# AgensBrowser Guide

Bitnine Global Inc.

October 08, 2021

---

## Copyright Notice

**Information of technical documentation**

**Title : AgensBrowser Guide**

**Published date : 08 11, 2021**

**S/W version : AgensBrowser 2.1**

**Technical documentation version : v1.0**

---

# Introduction

## AgensBrowser is

AgensBrowser is a web browser providing a user interface for management of AgensGraph. The status of each component of AgensGraph DB can be monitored and controlled through AgensBrowser.

Its main features include:

- Support for RESTful-API
- Provision of meta information and automatic refresh
- Creating and deleting labels
- Input, execution, storage of Cypher and ANSI-SQL
- Project Load/Save/Delete

- Graph navigation tool, styling, summary functions
- Import/export of Graphson and image export
- Retrieval of the query history
- LiveShare For Report

## System Requirements

The system environment required to run AgensBrowser is as follows:

- Server
    - OS : Windows 8 or higher, OS X 10 or higher, All kinds of Linux/Unix OSes

- Memory : 8GB or more
- Java version : 1.8

- Client
  - Browsers : Javascript ES6 Browsers (Chrome, Firefox, Edge, Safari etc.)
    - * • Internet Explorer not supported

# Installation

## Environment Setting

- Install Java 1.8

- Install AgensGraph (AgensGraph v1.3 or later)
  - Install DB for AgensBrowser management
  - Create an AgensBrowser Admin account

- Prepare binary

```
agens-browser-web-2.1
```

## Installation and Startup

1. Unzip and check the installation files.
   - agens-browser-web-2.1.jar
   - agens-config.yml
2. Modify .yml file.
   Open agens-config.yml and modify the connection information in consideration of the actual user environment. You are then connected to the graph via graph_path specified in the yml file; the login account should be the owner of graph_path. graph_path specified at service startup cannot be changed.

- server
  - port : AgensBrowser web server port

- logging
  - config : Sets the logs of AgensBrowser. This generates logs by date in the logs directory under the current folder

- agens
  - api :

  * query_timeout: Sets query timeout (Default: 600000, 10Min)

- jwt :
  * expiration : Sets client connection expiration time (Default: 43200, 12H)

- inner :
  * url : Used to store the query logs and projects in a file “agens-db.db” under the current folder
    (Regenerated with initial data if deleted)

- outer :
  * url
    - IP: IP of the database server
    - Port: Port of the database server
    - Database: Name of the database to connect to
  * graph_path: Graph name created in AgensGraph

  * username: User name of the database to connect to

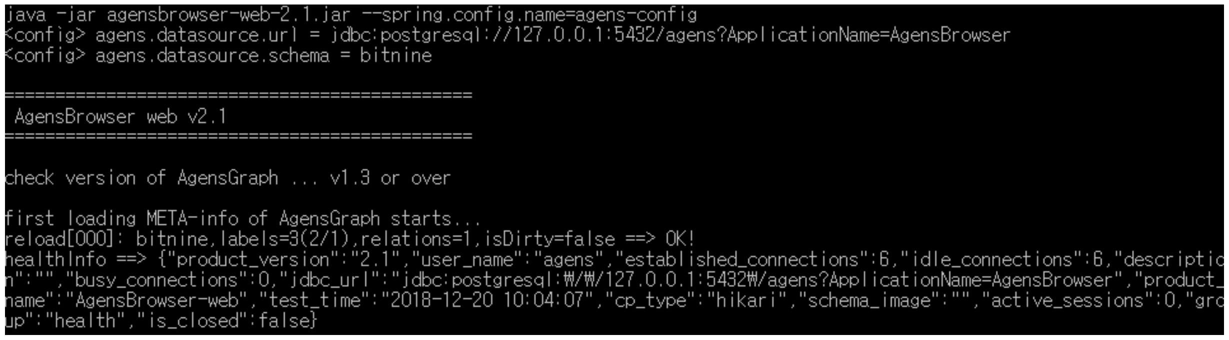
  * password: User password of the database to connect to
  * max_rows : max_rows: The maximum number of rows to return (1000 or less is recommended)
- client :
  * guest-key: Public key needed when using the saved project for reporting.
    `ex) http://192.168.0.56:8084/#/report/agens/1130`
  * animation-enabled: Whether to enable animation when the client's graph layout is used.
  * title-shown : title-shown: Whether the nodevertex title should be displayed when the client outputs the graph
    (It is recommended to set it False by default; if set to True, the rendering performance will degrade)

```
## config context path to "/" by setting an empty string
server:
  port: 8085    -- AgensBrowser Web server port

logging:
  config: classpath:logback-agens.xml

spring:
  pid:
    file: agensbrowser.pid
  main:
    banner-mode: "off"
  resources:
    cache:
      period: 3600
    static-locations: classpath:/META-INF/resources/,classpath:/resources/, \
                      classpath:/static/,classpath:/public/
  servlet:
    multipart:                    ## MULTIPART (MultipartProperties)
      enabled: true               # Enable multipart uploads
      file-size-threshold: 2KB  # Threshold after which files are written to disk.
      max-file-size: 200MB      # Max file size.
      max-request-size: 215MB   # Max Request Size

agens:
  api:
    base-path: /api
    query-timeout: 600000       # 1000 ms = 1 sec     -- Timeout 10 min
  jwt:
    header: Authorization       # not used
    secret: agensBrowserKey
    expiration: 43200           # unit: sec (12 Hour = 43200 sec)
  inner:
    datasource:
      driverClassName: org.h2.Driver
      url: jdbc:h2:file:./agens-h2;DB_CLOSE_DELAY=-1;MODE=MySQL
      username: sa
      password:
  outer:
    datasource:
      driverClassName: org.postgresql.Driver
      url: jdbc:postgresql://127.0.0.1:5432/agens?ApplicationName=AgensBrowser
      graph_path: bitnine
      username: agens
      password: agens
      max-rows: 1000    -- The maximum no. of rows to return
  file:
    download-dir: ./downloads   # for image file, etc..
    upload-dir: ./uploads       # for Graphson, Graphml, etc.. (not yet support EXCEL)
  client:
    mode: prod                  # mode : dev or prod
    guest-key: agens            # url middle-value for report output
                                -- Public key needed when using
                                the saved project for reporting
    animation-enabled: true     # to use animation at applying layout
                                -- Whether animation is applied
                                when client's graph layout is used.
    title-shown: false          # to show node title by default
                                -- Whether the title of the nodevertex
                                is printed when the client's graph is output.
                                -- (Rendering performance degrades
                                if set to True, False is recommended by default)

  product:
    name: AgensBrowser-web
    version: 2.1
    hello-msg: AgensBrowser web v2.1    -- hello message
```

