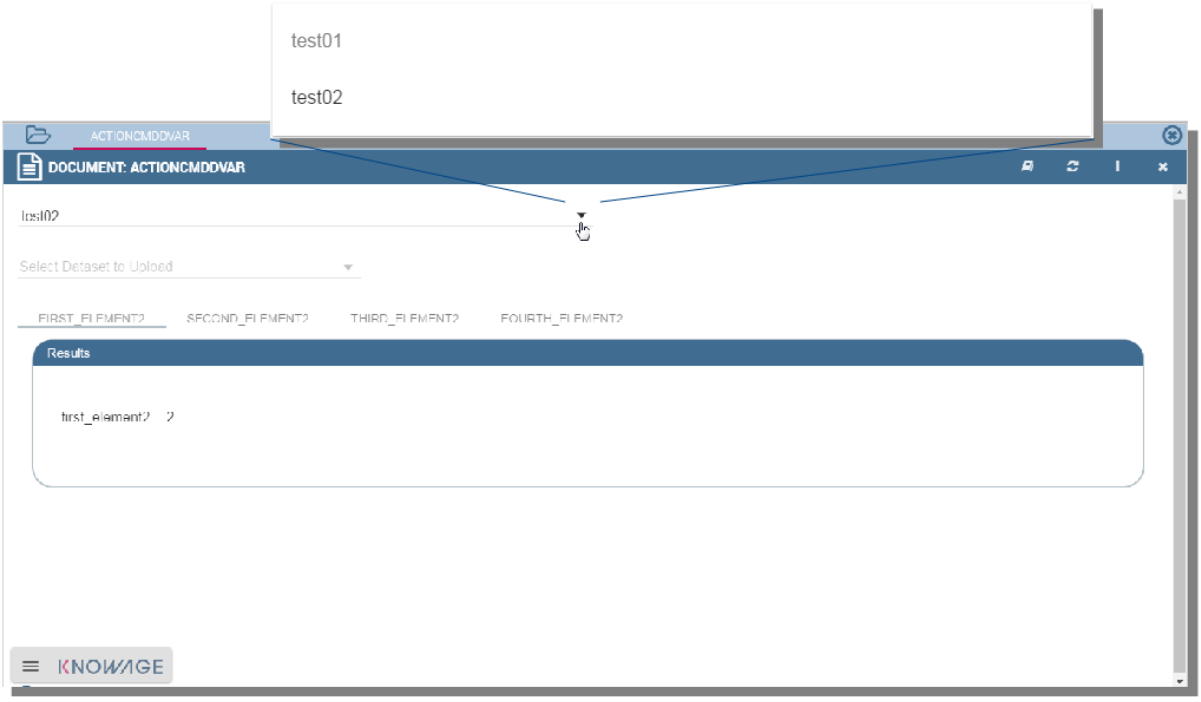


Fig. 5.332: Selecting the command to be applied.

```
<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(m)" mode="manual" label="second_element"/>
            <OUTPUT type="text" name="third_element" value="m"  function="fib" mode="manual" label="third_element"/>    <!-- problema-->
            <OUTPUT type="text" name="fourth_element" value=""  function="inca()" 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(m)" mode="manual" label="second_element2"/>
            <OUTPUT type="text" name="third_element2" value="m"  function="fib" mode="manual" label="third_element2"/>  <!-- problema-->
            <OUTPUT type="text" name="fourth_element2" value=""  function="inca()" mode="manual" label="fourth_element2"/>
        </OUTPUTS>
    </COMMAND>
</COMMANDS>
```

Fig. 5.333: The command tags.

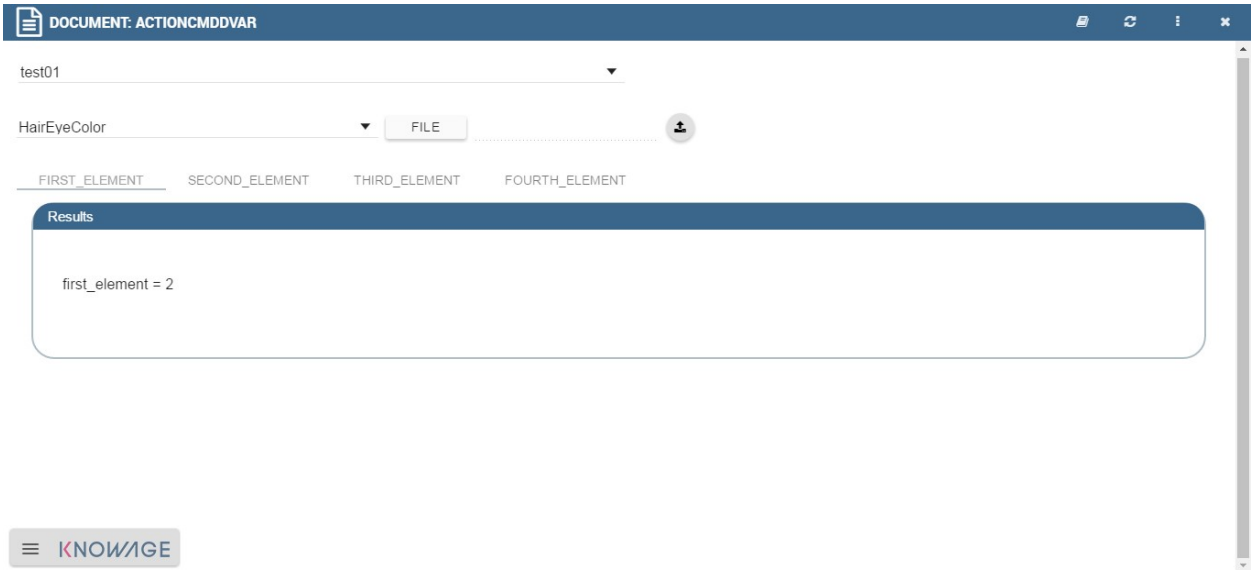


Fig. 5.334: Changing dataset related to a command.

Fig. 5.335: 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. 5.336: Setting parameters.

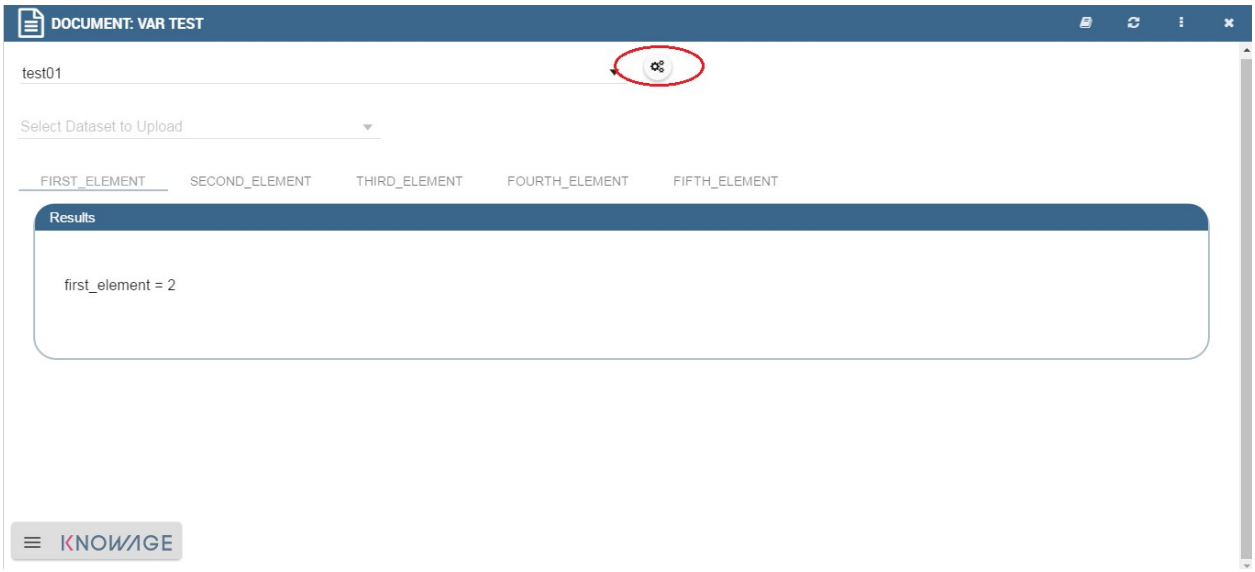


Fig. 5.337: Click on the double gear button to change the .

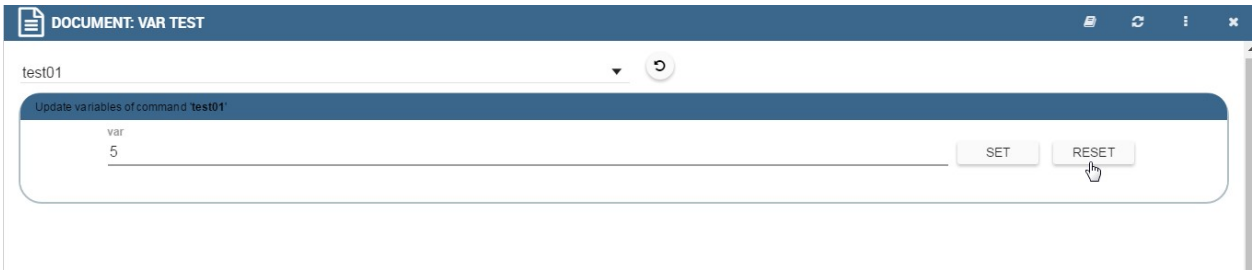


Fig. 5.338: Changing the parameter values.

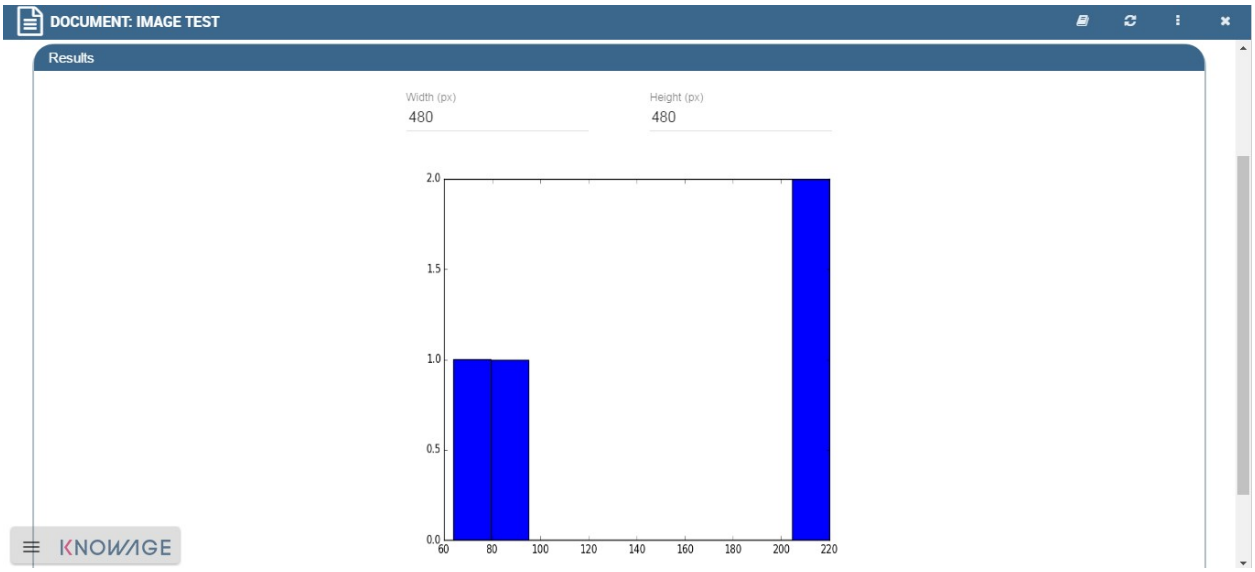


Fig. 5.339: Image outputs.

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.

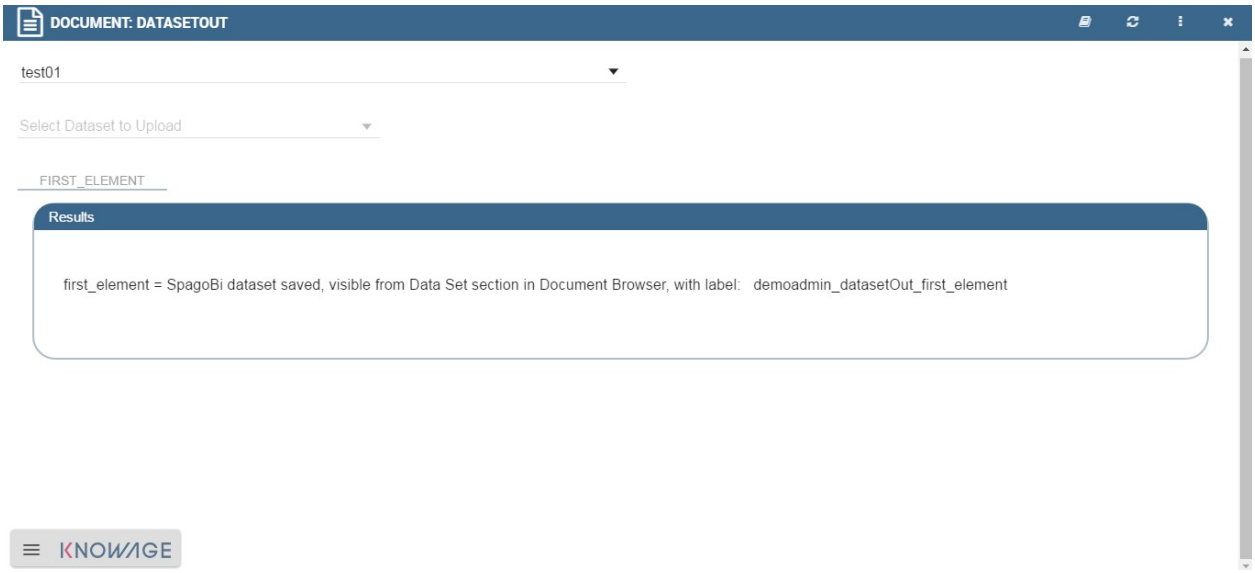


Fig. 5.340: Dataset outputs.

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, indipendently 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 esily 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.

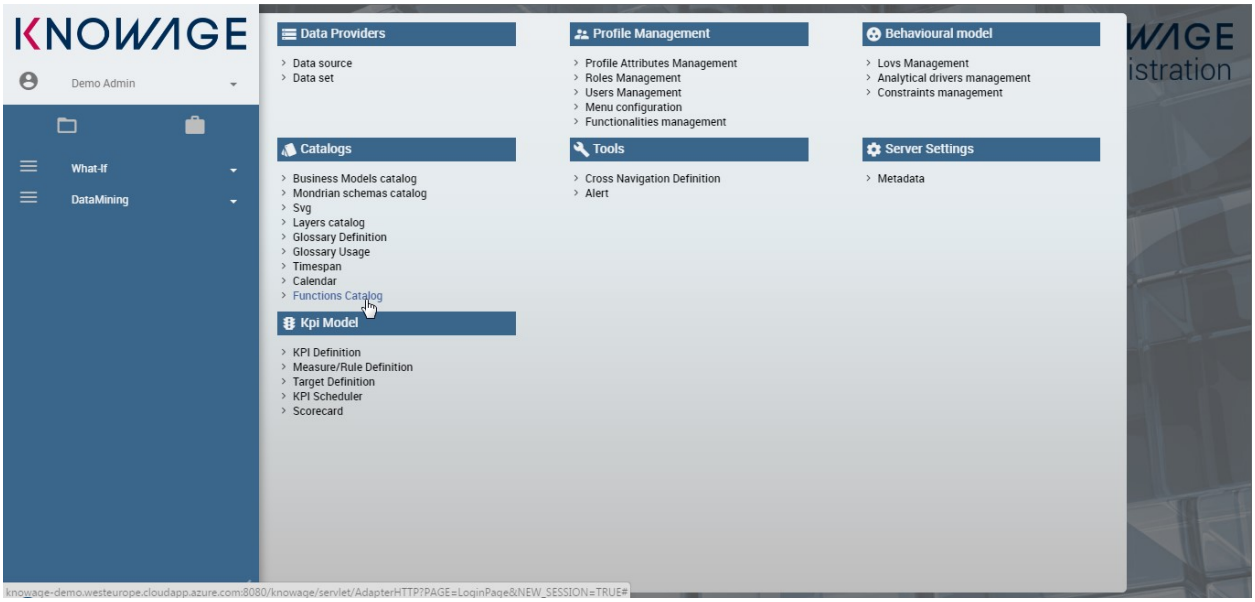


Fig. 5.341: Functions Catalog from Knowage menu.

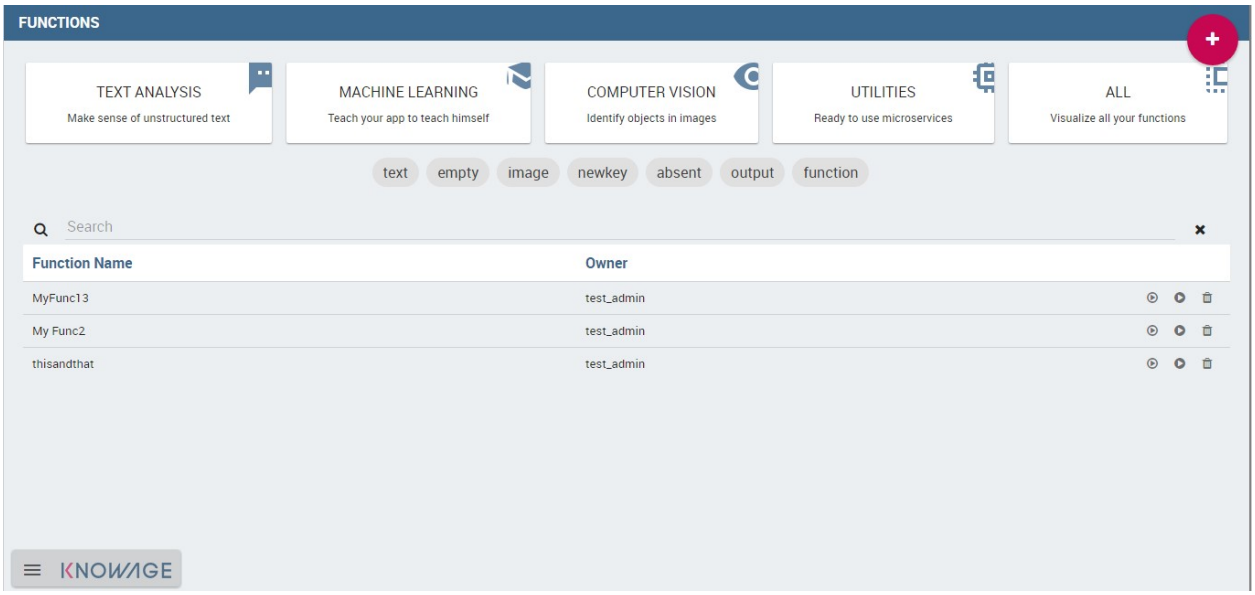


Fig. 5.342: Functions Catalog interface.

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.

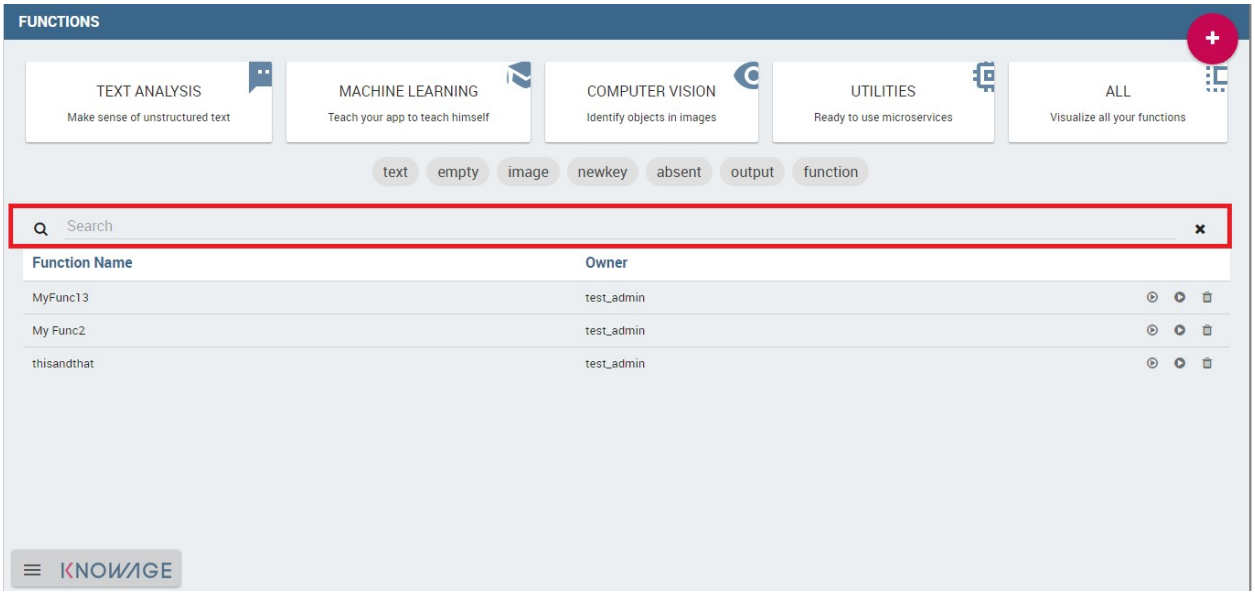


Fig. 5.343: Search box to look for a function.

We suppose here to select one category, which means to click on the category box, in order to be able to analize 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 fuctions associated to that category. Viceversa 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.

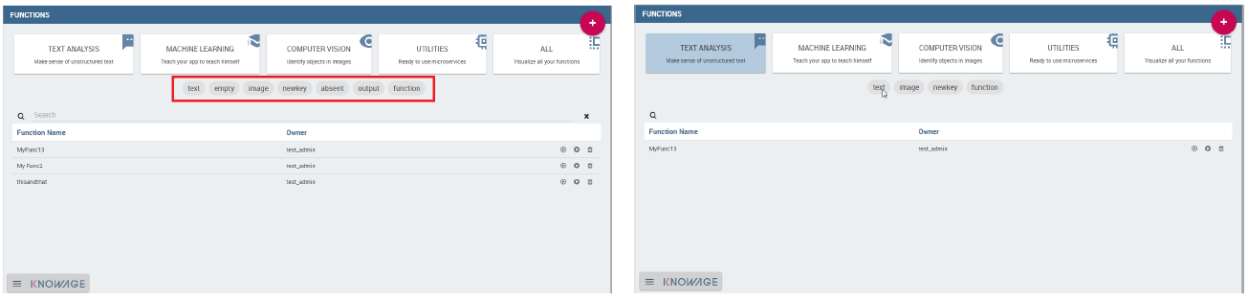


Fig. 5.344: Using tags and categories to look for functions.

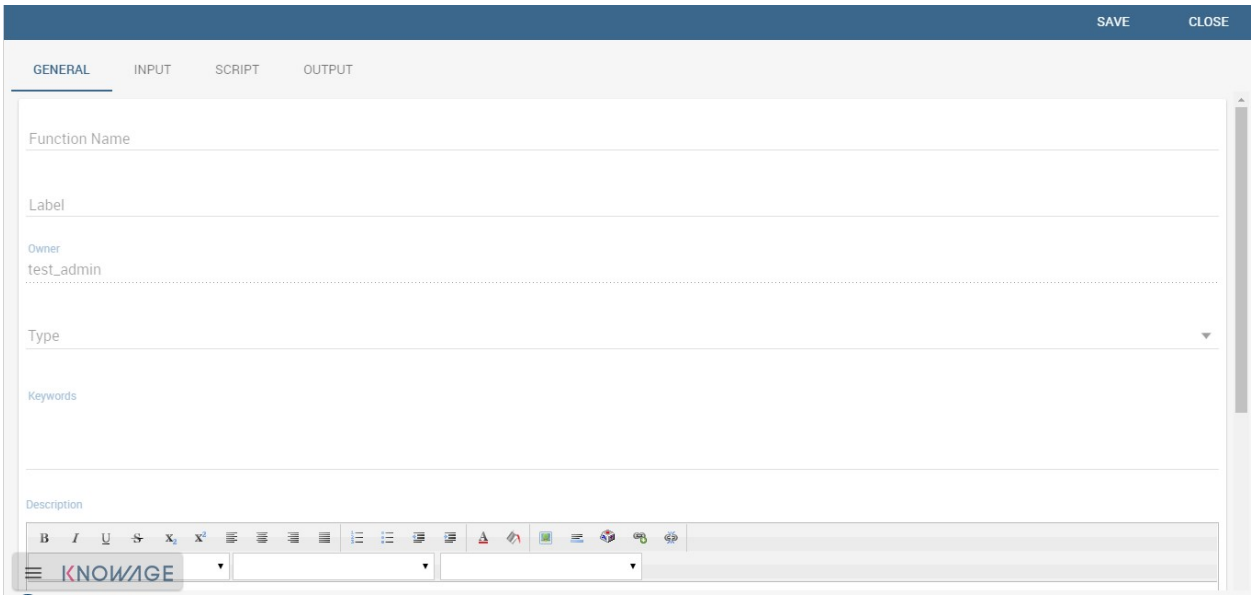


Fig. 5.345: Creating a new function.

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 were 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.

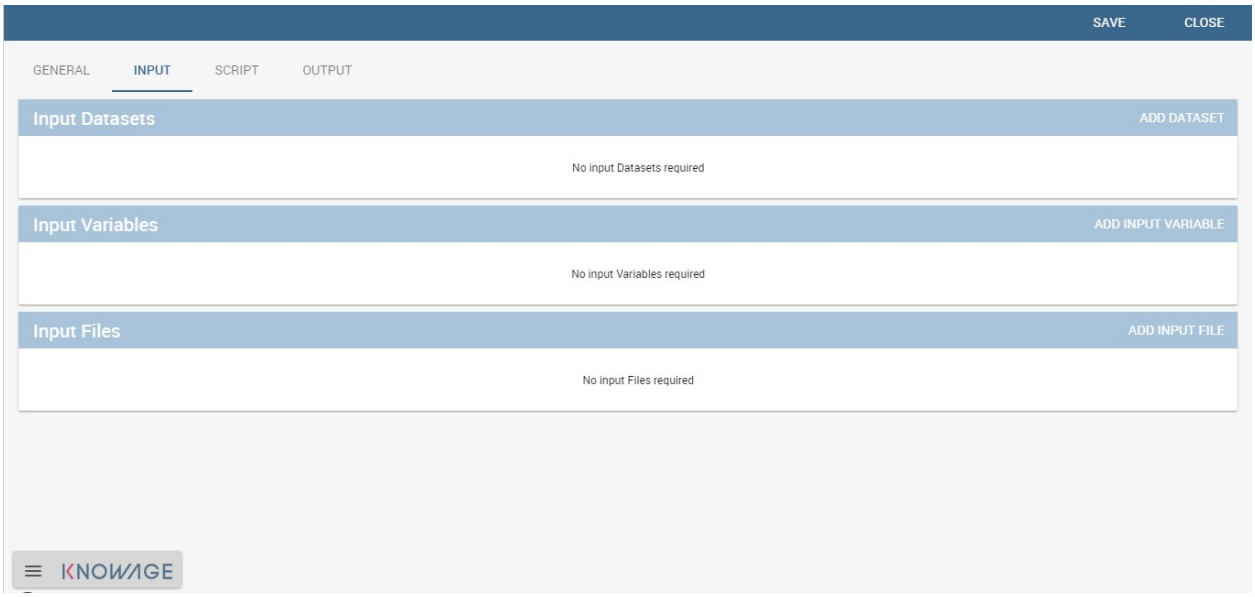


Fig. 5.346: Input tab.

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. 5.347: 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.

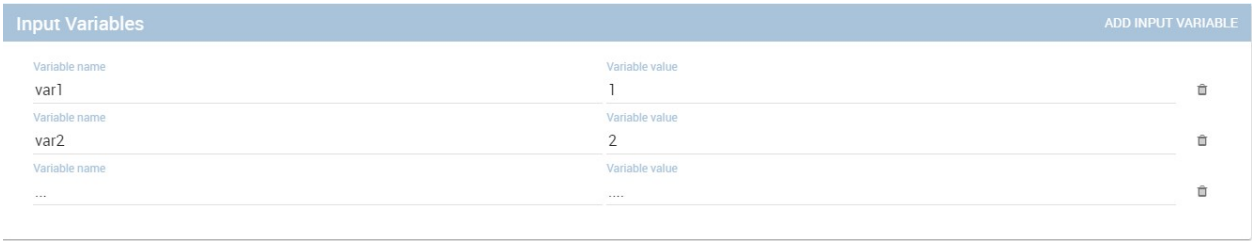


Fig. 5.348: The variable input of the function settings.

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.

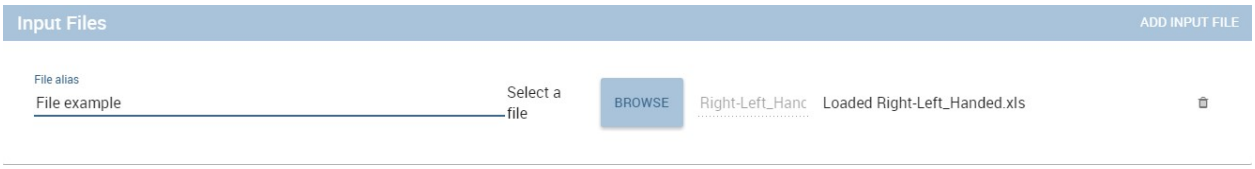


Fig. 5.349: The file input of the function settings.

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**.

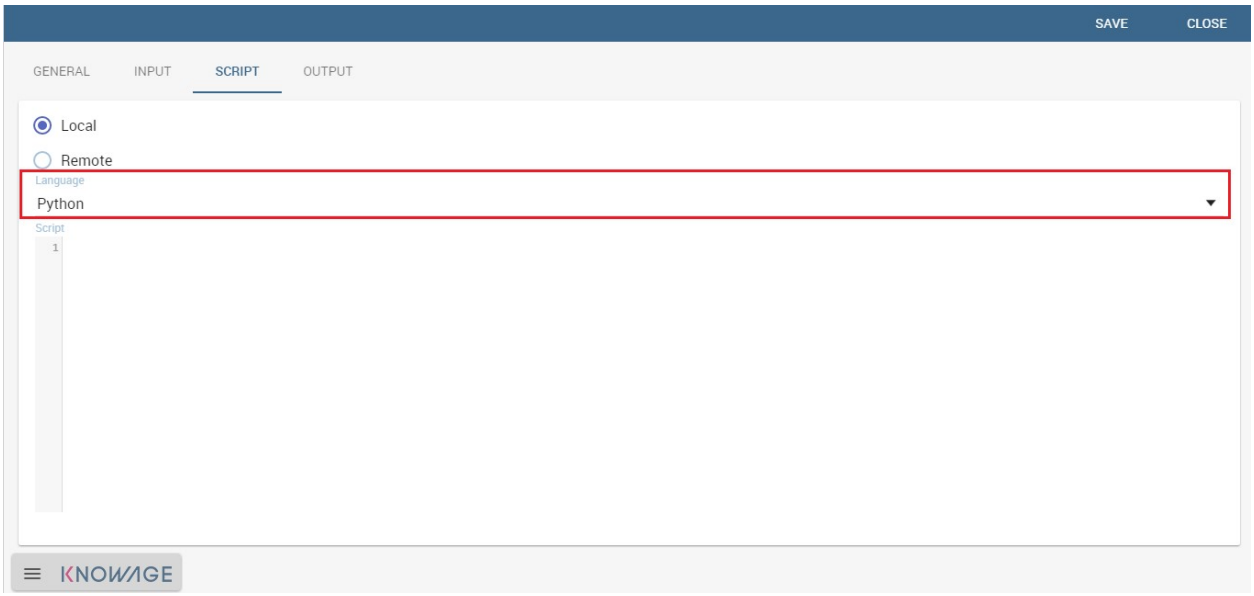


Fig. 5.350: The script tab.

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.

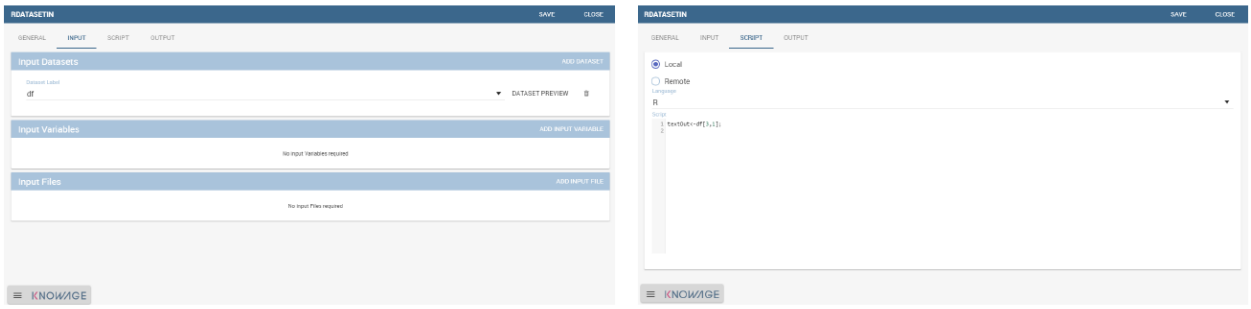


Fig. 5.351: Using the dataset dataframe generated by the software to edit the R script.

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 5.27: 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

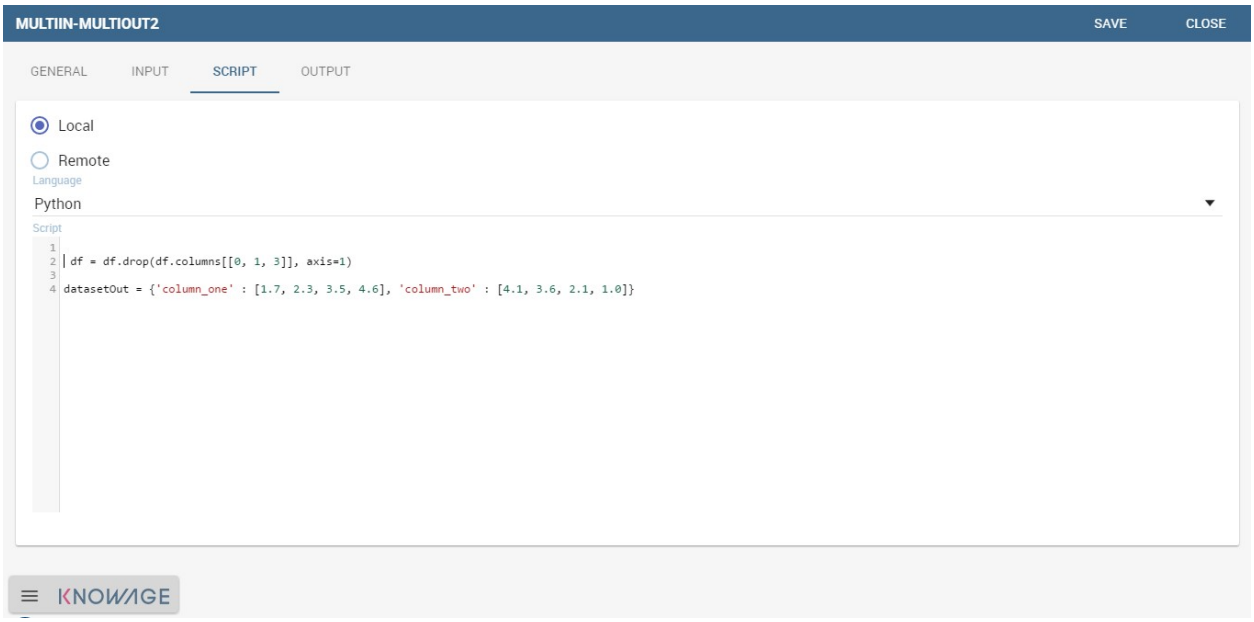


Fig. 5.352: Using the dataset dataframe of the pandas libraries generated by the software to edit the Python script.

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.

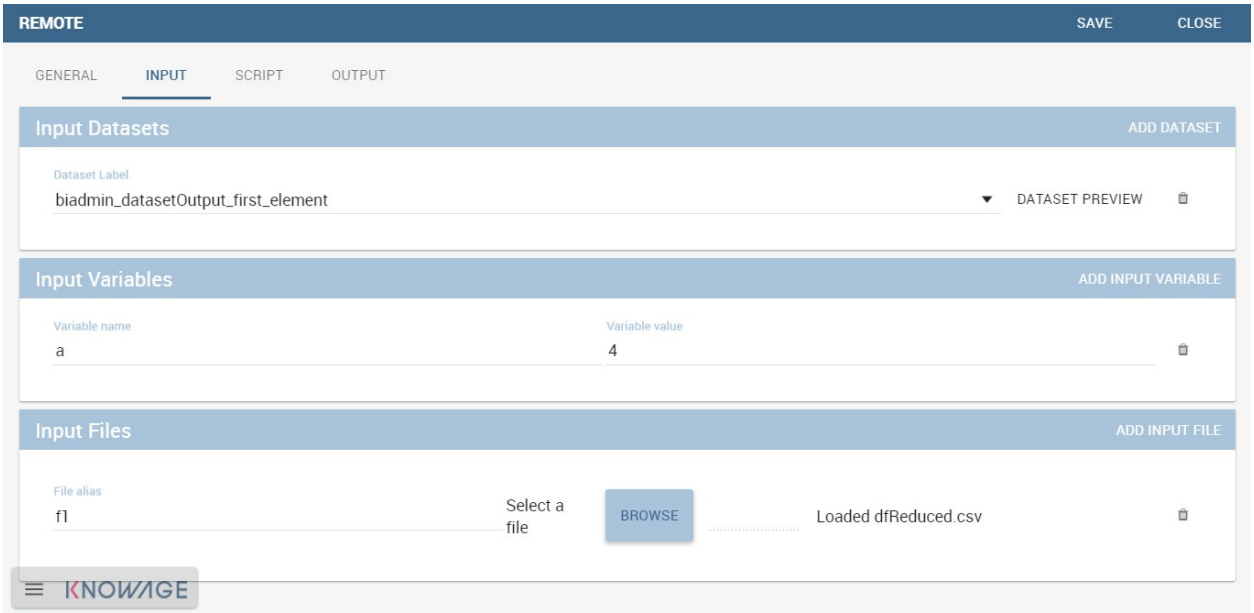


Fig. 5.353: Input definition for remote function.

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 5.28: JSON format for remote function

```
[
    {
        "type":"variablesIn",
        "items":
        {
           "demoVarName1":"3",
           "demoVarName2":"3"
        }
    },
    {
        "type":"datasetsIn",
        "items":
        {
            "demoDsName1":"df1",
            "demoDsName2":"df2"
        }
    },
    {
        "type":"filesIn",
        "items":
        {
            "demoFileAlias1":
        {
                "filename:filename1,
                base64 :..
```

(continues on next page)

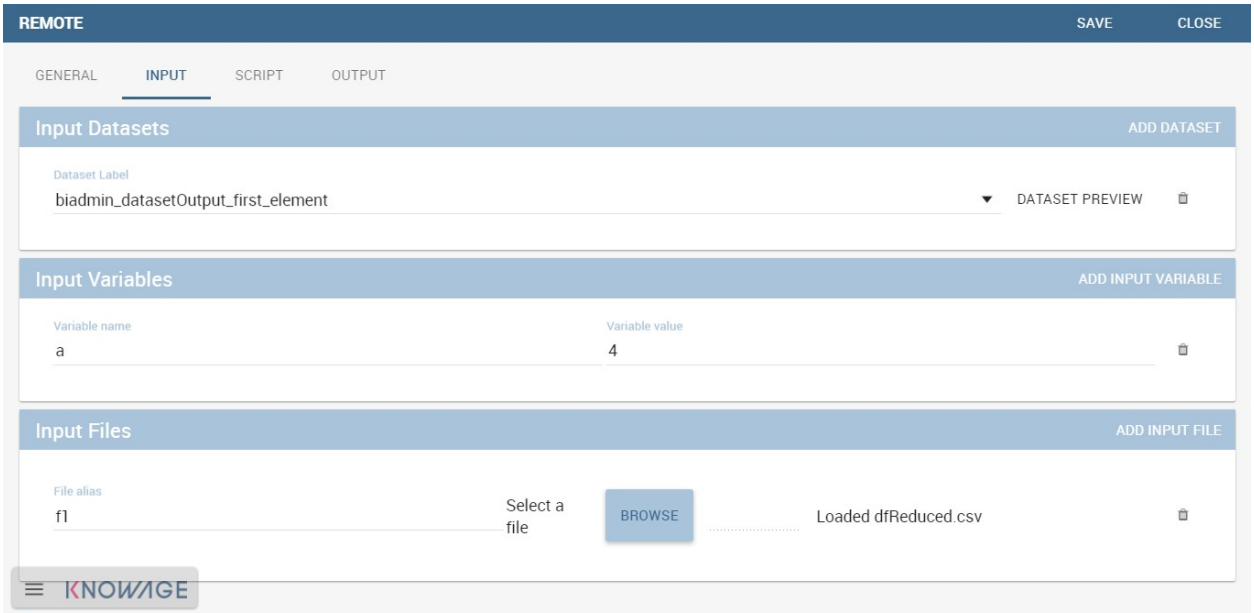


Fig. 5.354: Remote function definition.

(continued from previous page)

```
        },
            "demoFileAlias2":
        {
                "filename:filename2,
                base64 :..
        }
        }
    }
]
```

When the call runs successfully, the remote function must answer answer with a JSON element like the one exhibited in Code below.

Listing 5.29: JSON answer of a remote function

```
{
"resultType":"Image",
"result":".image content in base64.",
"resultName":"res"
},
{
"resultType":"Dataset",
"result":"outDatasetLabel",
"resultName":"datasetName"
},
{ "resultType":"File", "result":
{
"filesize":"54836", --optional
"filetype":"image/jpeg", --optional
"filename":"chart.jpg", --optional
"base64":".file content in base64." },
"resultName":"fileToBeSave"
}
```

If an error occur the function must returns the lines as shown in JSON format for remote function.

Listing 5.30: JSON answer of a remote function

```
{
  "service":"",
  errors":[
    {
     "message":"Here the error message."
     ]
}
```

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:

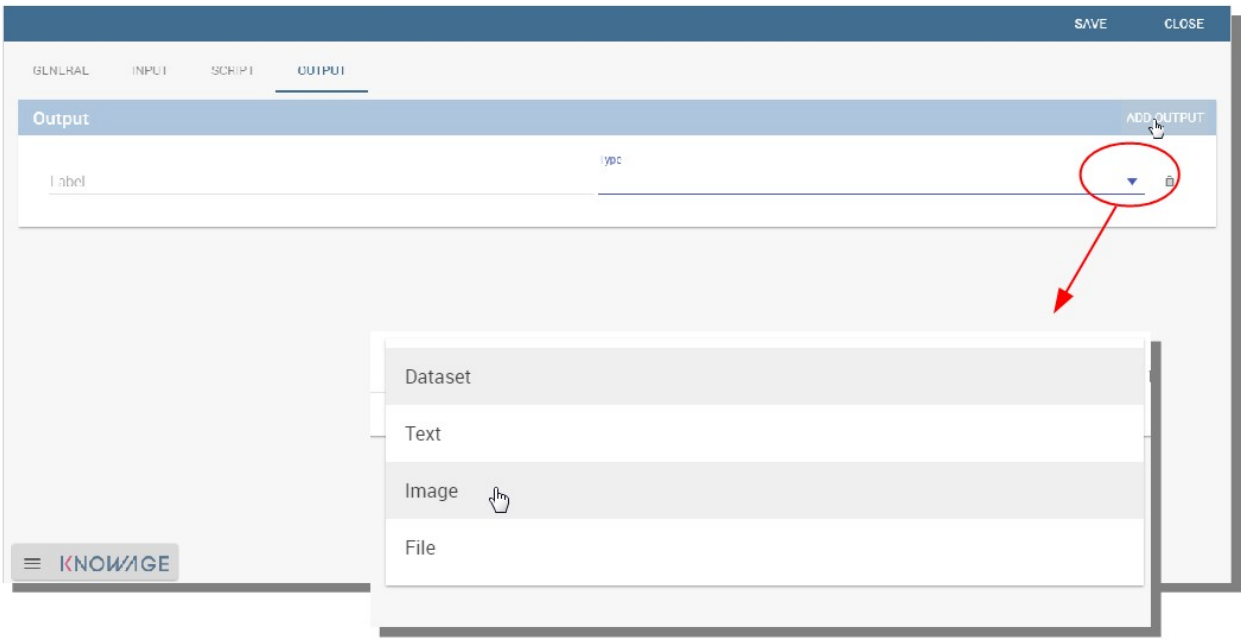


Fig. 5.355: Choosing the output type in the function definition.

- **Dataset**: the function will return a set of records as a 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.