3. Create an executable file.
   Enter the following in the text file and save it with an extension .sh (unix) or .bat (windows).

(e.g. agens-browser.sh / agens-browser.bat)

```sql
```

```
java -jar agens-browser-web-2.1.jar --spring.config.name=agens-config
```
```

4. Run the service.
   `{}     $ sh agens-browser.sh  -- linux     $ agens-browser.bat    -- windows`
5. Verify the service execution.



```
java -jar agensbrowser-web-2.1.jar --spring.config.name=agens-config
<config> agens.datasource.url = jdbc:postgresql://127.0.0.1:5432/agens?ApplicationName=AgensBrowser
<config> agens.datasource.schema = bitnine

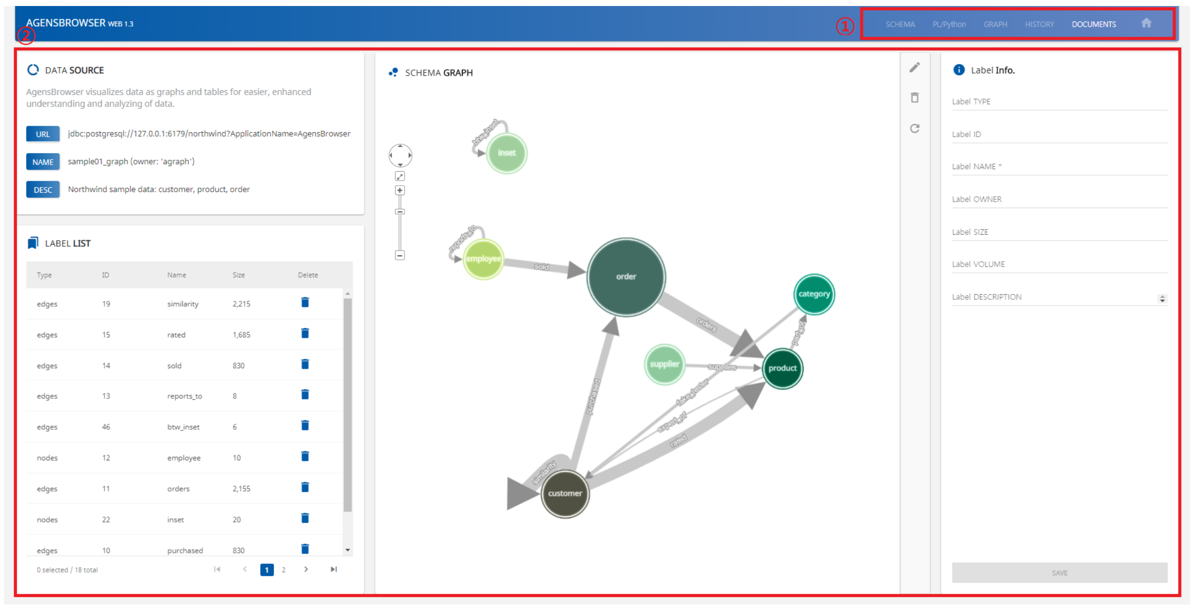
=============================================
 AgensBrowser web v2.1
=============================================

check version of AgensGraph ... v1.3 or over

first loading META-info of AgensGraph starts...
reload[000]: bitnine,labels=3(2/1),relations=1,isDirty=false ==> OK!
healthInfo ==> {"product_version":"2.1","user_name":"agens","established_connections":6,"idle_connections":6,"descriptio
n":"","busy_connections":0,"jdbc_url":"jdbc:postgresql:₩/₩/127.0.0.1:5432₩/agens?ApplicationName=AgensBrowser","product_
name":"AgensBrowser-web","test_time":"2018-12-20 10:04:07","cp_type":"hikari","schema_image":"","active_sessions":0,"gro
up":"health","is_closed":false}
```

## Login

Automatic login is made as a token-based authentication using AgensGraph account (ID/PW) specified in config file. The database access account should be the admin account.

To connect to AgensBrowser, enter URL in the address bar of your web browser with the following format.

```
http://DB_SERVER_IP_ADDRESS:WEB_SERVER_PORT/index.html
```

The following is an example of using a local database and a web server port 8085:

```
http://localhost:8085/index.html
```

## Logout

If you exit the browser window, you will be logged out. You can set the client connection expiration time by setting expiration in config file.
```

## Screen Layout

If the login is successful, it redirects you to the main screen as shown below. The main screen shows the information about database connection and meta graph(schema graph) and contains available menu (functions) provided by AgensBrowser.



DATA SOURCE

AgensBrowser visualizes data as graphs and tables for easier, enhanced understanding and analyzing of data.

URL jdbc:postgresql://127.0.0.1:6179/northwind?ApplicationName=AgensBrowser

NAME sample01_graph (owner: 'agraph')

DESC Northwind sample data: customer, product, order

LABEL LIST

| Type | ID | Name | Size | Delete |
|---|---|---|---|---|
| edges | 19 | similarity | 2,215 | 🗑 |
| edges | 15 | rated | 1,685 | 🗑 |
| edges | 14 | sold | 830 | 🗑 |
| edges | 13 | reports_to | 8 | 🗑 |
| edges | 46 | btw_inset | 6 | 🗑 |
| nodes | 12 | employee | 10 | 🗑 |
| edges | 11 | orders | 2,155 | 🗑 |
| nodes | 22 | inset | 20 | 🗑 |
| edges | 10 | purchased | 830 | 🗑 |

0 selected / 18 total

The main screen is divided into two areas:

- Menu Area (①)
  Provides the main menu for each group of the functions used in AgensBrowser. You may select a function provided by AgensBrowser from the menu area and move to a specific submenu and/or the corresponding screen.
  Each menu icon is described in the table below:

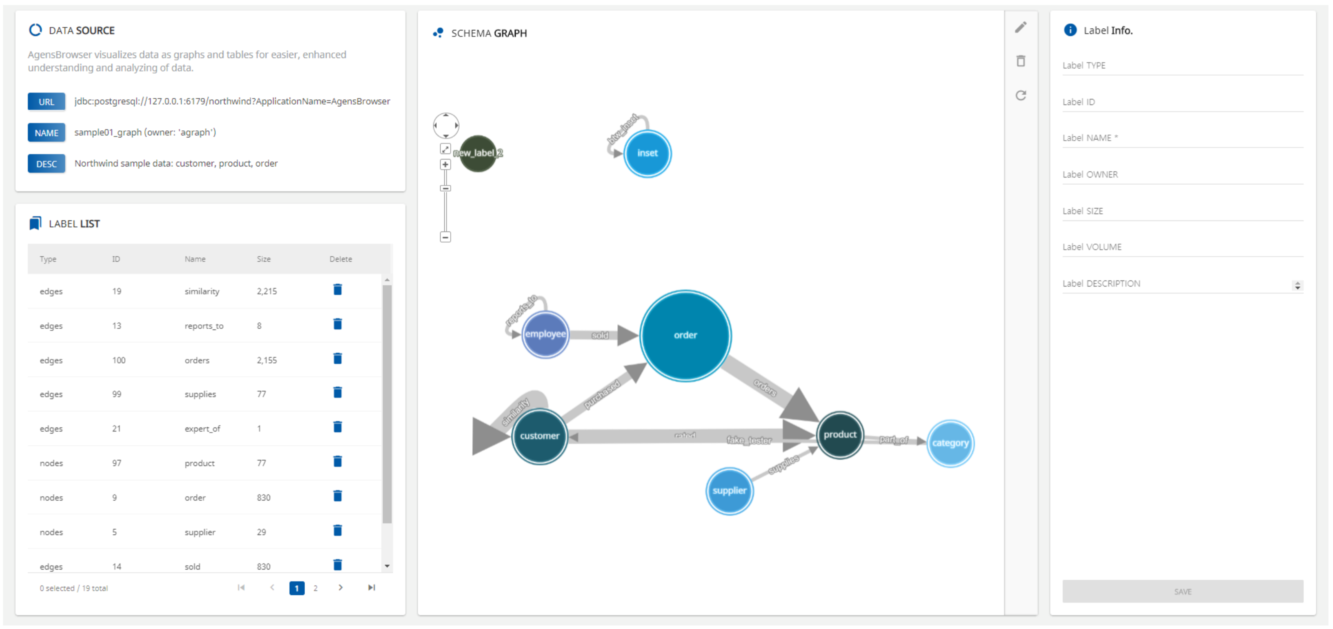
| Icon | Description |
|---|---|
| SCHEMA | To check the connection and metagraph information. |
| PL/Python | To create PL/Python functions and check the list. |
| GRAPH | To run Query Editor, Query Output, Graph Visualization, Project Save etc. |
| HISTORY | To check the query logs. |
| DOCUMENTS | To refer to the AgensBrowser manual. |