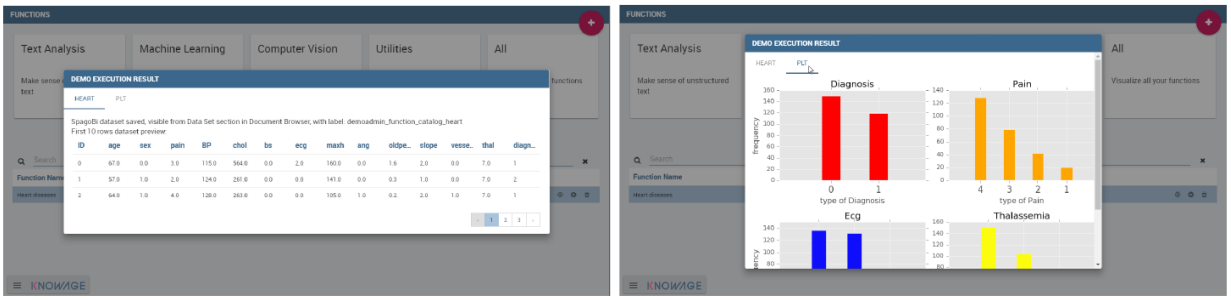


Fig. 5.356: Execution of demo for a function.

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.

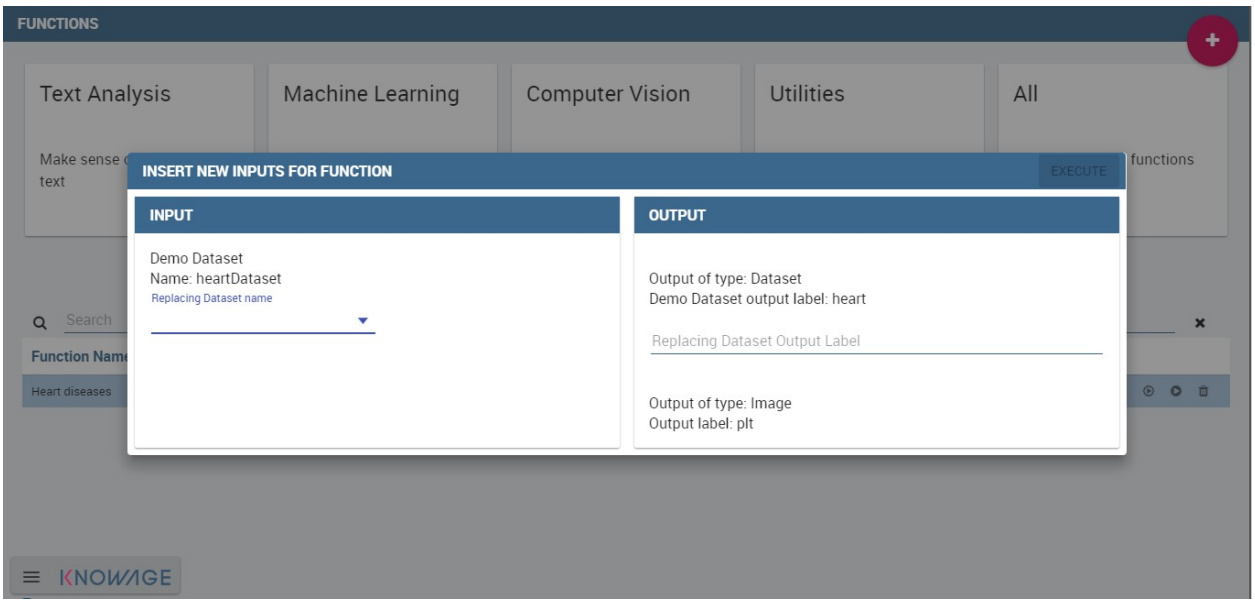


Fig. 5.357: Inserting new data values for function.

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’ stucture (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.

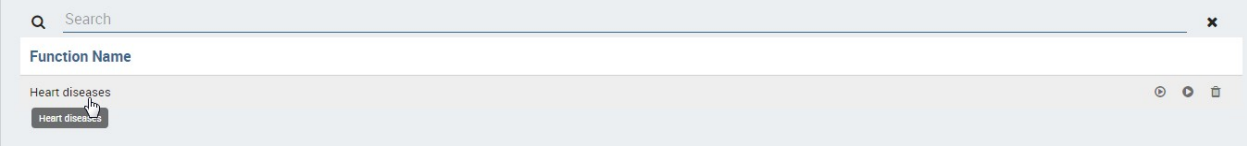


Fig. 5.358: Clicking on function name to modify it.

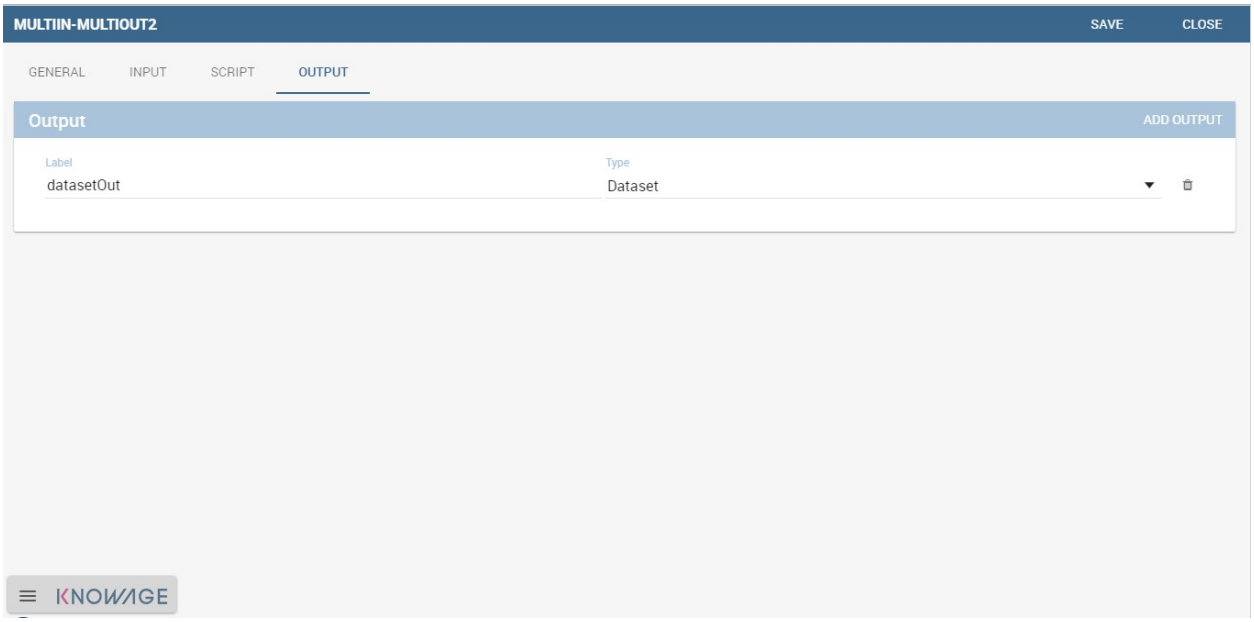


Fig. 5.359: Defining the output example.

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 lbrary 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 istalled 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 5.31: Linux and Tomcat example

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<DATA_MINING>
<DATASETS>
        <DATASET name="fileDS" readType="table" type="file" label="Dataset_Label_01 " canUpload="true
↪">
        <![CDATA[ ...read_options...]]>
</DATASET>
</DATASETS>

<SCRIPTS>
        <SCRIPT name="Script_Name_01" datasets="fileDs"  label="Script_Label_01">
        <![CDATA[.... action_to_call<-function(x){ ... }]]>
</SCRIPT>
        <SCRIPT name="Script_Name_02" datasets="fileDs"  label="Script_Label_01">
           <![CDATA[... z1<-'$P{var1}' ... ]]>
</SCRIPT>
        <SCRIPT name="Script_Name_03" label="Script_Name_03">
        <![CDATA[... z2<-$P{var2} ... ]]>
</SCRIPT>
</SCRIPTS>

<COMMANDS>
     <COMMAND name="command1" scriptName="Script_Name_01" label=" Command_Label_01" mode="auto">
↪                                    <OUTPUTS>
          <OUTPUT type="image" name="a" value="x" function="plot" mode="auto" label="Output_Label_01"/
↪>                                   <OUTPUT type="image" name="c" value="z,k" function="biplot" mode=
↪" manual" label="Output_Label_02"/>                                    <OUTPUT type="text" name="d" value="y
↪" mode="manual" label="   Output_Label_03"/>                                    </OUTPUTS>
     </COMMAND>
<COMMAND name="command2" scriptName="Script_Name_02" label=" Command_Label_01" mode="manual" action=
↪"function1(x)">                    <VARIABLES>
          <VARIABLE name="var1" default="valuevar1"/>
     </VARIABLES>
     <OUTPUTS>
          <OUTPUT type="text" name="c" value="z" function="function2(y,z)"  mode=" manual" label=
↪"Output_Label_01"/>                    </OUTPUTS>
</COMMAND>
  <COMMAND name="command3" scriptName="Script_Name_03" label=" Command_Label_03" mode="manual"action=
↪"action_to_call">                    <OUTPUTS>
        <OUTPUT type="text" name="e" value="z2" mode="manual" label=" Output_Label_01">
↪                                    <VARIABLES>
        <VARIABLE name="var2" default="valuevar2"/> </VARIABLES>
     </OUTPUT>
          <OUTPUT type="image" name="f" value="" function="rectf(z)" mode="auto" label="Output_Label_02
↪"/>                                 </OUTPUTS>
</COMMAND>
</COMMANDS>
</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 applyed 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 fileds as label, name, engine and datasource. Then you must upload the template file cliking on the icon available at the bottom of the form, highlighted below.

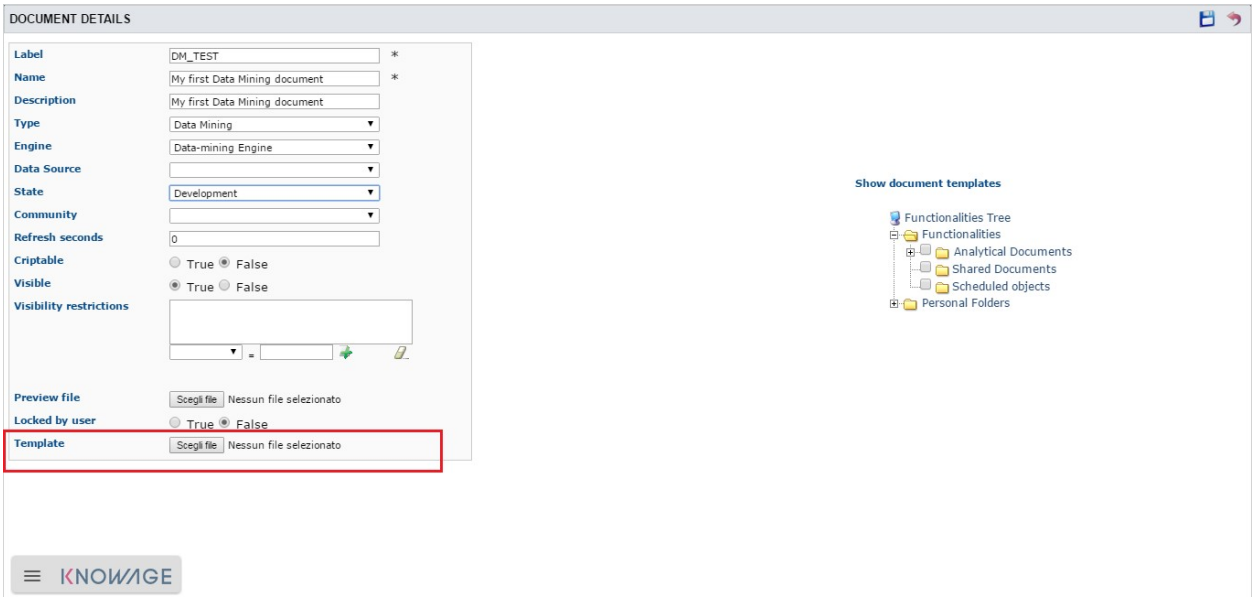


Fig. 5.360: Creating a new function.

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.

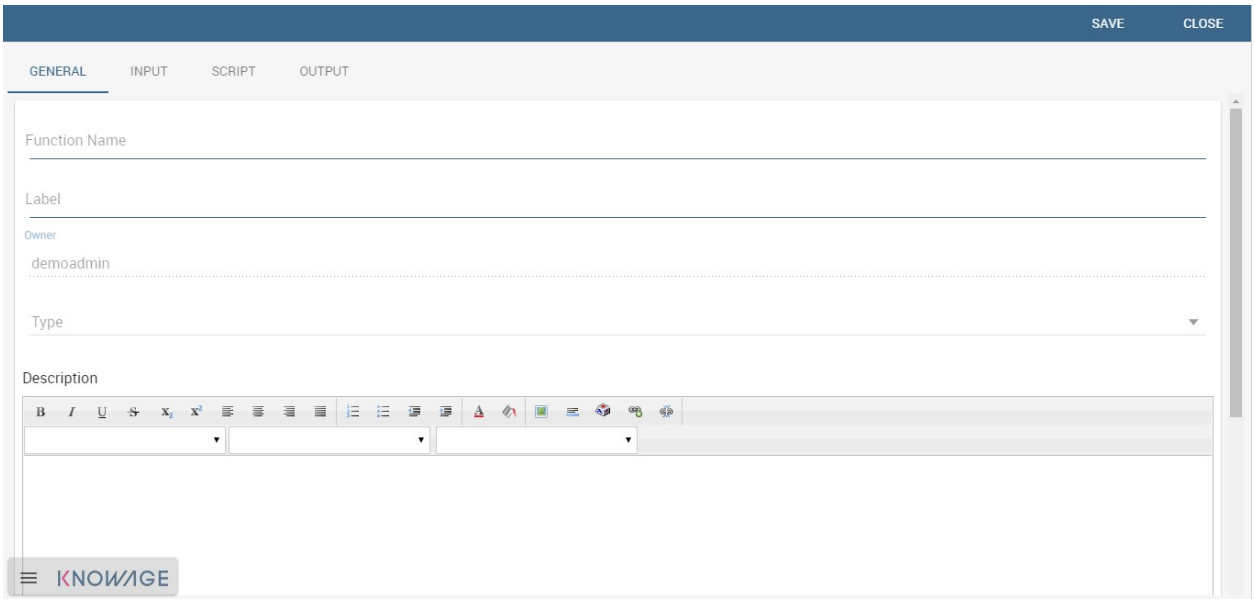


Fig. 5.361: Creating a new function.

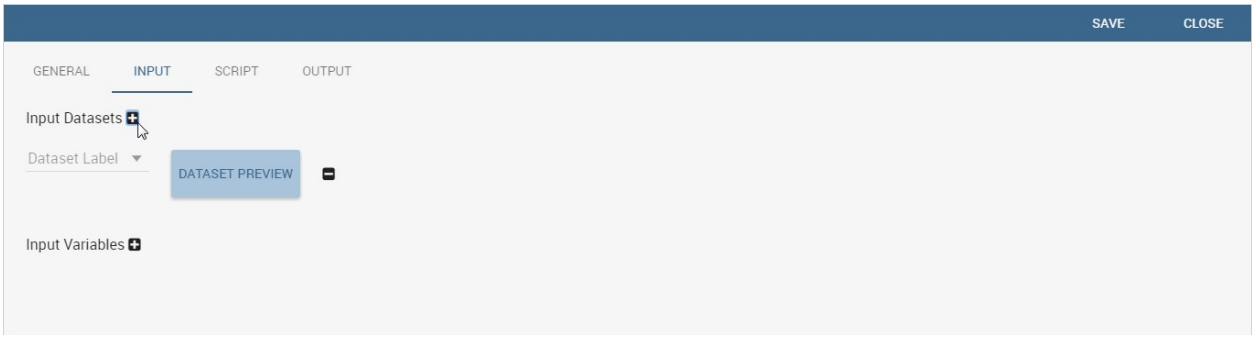


Fig. 5.362: Input tabs.

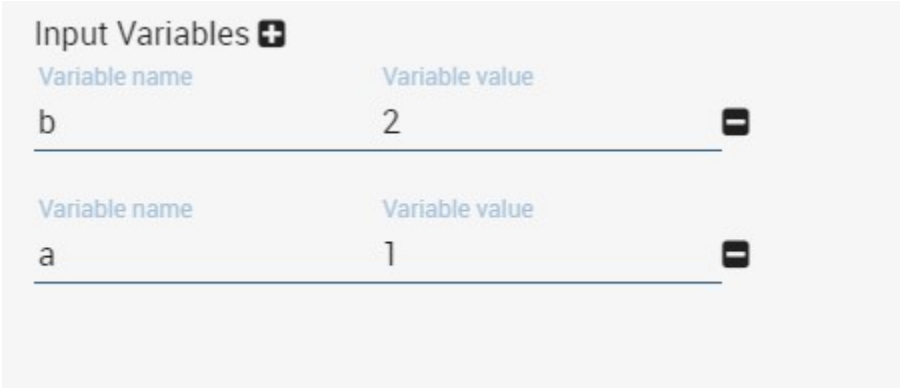


Fig. 5.363: Inserting variables.

- **Script**: here is where the user is required to have knoledge of R or Python language. Figure below shows an example.

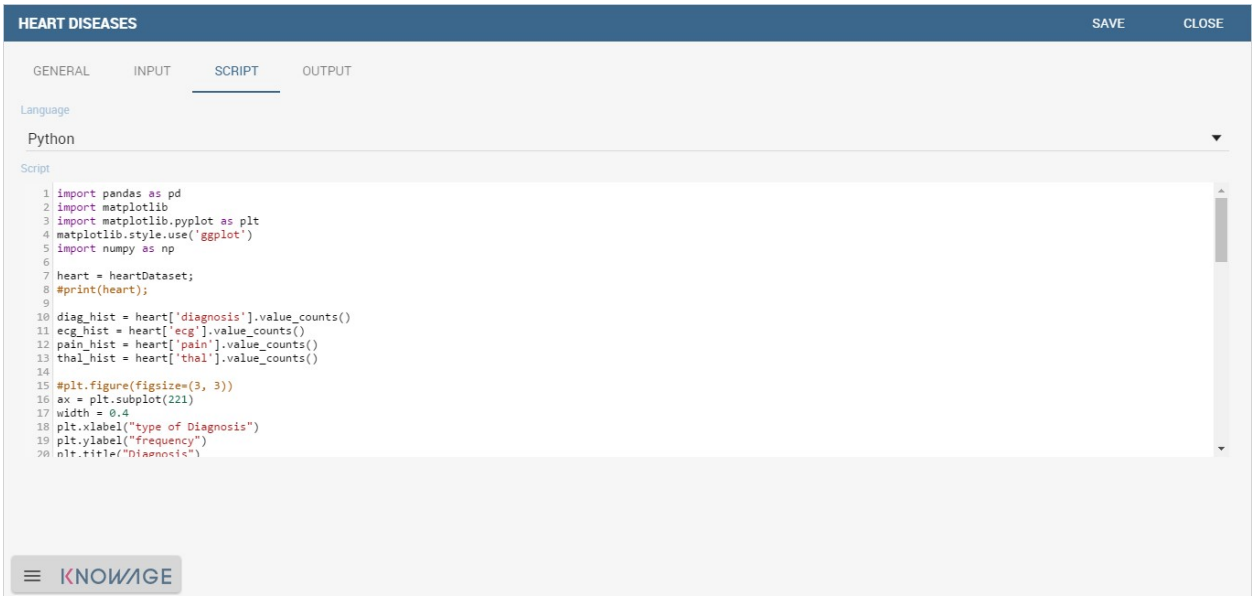


Fig. 5.364: Typing Python script

- **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. 5.365: Choosing how to visualise outputs.

Then save and you are ready to use the function.

5.15 Network

Network theory is the study of graphs as a representation of either symmetric relations or, more generally, of asymmetric relations between discrete objects. It has applications in many disciplines including statistical physics, particle physics, computer science, electrical engineering, biology, economics, operations research, climatology and sociology. Applications of network theory include logistical networks, the World Wide Web, Internet, gene regulatory networks, metabolic networks, social networks, epistemological networks, etc.; see List of network theory topics for more examples.

Using Knowage you can perform Network Analysis with knowagenetworkengine. The engine allows you to generate a network starting from your data. Now let see how. What is a network? It's a set of nodes linked by relations. So it's a 3-tuple where the entities are:

- Source node;
- Destination node;
- Relation.

In Knowage a document is the composition of a template and the data.

- In this case data are a dataset with at least the three columns described above.
- The template is the mapping of the columns of the dataset on graph properties (column *?* source node, column *?* destination node,)

5.15.1 Template

Before entering this section we underlight that Knowage uses cytoscapeweb library (http: //cytoscapeweb.cytoscape.org/) to render the network, so sometimes in the description we will refer to cytoscapeweb documentation

knowagenetworkanalysisengine supports different types of network definitions. If you have your network defined in GRAPHML or XGMML notation you can upload that file as template of the document and that's it. After this operation you can open your network using Knowage.

On the other hand if you want to create a dynamic network that gets data from your datasource you should create a dynamic document. In this case the structure of a template is typed in code below:

Listing 5.32: Template structure.

```
<NET>
      <NETWOK_DEFINITION>
         Definition of the network with mapping and rendering options

         <options>
            general rendering options: tooltip, shape of the network, interaction buttons,...
         </options>

         <dataset_mapping_LIST>
            mapping between dataset columns and network properties: data, node shapes, node colors,....
         </dataset_mapping_LIST>
      </NETWOK_DEFINITION>
      <info>
            info of the document: left tab content
      </info>
<drill>
      Cross navigation configuration
</drill>
</NET>
```

In the following we explain shortly the meaning of the cobe above.

The **NETWOK_DEFINITION** contains the definition of the network: nodes, edges, shapes, colours It has two children:

- **Options**: it contains the general options of the network. There are 2 type of options:
  - Network options. These are options that drive the rendering of all the network. For example where to put the navigation commands or the shape of the network (circular, radial,). You can find the list of available properties here http://cytoscapeweb.cytoscape.org/documentation/visualization and the list of layouts here http://cytoscapeweb.cytoscape.org/documentation/layout

  - Edge/nodes properties. General visual properties for the nodes and edges. The syntax for these settings is in next Syntax for edge/nodes properties:

Listing 5.33: Syntax for edge/nodes properties.

```
<options>
   <visual_style>
     <nodes>
        Nodes properties
     </nodes>
     <edges
        Edges properties
     </edges>
   </visual_style>
</options>
```

Tooltip is a special Edge/node property. The tooltip contains a set of proerty/value couple and the syntax is typed in the next Syntax for tooltip,an edge/nodes property:

Listing 5.34: Syntax for tooltip,an edge/nodes property.

```
<nodes (or edge)>
  <tooltip_LIST>
  <tooltip property="OBJ PROPERTY" text="PROPERTY LABEL TEXT"/>
  </tooltip_LIST>
</nodes (or edge)>
```

Where OBJ PROPERTY property is the name of the property (for example id) and PROPERTY LABEL TEXT is the text you'll see as label of the property in the tooltip. You can find the list of available properties here: http://cytoscapeweb. cytoscape.org/documentation/visual_style

- **Dataset_mapping_LIST**: this section maps the columns of the dataset on properties of the graph. This is done with the tag dataset_mapping. There are two possibilities:
  - Map a column of the dataset on a property of the graph and the syntax is showed in Listing 5.32:

Listing 5.35: Syntax for tooltip,an edge/nodes property.

```
<dataset_mapping element="source" column="sourceId" property="id"/>
```

Where:

- – element: is the element where we want to apply the property. It can be source, target (for nodes) and edge;
- – property: the property of the network object we wan to set;
- – column: the label of the dataset column we want to map.

The list of available node and edge properties is here http://cytoscapeweb.cytoscape.org/documentation/elements

- Set a fixed value to a property. The syntax is showed in Listing 5.32.

Listing 5.36: Syntax for tooltip,an edge/nodes property.

```
<dataset_mapping element="source" value="#caabff" property="color"/>
```

Where:

- – value is the fixed value of the property we want to set.
- **info**: contains some text/html that can help the user understanding the network. Since the syntax of the template is XML if

  you want to insert HTML you should envelop it into a CDATA tag. For example refer to Listing 5.32:

```
<![CDATA[ .......
  ]]>
```

- **drill**: is used to link the network to another document. The structure of the tag is showed in Template structure

```
<DRILL document="LINKED_DOCUMENT ">
  <PARAM name="PAR_NAME" type="TYPE" property =PROPERTY/>
</DRILL>
```

Where:

- DOCUMENT: is the label of the destination document;
- PAR_NAME: is the destination document parameter label; – TYPE: parameter type
  - ABSOLUTE/RELATIVE,
  - EDGE: the parameter will get an edge property value,
  - NODE: the parameter will get an node property value;
- PROPERTY: property of the object (node/edge) to bind to parameter.

5.15.2 An example*

Lets try to create a network that shows where the customers of Mexico usually go shopping.

Here, in the query on the foodmart demo data:

Listing 5.37: Foodmart demo data.

```
SELECT s.store_city store
      ,c.city customer
      ,c.city customer_city
      ,count(*) number_sales
      ,((length(s.store_city) \* 7) + 10) textlenght
      ,CONCAT (s.store_city,'-',c.city) rel_id
FROM sales_fact_1998 sf
JOIN customer c ON (c.customer_id = sf.customer_id)
JOIN store s ON (s.store_id = sf.store_id)
WHERE c.country = 'Mexico'
GROUP BY store
     ,customer
     ,rel_id
```

Now we can collect all these information and build our first network template. In our example the nodes are the cities and the relations represent where the customer of a city go to shop. Template for foodmart demo shows a simply template for this document:

Listing 5.38: Template for Foodmart demo.

```
<NET>
   <NETWOK_DEFINITION>
        <options pan_Zoom_Control_Position="topLeft">
        </options>
        <dataset_mapping_LIST>
          <dataset_mapping element="source" column="customer" property="id"/>
          <dataset_mapping element="target" column="store" property="id"/>
          <dataset_mapping element="edge" column="rel_id" property="id"/>
        </dataset_mapping_LIST>
   </NETWOK_DEFINITION>
</NET>
```

Now we try to make the graph “nicer”. We want to:

- see the name of the cities,
- see the number of sales of customers coming from city A to shop in city B,
- add some image as background of the nodes The template will look like Improved template for foodmart demo:

Listing 5.39: Improved template for foodmart demo.

```
<NET>
   <NETWOK_DEFINITION>
         <options edgeLabelsVisible="true" pan_Zoom_Control_Position="topLeft" nodeTooltipsEnabled="true" layout="Circle">
             <visual_style>
                <edges directed="true">
                  <label>
                    <passthrough_Mapper attrName="number_sales"/>
                  </label>
                </edges>
             </visual_style>
         </options>

         <dataset_mapping_LIST>
            <dataset_mapping element="source" column="customer"property="id"/>
            <dataset_mapping element="source" property="size" value ="50"/>
            <dataset_mapping element="source" column="customer_city "property="label"/>
            <dataset_mapping element="source" property="image"value="../img/city2.png"/>
            <dataset_mapping element="source" property="labelFontSize" value="12"/>
            <dataset_mapping element="source" property="labelFontWeight" value="bold"/>
            <dataset_mapping element="target" column="store"property="id"/>
            <dataset_mapping element="target" property="labelFontWeight" value="bold"/>
            <dataset_mapping element="target" property="labelFontSize" value="12"/>
            <dataset_mapping element="edge" column="rel_id"property="id"/>
            <dataset_mapping element="edge" column="number_sales"property="number_sales"/>
            <dataset_mapping element="edge" value="ARROW" property="sourceArrowShape"/>
         </dataset_mapping_LIST>
   </NETWOK_DEFINITION>
</NET>
```

Remark: The path ../img/city2.png is relative to the context of the web application, so it refers to the folder img inside the web application knowagenetworkengine Finally, the result is showed in next figure:

5.16 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.

5.16.1 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.

Here an example.

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

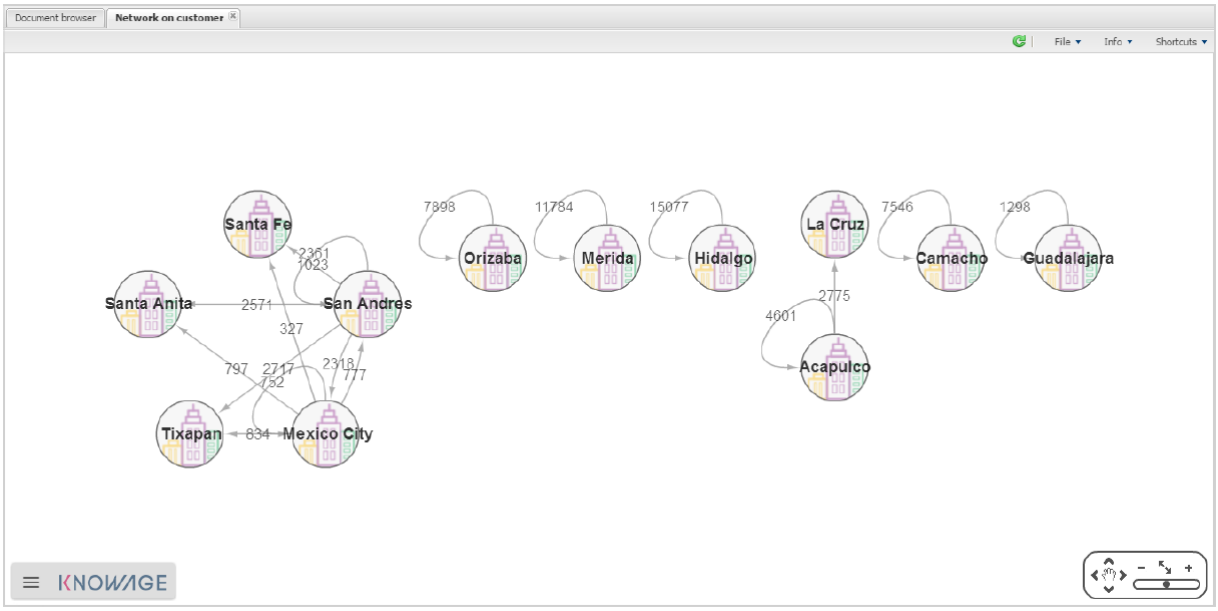


Fig. 5.366: Network for foodmart demo example.

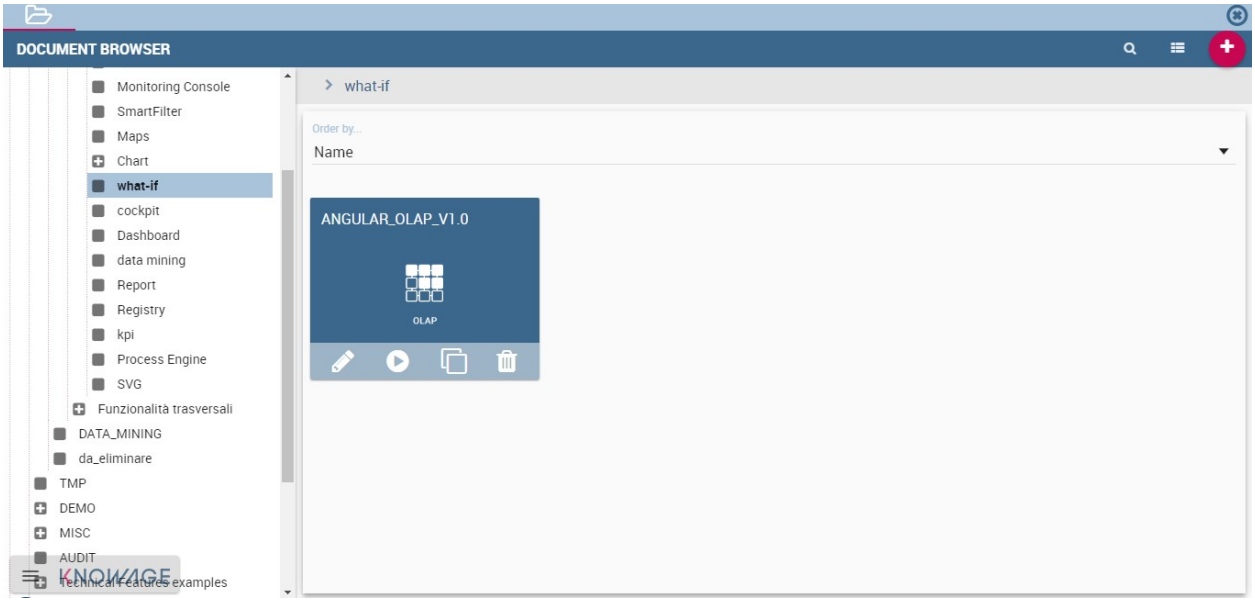


Fig. 5.367: Browse the documents and select an OLAP document.

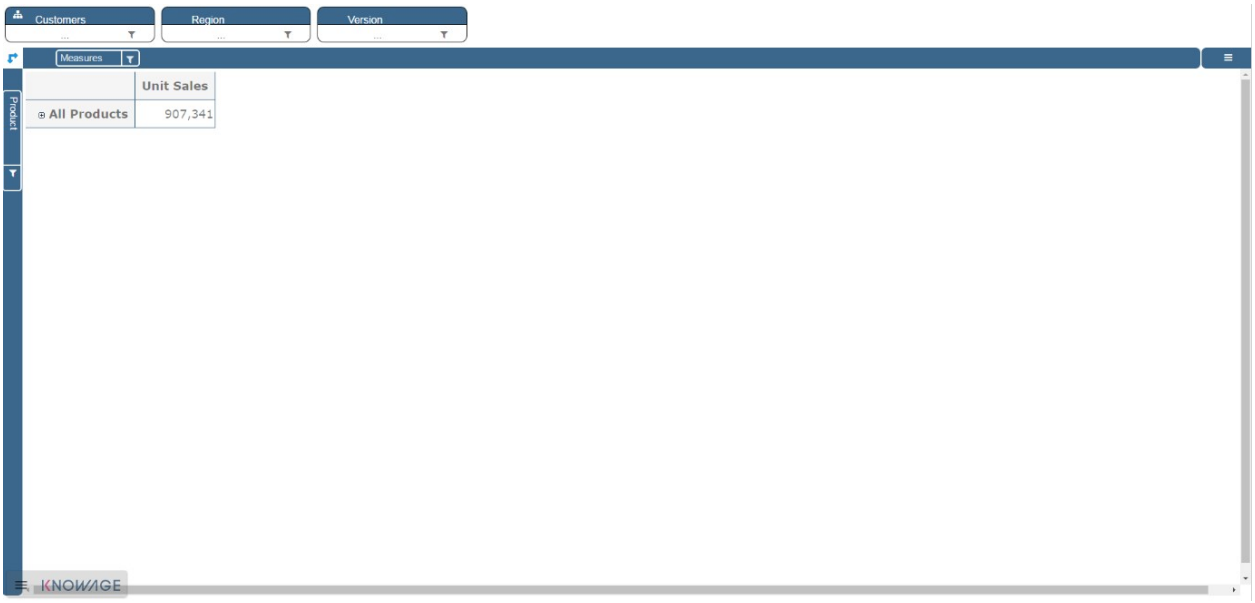


Fig. 5.368: Exploring an existing OLAP document.

5.16.1.1 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.

Fig. 5.369: The filter panel.

5.16.1.2 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.

Fig. 5.370: The filter card inside the filter panel.

Filter cards are used to:

- inform the user about available dimensions defined in OLAP schema,
- inform the user about dimension's name,
- perfom 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:

- an icon for opening dimension chooser dialog (a),
- a dimension name (b),
- icon slicer (c)+(e),
- choosed filter name (d),

Fig. 5.371: Features of a filter card.

5.16.1.3 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).

5.16.1.4 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,

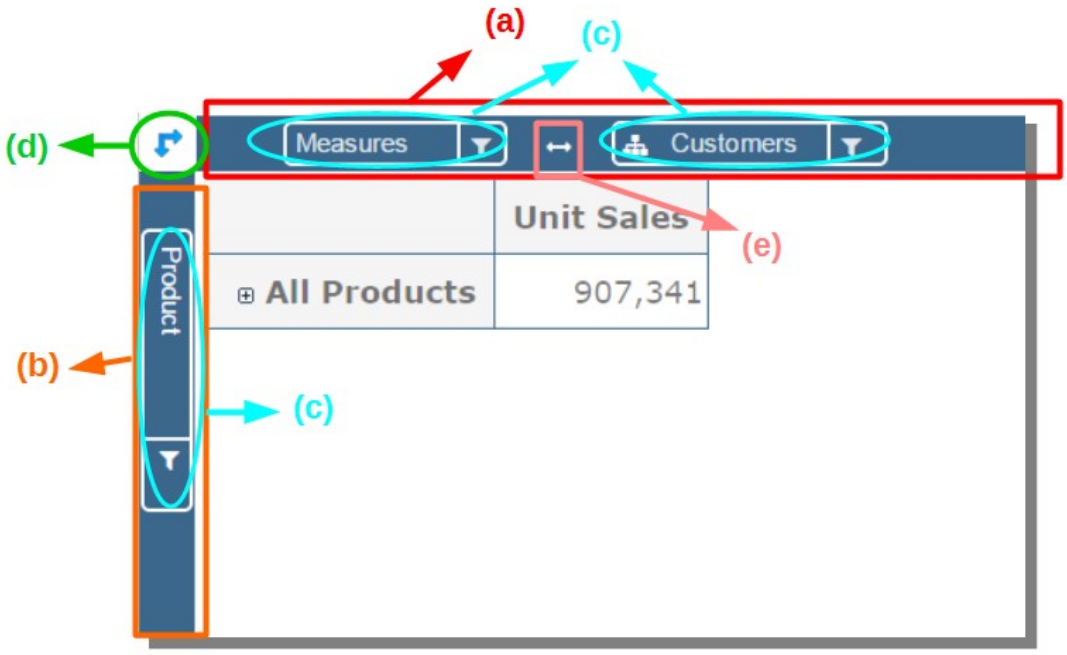


Fig. 5.372: Axes panel features.

	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. 5.373: Pivot table.

- drill through,
- show properties of a particular member,
- sort data,
- 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).

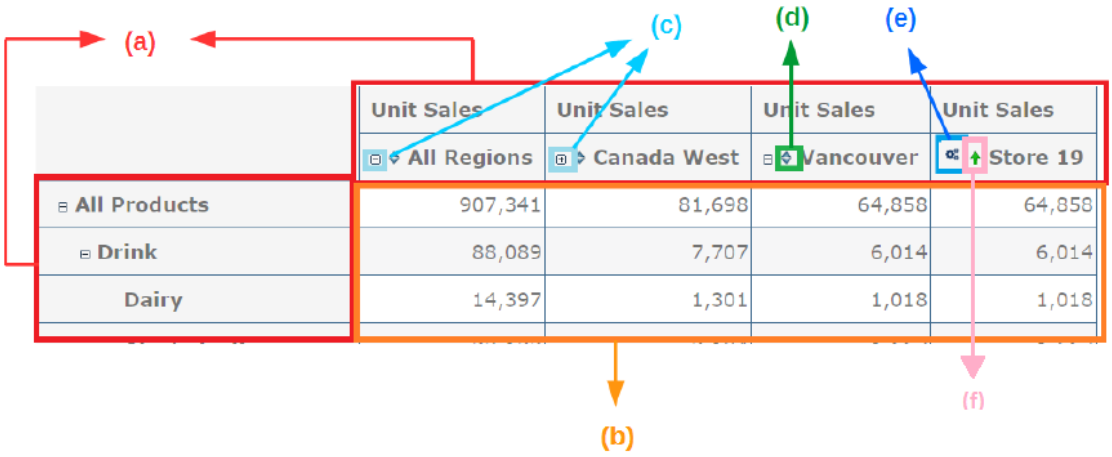


Fig. 5.374: Pivot table features.

5.16.1.5 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).

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.

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),

Fig. 5.375: Open the side bar.

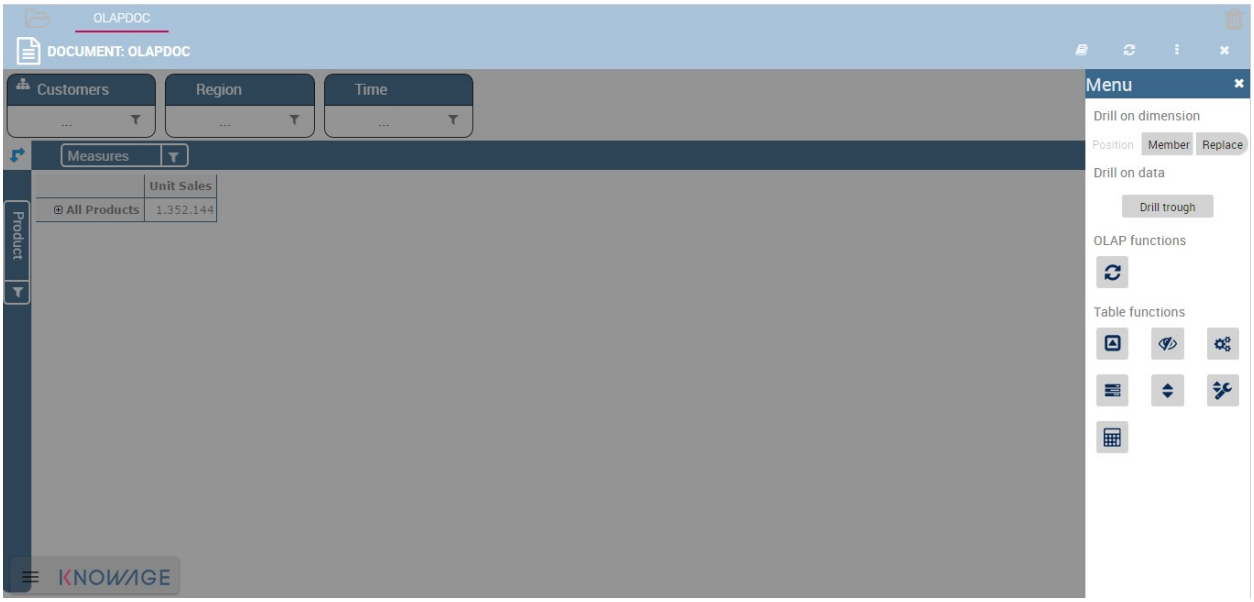


Fig. 5.376: Side bar.

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

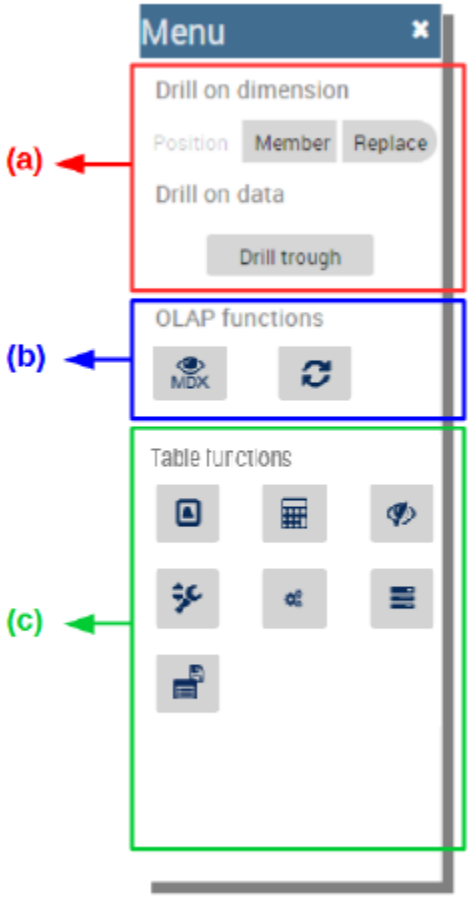


Fig. 5.377: Side bar Menu.

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),

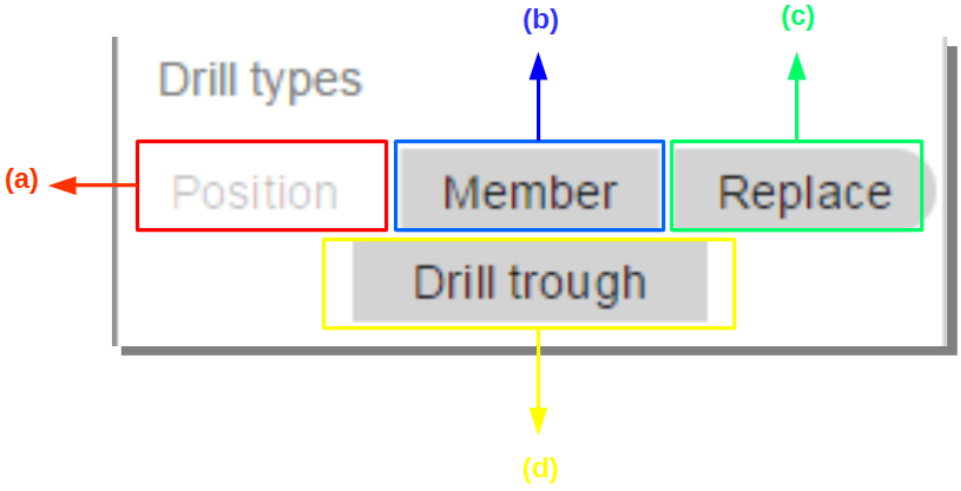


Fig. 5.378: Drill types.

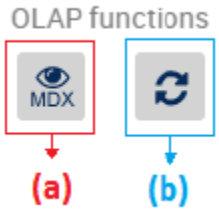


Fig. 5.379: OLAP functions.

- sorting settings (b),
- save customized view (c),
- show properties (d),
- suppress empty rows/columns (e),
- hide spans (f),
- calculated field wizard (g).