- Main Area (②)
  The work space that provides control/monitoring information of the tasks performed through the menu.

# Functions Detail

## Schema Menu

Displays the user's access information and graph meta data of AgensGraph.



DATA SOURCE

AgensBrowser visualizes data as graphs and tables for easier, enhanced understanding and analyzing of data.

| | |
|---|---|
| URL | jdbc:postgresql://127.0.0.1:6179/northwind?ApplicationName=AgensBrowser |
| NAME | sample01_graph (owner: 'agraph') |
| DESC | Northwind sample data: customer, product, order |

LABEL LIST

| Type | ID | Name | Size | Delete |
|---|---|---|---|---|
| edges | 19 | similarity | 2,215 | |
| edges | 13 | reports_to | 8 | |
| edges | 100 | orders | 2,155 | |
| edges | 99 | supplies | 77 | |
| edges | 21 | expert_of | 1 | |
| nodes | 97 | product | 77 | |
| nodes | 9 | order | 830 | |
| nodes | 5 | supplier | 29 | |
| edges | 14 | sold | 830 | |

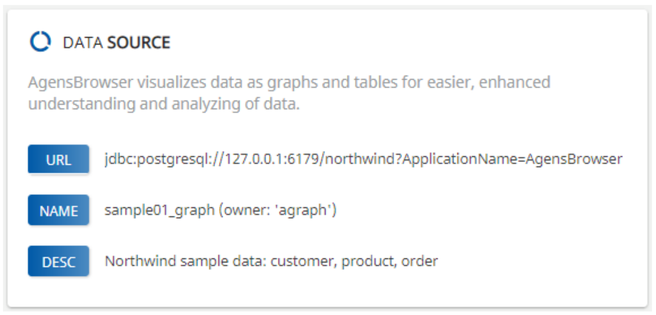
0 selected / 19 total

Schema consists of:

- Data Source
- Label List
- Schema Graph & Label Info

### Data Source

Data Source shows the connection information.



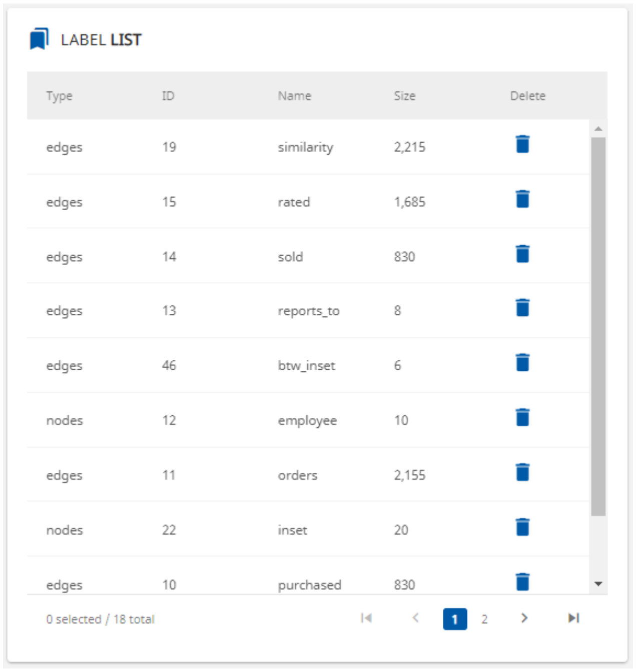
DATA SOURCE

AgensBrowser visualizes data as graphs and tables for easier, enhanced understanding and analyzing of data.

| | |
|---|---|
| URL | jdbc:postgresql://127.0.0.1:6179/northwind?ApplicationName=AgensBrowser |
| NAME | sample01_graph (owner: 'agraph') |
| DESC | Northwind sample data: customer, product, order |

### Label List

The details of the node/edge labels can be checked here. You may delete a node(or edge) label by selecting Delete(🗑).

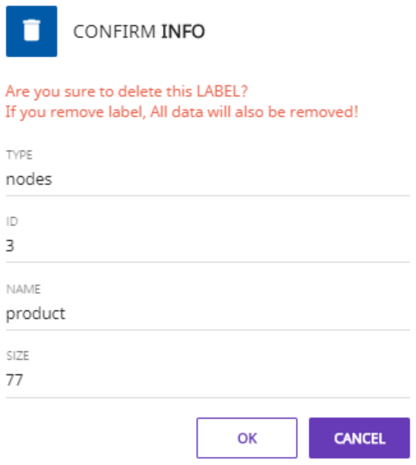


LABEL LIST

| Type | ID | Name | Size | Delete |
|---|---|---|---|---|
| edges | 19 | similarity | 2,215 | 🗑 |
| edges | 15 | rated | 1,685 | 🗑 |
| edges | 14 | sold | 830 | 🗑 |
| edges | 13 | reports_to | 8 | 🗑 |
| edges | 46 | btw_inset | 6 | 🗑 |
| nodes | 12 | employee | 10 | 🗑 |
| edges | 11 | orders | 2,155 | 🗑 |
| nodes | 22 | inset | 20 | 🗑 |
| edges | 10 | purchased | 830 | 🗑 |

0 selected / 18 total

- Delete a Label
  If the Delete icon is clicked, it shows information about the label to be deleted as shown below and asks if you want to proceed with the deletion. Be careful in using the Delete option; if you delete a label, any relevant data will also be deleted. You can proceed with Delete or cancel Delete by pressing OK or CANCEL button at the bottom.



CONFIRM INFO

Are you sure to delete this LABEL?
If you remove label, All data will also be removed!

TYPE
nodes

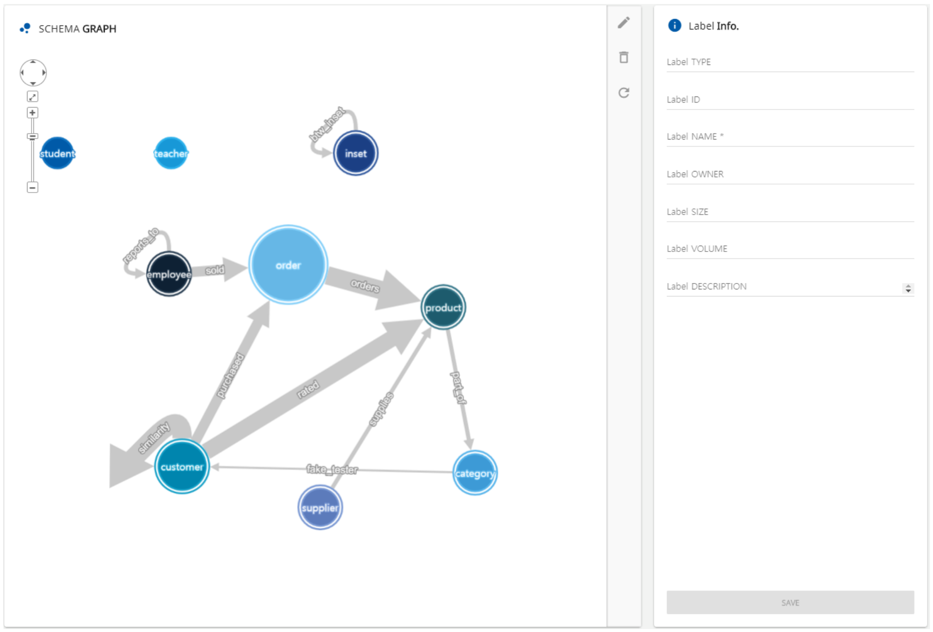
ID
3

NAME
product

SIZE
77

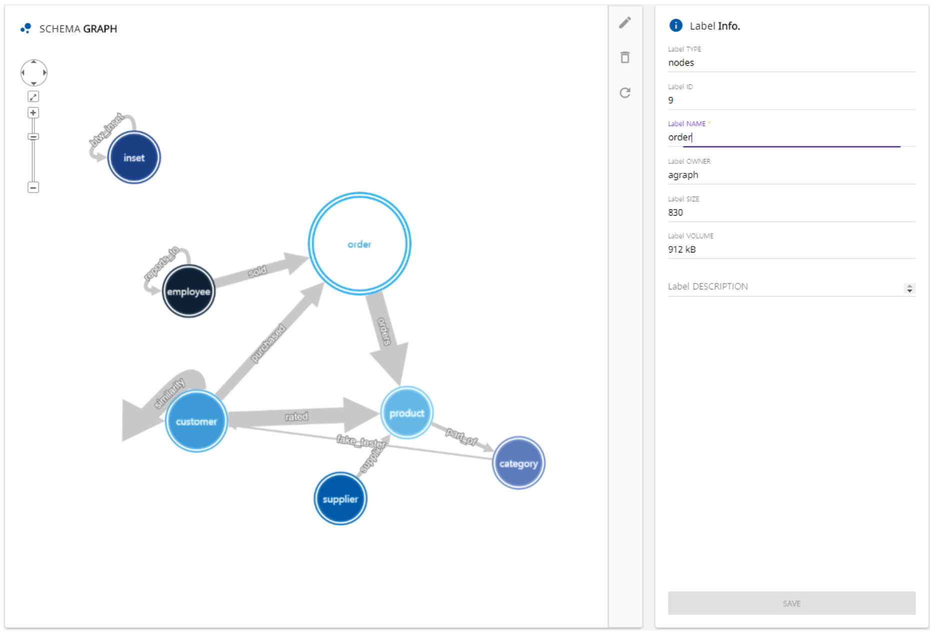
### Schema Graph & Label Info

The relationship between the node labels and the edge labels in metadata is described in graph.