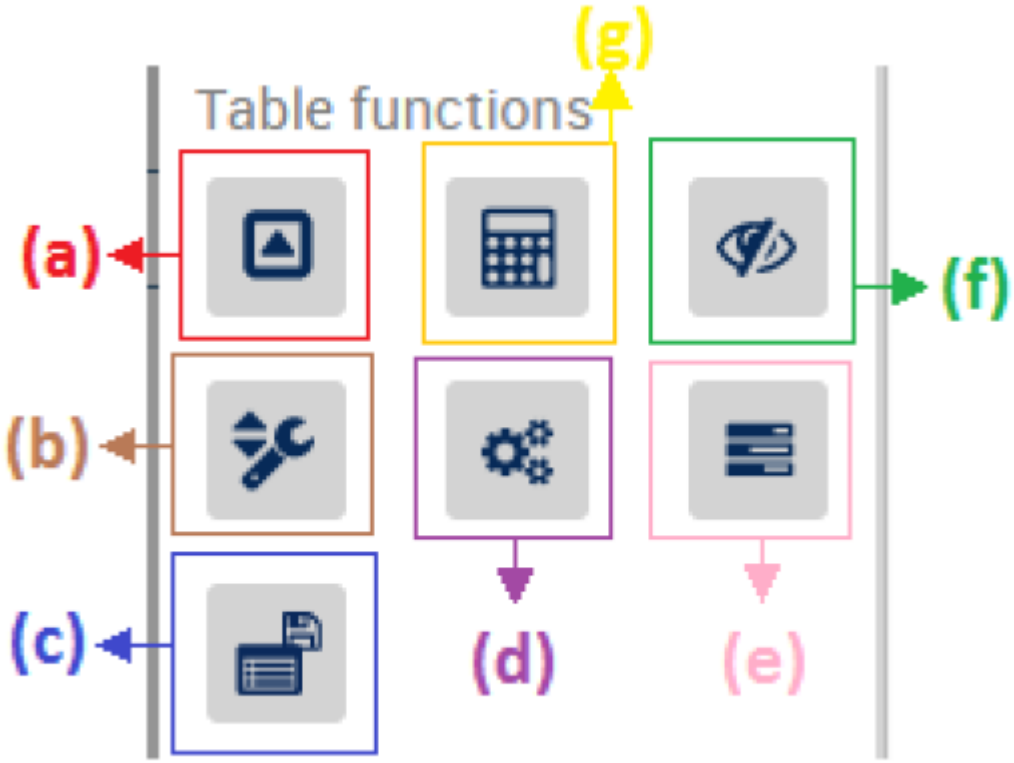


Fig. 5.380: Table functions.

Referring to next figure, what if consists of:

- lock/unlock model (a),
- delete versions (b),
- select an algotithm (c),
- output wizard (d),
- save as new version (e),
- undo (f),
- export excel for edit (g).

Fig. 5.381: Table functions.

5.16.2 Functionalities

5.16.2.1 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

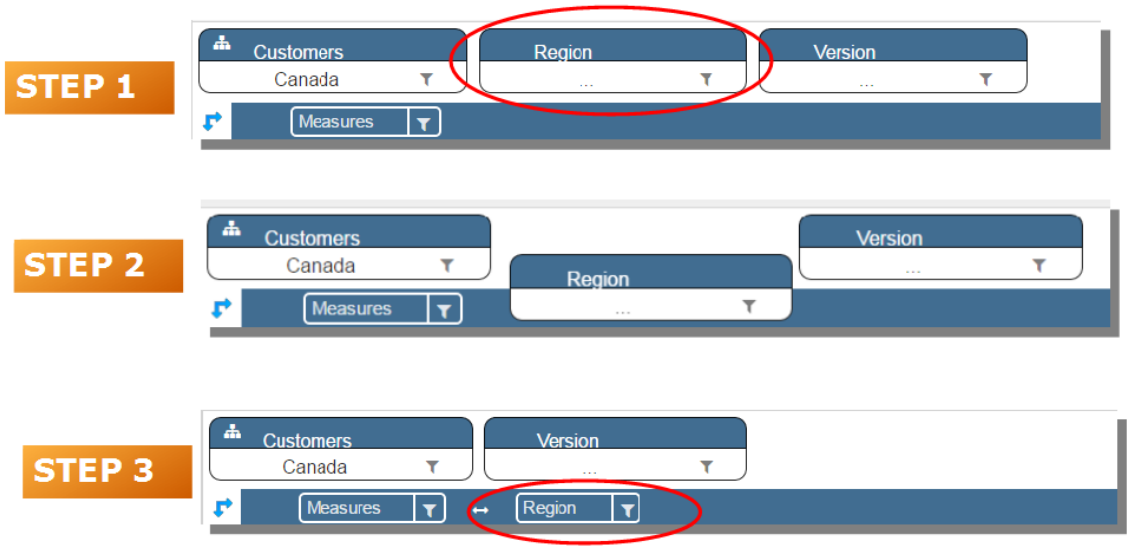


Fig. 5.382: Move a hierarchy to the columns axis.

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.

5.16.2.2 Swaping axes

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

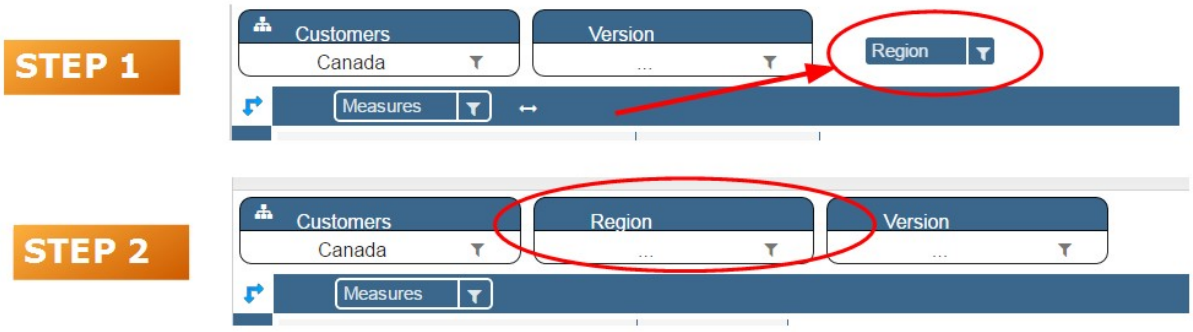


Fig. 5.383: Move a dimension from the columns axis to the filter panel.

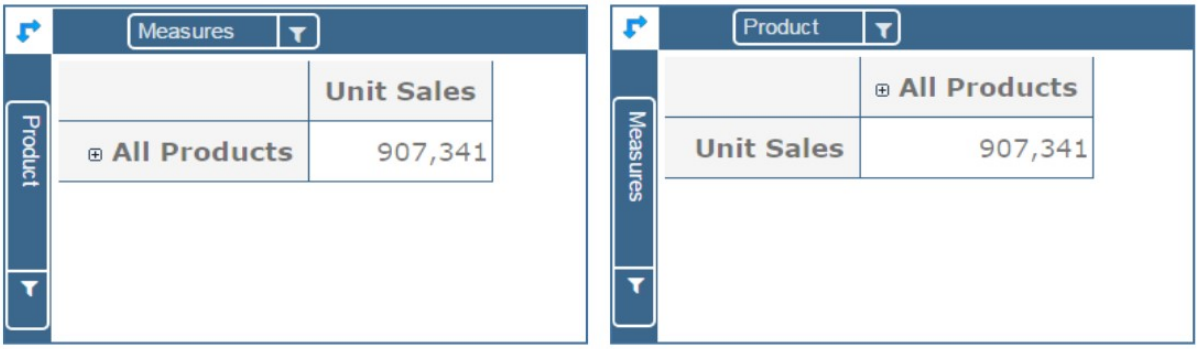


Fig. 5.384: Swap axes.

5.16.2.3 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.

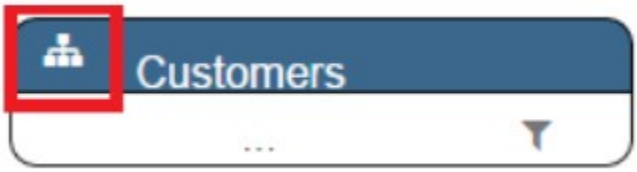


Fig. 5.385: Hierarchies icon.

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).

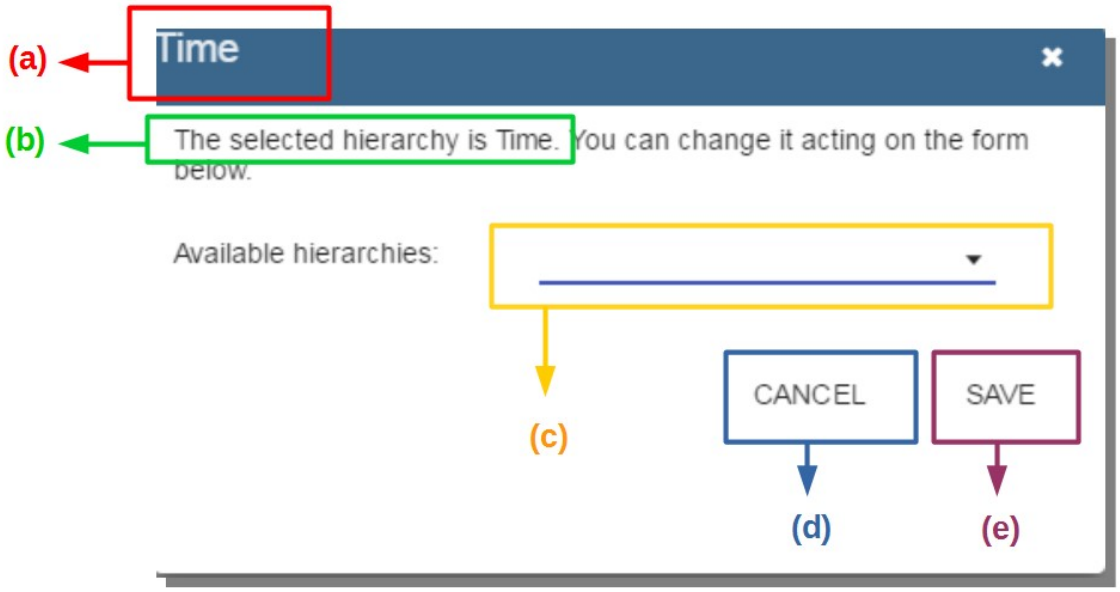


Fig. 5.386: Hierarchies dialog pop up.

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 hieararchies and has the chance to change it if needed to, as shown below.

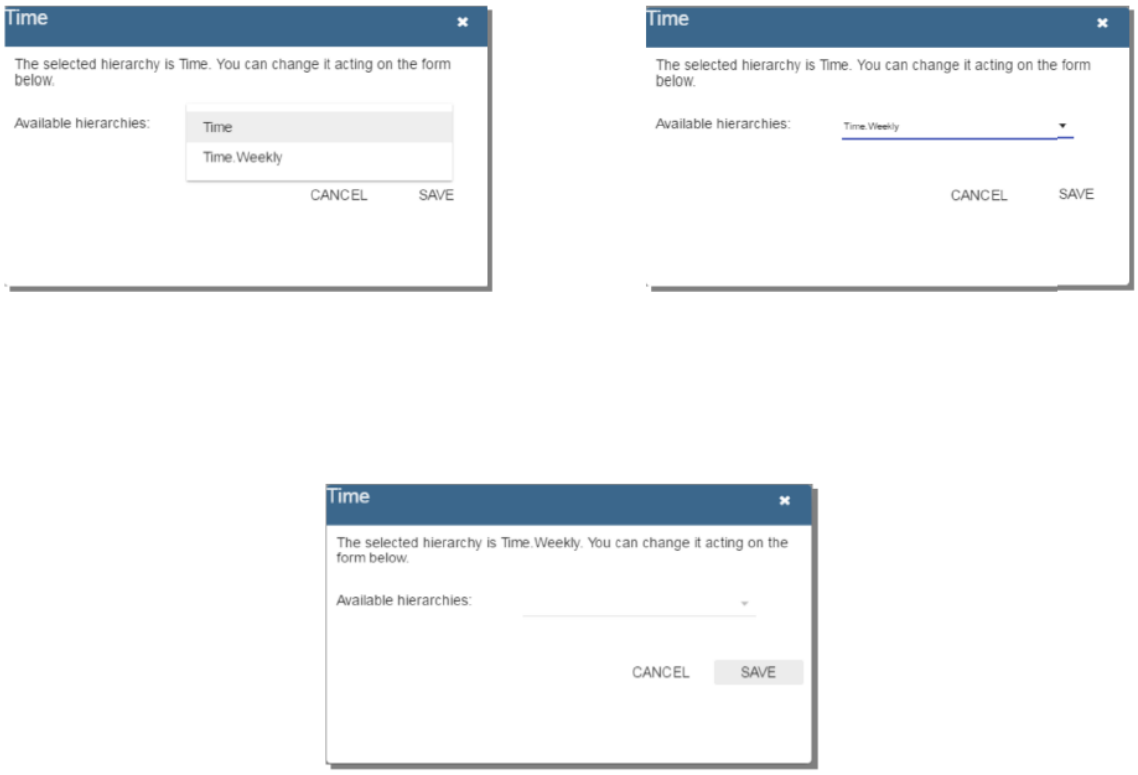


Fig. 5.387: Changing the hierarchies.

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. 5.388: Time hierachy: the table shows days in the month.

5.16.2.4 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 dimesion 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. 5.389: 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. 5.390 (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.

5.16.2.5 Filtering

To filter dimension members in a pivot table, the user should click on a button (see Fig. 5.371) 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.

5.16.2.6 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.

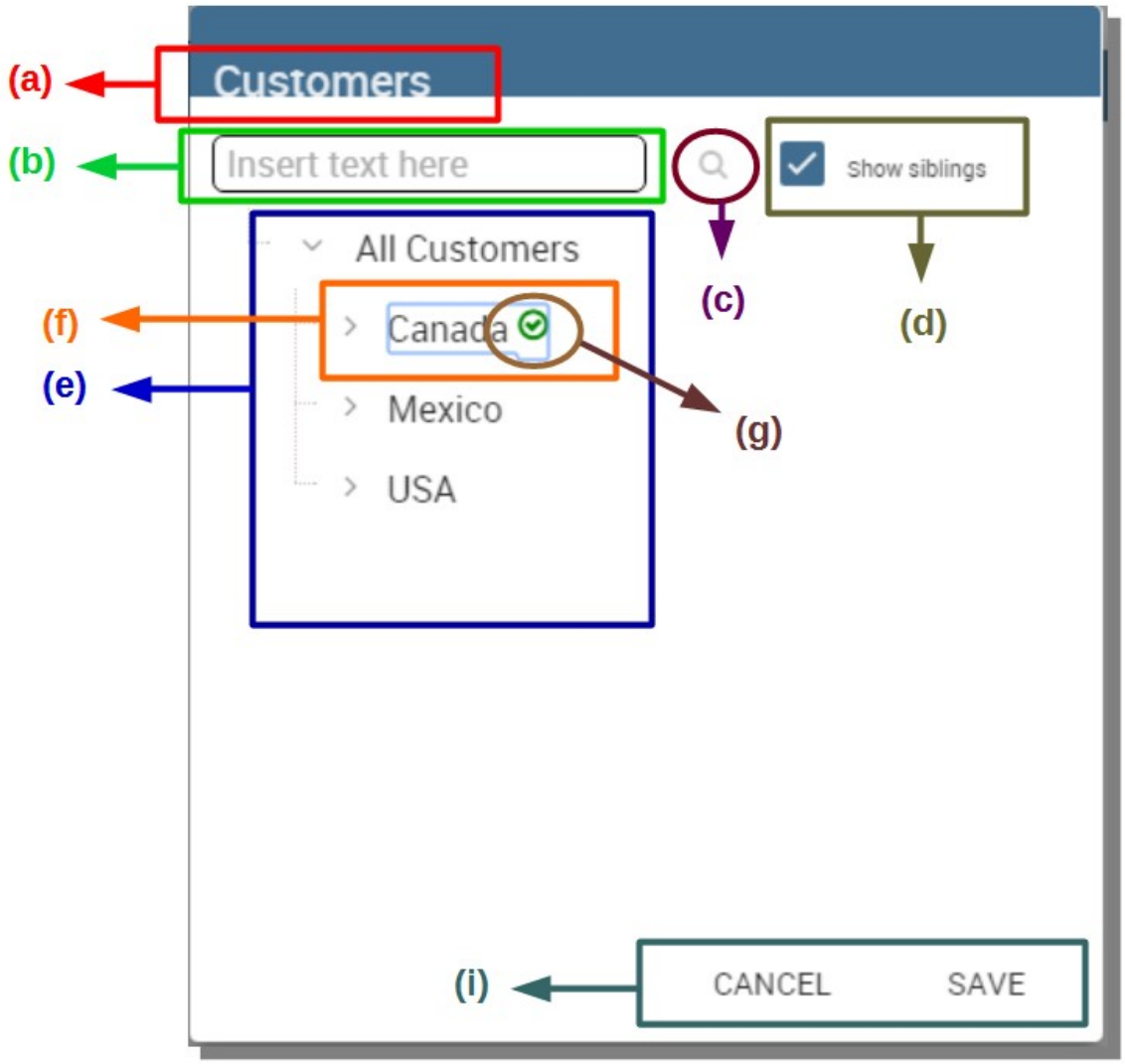


Fig. 5.390: Dialog for slicer choosing.

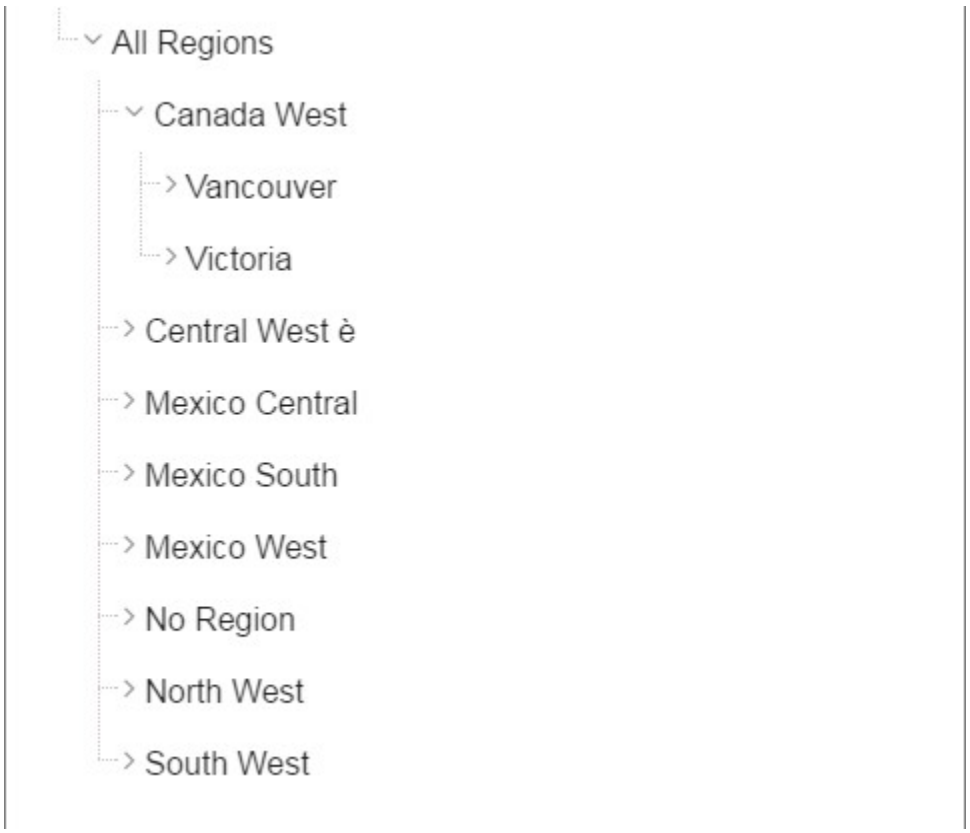


Fig. 5.391: Browsing the member tree.

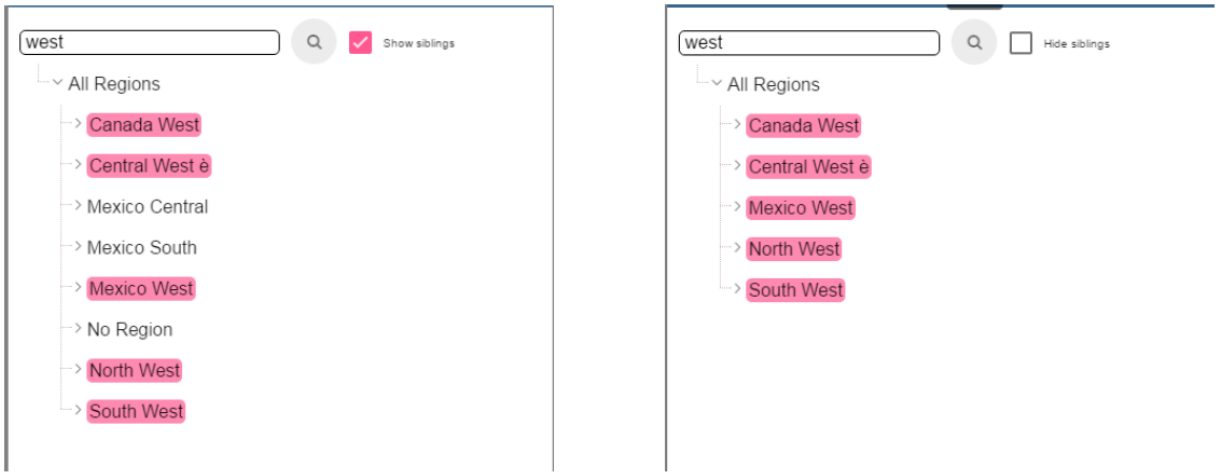


Fig. 5.392: Using the research box.

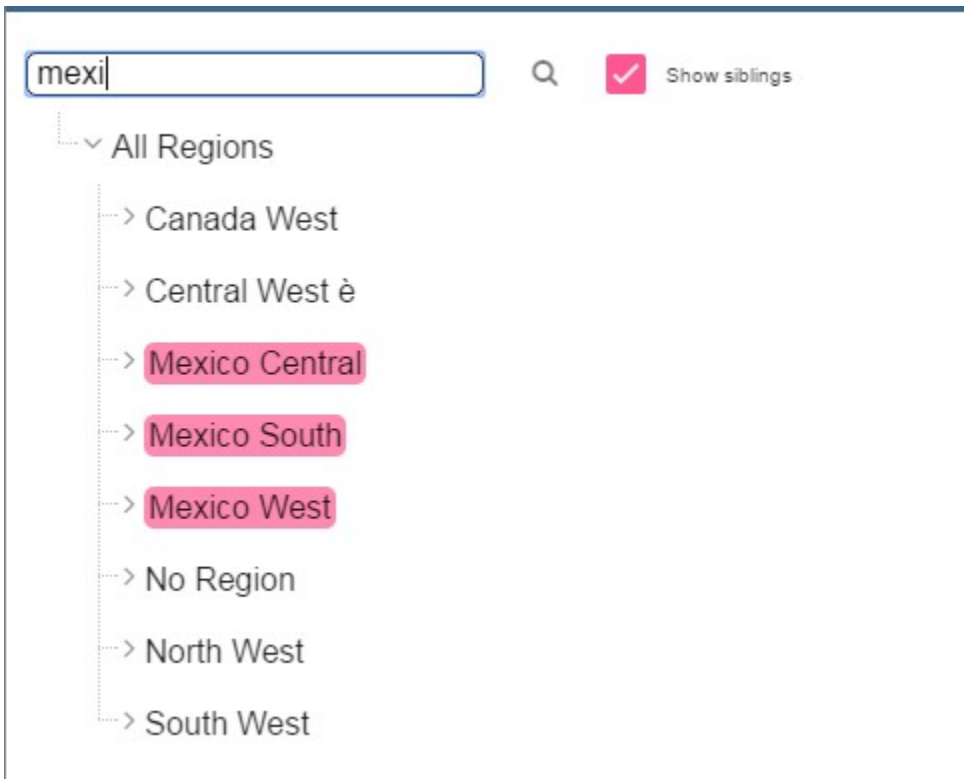


Fig. 5.393: Using the research box after a first investigation.

5.16.2.7 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,

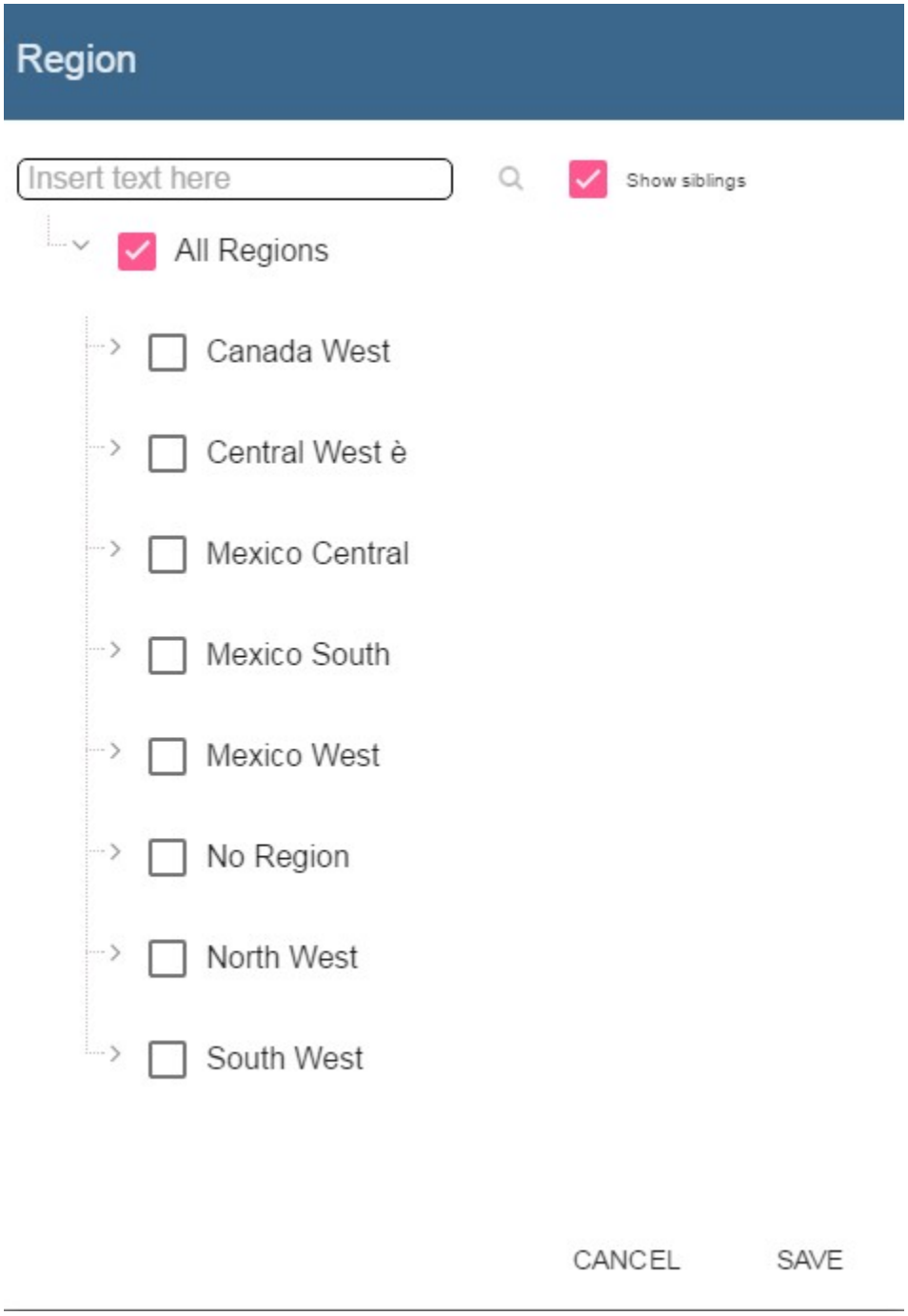


Fig. 5.394: Filter dialog.

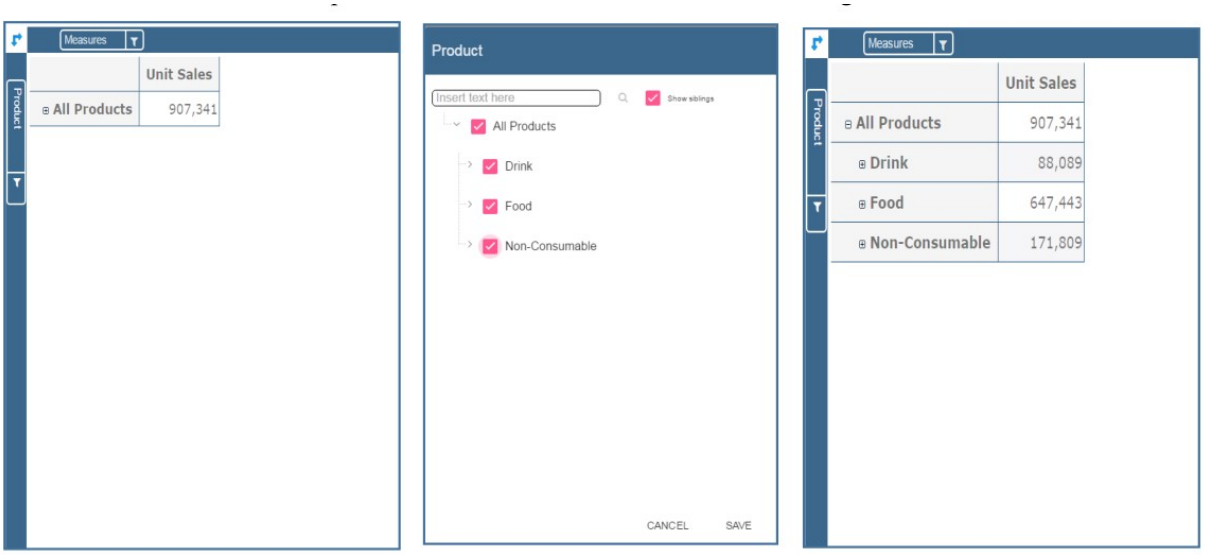


Fig. 5.395: Filter effects on pivot table.

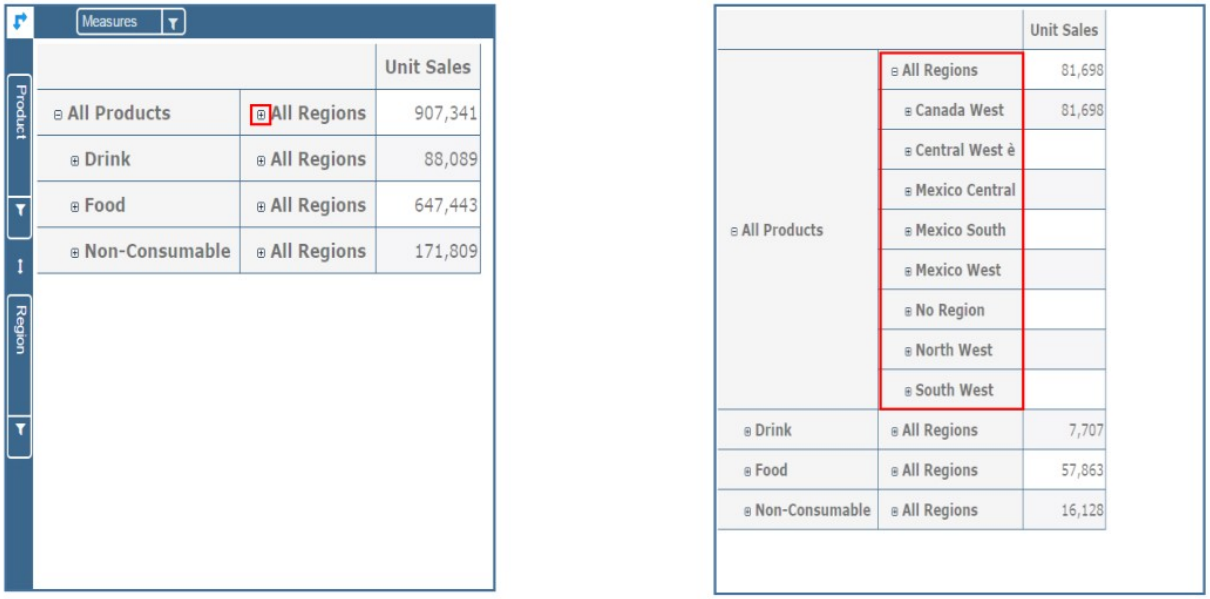


Fig. 5.396: “Position” drill down.

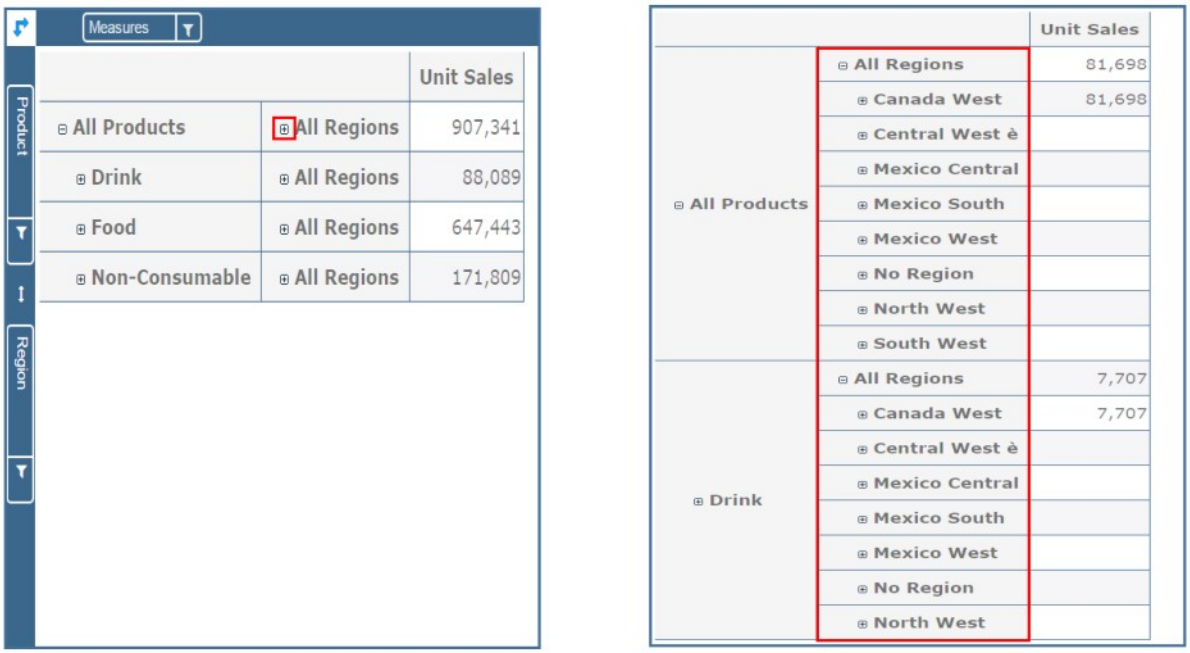


Fig. 5.397: "Member" drill down.

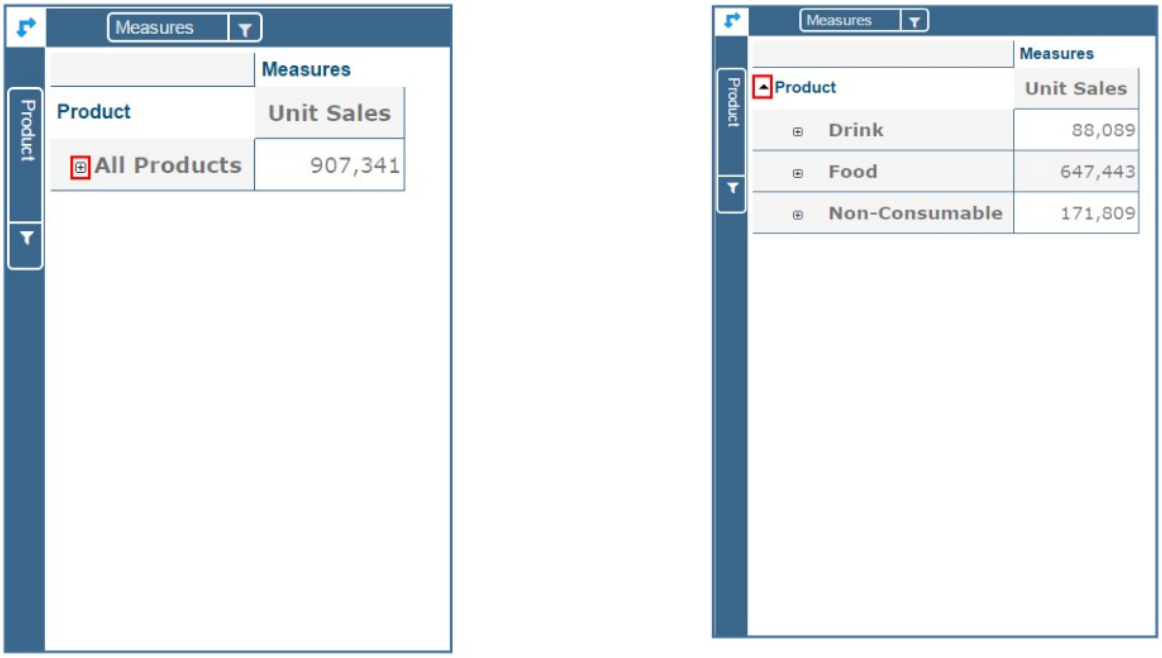


Fig. 5.398: "Replace" drill down.

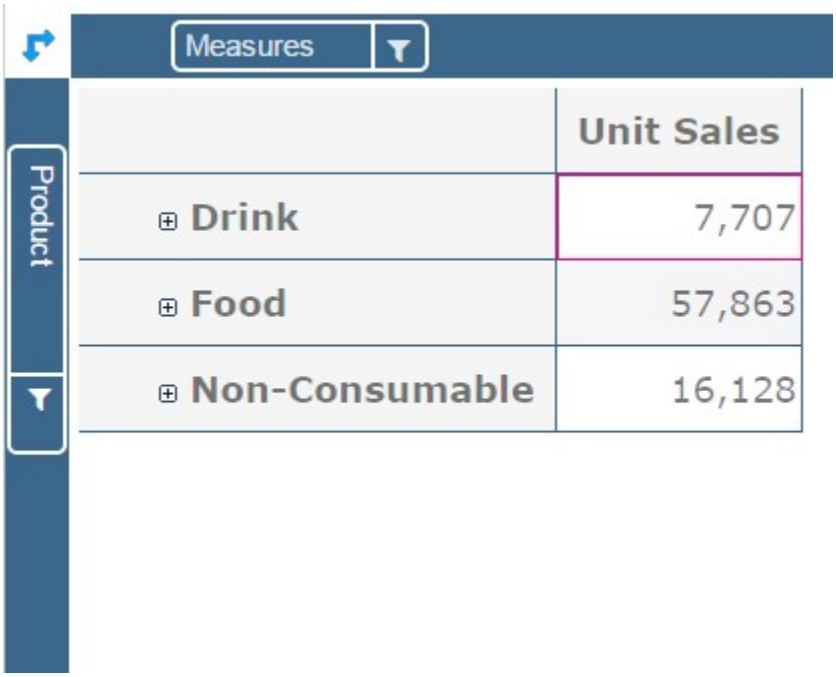


Fig. 5.399: Drill thorugh option.

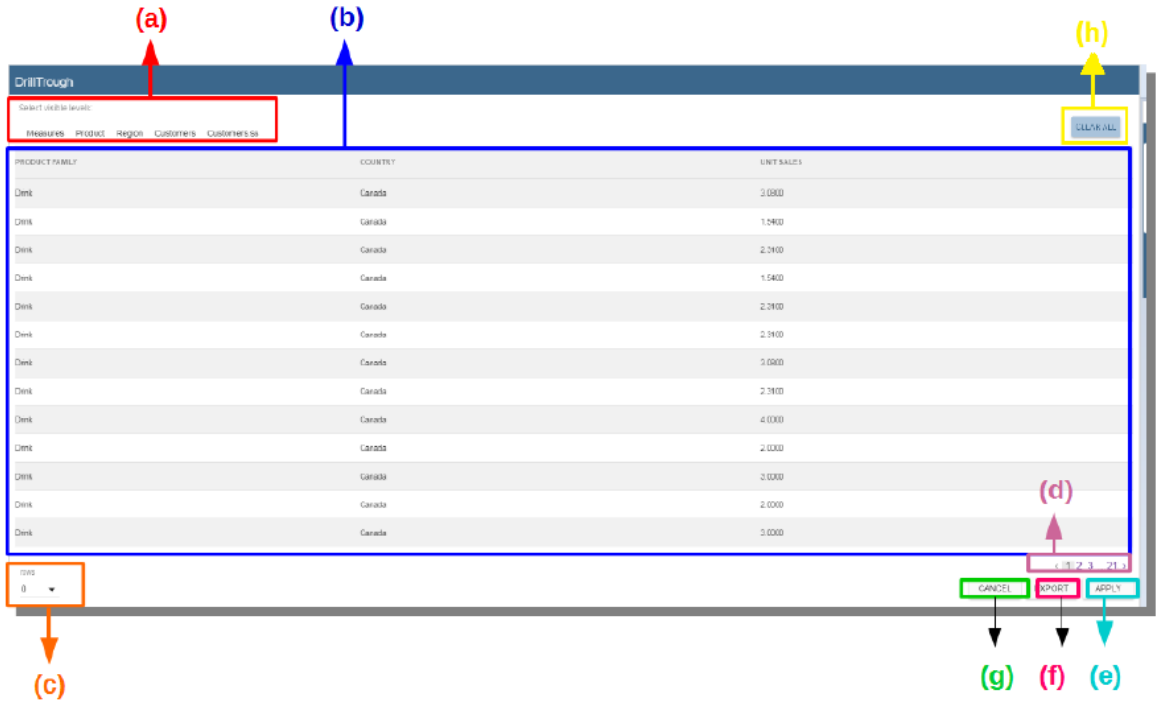


Fig. 5.400: Drill thorugh window.

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.

5.16.2.8 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.

5.16.2.9 Showing MDX

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

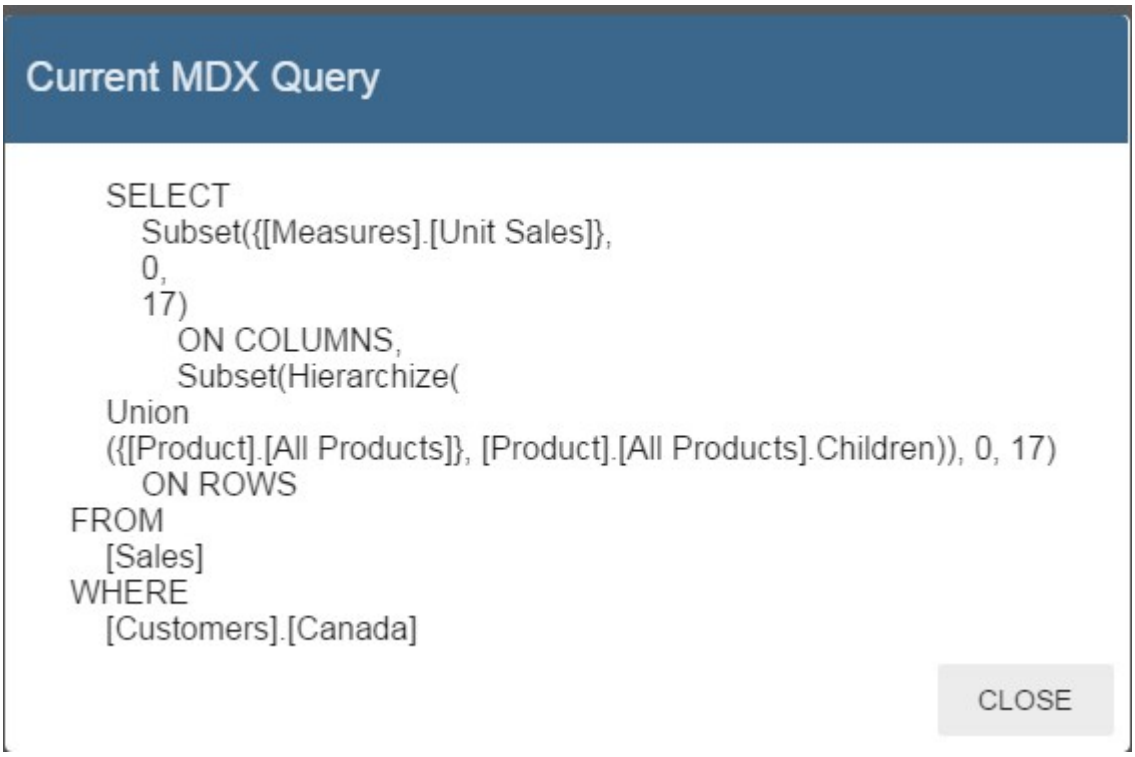


Fig. 5.401: Showing MDX query example.

5.16.2.10 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 responsable for entering *valid* MDX query.

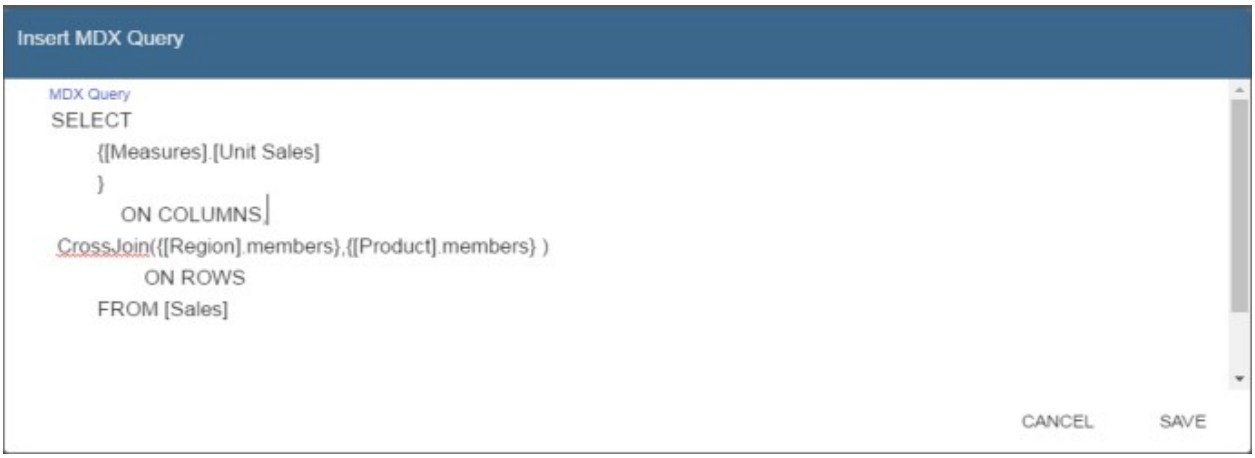


Fig. 5.402: Sending MDX query example.

5.16.2.11 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.

5.16.2.12 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.

5.16.2.13 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 colunms/rows which has property set, a table icon appears. This icon lets the property pop up opens. Figure below shows an example.

5.16.2.14 Suppressing empty colunms/rows

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

5.16.2.15 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" hiearchy 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. 5.403: Sending MDX query example.

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

Fig. 5.404: 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. 5.405: 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. 5.406: 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. 5.407: Show properties.

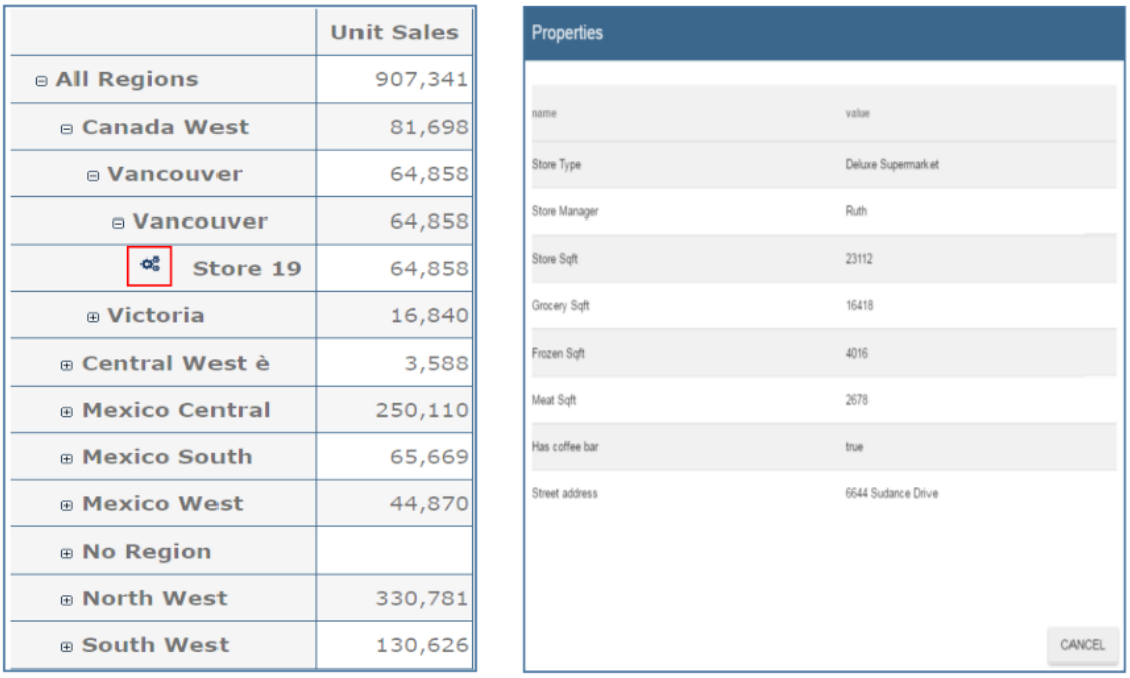


Fig. 5.408: Show properties summarized in a pop up.

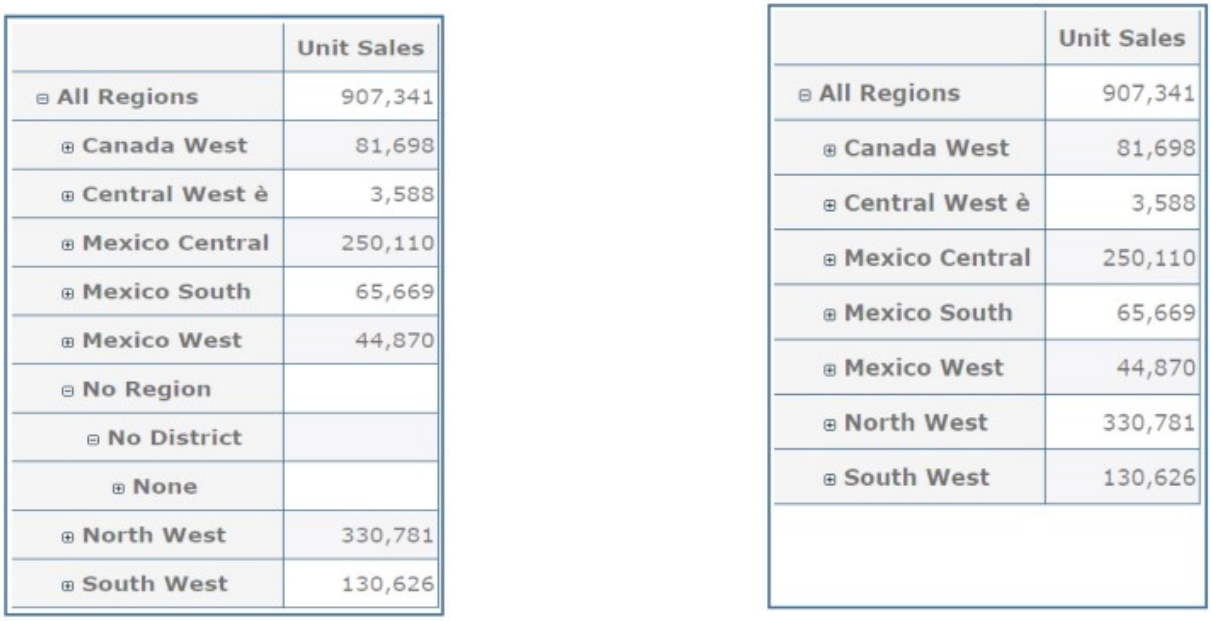
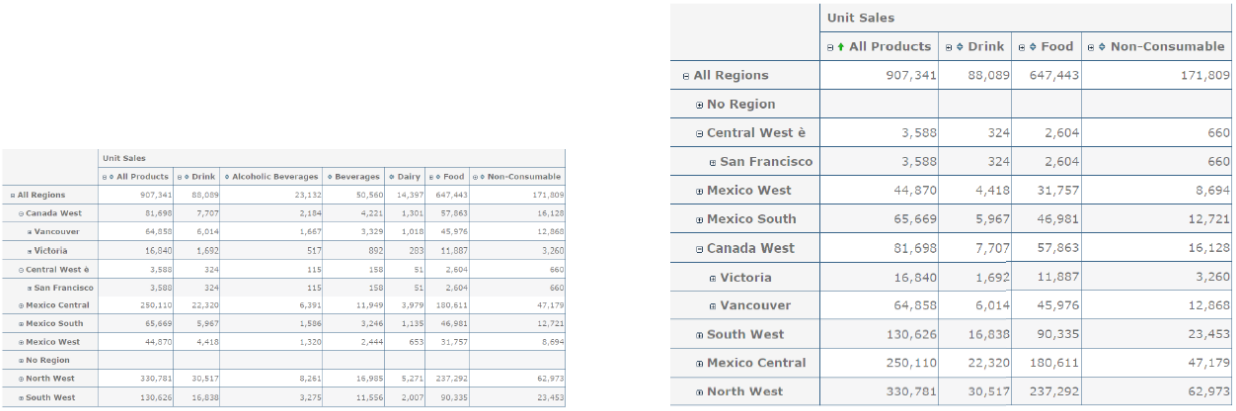


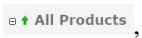
Fig. 5.409: Suppressing empty colunms/rows.



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

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

Fig. 5.410: Member sorting.

To sort members the user needs to click on the sorting command  , available next to each member of the pivot table. Note that the sorting criteria is ascending at first 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.

5.16.2.16 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),

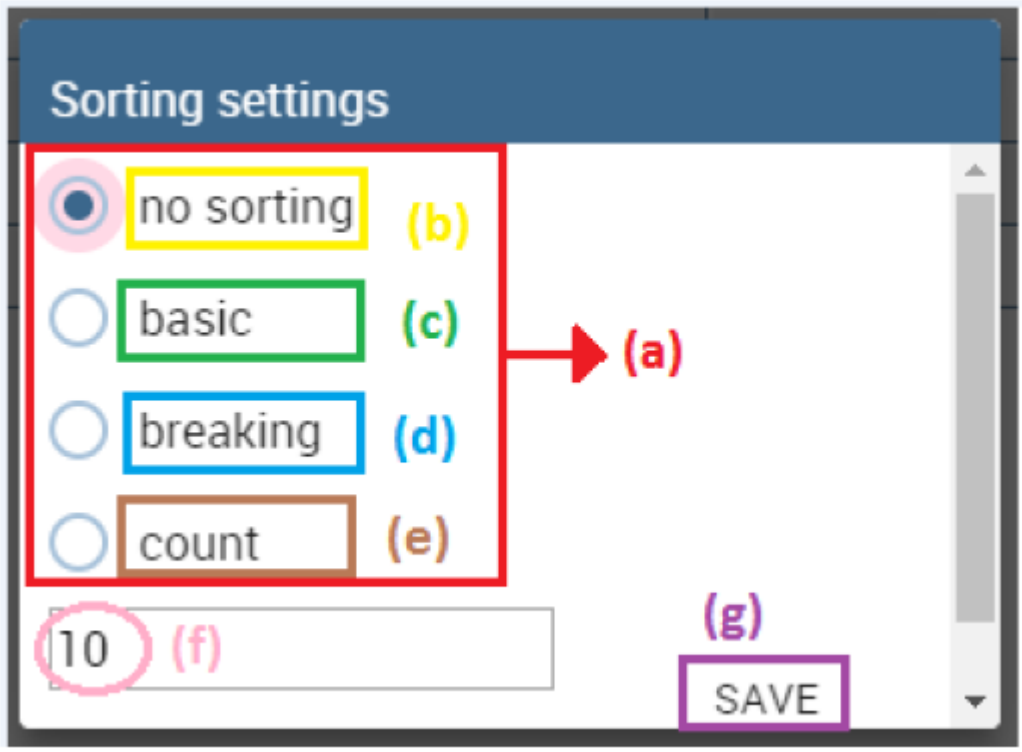


Fig. 5.411: Sorting settings window.

	Unit Sales
⊟ All Products	907,341
⊞ Drink	88,089
⊞ Food	647,443
⊞ Non-Consumable	171,809

Fig. 5.412: 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).

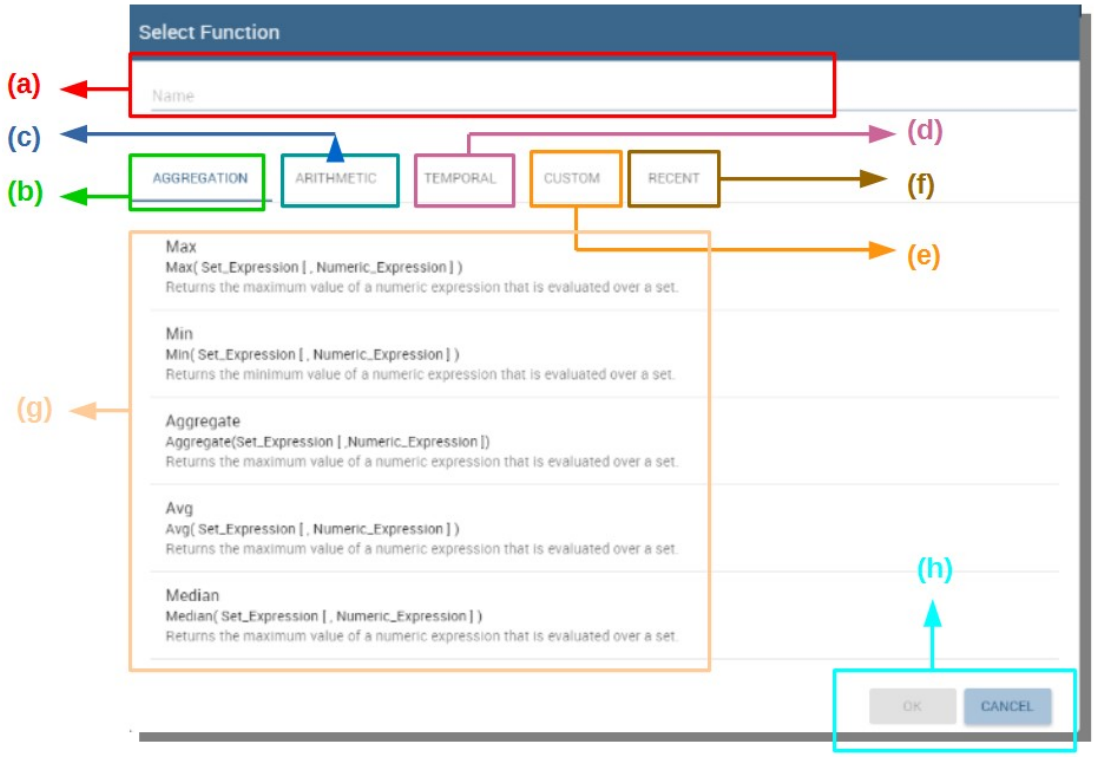


Fig. 5.413: Select function dialog.

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 defintion 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. 5.414: 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.

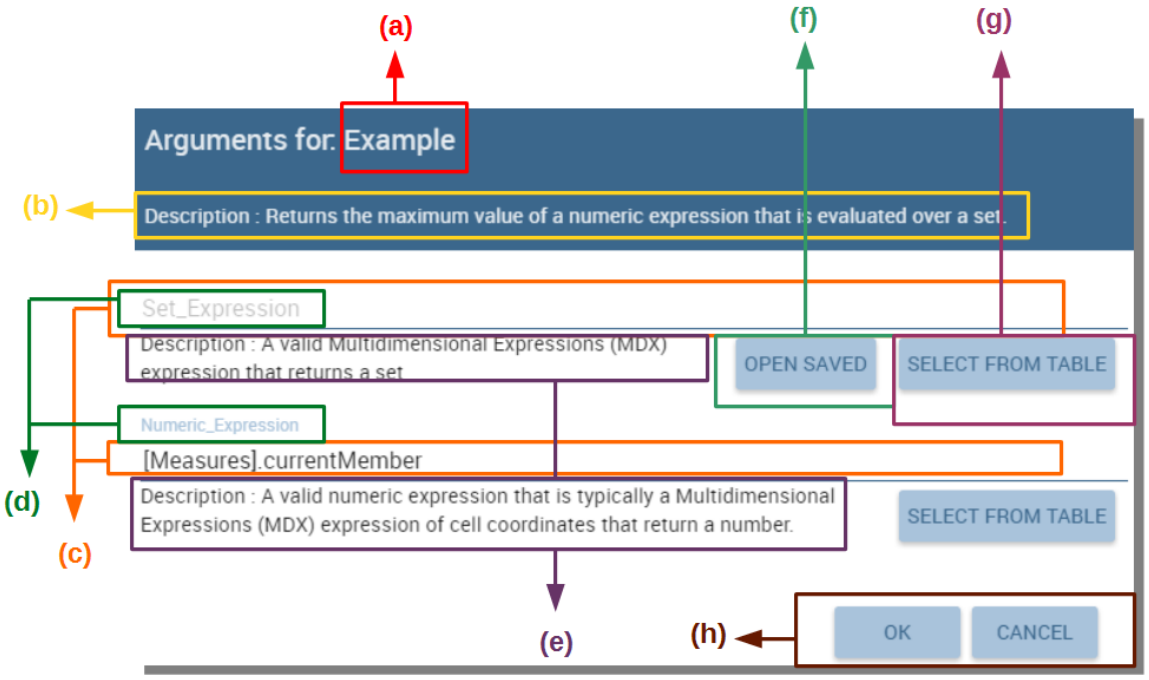


Fig. 5.415: Argument defintion dialog.

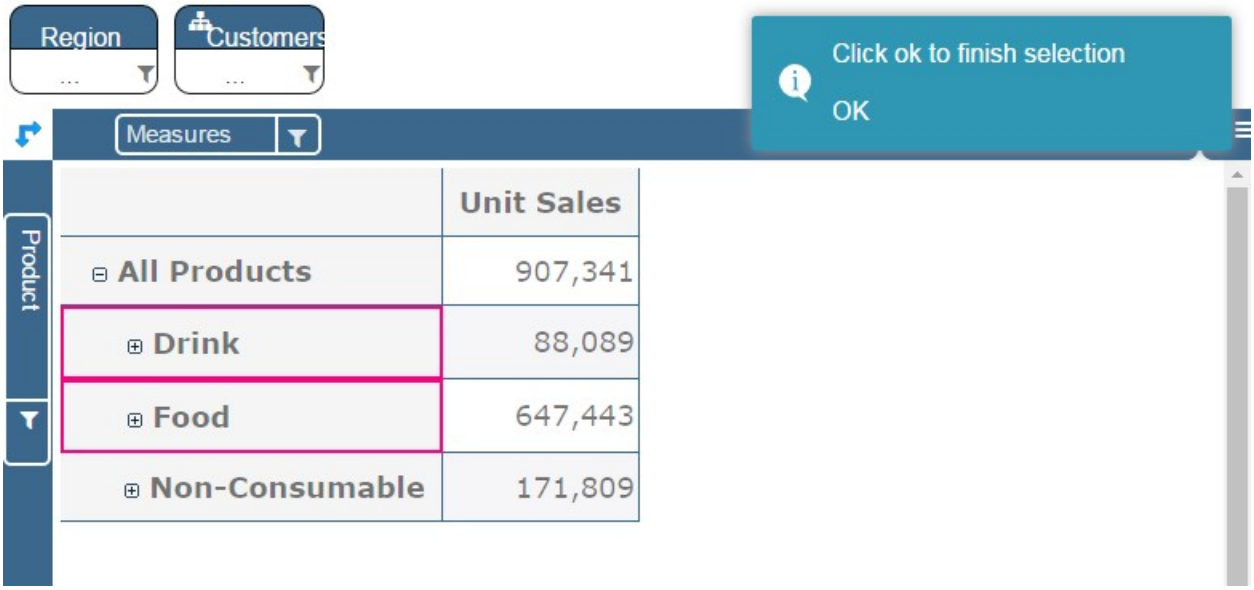


Fig. 5.416: Selecting members.

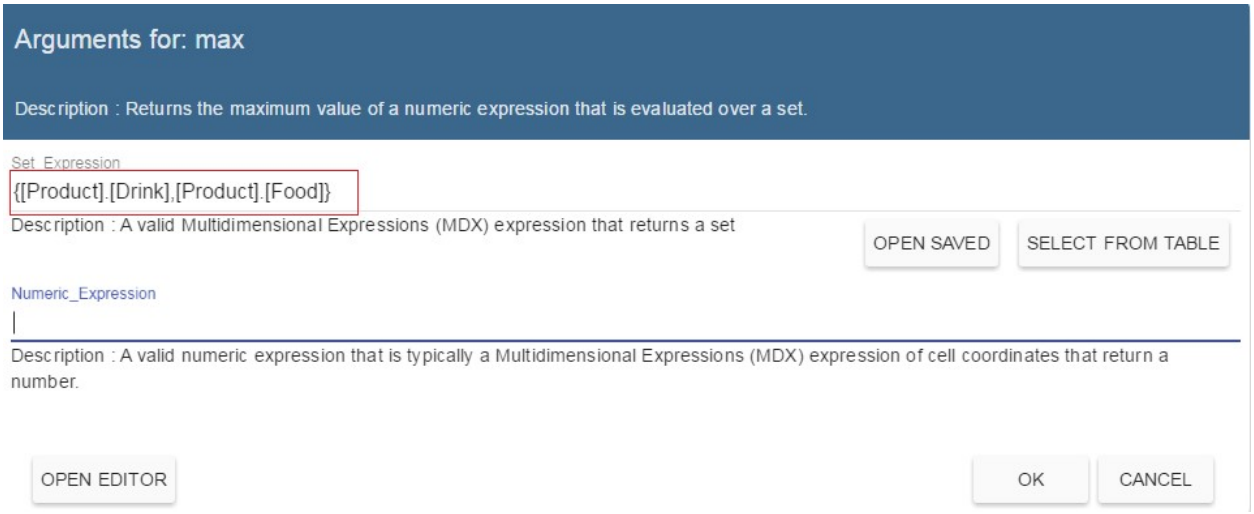


Fig. 5.417: Expression of the selected members.

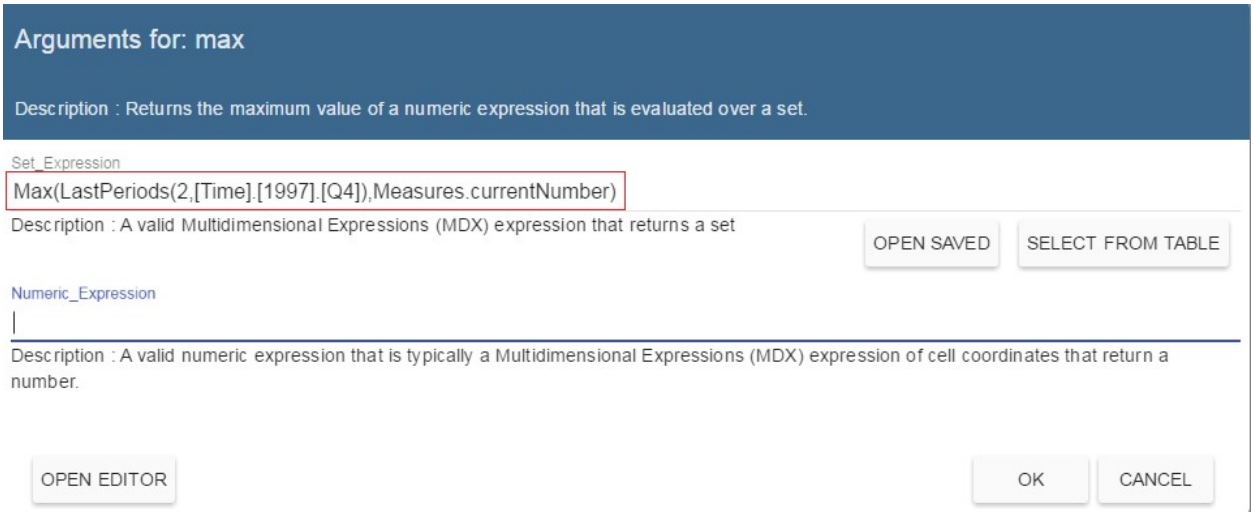


Fig. 5.418: Saved sets dialog.

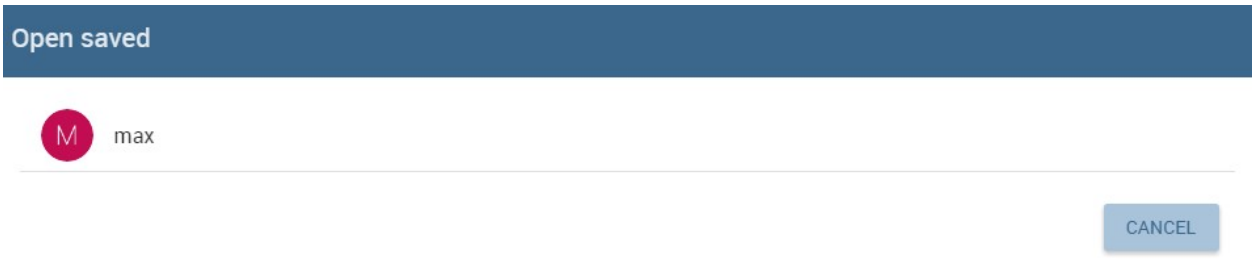


Fig. 5.419: Expression of the saved/calculated member/set.

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.

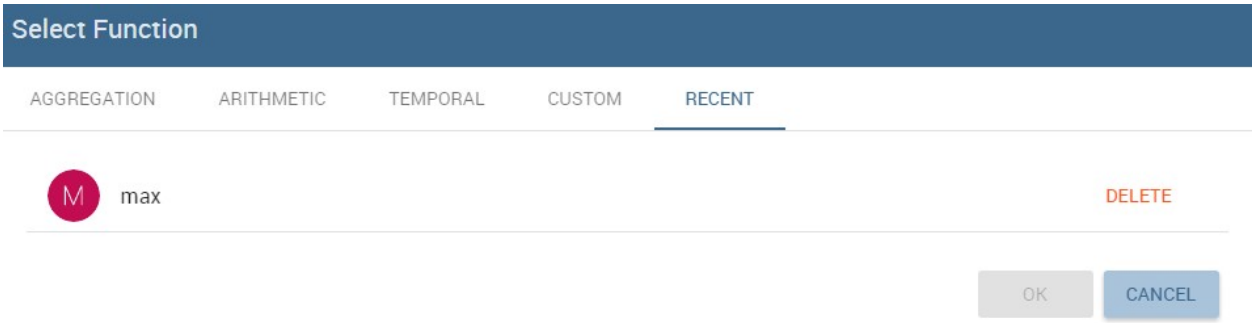


Fig. 5.420: Edit a calculated member.

Deleting is done by Delete button as shown in figure above.

5.16.3 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.

5.16.3.1 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.

5.16.3.2 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.

5.16.3.2.1 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 5.40: Mondrian schema example

```
<?xml version="1.0"?>
 <Schema name="FoodMart">
       <!-- Shared dimensions -->
       <Dimension name="Customers">
          <Hierarchy hasAll="true" allMemberName="All Customers"
                     primaryKey=" customer_id">
              <Table name="customer"/>
              <Level name="Country" column="country" uniqueMembers="true"/>
              <Level name="State Province" column="state_province"
                     uniqueMembers="true"/>

              <Level name="City" column="city" uniqueMembers="false"/>

          </Hierarchy> ...

       </Dimension> ...

       <!-- Cubes -->
       <Cube name="Sales">

          <Table name="sales_fact_1998"/>

          <DimensionUsage name="Customers" source="Customers"
                          foreignKey="customer_id" /> ...

          <!-- Private dimensions -->

          <Dimension name="Promotion Media" foreignKey="promotion_id">

              <Hierarchy hasAll="true" allMemberName="All Media"
                         primaryKey="promotion_id">
                  <Table name="promotion"/>
                  <Level name="Media Type" column="media_type" uniqueMembers="true"/>
```

(continues on next page)

(continued from previous page)

```
                </Hierarchy>

            </Dimension> ...

            <!-- basic measures-->

            <Measure name="Unit Sales" column="unit_sales" aggregator="sum"
                     formatString="#,###.00"/>

            <Measure name="Store Cost" column="store_cost" aggregator="sum"
                     formatString= "#,###.00"/>

            <Measure name="Store Sales" column="store_sales" aggregator="sum"
                     formatString="#,###.00"/>
            ...

            <!-- derived measures-->

            <CalculatedMember name="Profit" dimension="Measures">
                <Formula>
                    [Measures].[Store Sales] - [Measures].[Store Cost]
                </Formula>

                <CalculatedMemberProperty name="format_string" value="$#,##0.00"/>
            </CalculatedMember>

        </Cube>
    ...
</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/.

5.16.3.2.1.1 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 you XML schema file and choosing **Name** and **Description**. When creating a new OLAP template, you will choose among the available cubes defined in the registered schemas.

Note that the Lock option forbids other technical users to modify settings.

5.16.3.2.2 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 5.41: Mapping template example

```
<?xml version="1.0" encoding="UTF-8"?>
<olap>
   <!-- schema configuration -->
   <cube reference="FoodMart"/>

   <!-- query configuration -->
   <MDXquery>
       SELECT {[Measures].[Unit Sales]} ON COLUMNS
       , {[Region].[All Regions]} ON ROWS
       FROM [Sales]
       WHERE [Product].[All Products].[${family}]
       <parameter name="family" as="family"/>
   </MDXquery>

   <MDXMondrianQuery>
       SELECT {[Measures].[Unit Sales]} ON COLUMNS
       , {[Region].[All Regions]} ON ROWS
       FROM [Sales]
       WHERE [Product].[All Products].[Drink]
   </MDXMondrianQuery>

   <!-- toolbar configuration -->
   <TOOLBAR>
       <BUTTON_MDX visible="true" menu="false" />
       <BUTTON_FATHER_MEMBERS visible="true" menu="false"/>
       <BUTTON_HIDE_SPANS visible="true" menu="false"/>
       <BUTTON_SHOW_PROPERTIES visible="true" menu="false"/>
       <BUTTON_HIDE_EMPTY visible="true" menu="false" />
       <BUTTON_FLUSH_CACHE visible="true" menu="false" />
       <BUTTON_SAVE visible="true" menu="false" />
       <BUTTON_SAVE_NEW visible="true" menu="false" />
       <BUTTON_EXPORT_OUTPUT visible="true" menu="false" />
   </TOOLBAR>

</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 5.42: Menu configurable options

```
   <BUTTON_DRILL_THROUGH visible="true"/>
   <BUTTON_MDX visible="true"/>
   <BUTTON_EDIT_MDX visible="true"/>
   <BUTTON_FATHER_MEMBERS visible="true"/>
   <BUTTON_CC visible="true"/>
   <BUTTON_HIDE_SPANS visible="true"/>
   <BUTTON_SORTING_SETTINGS visible="true"/>
   <BUTTON_SORTING visible="true" />
```

(continues on next page)

(continued from previous page)

```
    <BUTTON_SHOW_PROPERTIES visible="true"/>
    <BUTTON_HIDE_EMPTY visible="true"/>
    <BUTTON_FLUSH_CACHE visible="true"/>
    <BUTTON_SAVE visible="true"/>
    <BUTTON_SAVE_NEW visible="true"/>
    <BUTTON_UNDO visible="true"/>
    <BUTTON_VERSION_MANAGER visible="true"/>
    <BUTTON_EXPORT_OUTPUT visible="false"/>
```

5.16.3.2.3 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.

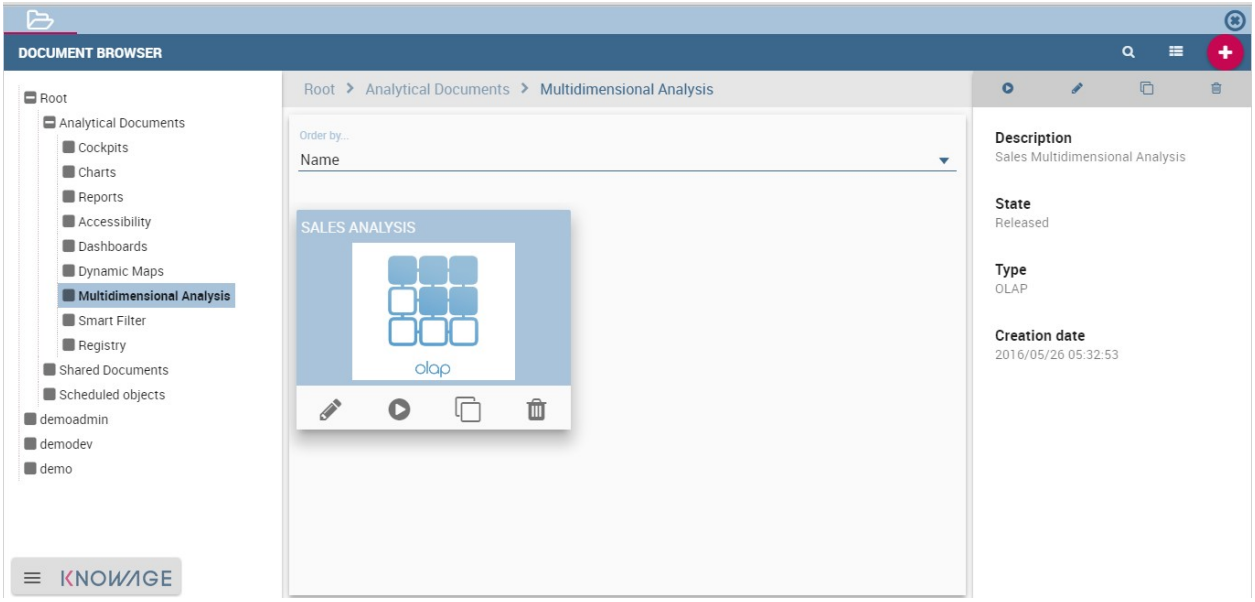


Fig. 5.421: OLAP document on server.

5.16.3.3 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.

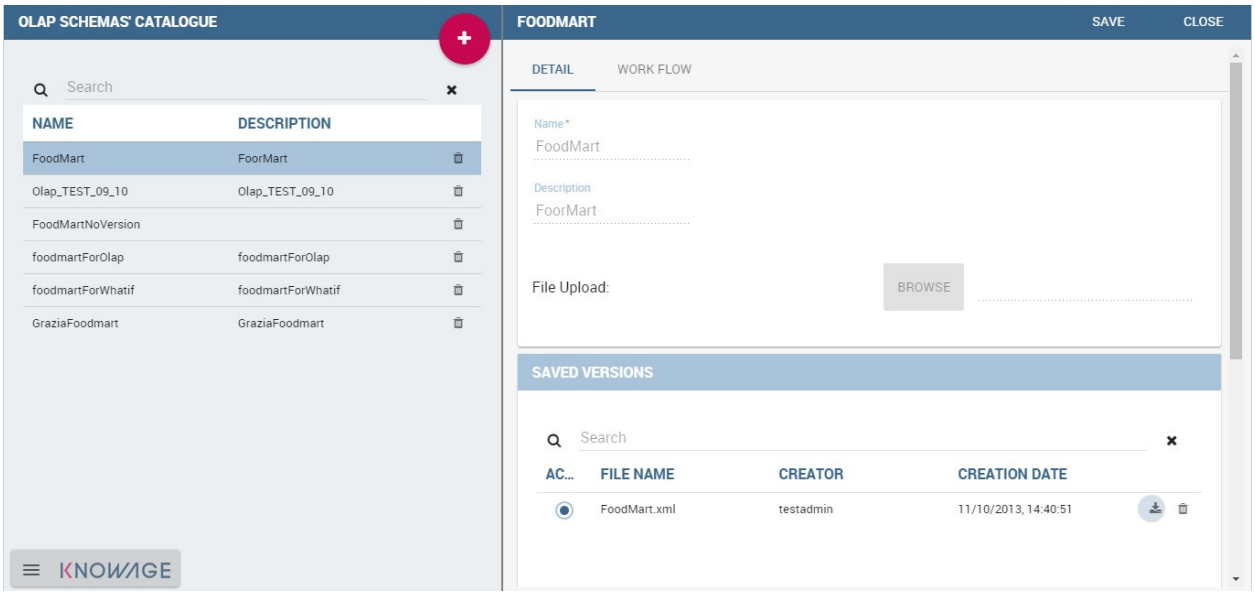


Fig. 5.422: Schema Mondrian from catalog.

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 opend 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

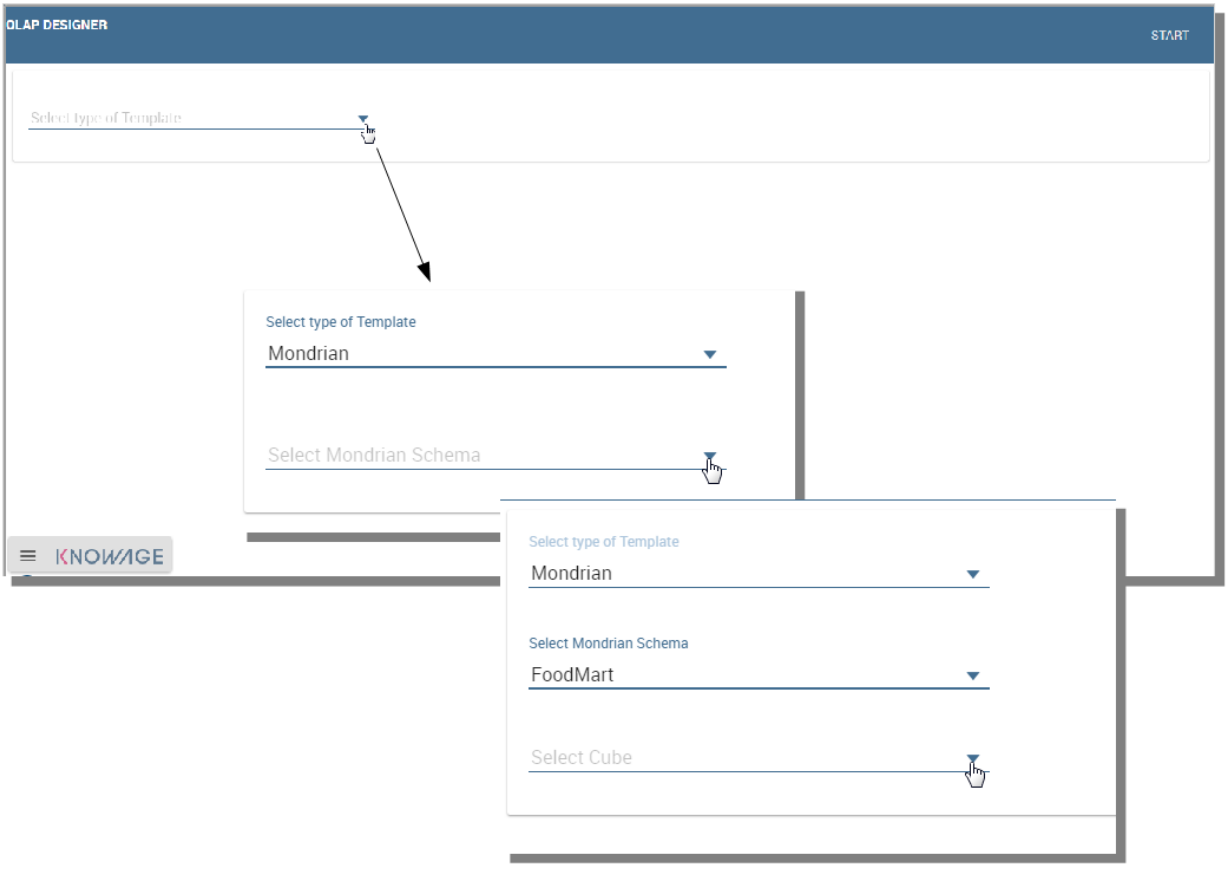


Fig. 5.423: OLAP core configuration.

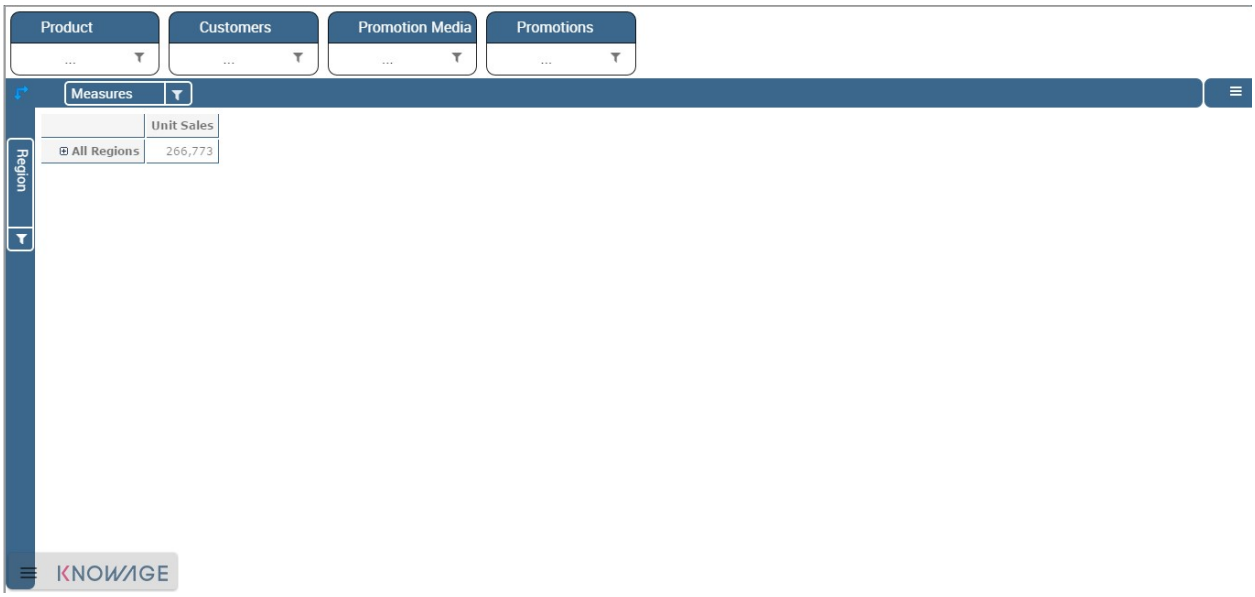


Fig. 5.424: Defining OLAP template.

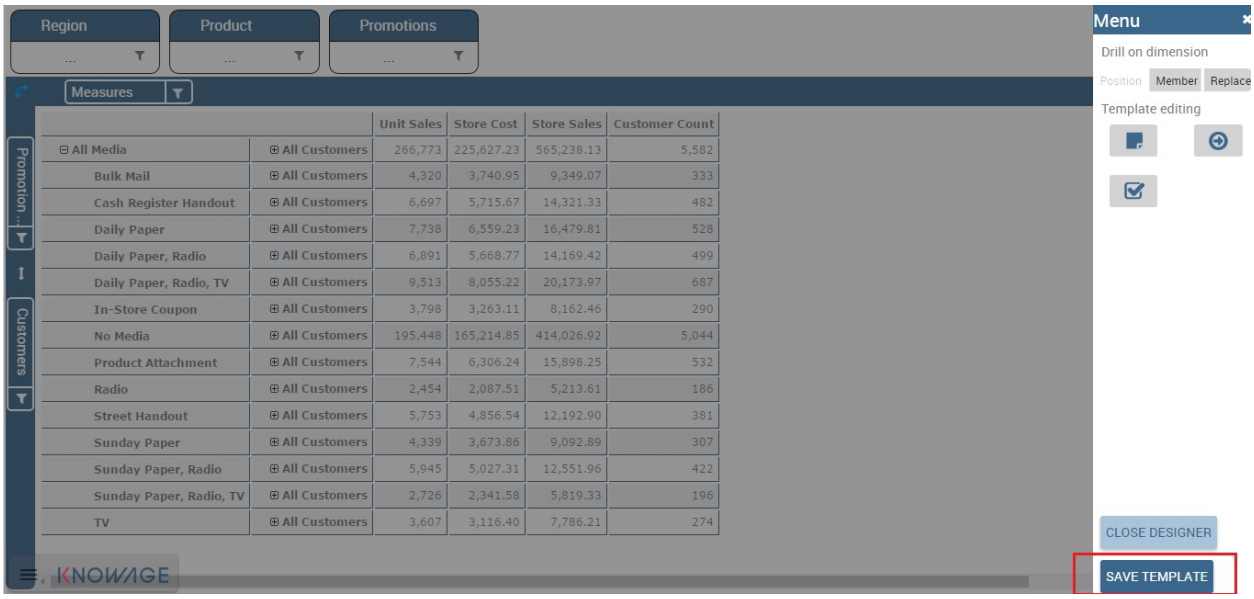


Fig. 5.425: Defining OLAP template.

lets the end-user the permission to edit and modify. Referring to 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 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. 5.426 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.

5.16.3.3.1 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.

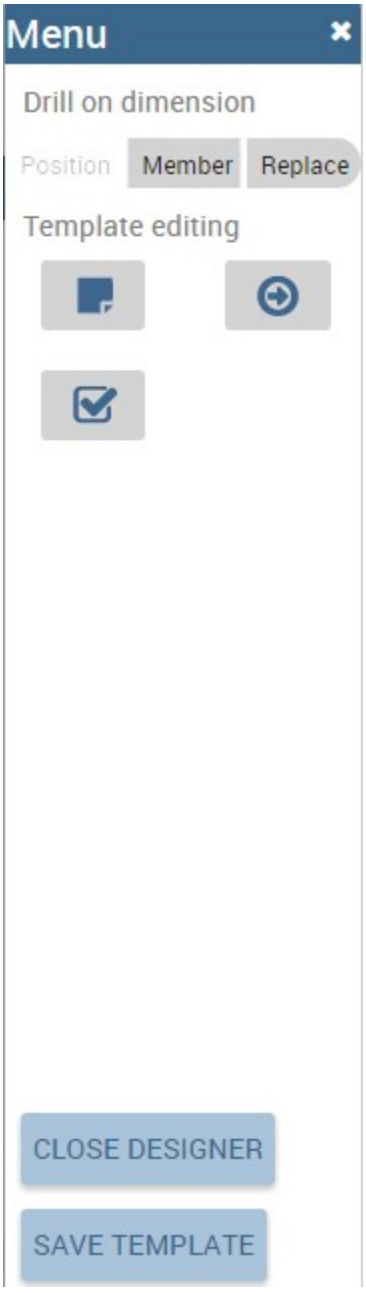


Fig. 5.426: Side panel features for the OLAP Designer.