You can choose each node or edge and see its details in Label Info.

If you select a node label named order, you will see this below screen:



Label TYPE
nodes

Label ID
9

Label NAME *
order

Label OWNER
agraph

Label SIZE
830

Label VOLUME
912 kB

Label DESCRIPTION

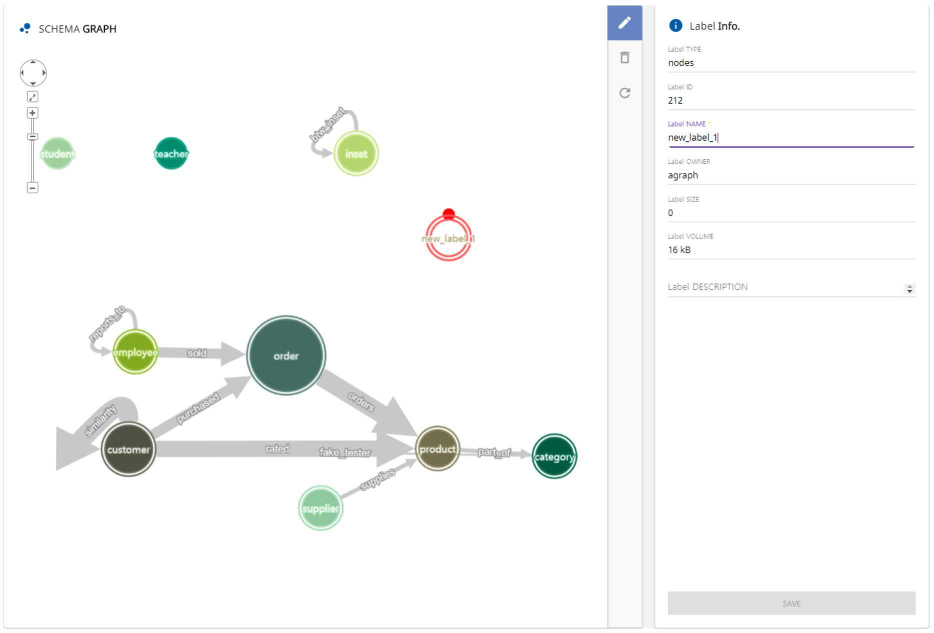
- Edit a Label

  Click Edit Schema( ) in the Schema Graph area and select a node or an edge to edit. Then, Label Info will be activated so that you can edit Label name and its description.

- Create a Label

  You may create a node by double-clicking the canvas in Edit mode, and generate an edge between different nodes by dragging the red dot of a node as shown below.

- Create node



Label Info.

Label TYPE
nodes

Label ID
212

Label NAME *
new_label_1

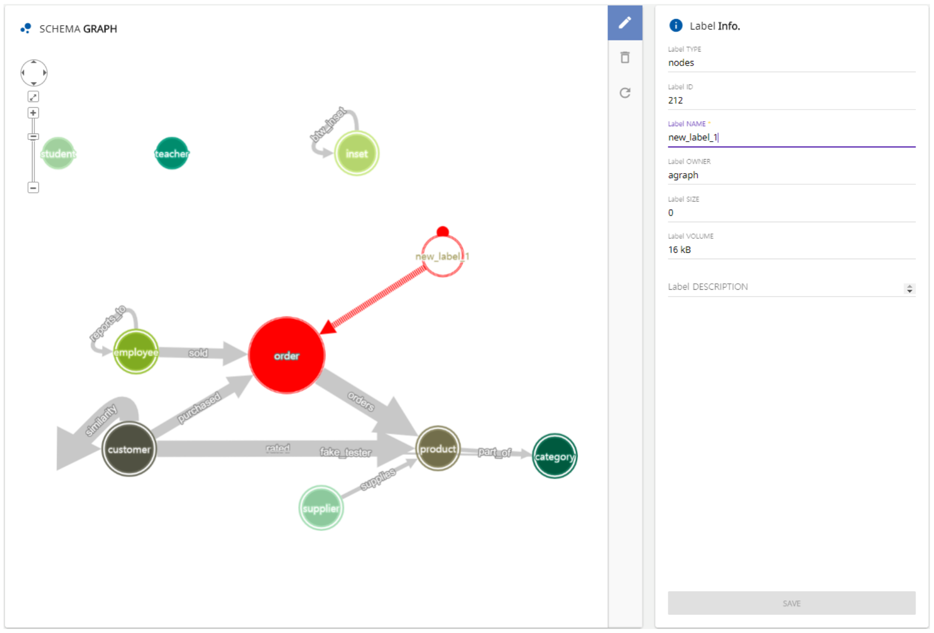
Label OWNER
agraph

Label SIZE
0

Label VOLUME
16 kB

Label DESCRIPTION

- Create edge



Label Info.