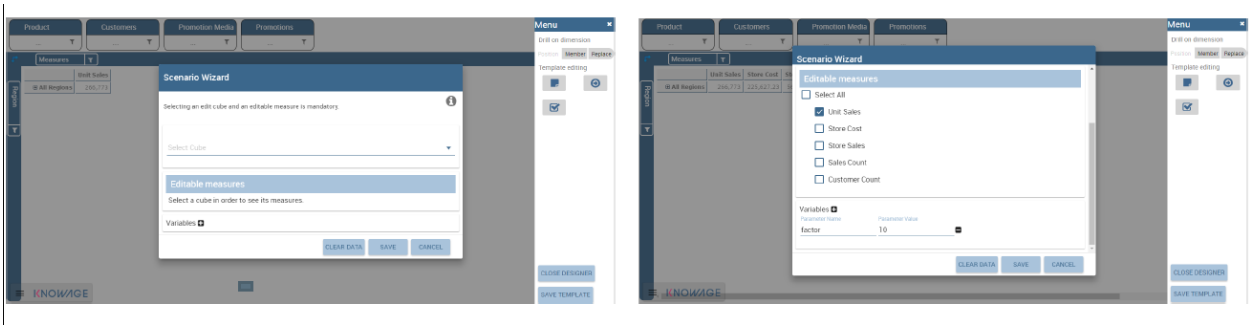


Fig. 5.427: Wizard to configure the scenario.

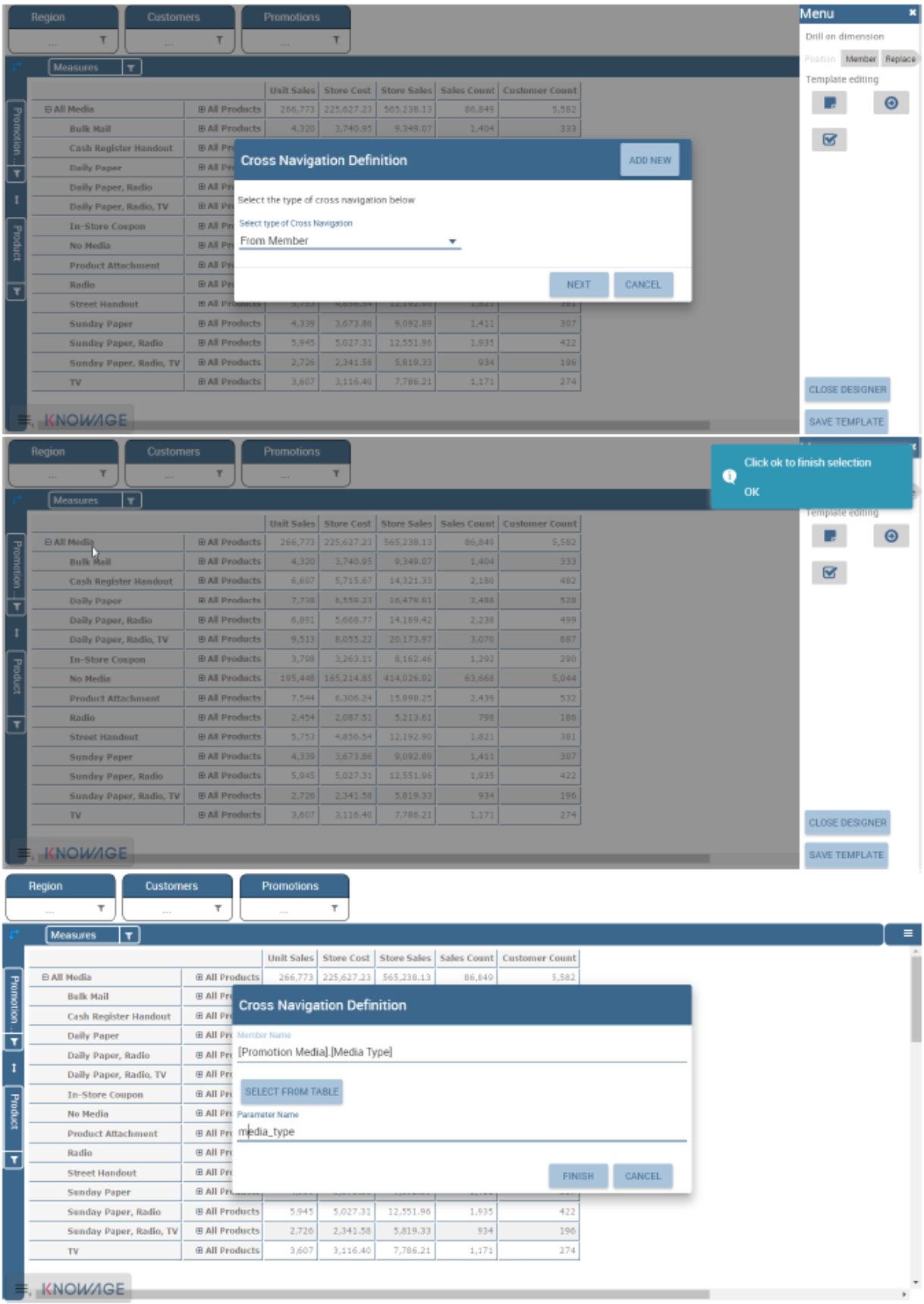
To set the filter, which is based on the attribute (or attributes) in the user's profile, the tecnical user has to type the Mondrian schema. We report Cube level profilation example as a reference guide. Note that data profiling is performed on the cube directly since the filter acts on the data retrieval 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 5.43: Cube level profilation example.

```
<?xml version="1.0"?>
<Schema name="FoodMartProfiled">
....
 <Cube name="Sales_profiled"> <Table name="sales_fact_1998"/>
 ...
   <!-- profiled dimension -->
   <Dimension name="Product" foreignKey="product_id">
    <Hierarchy hasAll="true" allMemberName="All Products" primaryKey="product_id">
        <View alias="Product">
          <SQL dialect="generic">
            SELECT pc.product_family as product_family, p.product_id as
            product_id,
            p.product_name as product_name,
            p.brand_name as brand_name, pc.product_subcategory as
            product_subcategory, pc.product_category as product_category,
            pc.product_department as product_department
            FROM product as p
            JOIN product_class as pc ON p.product_class_id = pc.
            product_class_id
            WHERE and pc.product_family = '${family}'
          </SQL>
        </View>

        <Level name="Product Family" column="product_family"
               uniqueMembers="false" />
        <Level name="Product Department" column="product_department"
               uniqueMembers="false"/>
        <Level name="Product Category" column="product_category"
              uniqueMembers=" false"/>
        <Level name="Product Subcategory" column="product_subcategory"
               uniqueMembers="false"/>
        <Level name="Brand Name" column="brand_name"
               uniqueMembers="false"/>
        <Level name="Product Name" column="product_name"
               uniqueMembers="true"/>
    </Hierarchy>
   </Dimension>
 </Cube>
 ...
</Schema>
```

In the above example, the filter is implemented within the SQL query that defines the dimension using the usual syntax



5.16. Multidimensional Analysis Fig. 5.428: Cross navigation definition.

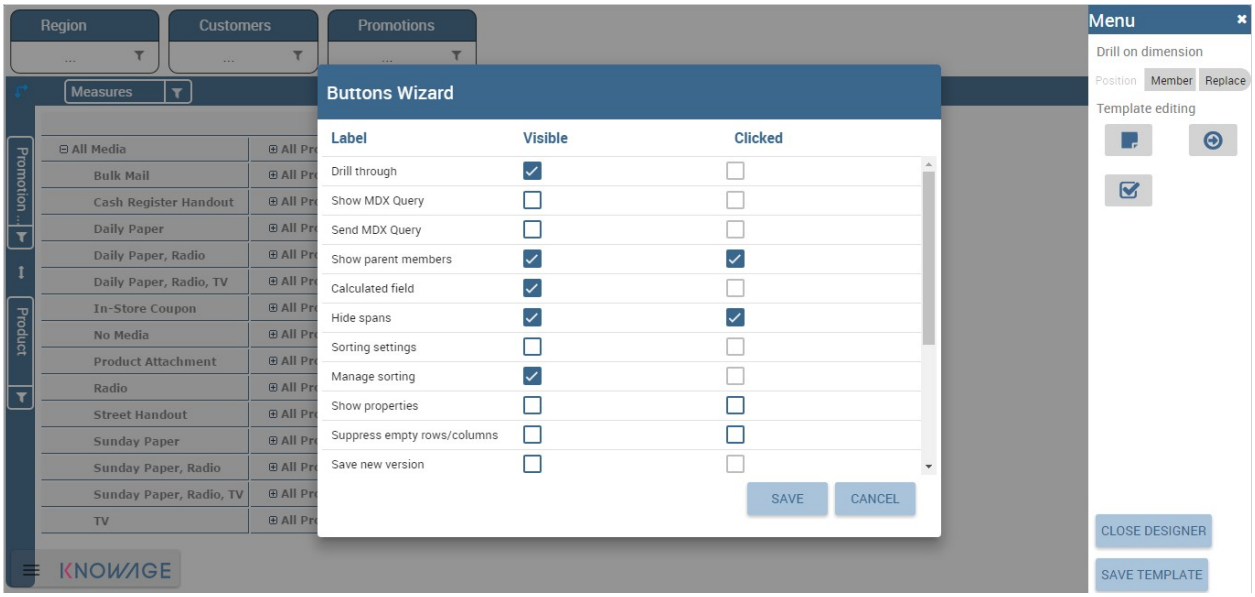


Fig. 5.429: Wizard to configure the scenario.

"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.

5.16.3.4 Cross Navigation

The cross navigation must be implemented at template level but also at analytical document level. The latter has been already wildly 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 declaire 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.

5.16.3.4.1 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.

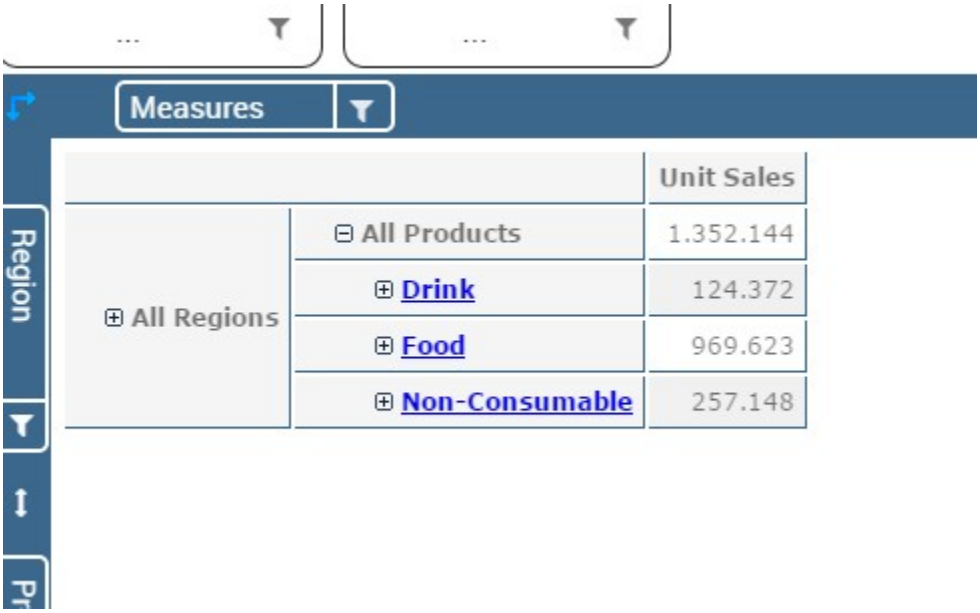


Fig. 5.430: Cross navigation on member.

If you open the template file you will read instructions similar to the ones reported in Syntax used to set cross navigation.

Listing 5.44: Syntax used to set cross navigation.

```
<MDXquery>
  select {[Measures].[Unit Sales]} ON COLUMNS,
  {([Region].[All Regions], [Product].[All Products])} ON ROWS from
  [Sales_V]
  <clickable uniqueName="[Product].[Product Family]" >
     <clickParameter name="family" value="{0}"/>
  </clickable>
</MDXquery>
```

5.16.3.4.2 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 creates two parameters one for family where dimension is product, hierarchy is product, level is product family and one for year parameter where dimension in type, hierarchy is time and level is year. Let 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 5.45: 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 displayed in each cell. User can click on that icon to start cross navigation from a cell.

5.17 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.

5.17.1 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. 5.431: What-If buttons inside the sidebar.

Fig. 5.432: The What-If is used by another user.

- The workflow is finished.

We briefly recall the fuctionality 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.

5.17.2 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 aggragation 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 "=", 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.

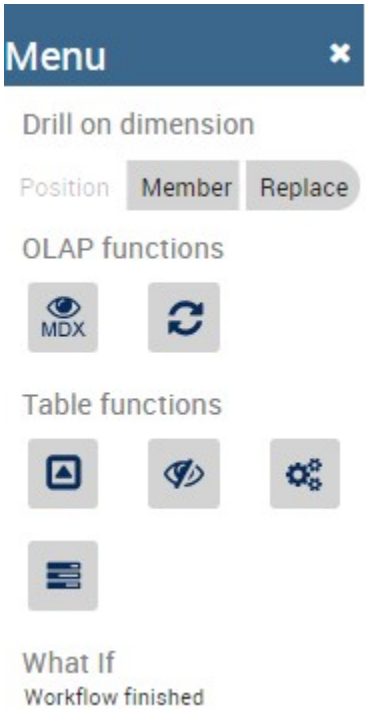


Fig. 5.433: The What-If is finished.

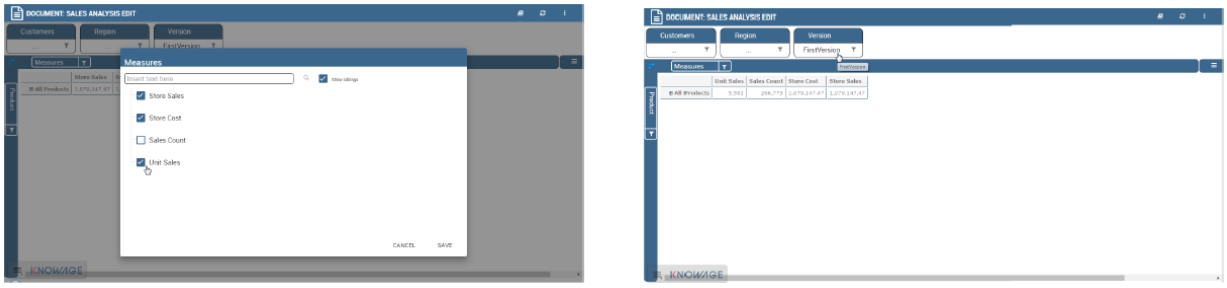


Fig. 5.434: Checking measures and selecting version.

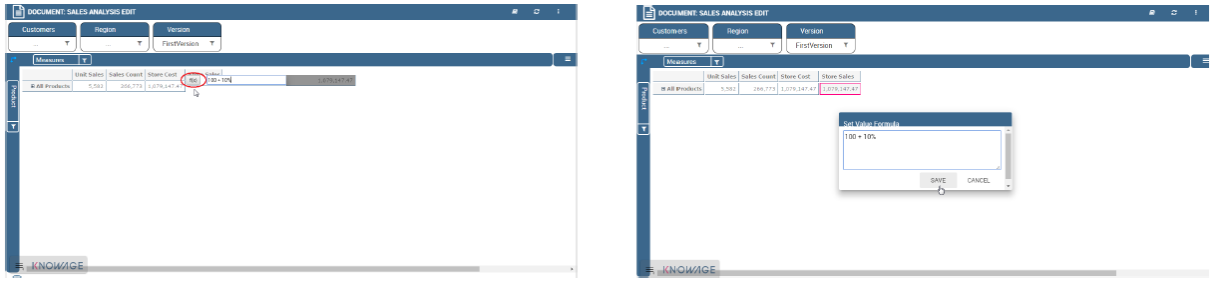


Fig. 5.435: Inserting formula and its convalidation.

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 5.46: Product.Deli

```
[Measures].[Store Sales], [Product].[Food].[Deli], [Version].[0],
[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 5.47: Product.Eggs

```
[Measures].[Store Sales], [Product].[Food].[Eggs], [Version].[0],
[Region].[Mexico Central], [Customers].[All Customers], [Customers].[All Customers]
```

Listing 5.48: Product.Eggs; Measures.Unit Sales

```
[Measures].[Unit Sales], [Product].[Food].[Eggs], [Version].[0],
[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 5.49: Referring to different members.

```
<dimension's name>.<member's name>or[<dimension's name>].[<member's name>]
```

Listing 5.50: Referring to different members of another hierarchy.

```
<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 5.51: Example of cell's tuple.

```
[Measures].[Store Sales], [Product].[Food].[Deli], [Version].[0],
[Region].[Mexico Central], [Customers].[All Customers], [Customers].[All Customers]
```

Note that you can refer to the tuple

Listing 5.52: Example of cell's tuple.

```
[Measures].[Store Sales], [Product].[Drink].[Alcoholic Beverages],
[Version].[0], [Region].[Mexico Central], [Customers].[All Customers],
[Customers].[All Customers]
```

just with:

Listing 5.53: Shorten syntax code.

```
[Product].[Drink.Alcoholic Beverages]
```

Another example from Code below

Listing 5.54: Example of cell's tuple.

```
[Measures].[Store Sales], [Product].[Food].[Deli].[Meat],
[Version].[0], [Region].[Mexico Central], [Customers].[All Customers],
```

to Code below

Listing 5.55: Example of cell's tuple.

```
[Measures].[Store Sales], [Product].[Drink].[Alcoholic Beverages].[Beer and Wine], [Version].[0],
[Region].[Mexico Central], [Customers].[AllCustomers], [Customers].[All Customers]
```

is as in the following code

Listing 5.56: Used expression.

```
[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 5.57: Further example.

```
[Product].[Food]
```

5.17.3 What-if analysis implementation

In this chapter we will deal with some technical fetaures of the What-If analysis that can be handled only by expert users.

5.17.3.1 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

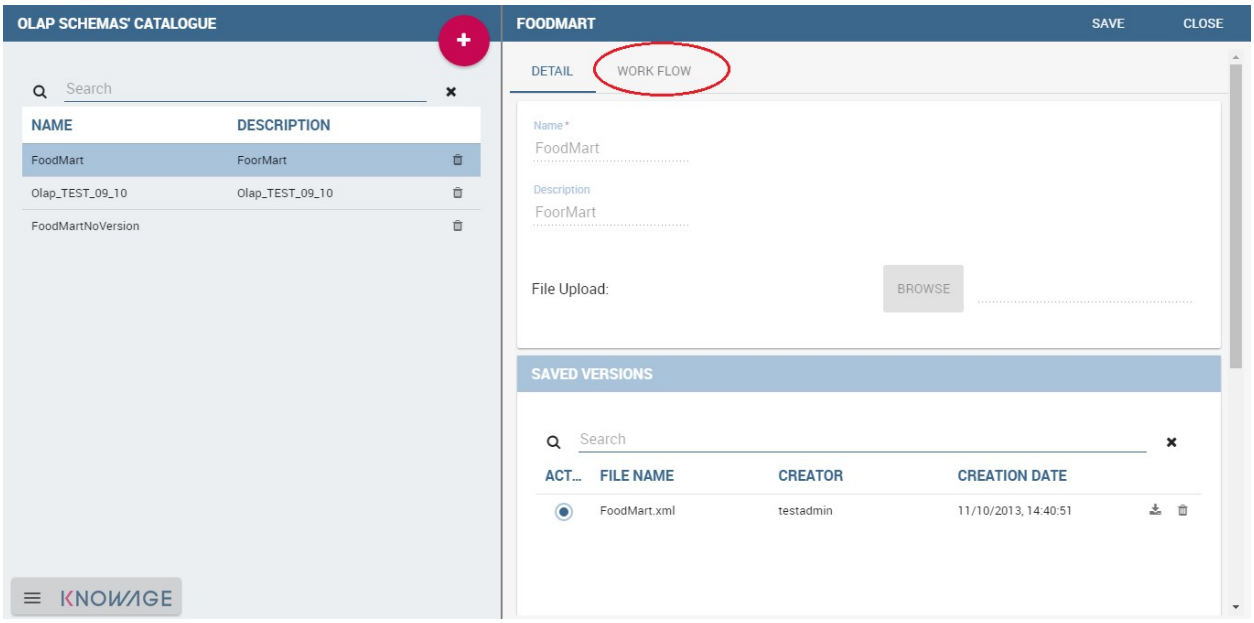


Fig. 5.436: Workflow tab.

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

5.17.3.2 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 belowthere 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”.

5.17.3.3 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 inChanging the Mondrian Schema.

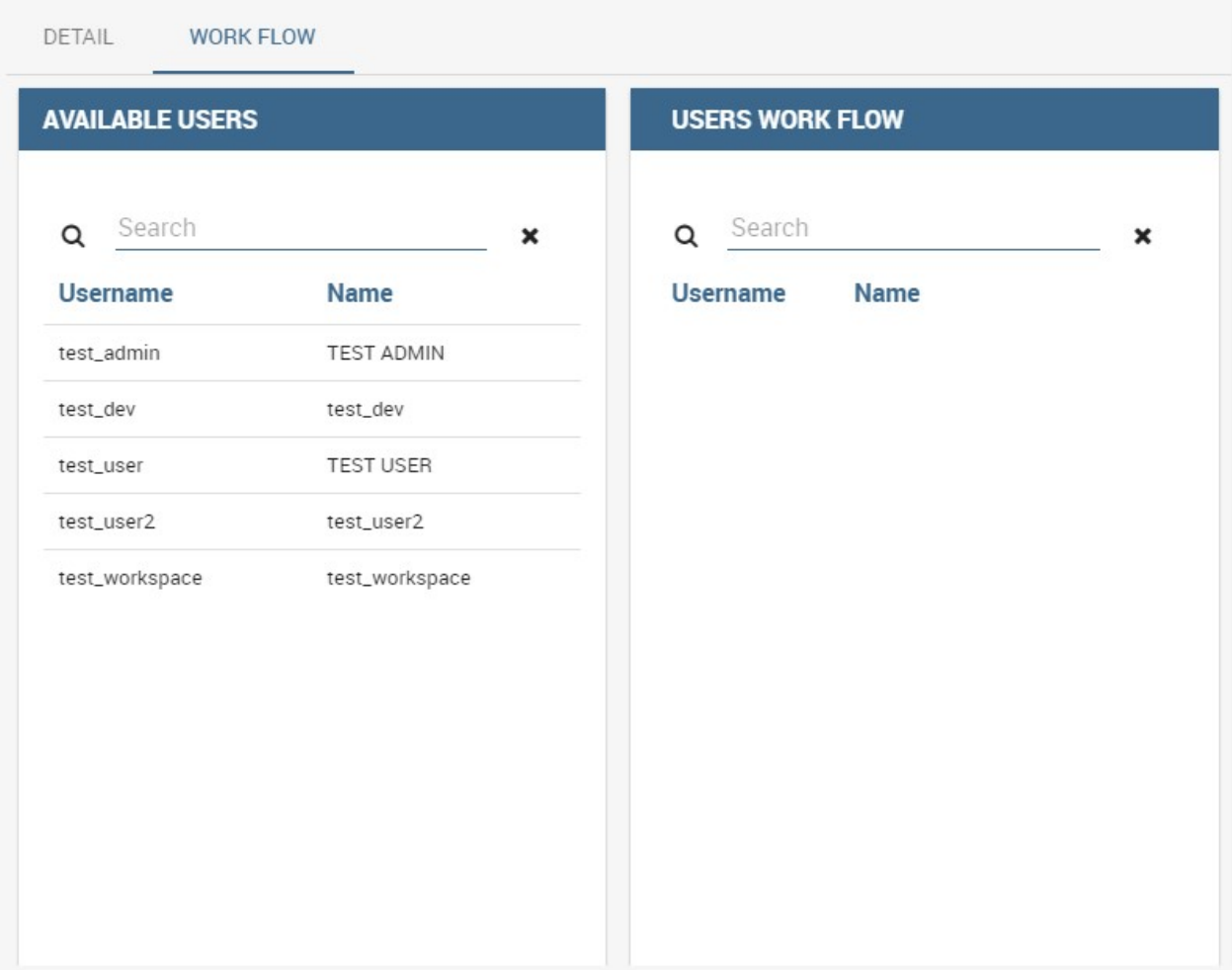


Fig. 5.437: Workflow tab interface.

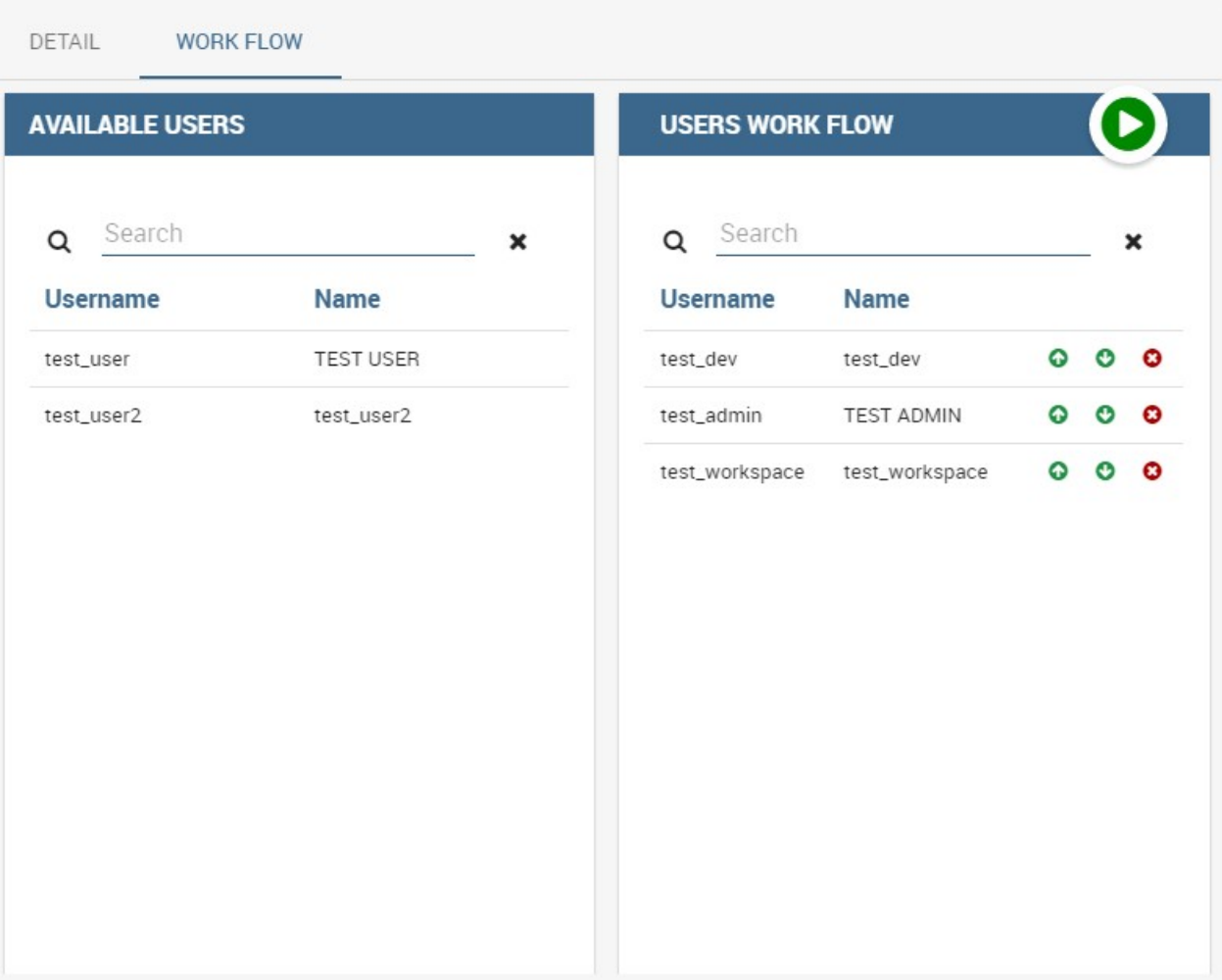


Fig. 5.438: Selecting users for workflows.

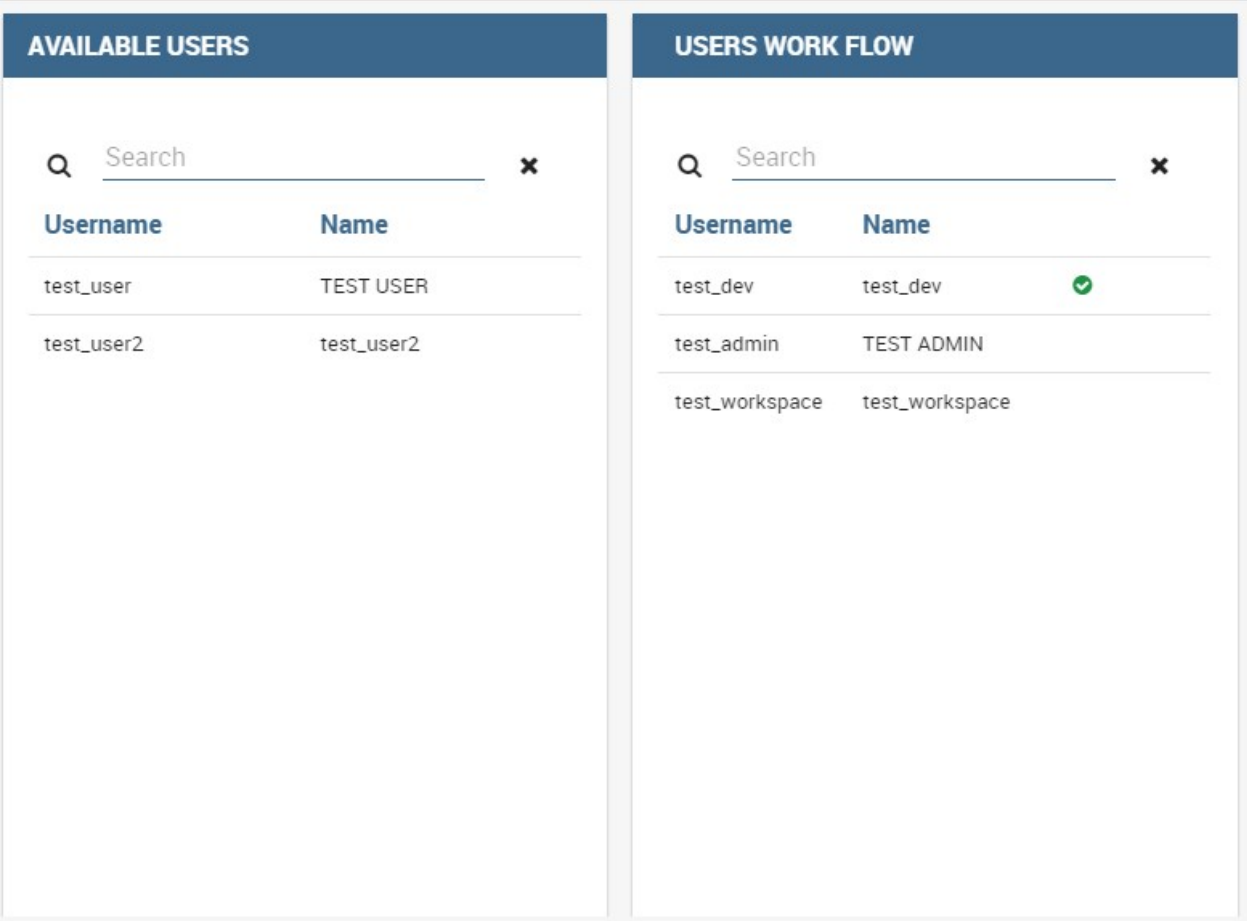


Fig. 5.439: Selecting users for workflows.

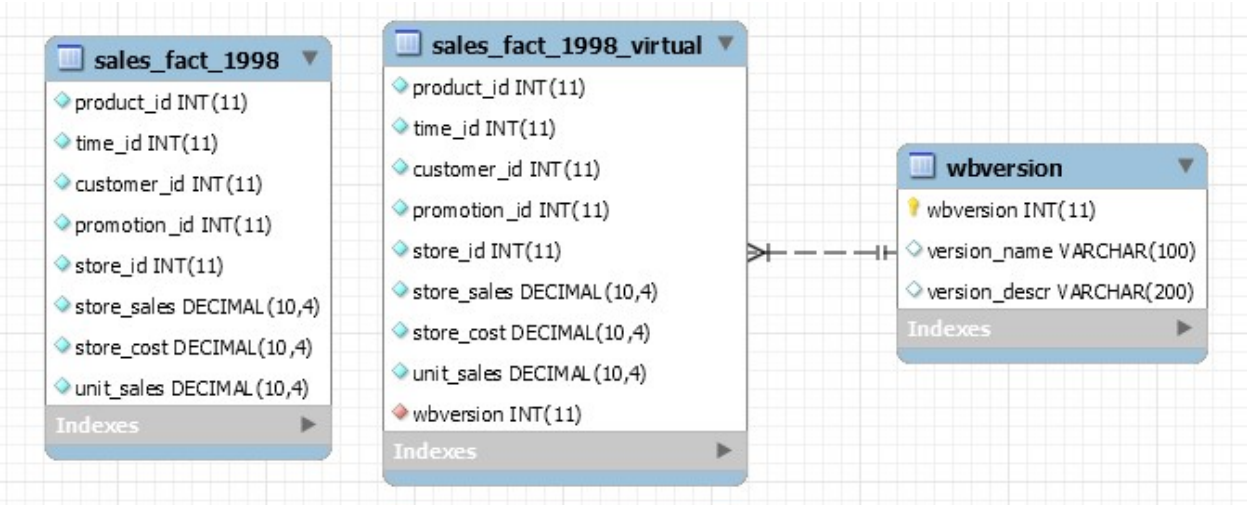


Fig. 5.440: Cube and new virtual table example.

Listing 5.58: Changing the Mondrian Schema.

```
<Dimension name="Version">
   <Hierarchy hasAll="false" primaryKey="wbversion"
   defaultMember="[Version ].[0]" >
   <Table name="wbversion"/>
   <Level name="Version" column="wbversion" uniqueMembers="true"
   captionColumn="version_name"/>
   </Hierarchy>
</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 5.59: Creating the mapping of the editable cube.

```
<Cube name="Sales_Edit">
   <Table name="sales_fact_1998_virtual"/>
   <DimensionUsage name="Product" source="Product"
                   foreignKey="product_id" />
   <DimensionUsage name="Region" source="Region"
                   foreignKey="store_id"/>
   <DimensionUsage name="Customers" source="Customers" foreignKey="customer_id"/>
   <DimensionUsage name="Version" source="Version"
   foreignKey="wbversion"/>
   <Measure name="Store Sales" column="store_sales" aggregator="sum"
   formatString="#,###.00"/>
</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 5.60: Creating the virtual cube.

```
<VirtualCube name="Sales_V">
   <CubeUsages>
      <CubeUsage cubeName="Sales_Edit" ignoreUnrelatedDimensions="true"/>
      <CubeUsage cubeName="Sales" ignoreUnrelatedDimensions="true"/>
   </CubeUsages>

   <VirtualCubeDimension cubeName="Sales" name="Customers"/>
   <VirtualCubeDimension cubeName="Sales" name="Product"/>
   <VirtualCubeDimension cubeName="Sales" name="Region"/>
   <VirtualCubeDimension cubeName="Sales_Edit" name="Customers"/>
   <VirtualCubeDimension cubeName="Sales_Edit" name="Product"/>
   <VirtualCubeDimension cubeName="Sales_Edit" name="Region"/>
   <VirtualCubeDimension cubeName="Sales_Edit" name="Version"/>
   <VirtualCubeMeasure cubeName="Sales" name="[Measures].[Unit Sales Original]" visible="false"/>
   <VirtualCubeMeasure cubeName="Sales" name="[Measures].[Sales Count Original]" visible="false"/>
   <VirtualCubeMeasure cubeName="Sales_Edit" name="[Measures].[Store Sales]" visible="true"/>
   <VirtualCubeMeasure cubeName="Sales_Edit" name="[Measures].[Store Cost]" visible="true"/>

   <CalculatedMember name="Sales Count" dimension="Measures">
      <Formula>VALIDMEASURE([Measures].[Sales Count Original])</Formula>
   </CalculatedMember>

   <CalculatedMember name="Unit Sales" dimension="Measures">
      <Formula>VALIDMEASURE([Measures].[Unit Sales Original])</Formula>
   </CalculatedMember>
</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 doesnt 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 solve 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.

5.18 Glossary and data lineage

The **Glossary** functionality offers a way to find documents by browsing an index page.

5.18.1 Glossary management

The Glossary management is split in two section. Once logged in, the user can find the two menu items: **Glossary Definition** and **Glossary Usage**, as showed below.

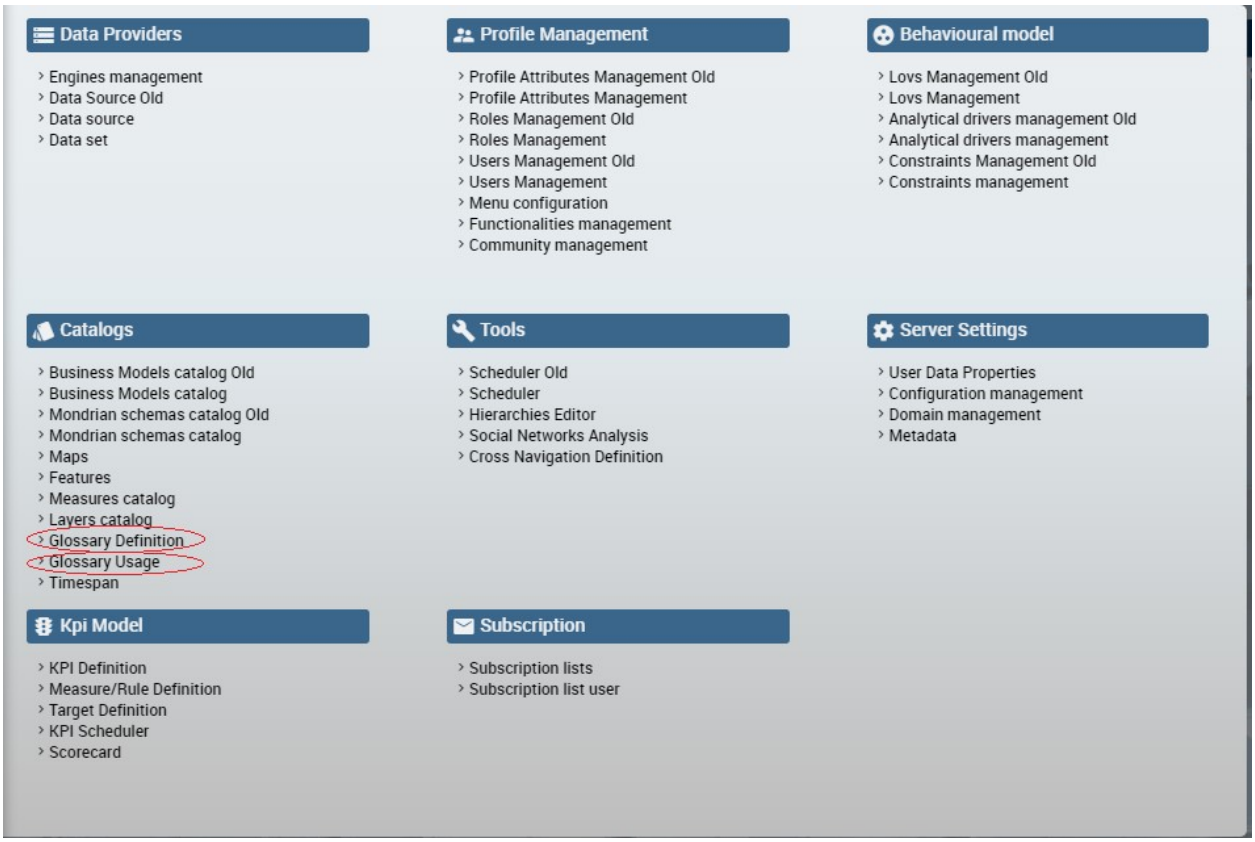


Fig. 5.441: Glossary menu items.

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. 5.442: 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.

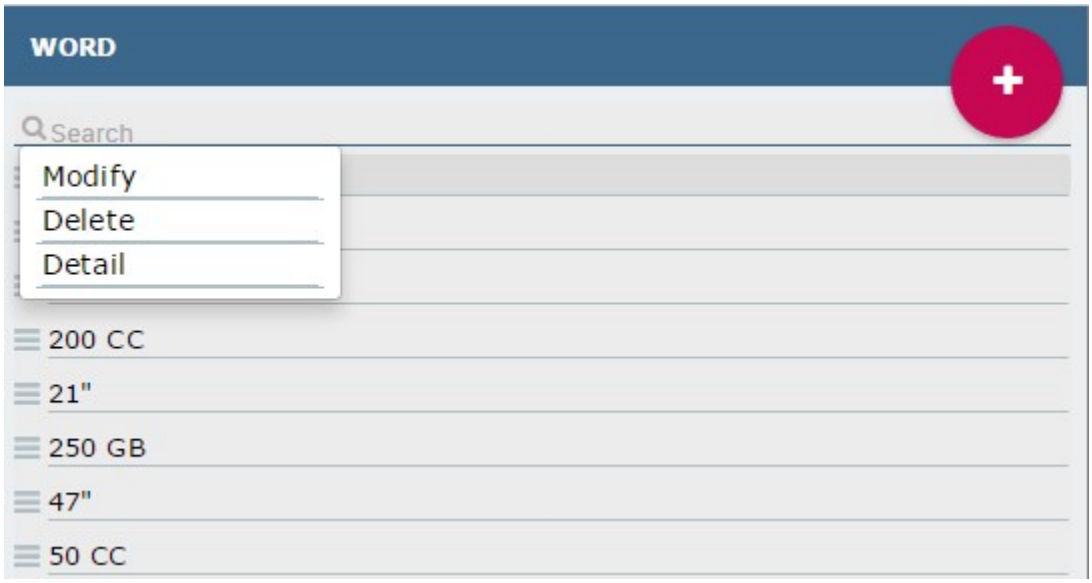


Fig. 5.443: Exploring an existing word.

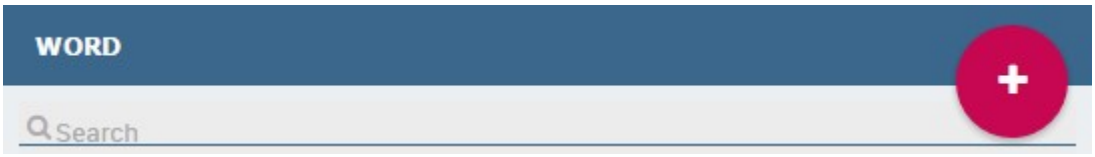


Fig. 5.444: Add a new word.

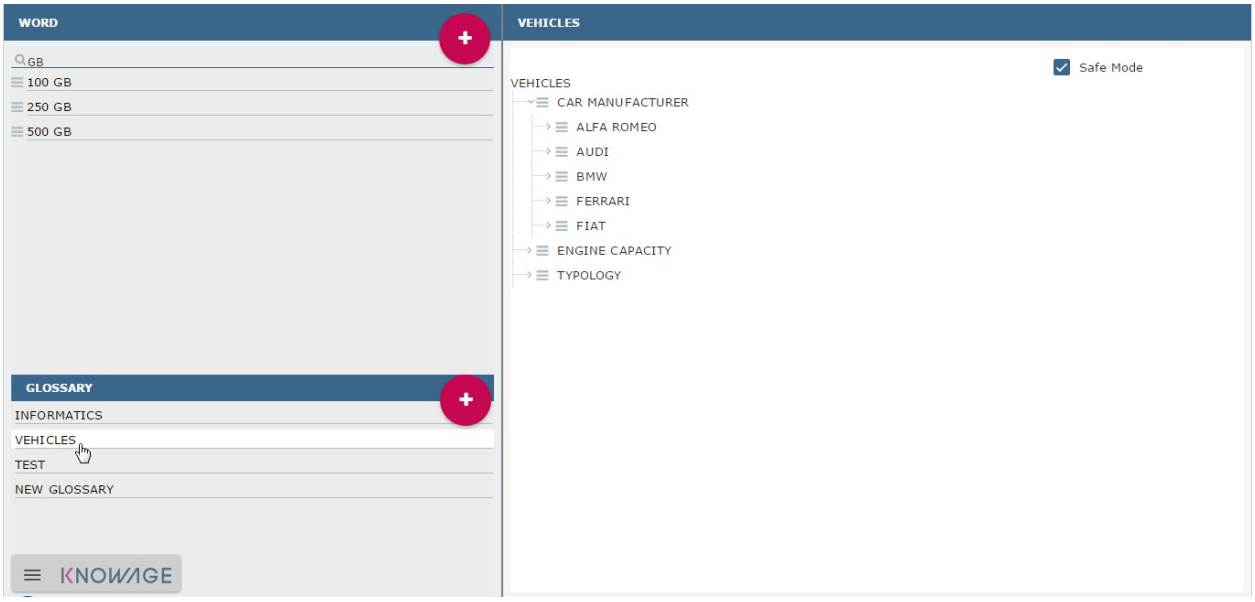


Fig. 5.445: Exploring a glossary from the menu.

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.

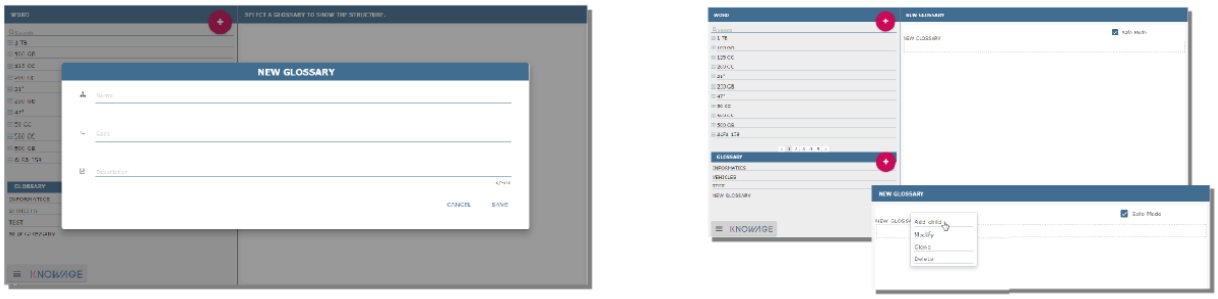


Fig. 5.446: (Left) New glossary wizard. (Right) Add a new child to the glossary.

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. 5.447: 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.

5.18.2 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**.

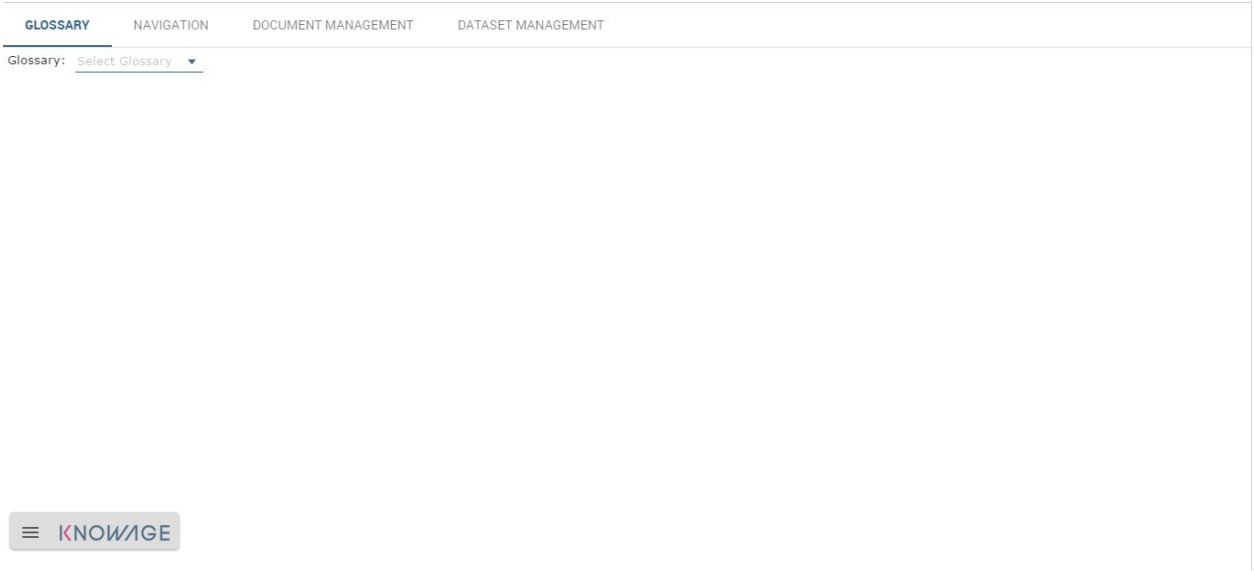


Fig. 5.448: Glossary Usage graphic interface.

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 colums: 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.

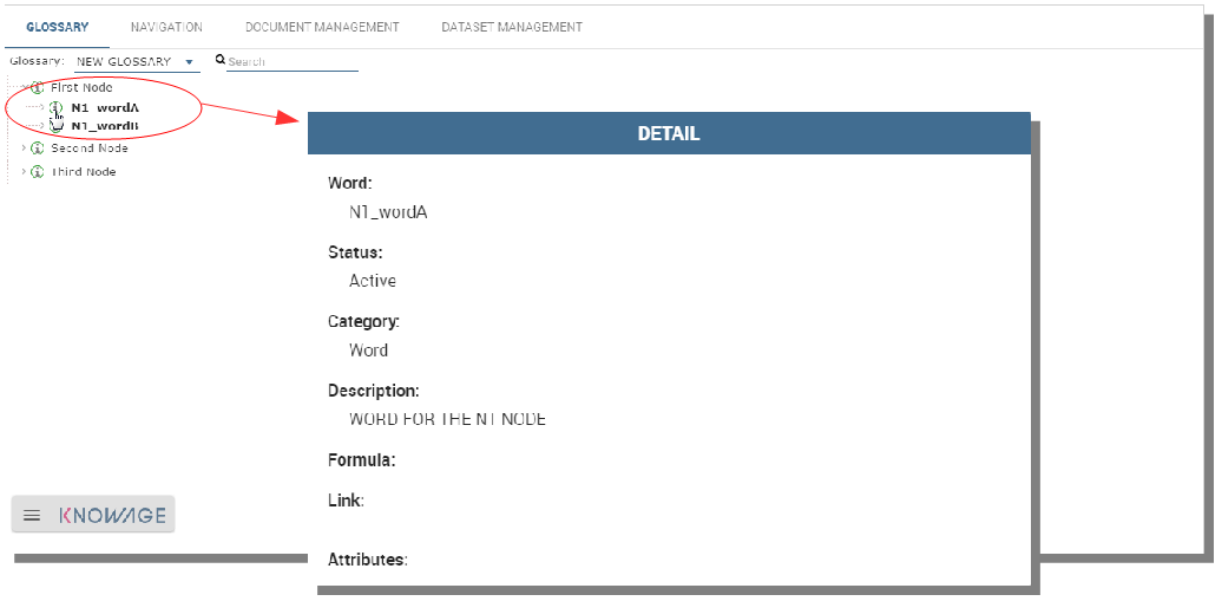


Fig. 5.449: Visualization of glossary details.

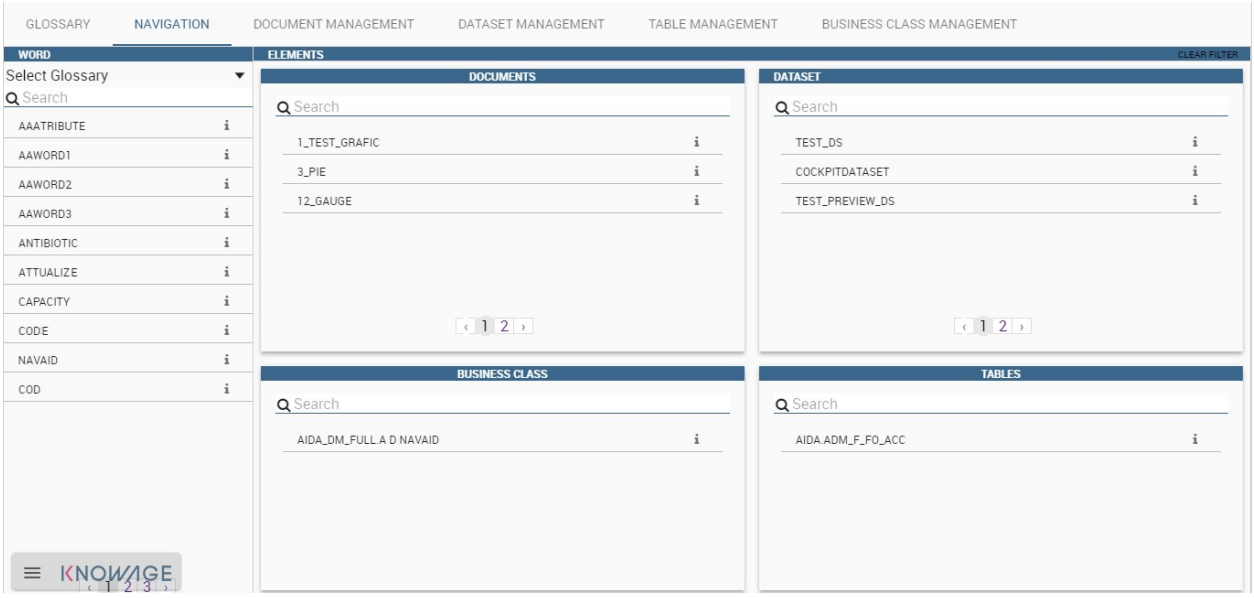


Fig. 5.450: Navigation tab window.

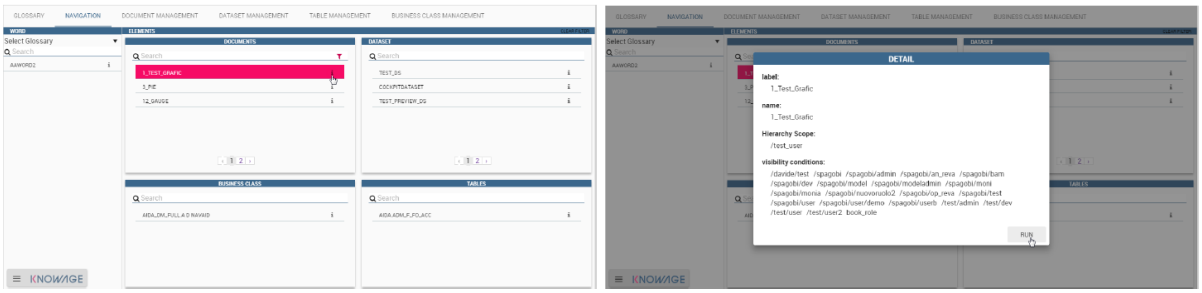


Fig. 5.451: Execution documents by means of the glossary.

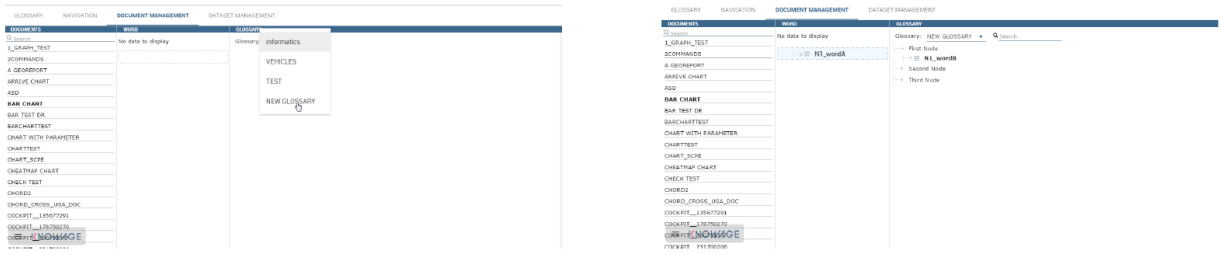


Fig. 5.452: Managing the association with a document: (Left) Select the documnet. (Right) Associate one (or more) word(s).

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.

5.18.3 Help Online functionality

The user can inspect the association of a specific analytical element (dataset, document or model) by using the **Help Online** funcitonality. 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.

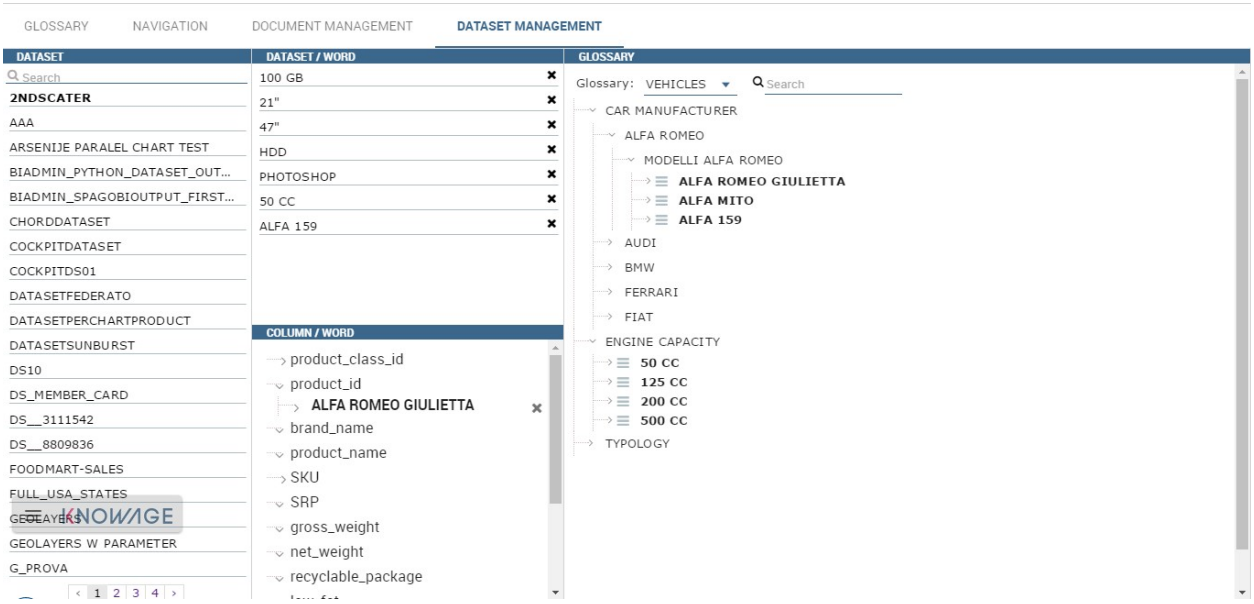


Fig. 5.453: Dataset management tab.

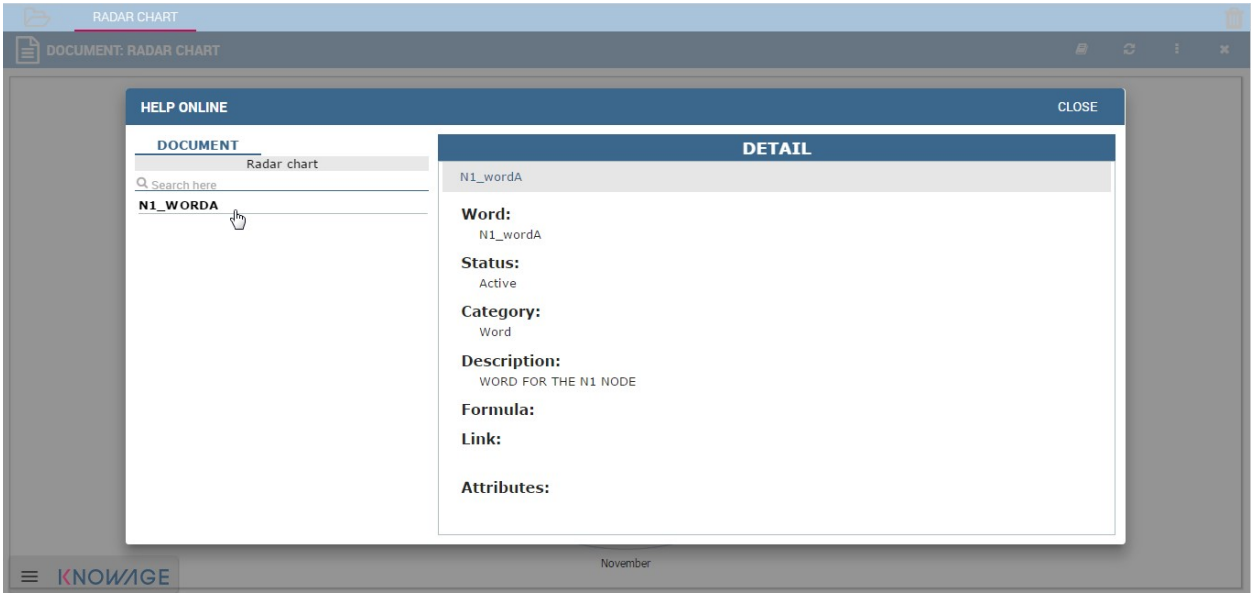


Fig. 5.454: Help Online wizard.

5.19 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).

5.19.1 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 implemetation.

- **Measure definition**: it is necessary to define first the measures and attributes, eventually with parameters, to compute the critical value of interest.
- **KPI definition**: here you compute the requested value through a formula using the measures and attributes set in previuos step and configure thresholds.