Label TYPE
nodes

Label ID
212

Label NAME *
new_label_1

Label OWNER
agraph

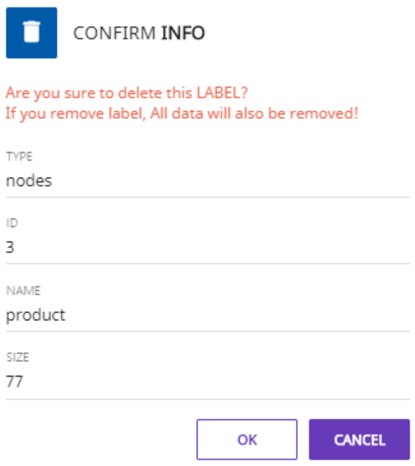
Label SIZE
0

Label VOLUME
16 kB

Label DESCRIPTION

- Delete a Label

  Select a nodevertex or an edge to delete and click Remove( ) in the Schema Graph area. It will show information about the label to be deleted and ask whether you want to proceed with Remove.


CONFIRM **INFO**

Are you sure to delete this LABEL?
If you remove label, All data will also be removed!

TYPE
nodes

ID
3

NAME
product

SIZE
77

## PL/Python Menu

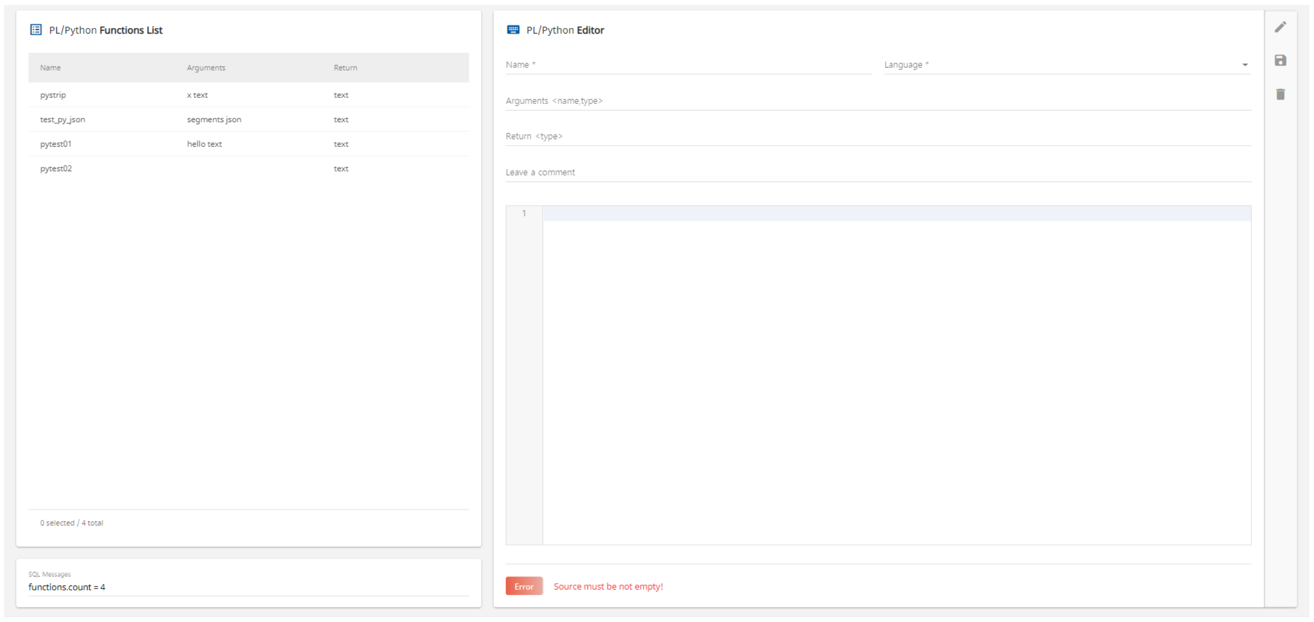
You may use PL/Python, a procedural language, in AgensGraph to write and execute the functions written in Python in the application.

To use PL/Python, you must create an extension.

```
agens=# CREATE EXTENSION plpython2u;
```

The Python syntax used in PL/Python is in line with the conventional Python language. See Developer Manual for more information on syntax.