- **Target**: it is possible to monitor the value of one or more KPIs comparing it to an additional fixed value.
- **Schedulation**: you can schedule the execution of one or more KPIs, eventually filtered by conditions.
- **Document creation**: finally, you develop the KPI document.

Therefore, we go into further details.

Measure definition. The first step is to create a new measure or rule. Select **Measure/Rule Definition** from the contextual menu of the main page, as shown 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 CRTL key and the space bar opens a contextual menu containing the available datasource columns and the database keywords. Refer to thefollowing 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 schedulation of the KPI (this procedure will be further examined in "Document creation" paragraph). The syntax to use in queries for setting a placeholder is columnName=@placeholderName, as the example in figure 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 fields returned by the query using the **Metadata** tab as highlighted in the next figure. The typology is required to associate a type to each

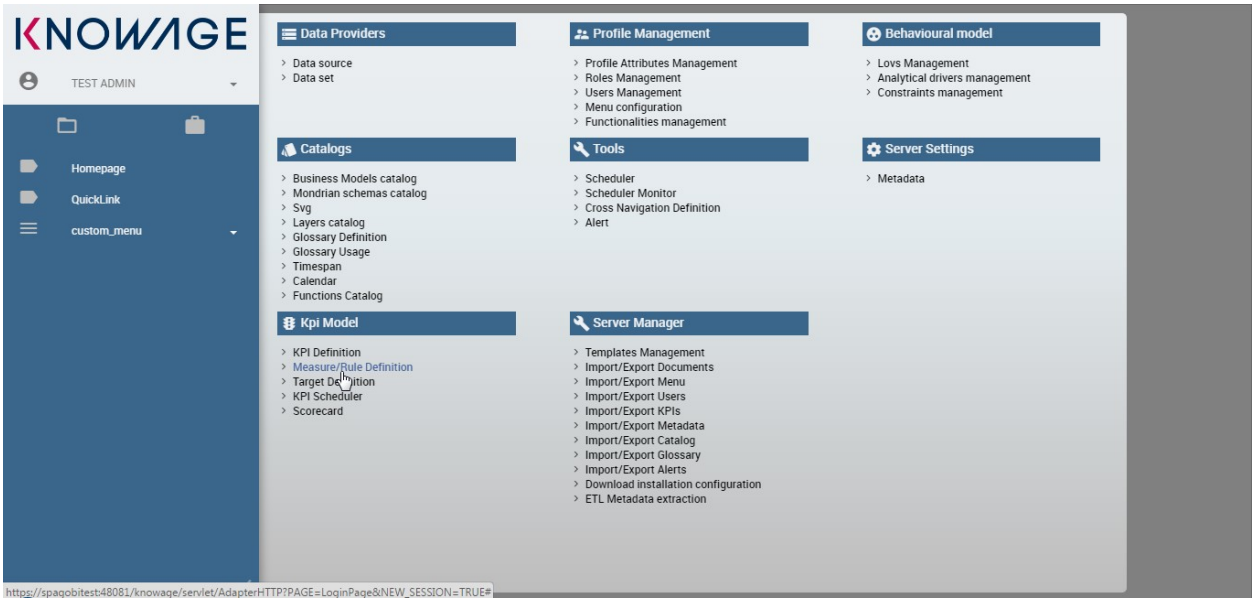


Fig. 5.455: Measure/Rule Definition menu item.

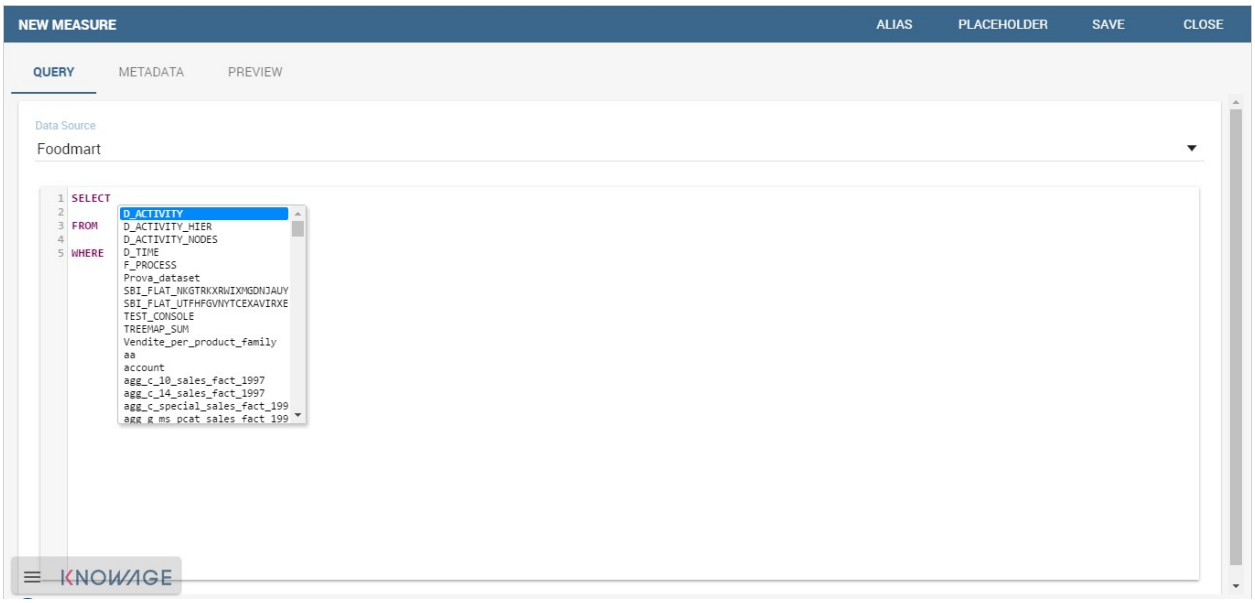


Fig. 5.456: Editing the query when defining a KPI.

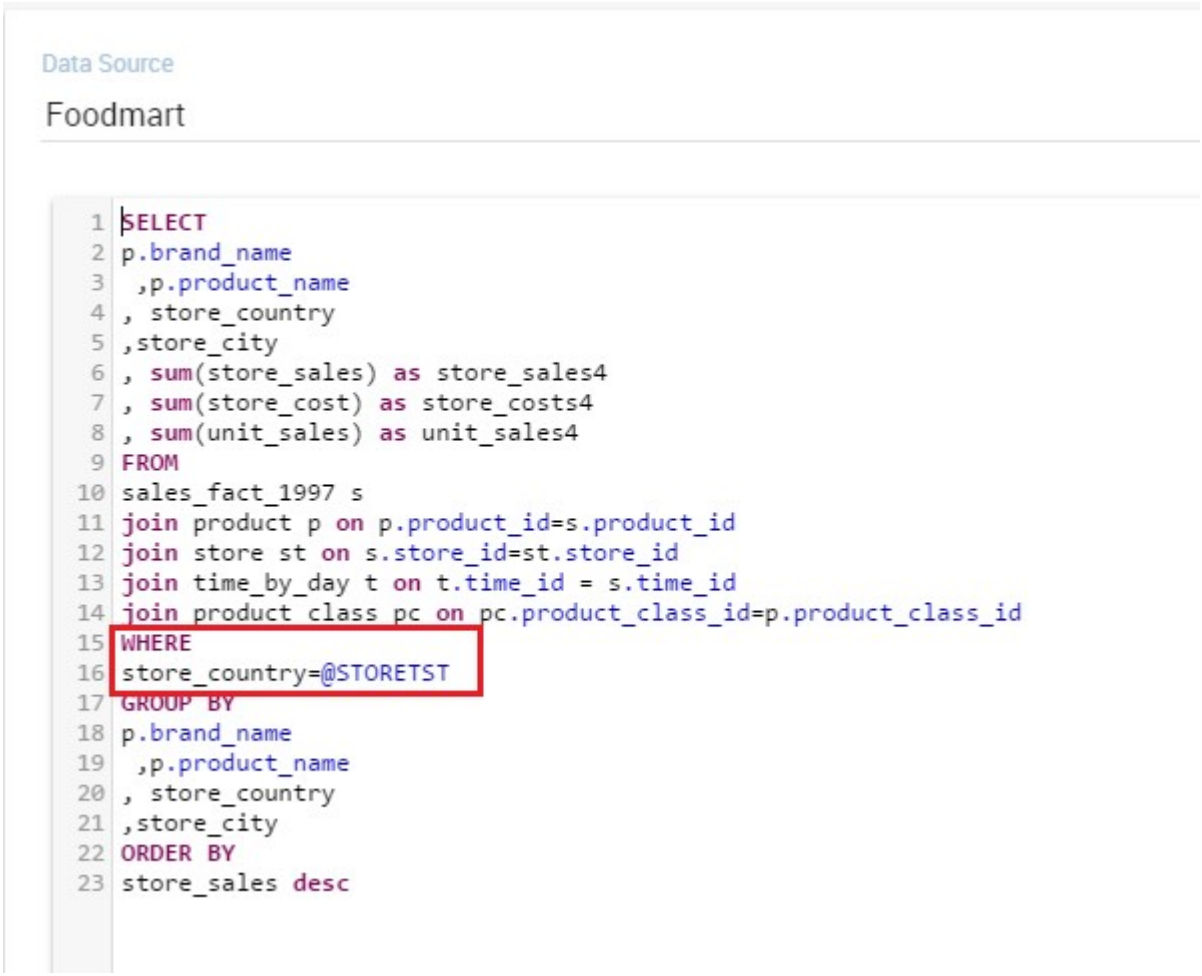


Fig. 5.457: Setting placeholder in query definition.

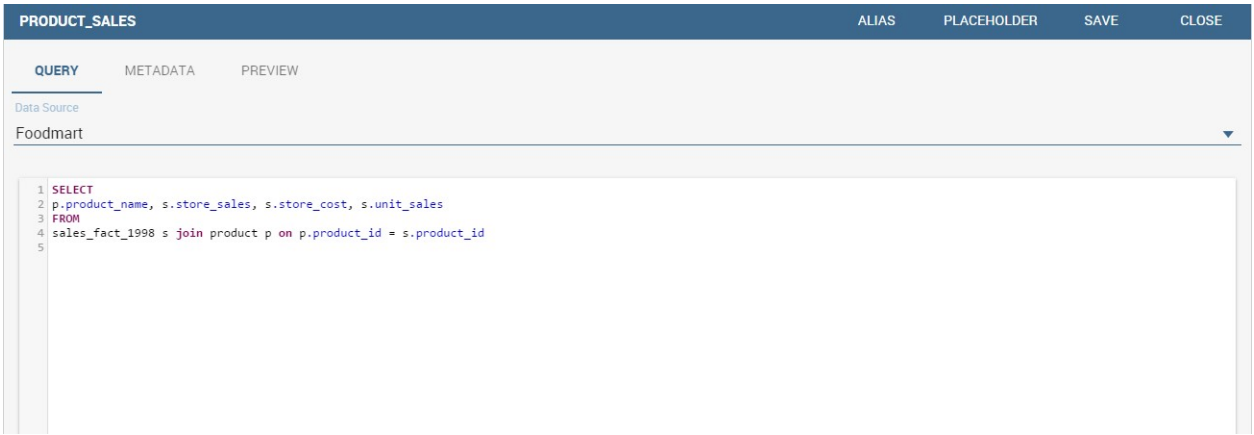


Fig. 5.458: Query definition.

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.

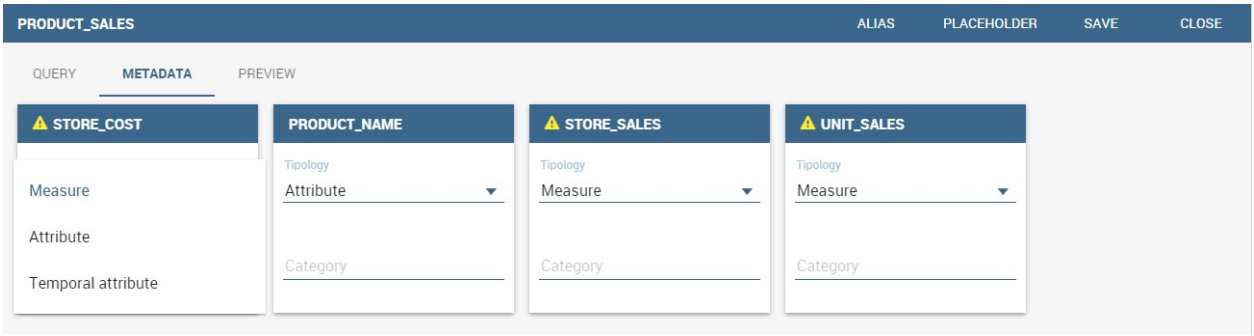


Fig. 5.459: Metadata settings.

We say in advance that, it is important to distinguish these metedata categories from the required field "Category" that occurs while saving the KPI definition (see next figure).

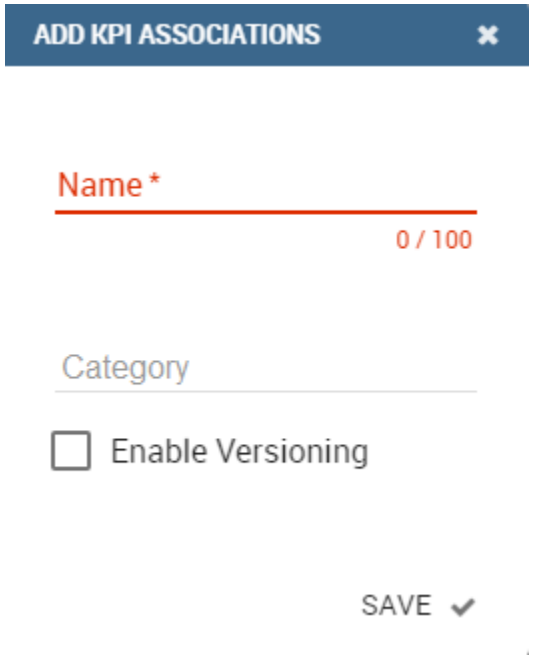


Fig. 5.460: Category assigned when saving a KPI definition.

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

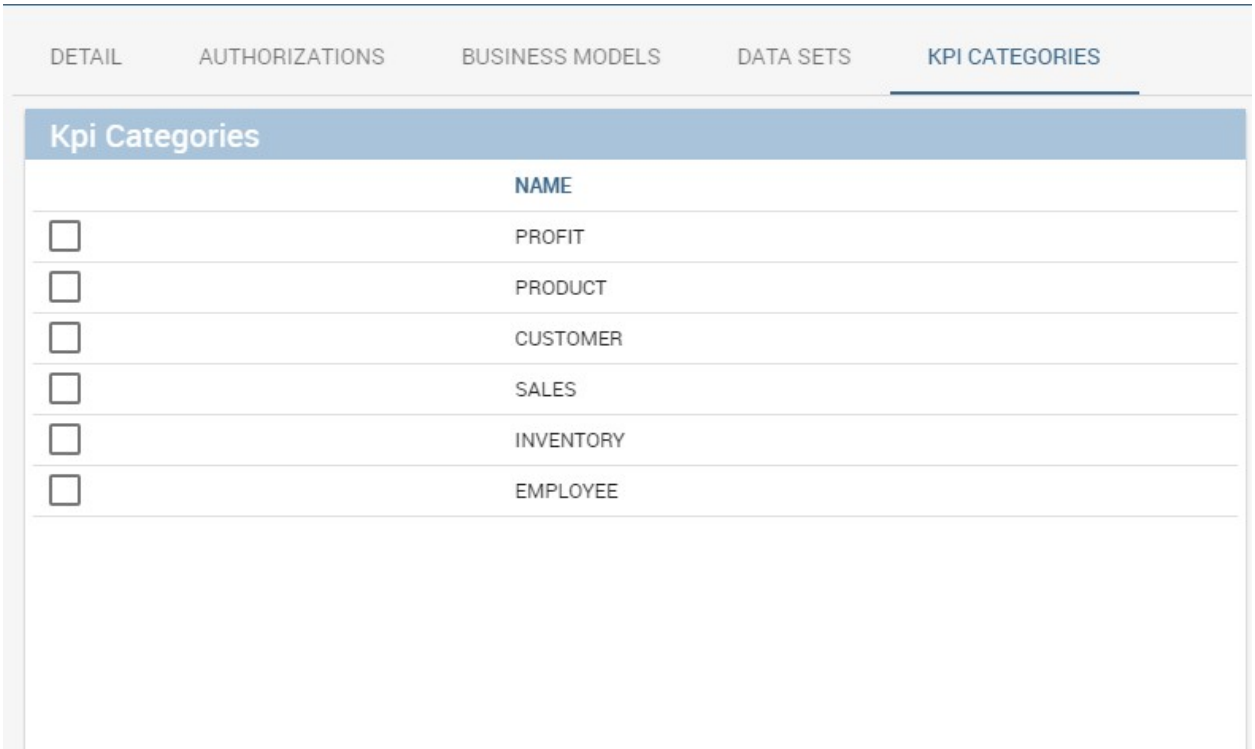


Fig. 5.461: KPI category.

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 colums saved as attribute are stored and showed. This is useful in order to avoid aliases already in use when defining a new rule. Indeed an alias can not be saved twice even if contained in different rules.
- **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.

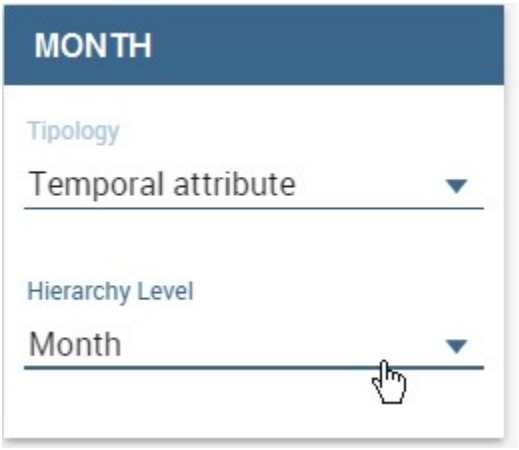


Fig. 5.462: Hierarchy level for temporal attributes.

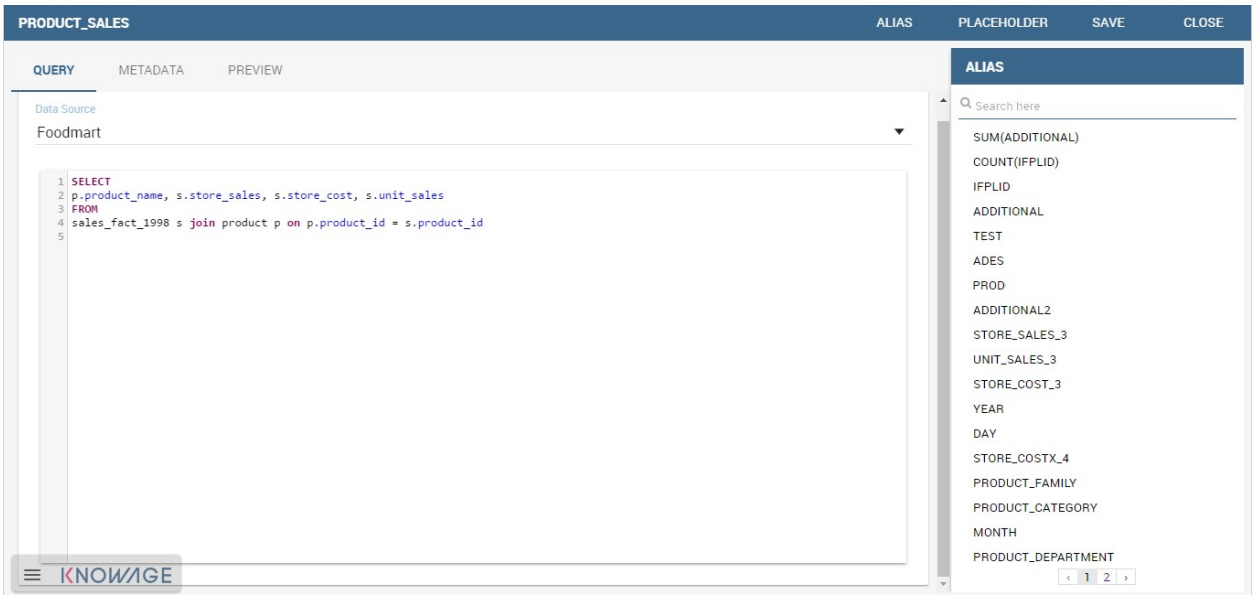


Fig. 5.463: Checking aliases.

Fig. 5.464: Setting placeholders in a query.

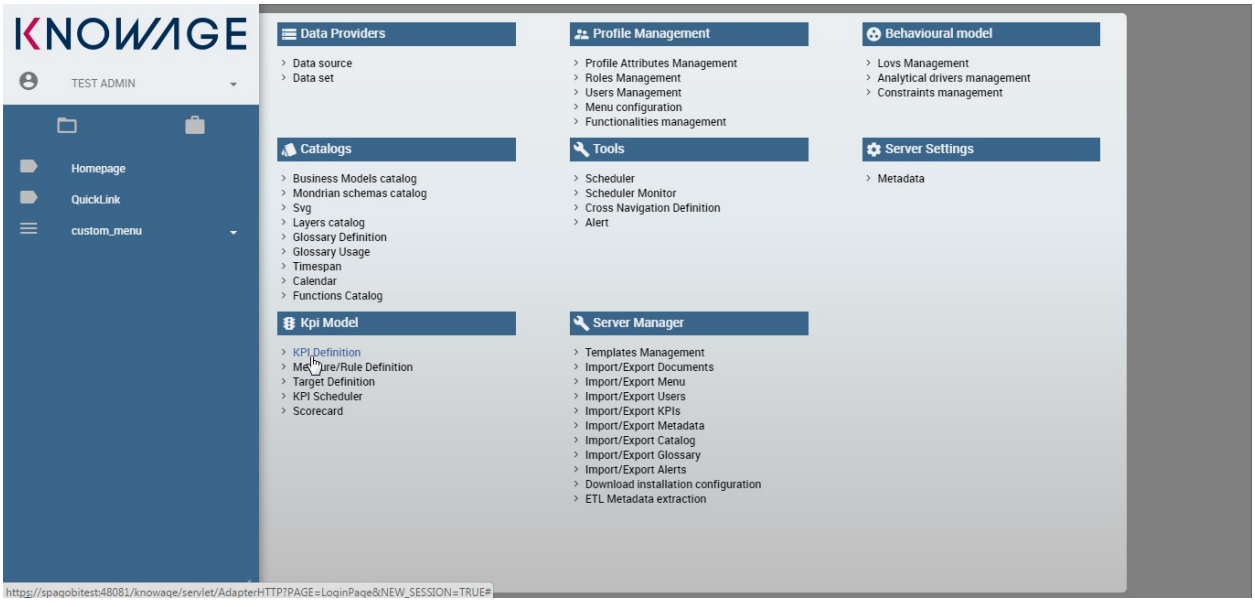


Fig. 5.465: Configure a new KPI.

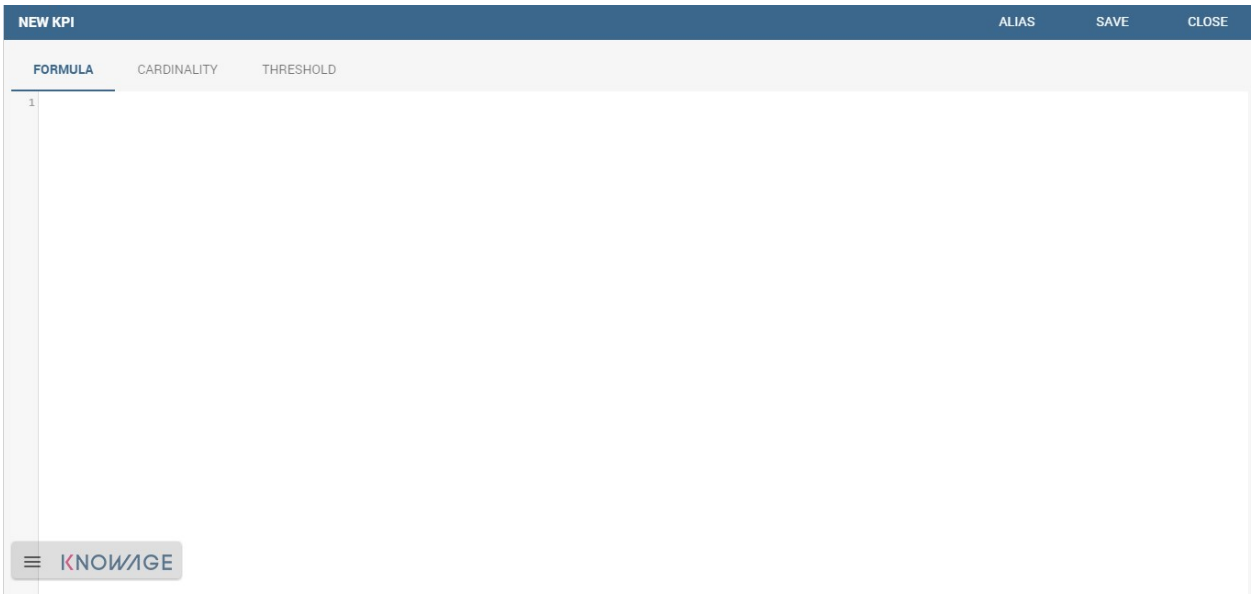


Fig. 5.466: Formula definition tab.

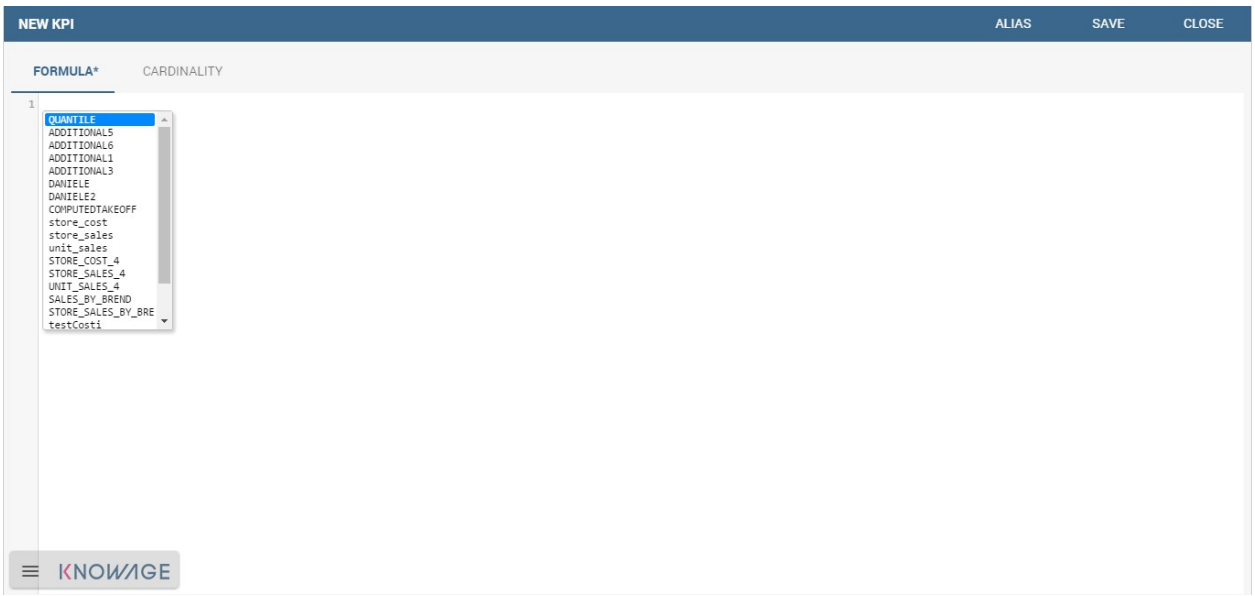


Fig. 5.467: Press crtl and space to get measures.

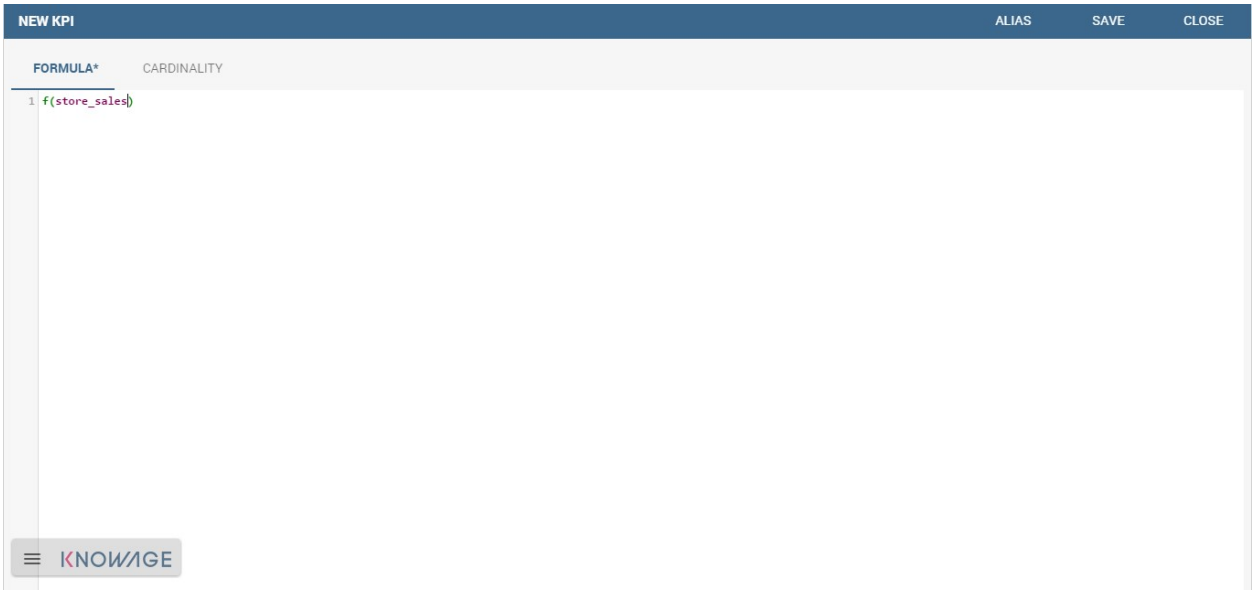


Fig. 5.468: Formula syntax.

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 sintax and you can go on editing it.

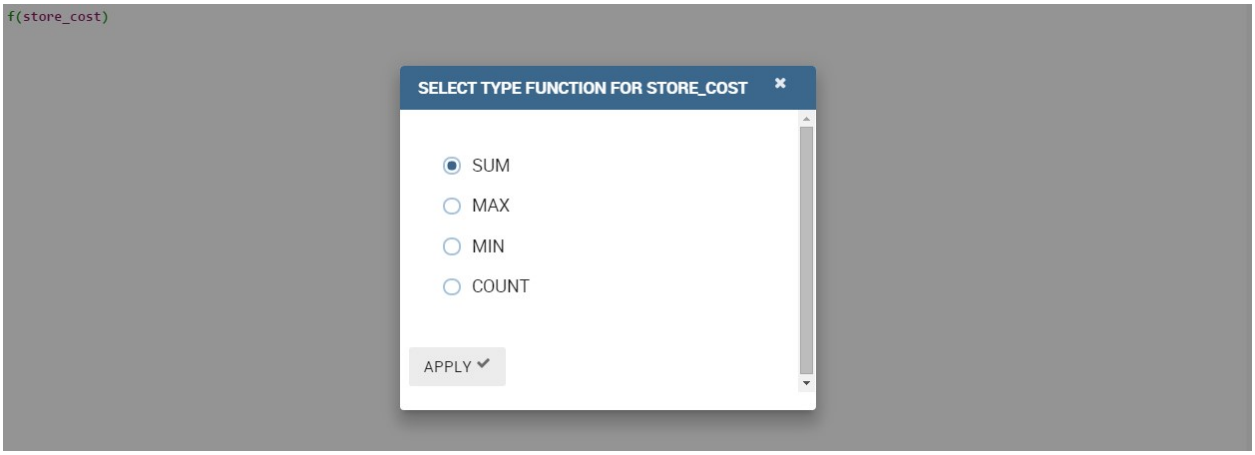


Fig. 5.469: Available functions.

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**

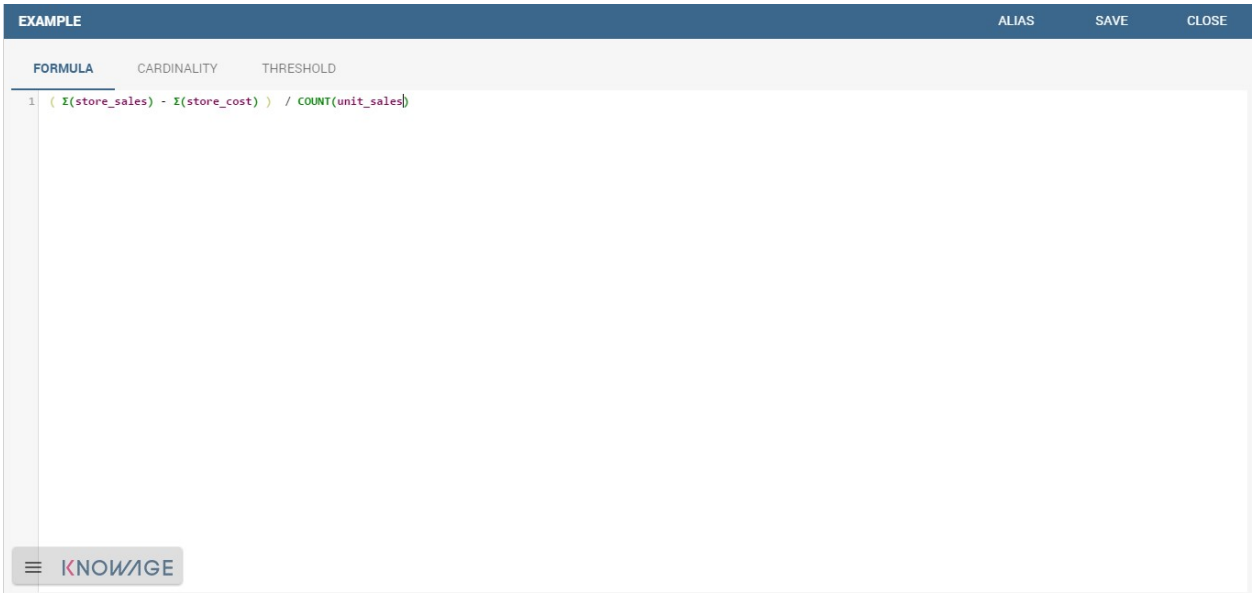


Fig. 5.470: Complete formula example.

Fig. 5.471: Cardinality settings example.

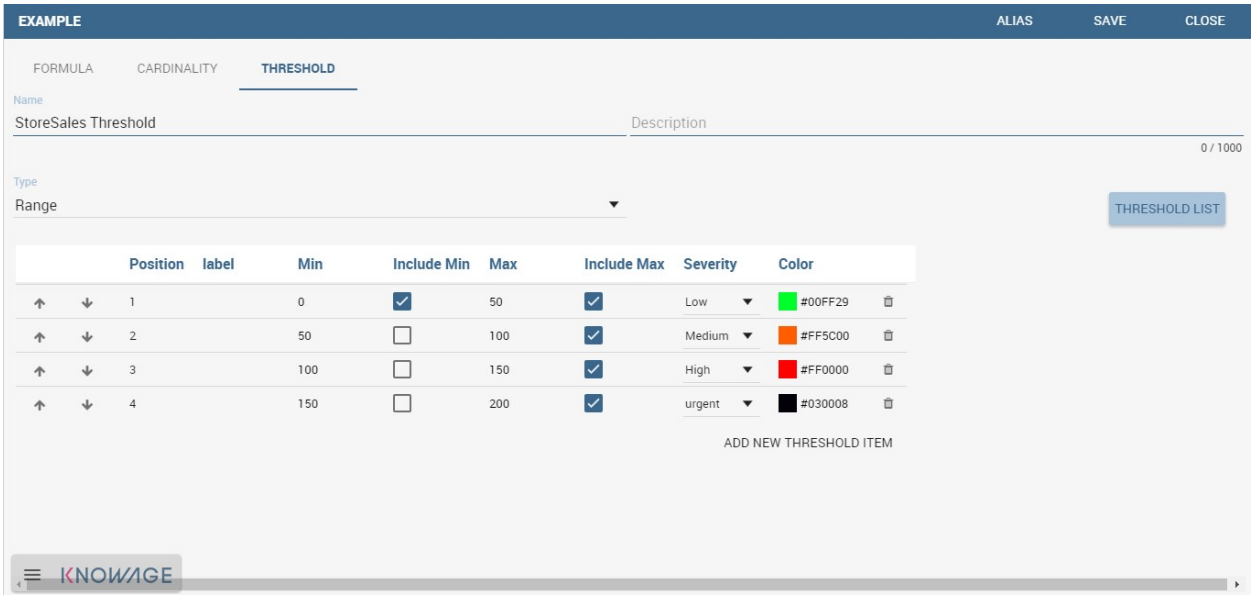


Fig. 5.472: Setting thresholds.

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

Well call **standard colors** for thresholds the ones listed below (in terms of hexadecimals):

- green: #00FF00,
- yellow: #FFF00,
- 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 permmissions to this specific category can access the KPI. Remember that it is possible to assign permissions over KPI when defining roles using the "Roles management" functionality available on Knowage main page. Furthermore, the user can check or uncheck the "**Enable Versioning**" button if he/she wishes to keep track of the rules/measures/targets that generate the KPI response at each KPI execution.

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 possibile to associate a target clicking on the item **Add KPI** and selecting the KPI of interest. Once the association is set, the "Value" box gets editable and you can insert the value you wish to send to the selected KPI. An example is given in figure 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.

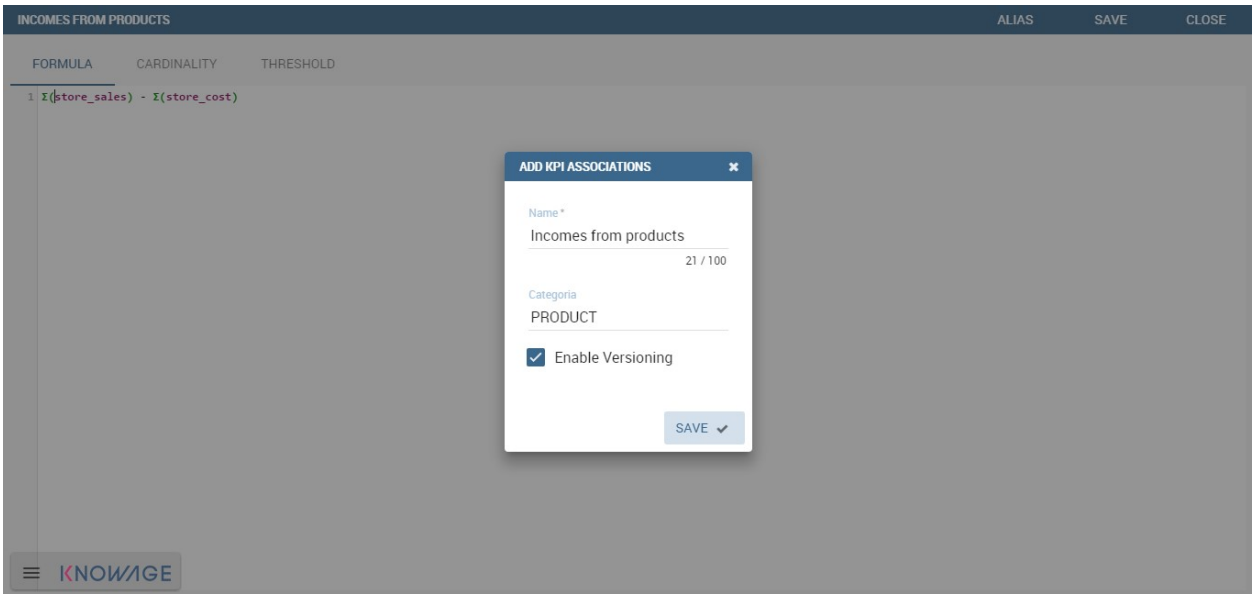


Fig. 5.473: Save the KPI definition and set category.

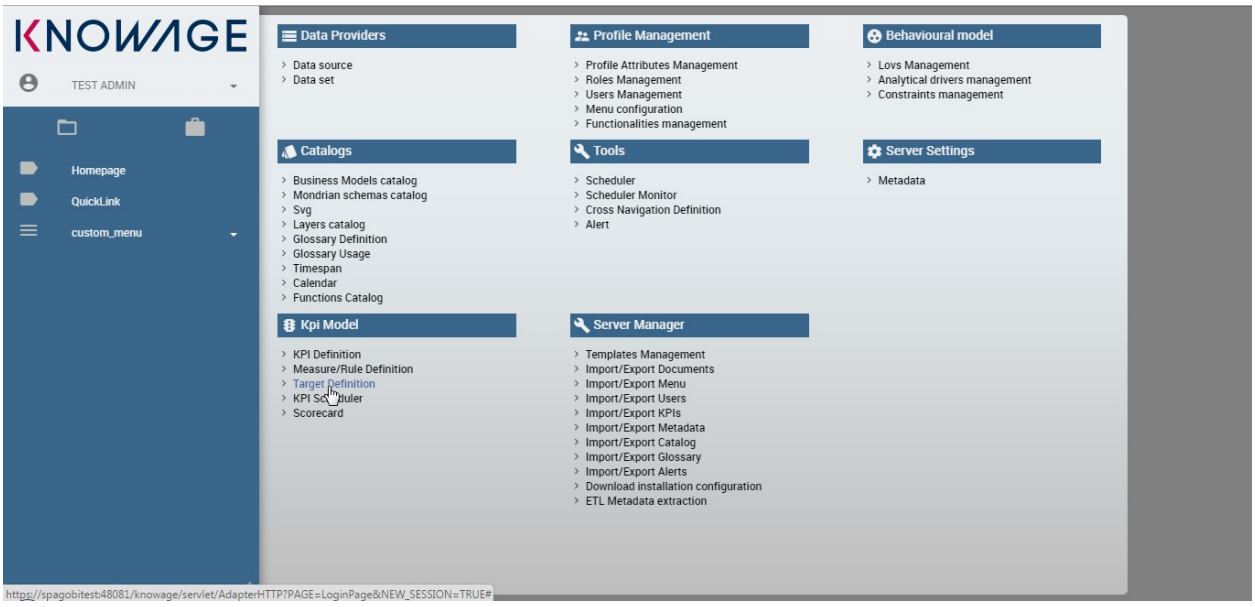


Fig. 5.474: Target Definition menu item.

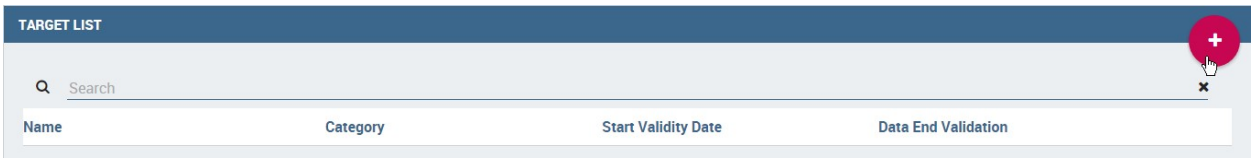


Fig. 5.475: Add a new target.

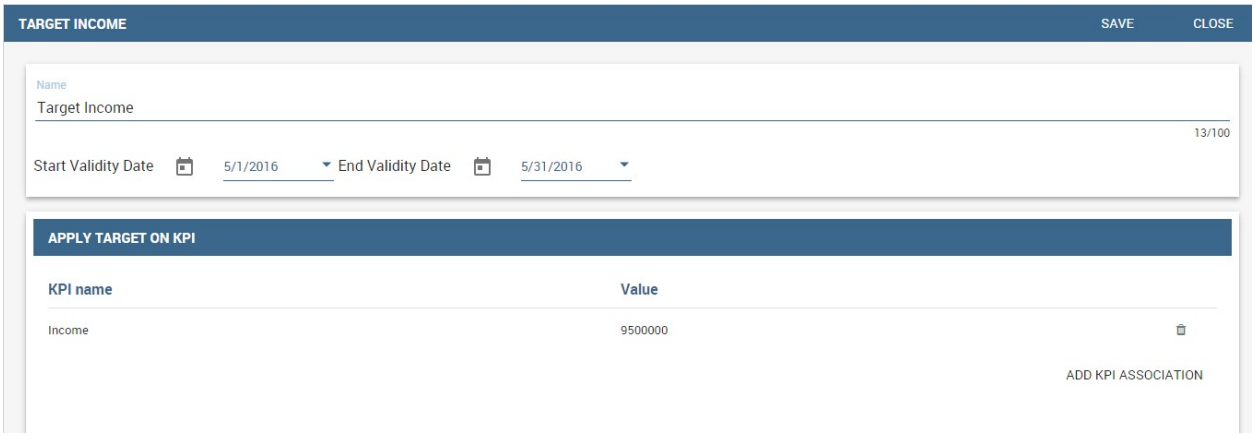


Fig. 5.476: KPI target association.

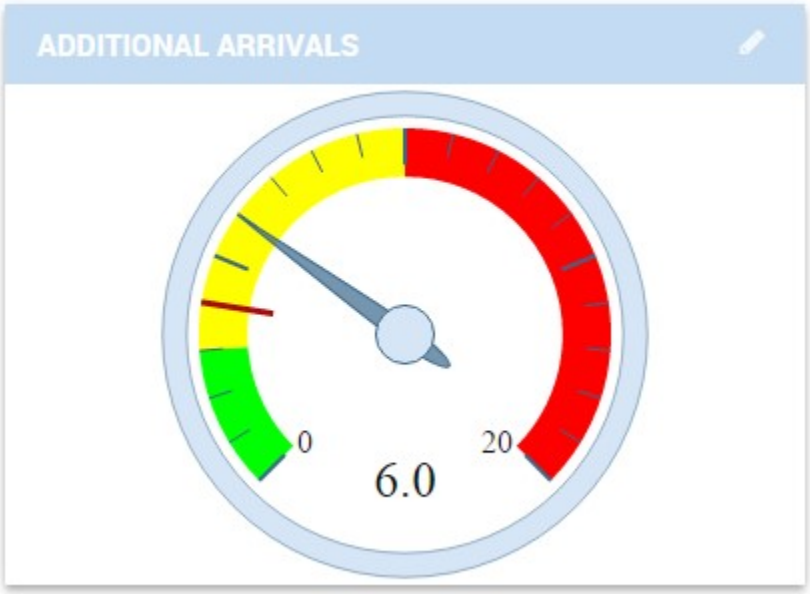


Fig. 5.477: Target mark in KPI scale of values.

Schedulation. Once the KPI has been defined, it is necessary to schedule them to proceed with the creation of an analytical document. For this purpose, click on the **KPI Scheduler** from the contextual menu that you can see below.

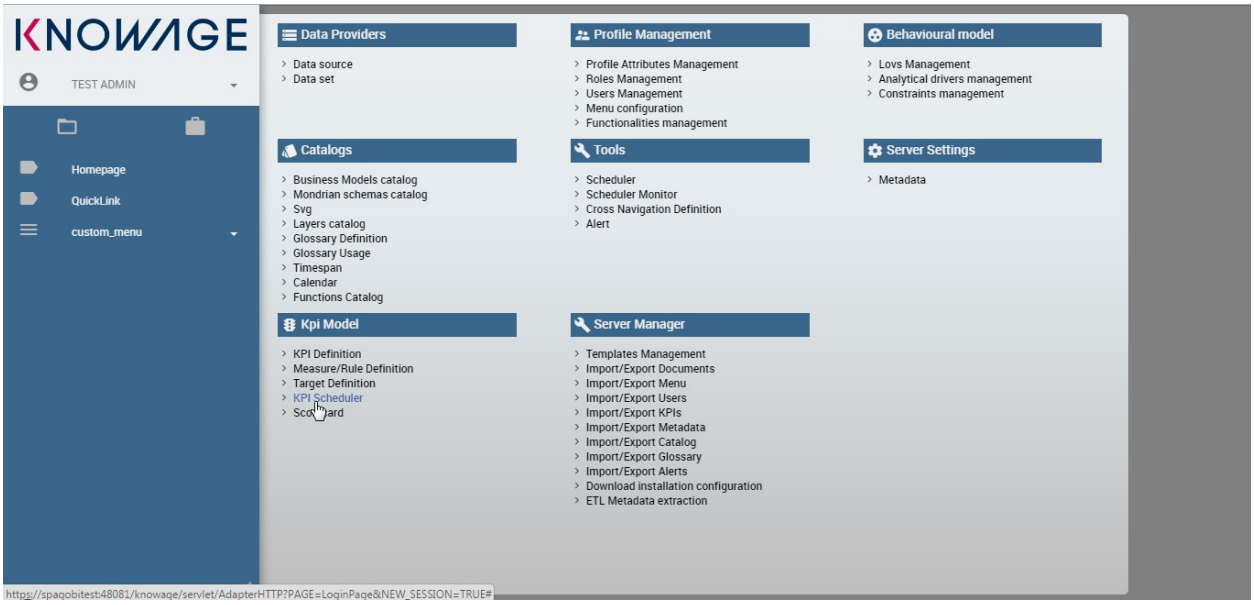


Fig. 5.478: KPI Scheduler menu item.

As for the other interfaces it is enough to click on the "Plus" icon to create a new schedulation. The new schedulation window presents several tabs.

- **KPI**: it is possible to associate one or more KPI to the schedulation clicking on "Add KPI Association".

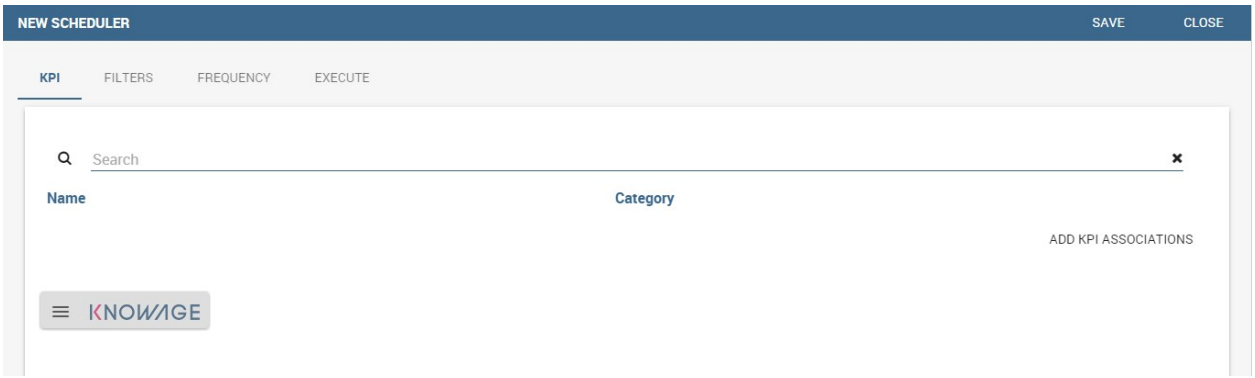


Fig. 5.479: KPI tab window.

- **Filters**: here you assign values to the filters (if configured) associated to the schedulation. Note that it is possibile to assign values to the filters by means of a LOV, a fixed list of values or a temporal function. In case the LOV option is chosen, remember that the LOV must return one unique value. This choice can be useful for profiling tasks.
- **Frequency**: here is the place where the schedulation 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

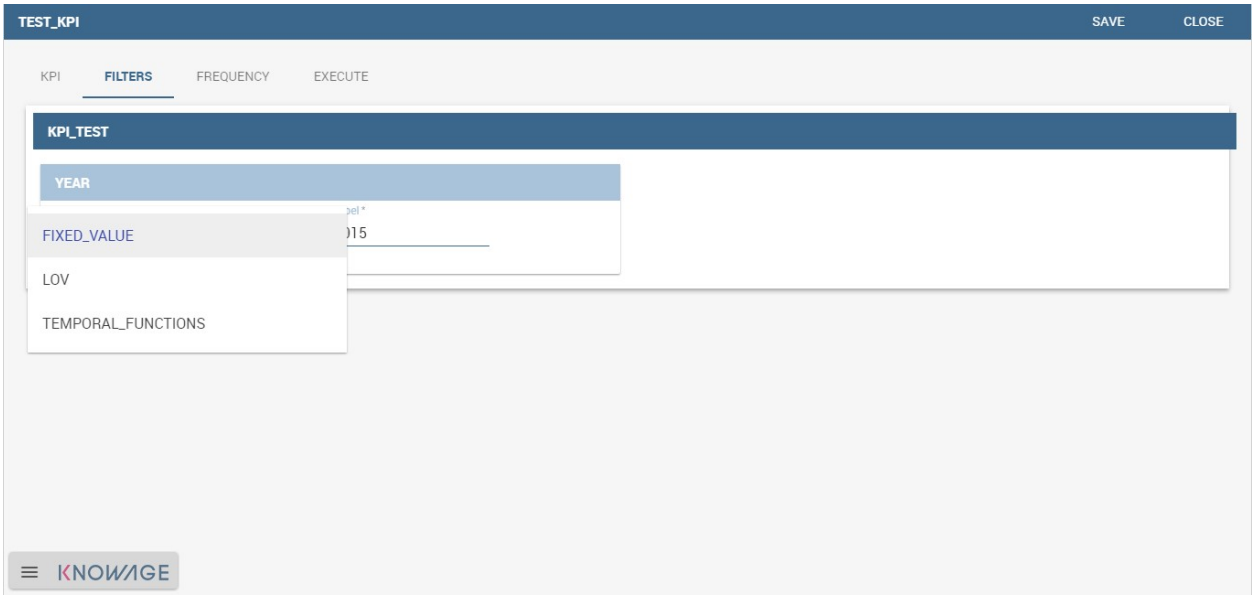


Fig. 5.480: Filters options.

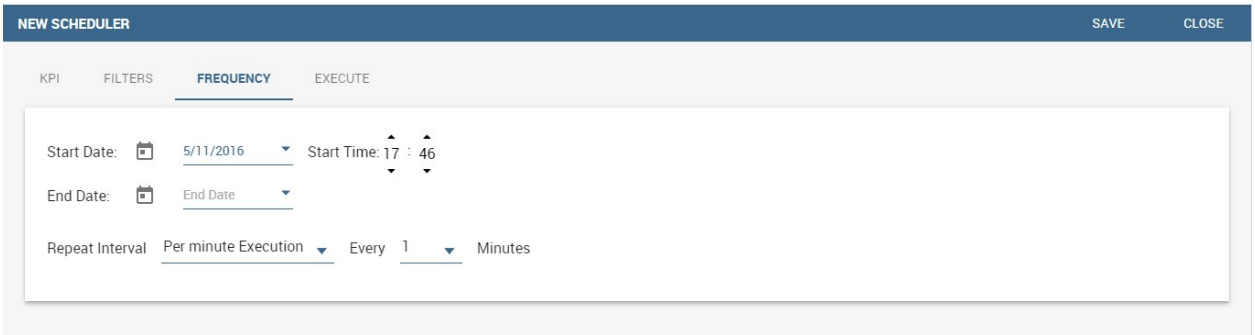


Fig. 5.481: Frequency tab window.

and all error log text files are available right beneath. While selecting **Delete and insert** the previous data are deleted.

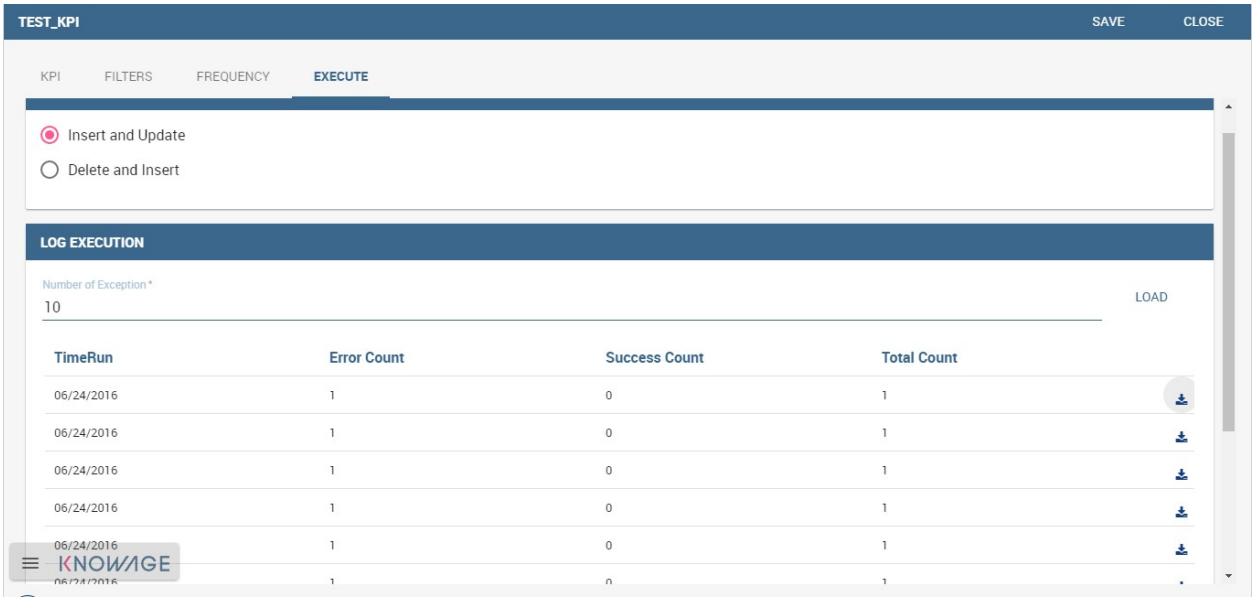


Fig. 5.482: Execute tab window.

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.

5.19.2 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

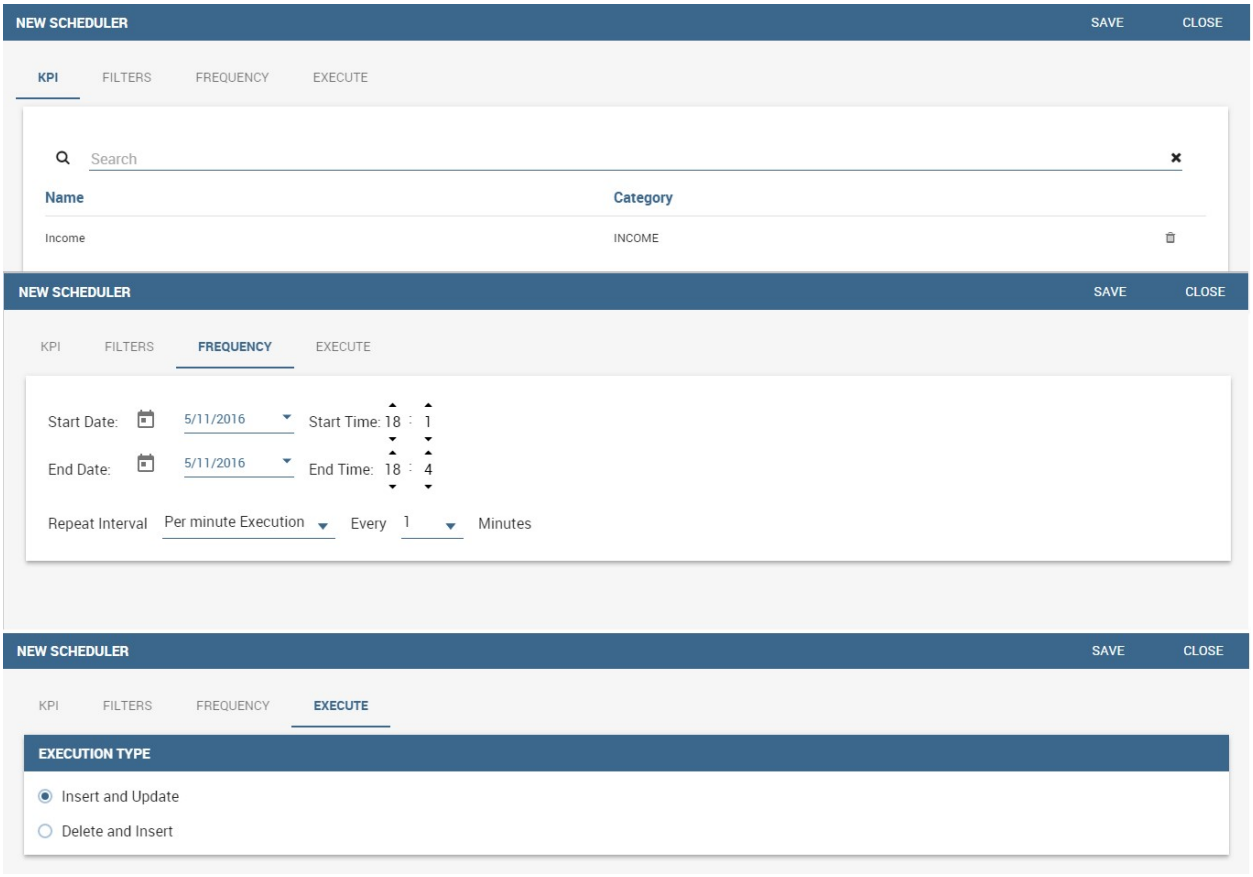


Fig. 5.483: Overview of the KPI case.

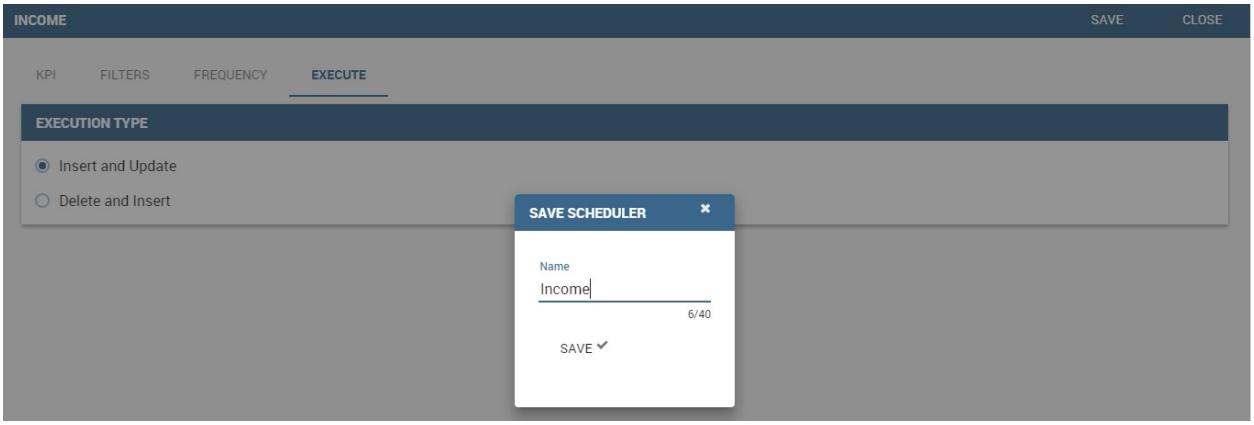


Fig. 5.484: Creation of a KPI Document.

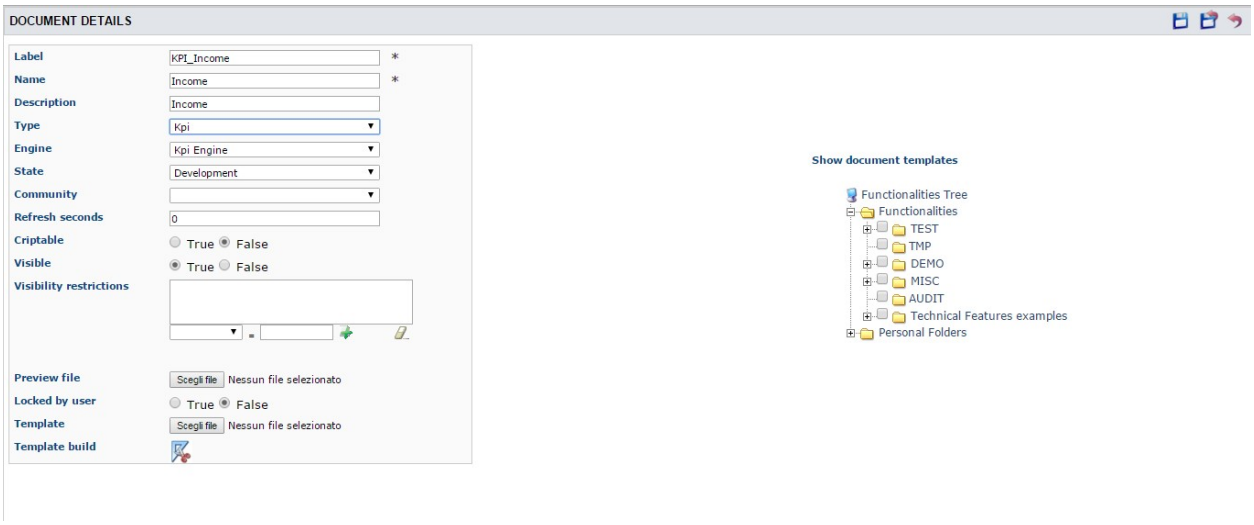


Fig. 5.485: Overview of the KPI case.

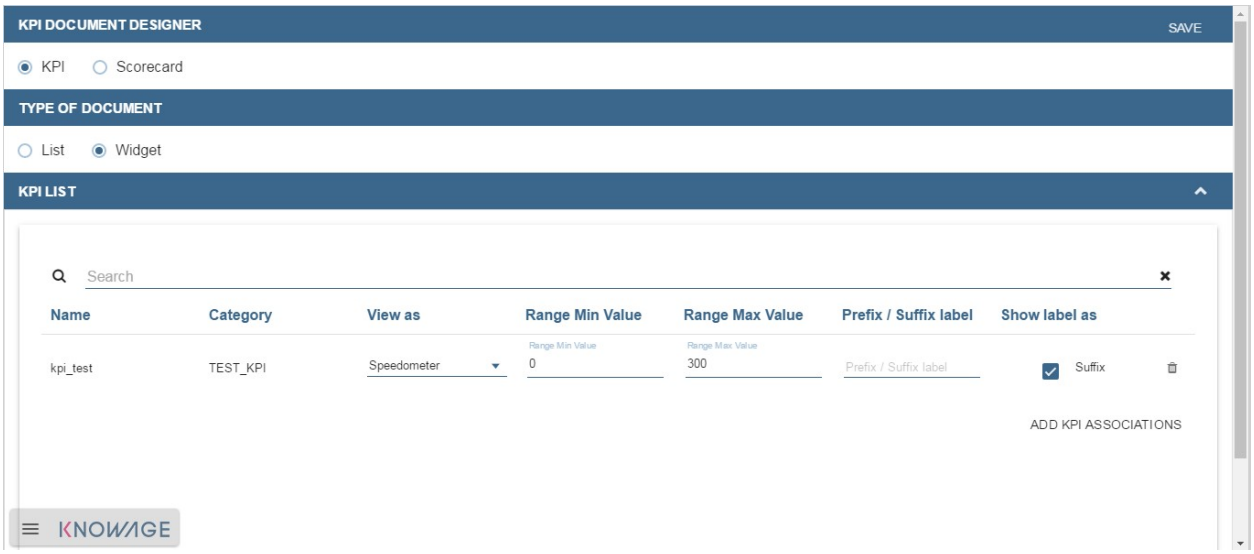


Fig. 5.486: Setting the KPI associations using the dedicated area.

Fig. 5.487: 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.

5.20 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.

5.20.1 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.

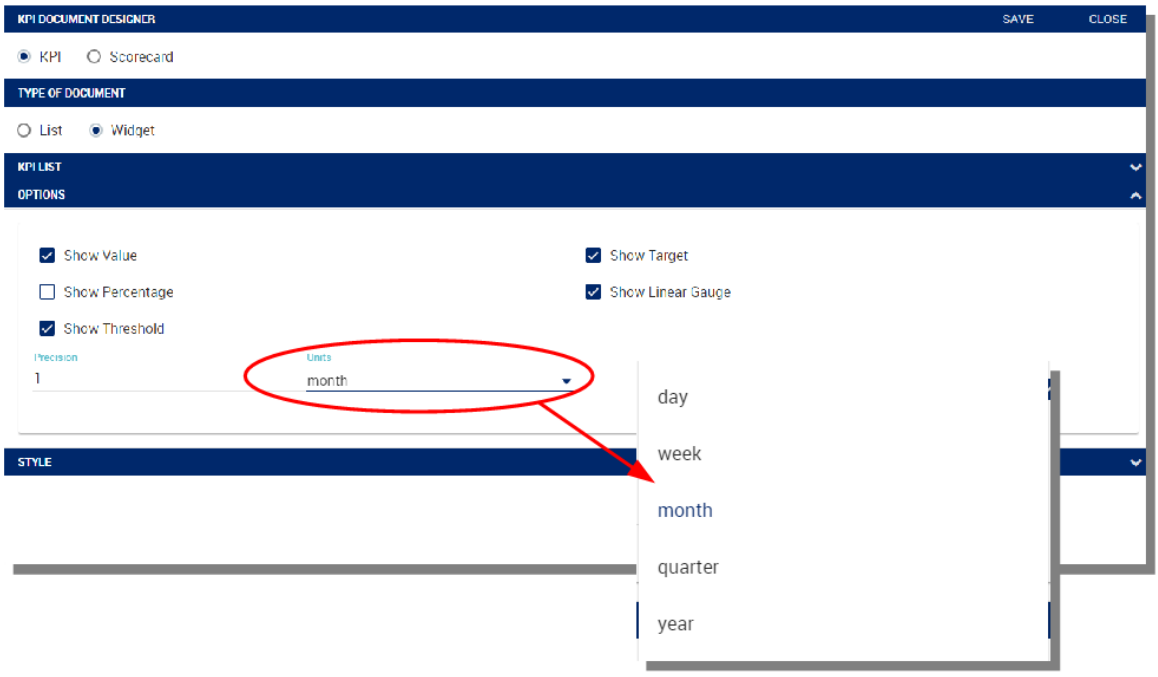


Fig. 5.488: Choose the time granularity.

Fig. 5.489: Style settings.

Fig. 5.490: Cardinality settings example.

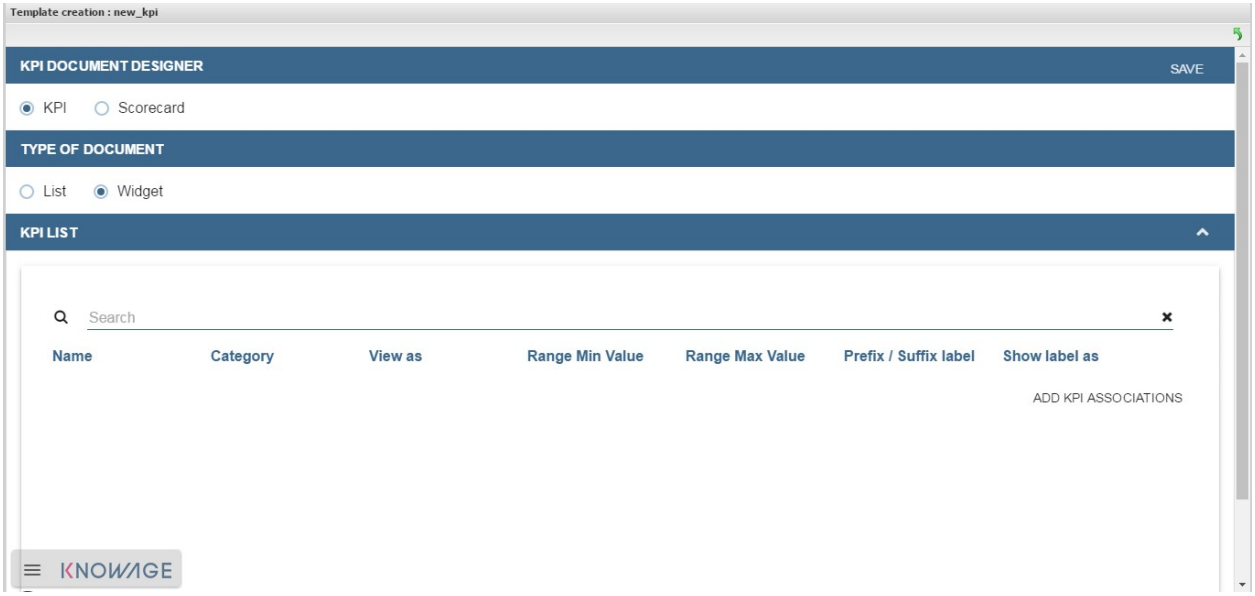


Fig. 5.491: KPI association.

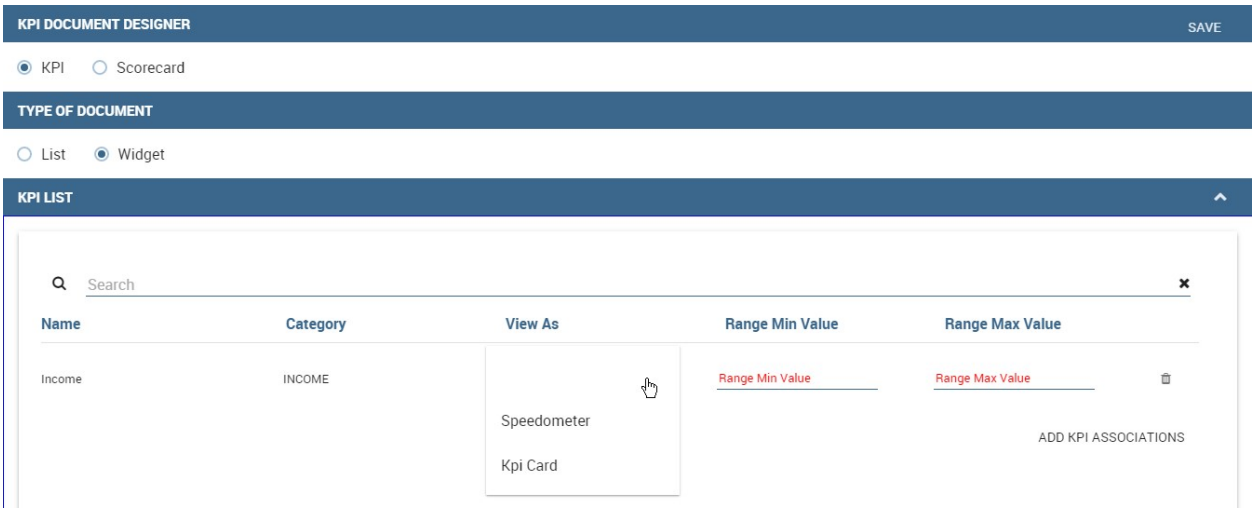


Fig. 5.492: Widget document.

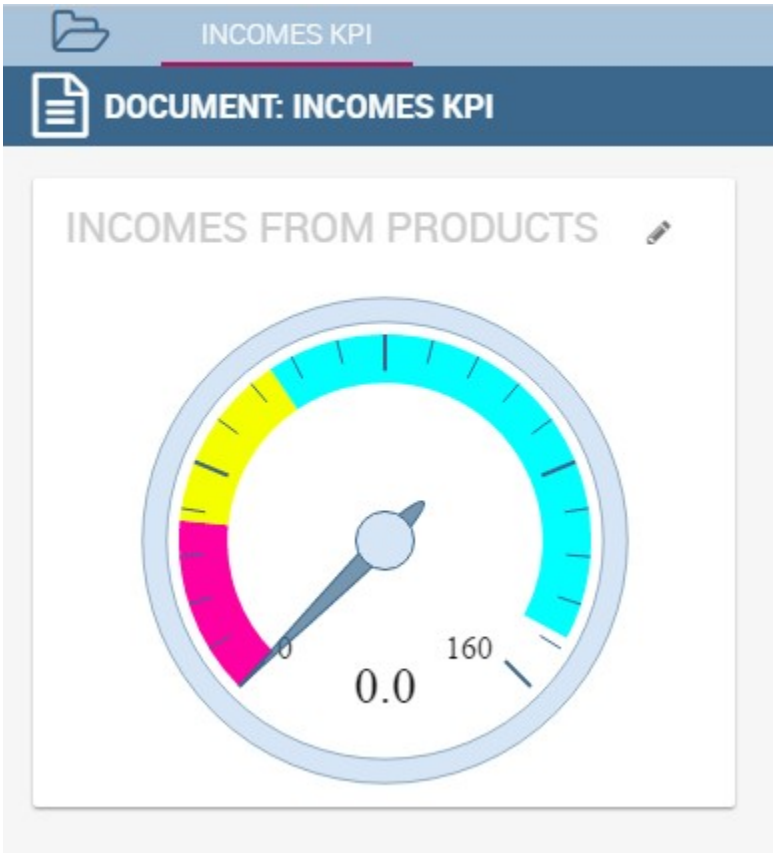


Fig. 5.493: KPI Speedometer.

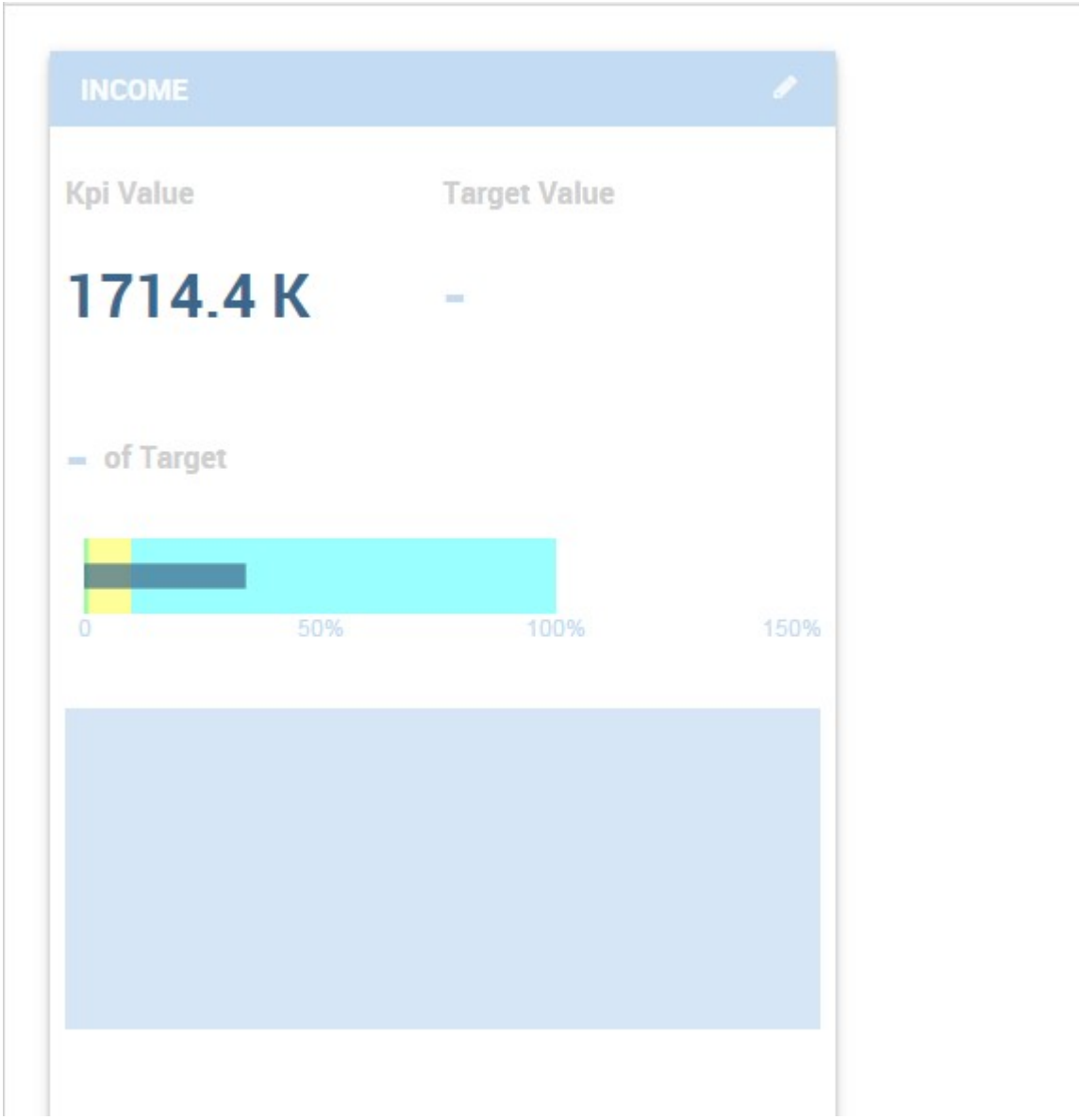


Fig. 5.494: KPI Card.

Fig. 5.495: KPI List.

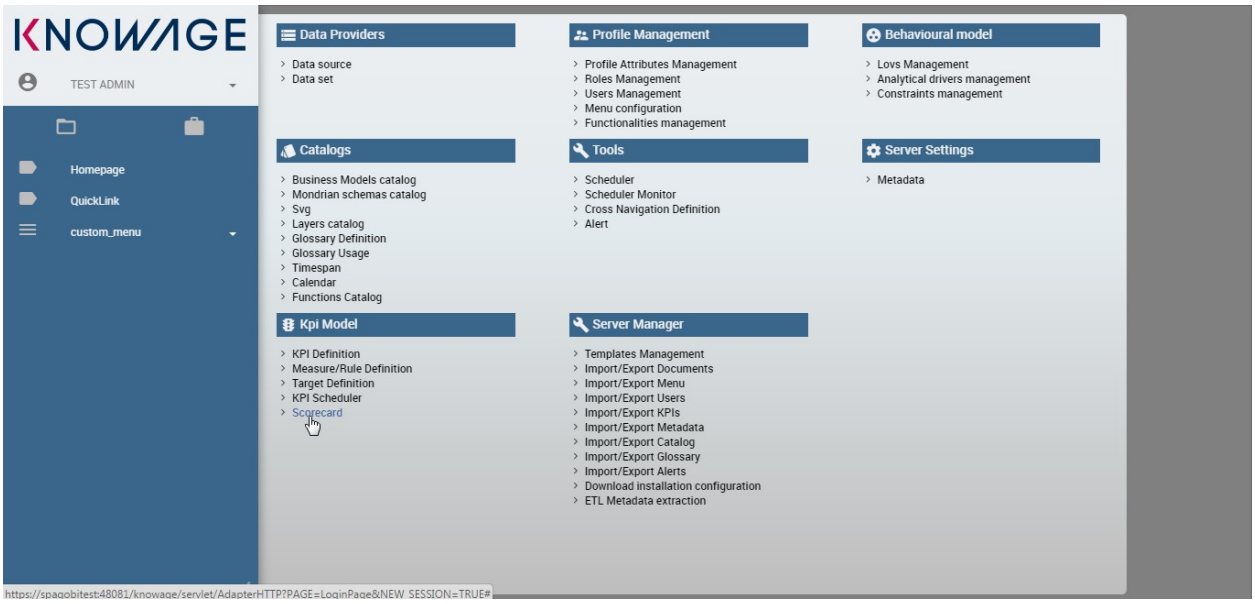


Fig. 5.496: Scorecard from the contextual menu.

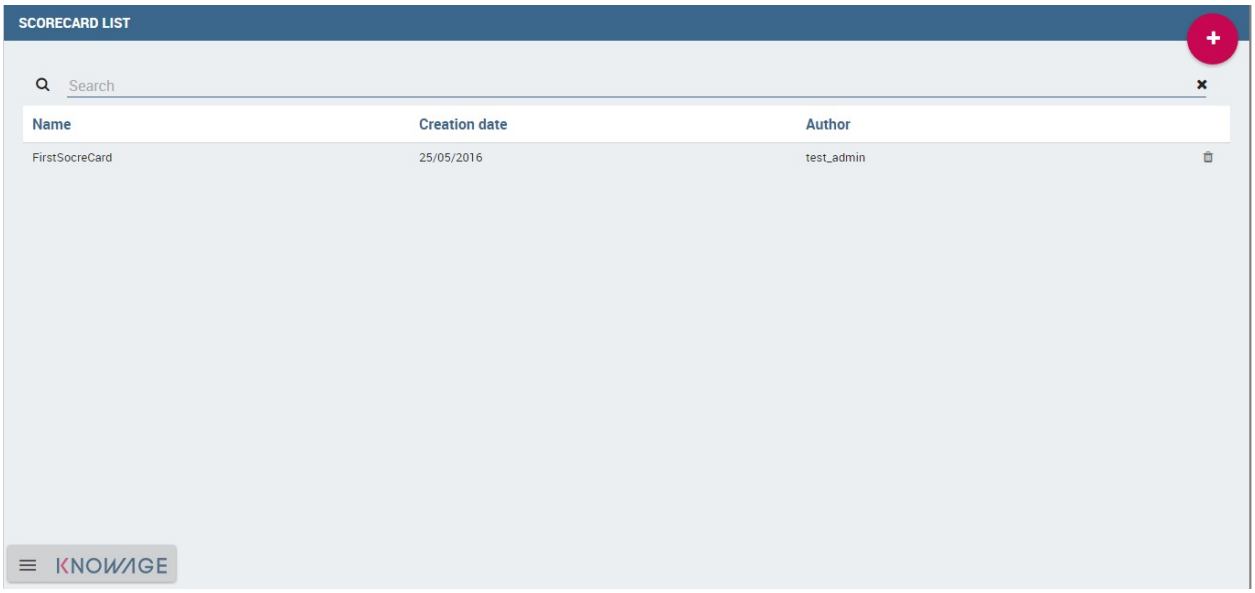


Fig. 5.497: Scorecard window.

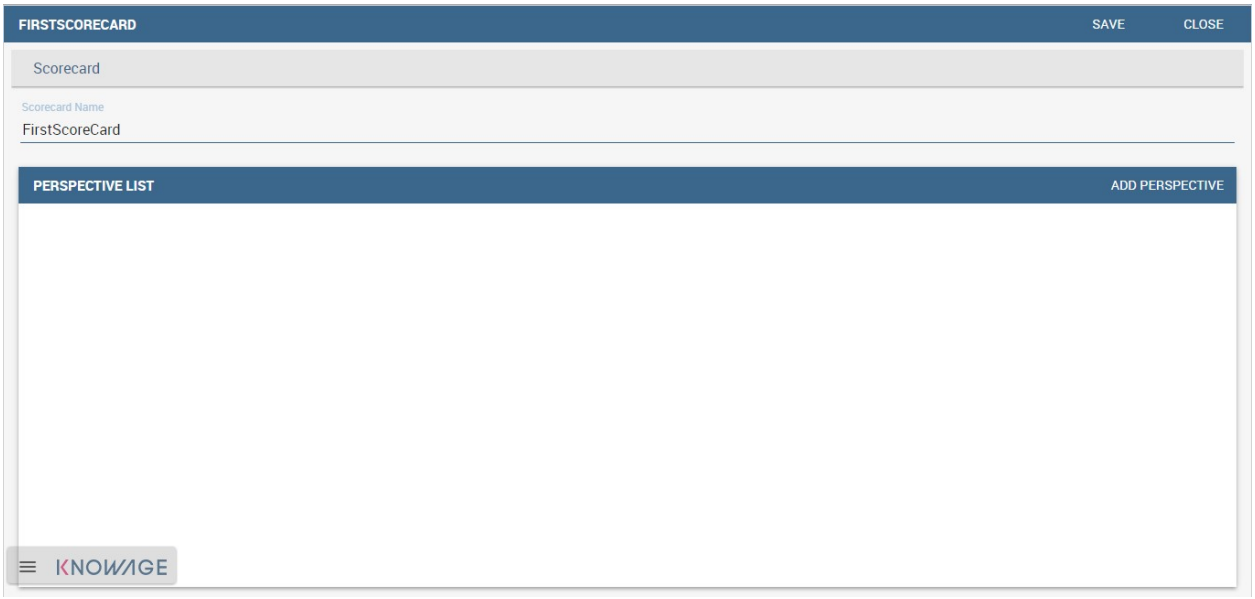


Fig. 5.498: Defining a new scorecard.

Fig. 5.499: Add perspective to the scorecard.

An example is given in the following figure.

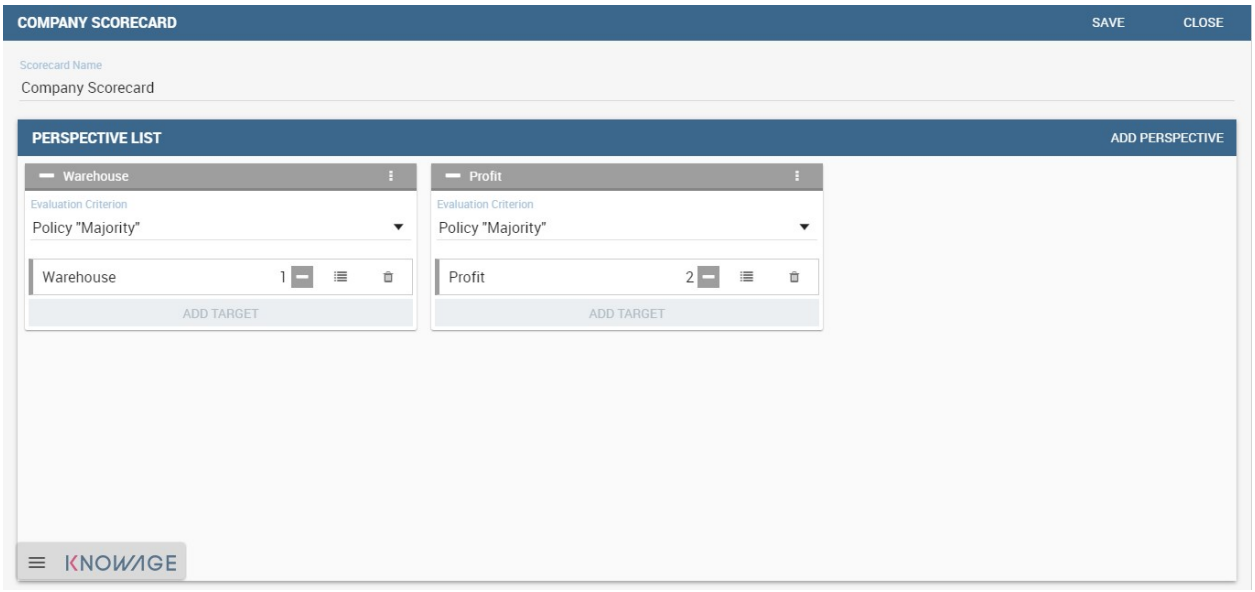


Fig. 5.500: Perspective list example.

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,

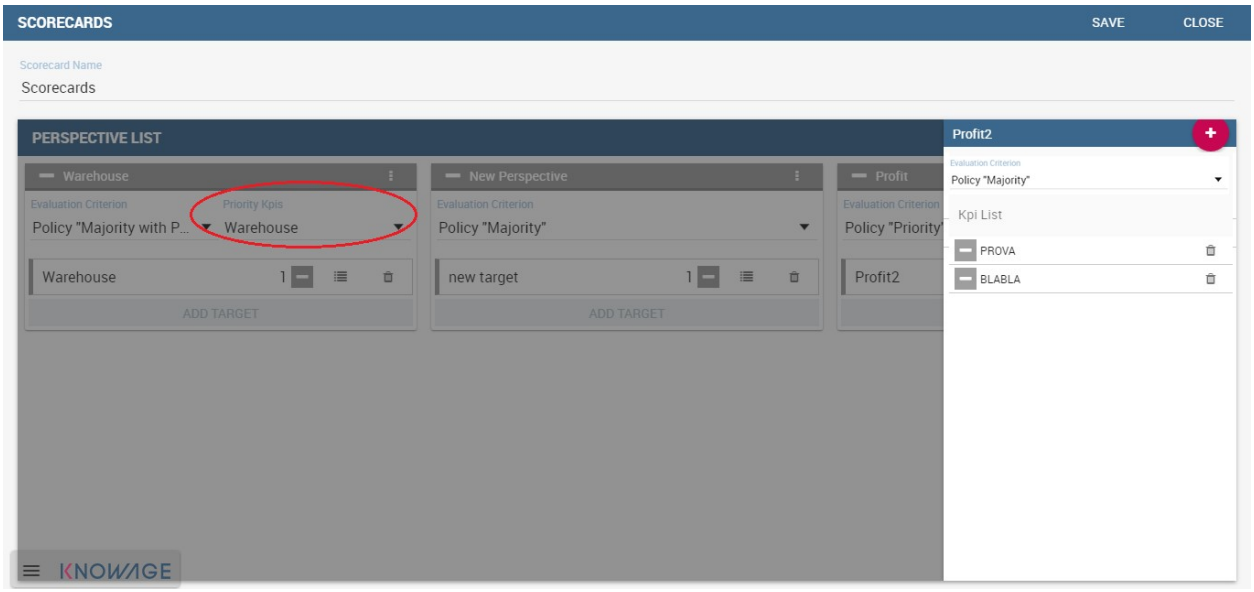


Fig. 5.501: Select the KPI with priority.

- **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.

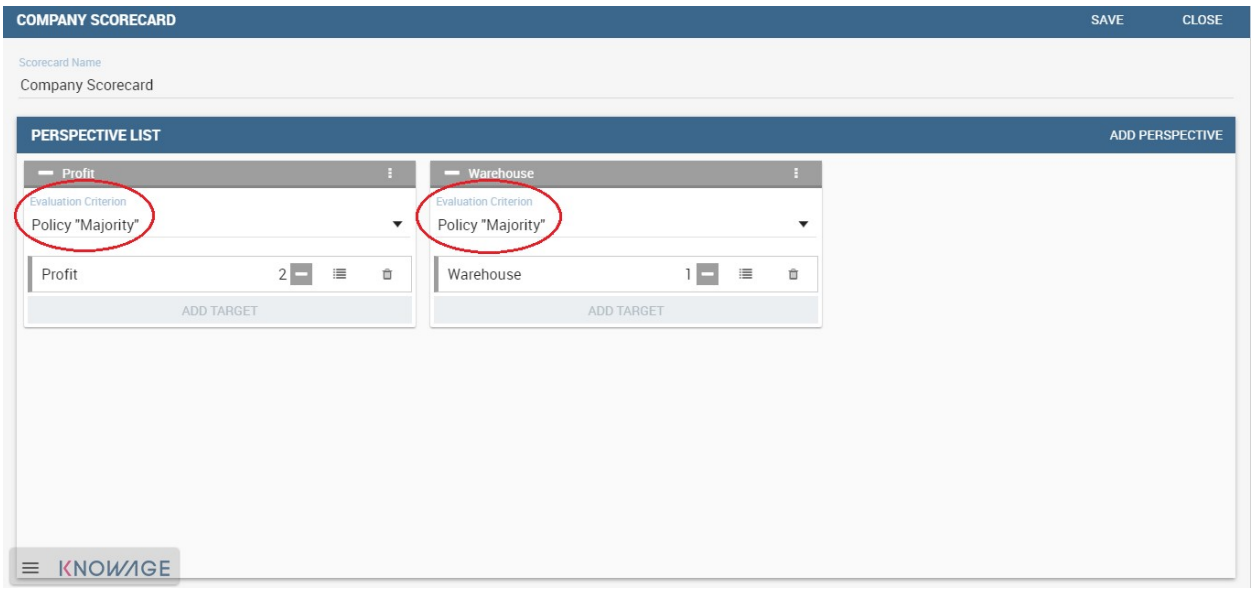


Fig. 5.502: Perspective policy.

Remember to save once perspectives and targets have been set.

5.20.2 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

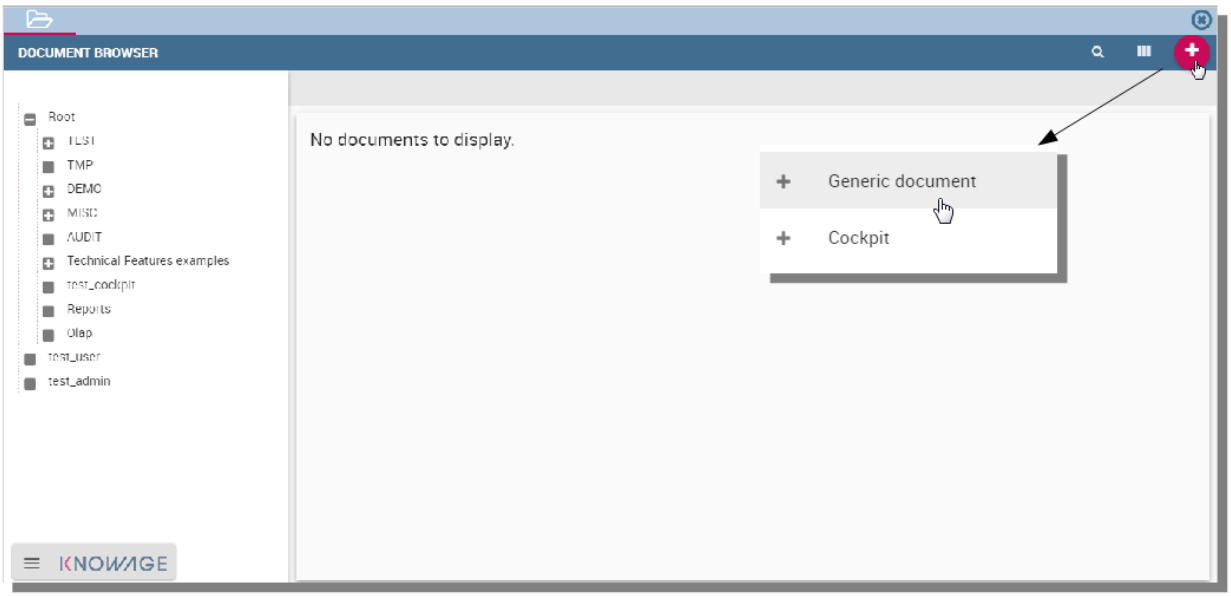


Fig. 5.503: Create a generic document from document browser.

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.

5.21 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.

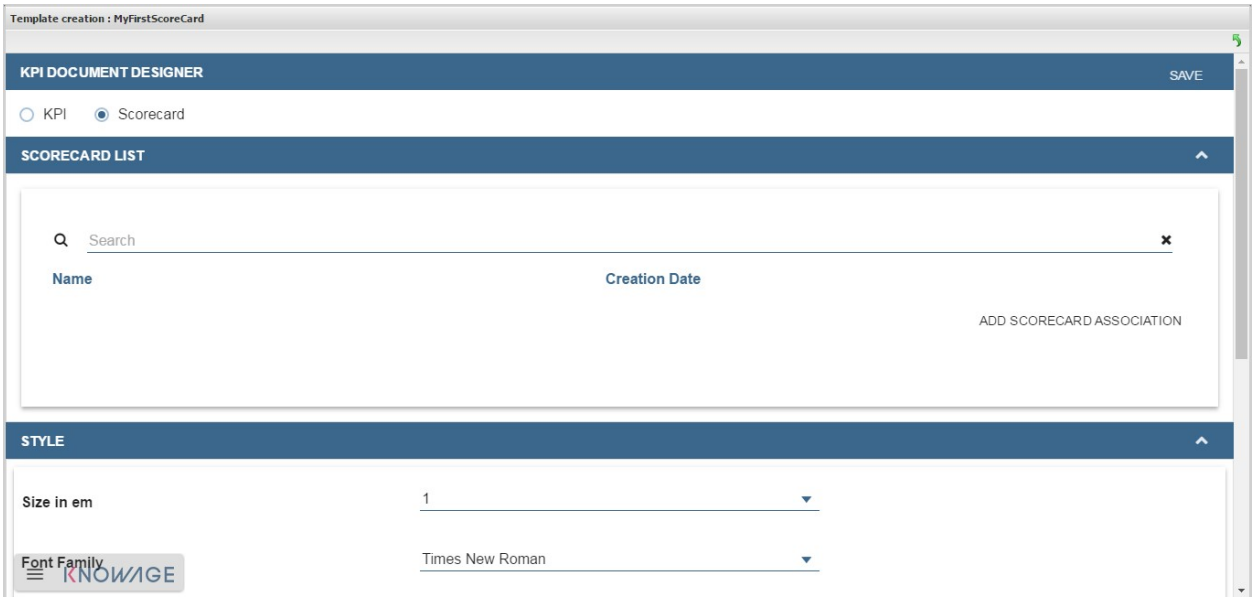


Fig. 5.504: Template creation window.

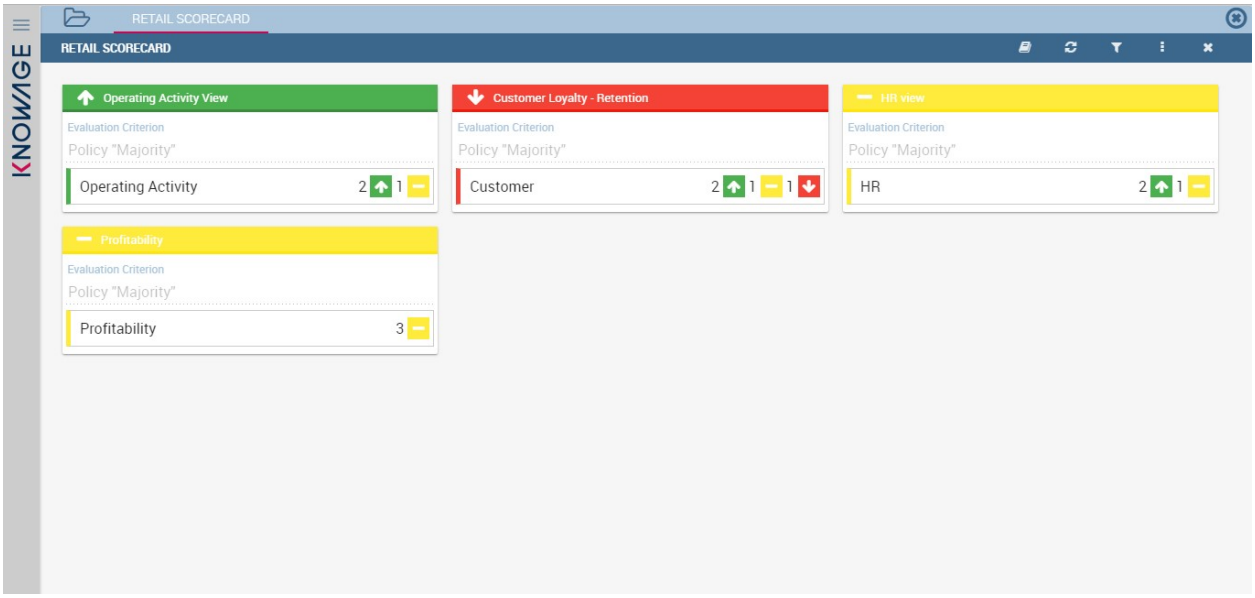


Fig. 5.505: Scorecard document interface.

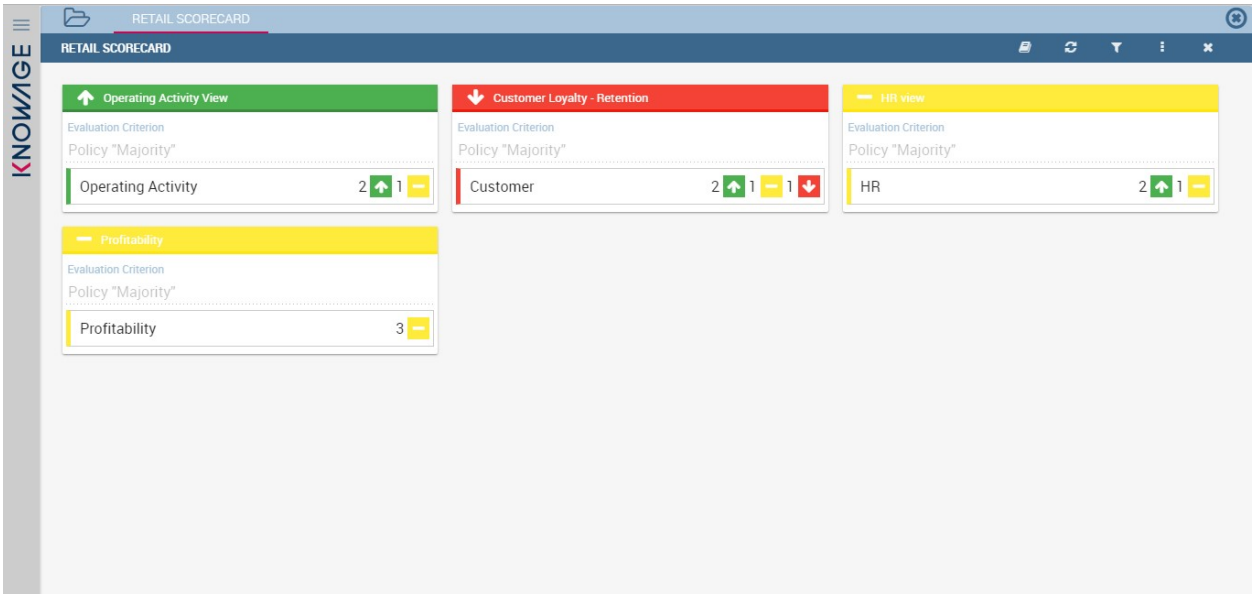


Fig. 5.506: Scorecard document interface.

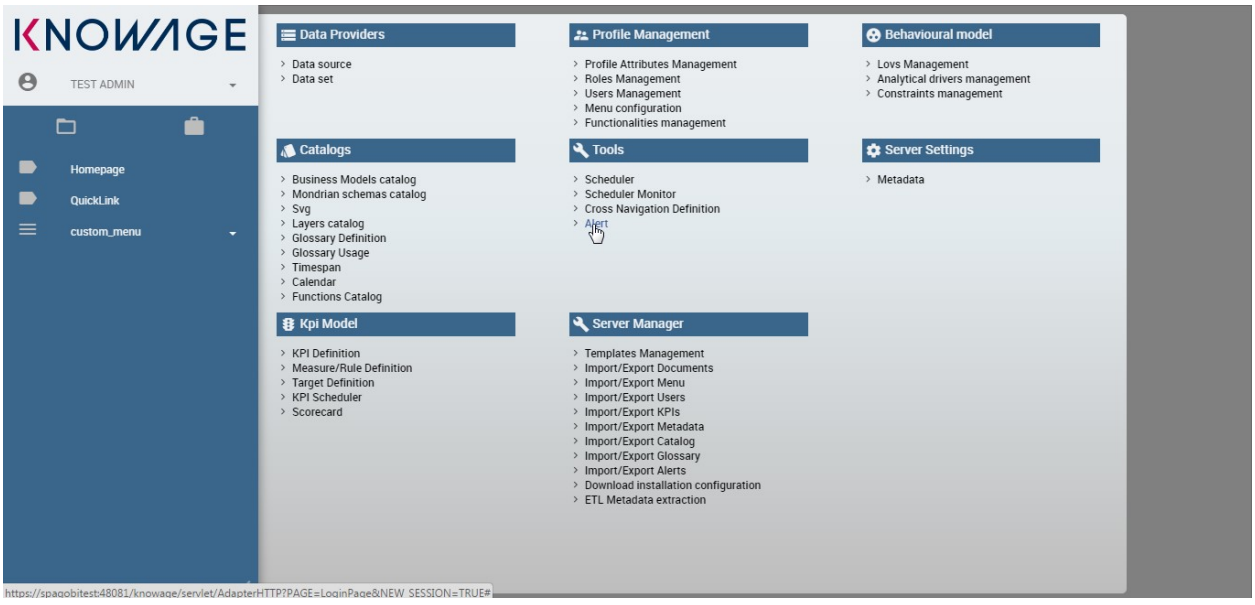


Fig. 5.507: Alert functionality from contextual menu.

5.21.1 Alert implementation

The Alert definition window (refer to the figure above) diplays 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.

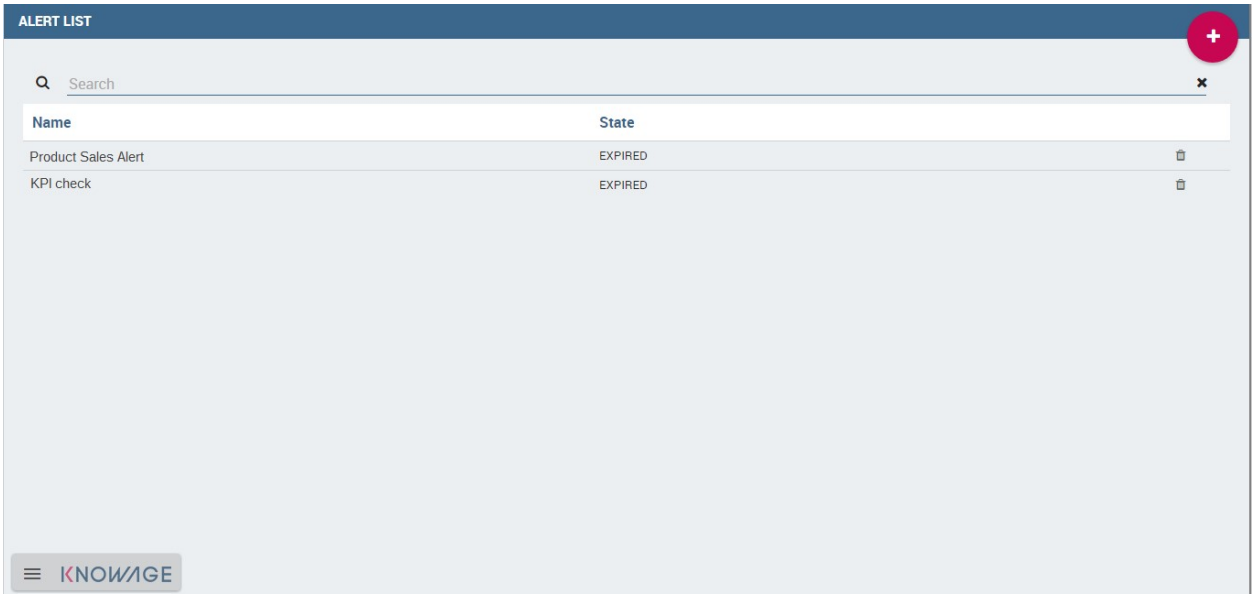


Fig. 5.508: Alert definition window.

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 occurence 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 overcomed thefunctionality 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 is 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 ovecomed. 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.

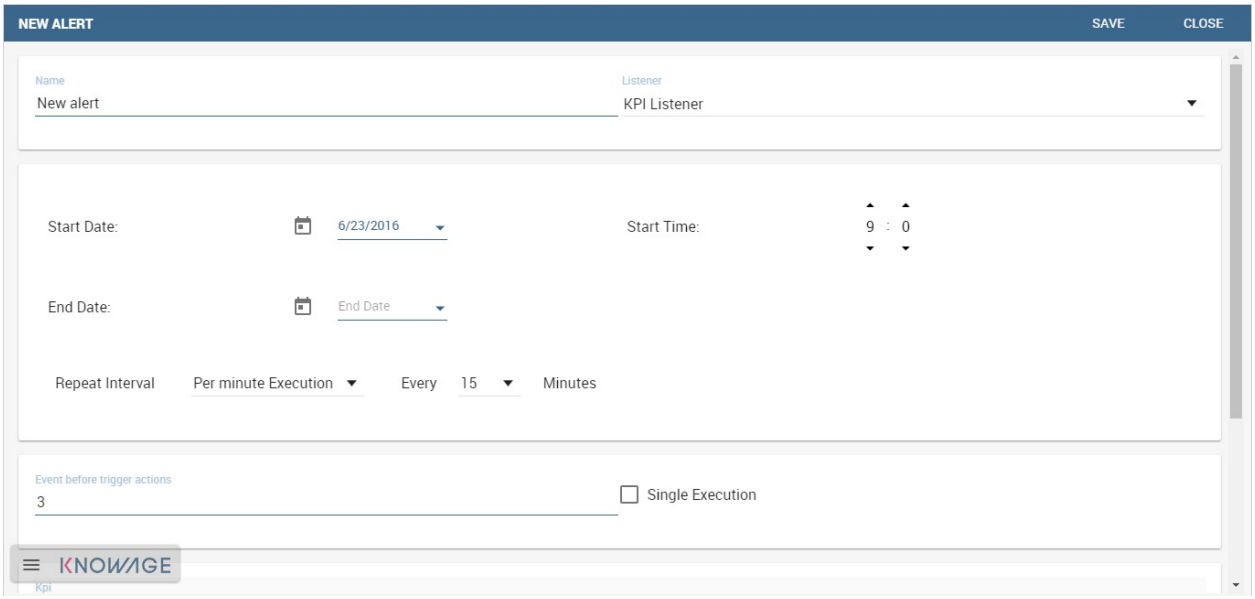


Fig. 5.509: Alert settings.

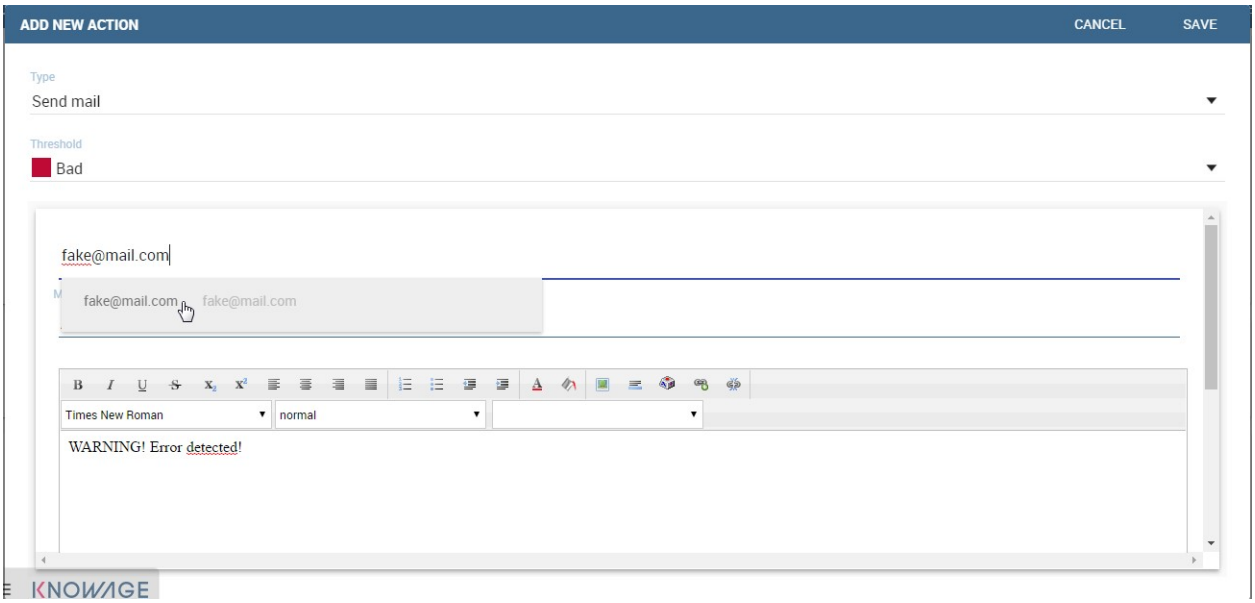


Fig. 5.510: Send email configuration.

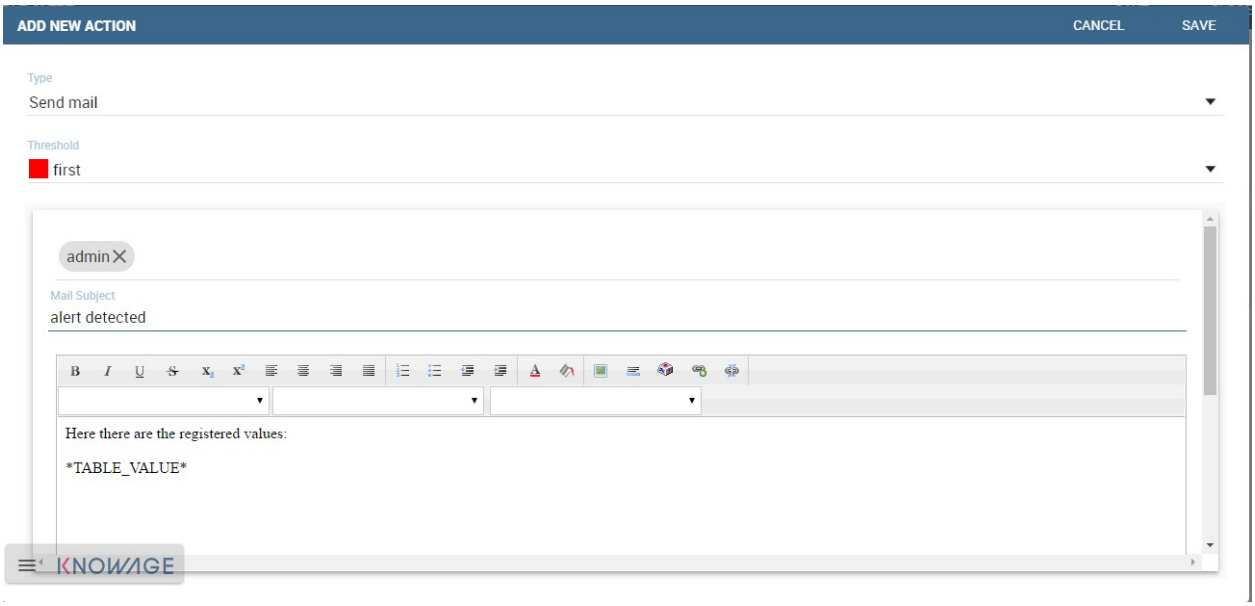


Fig. 5.511: Send email containing a table with detected values.

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.

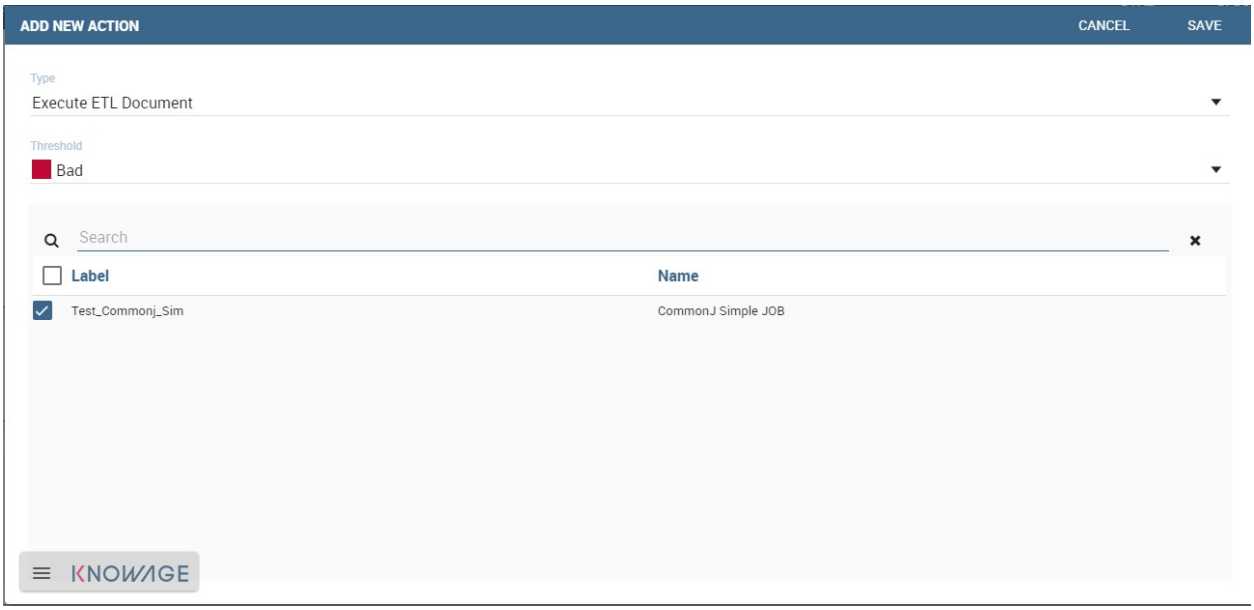


Fig. 5.512: Execute document configuration.