In the PL/Python area of AgensBrowser, you can retrieve the PL/Python function list created in Agens-Graph database and create functions.



| Name | Arguments | Return |
|---|---|---|
| pystrip | x text | text |
| test_py_json | segments json | text |
| pytest01 | hello text | text |
| pytest02 | | text |

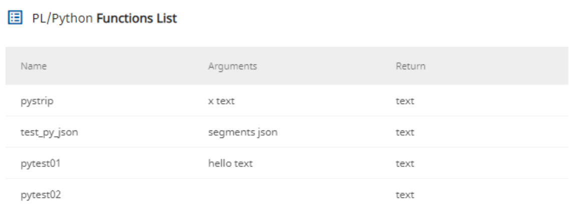
0 selected / 4 total

SQL Messages
functions.count = 4

Source must be not empty!

### PL/Python Functions List

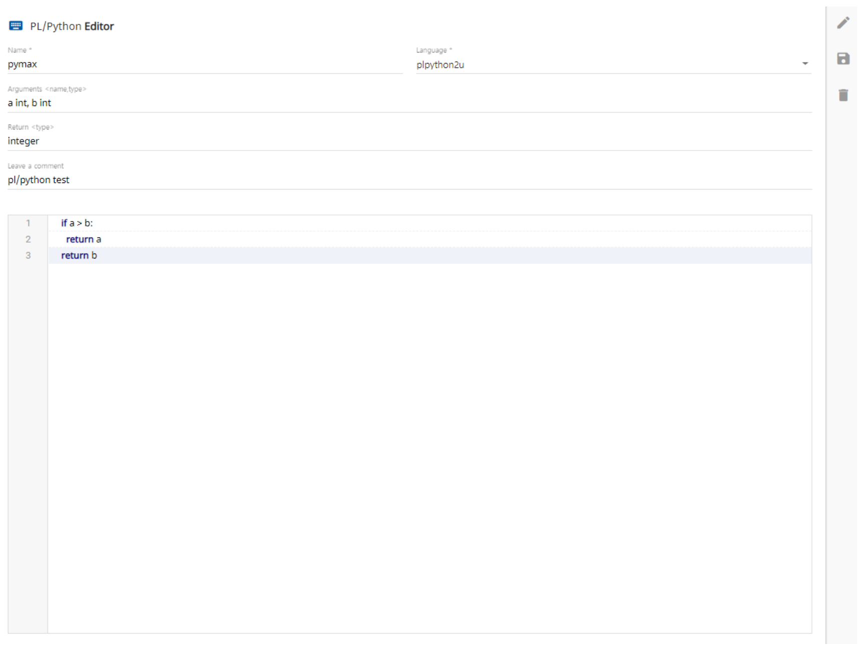
This is a list of PL/Python functions in AgensGraph. Click the created functions to see their details in the PL/Python Editor. A description on the function editor follows.



| Name | Arguments | Return |
|---|---|---|
| pystrip | x text | text |
| test_py_json | segments json | text |
| pytest01 | hello text | text |
| pytest02 | | text |

## PL/Python Editor

You can create functions using PL/Python Editor. Click new( ) to create a new function in PL/Python. You may also delete a created function by selecting it and clicking delete( ).



Name *
pymax

Language *
plpython2u

Arguments <name,type>
a int, b int

Return <type>
integer

Leave a comment
pl/python test

```
if a > b:
  return a
return b
```

1) Name
   Specify the function name.

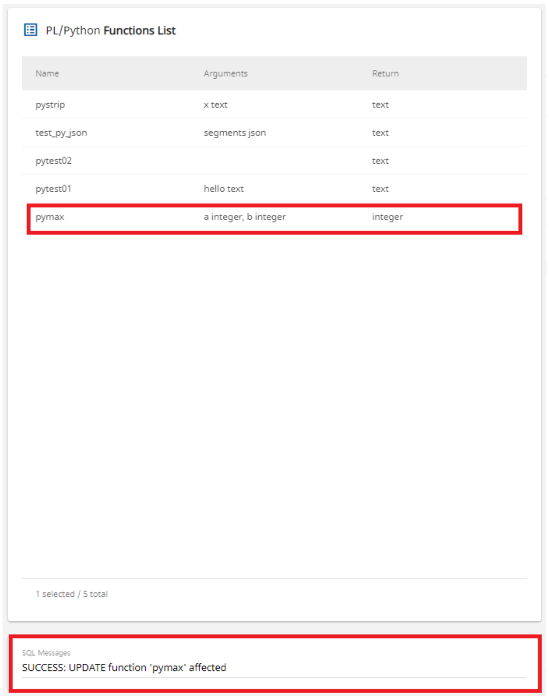
2) Language
   Choose one among Internal, C, SQL, PL/pgSQL, PL/Python(Only the PL/Python history is displayed in the Functions List).

3) Arguments <name, type>
   Specify the argument name and type to be used in the function.

4) Return
   Specify the Type of result to be returned.

5) Leave a comment
   Enter a short description of the function.

6) After coding a function in this section, press save( ) to create the function and update it in the list.



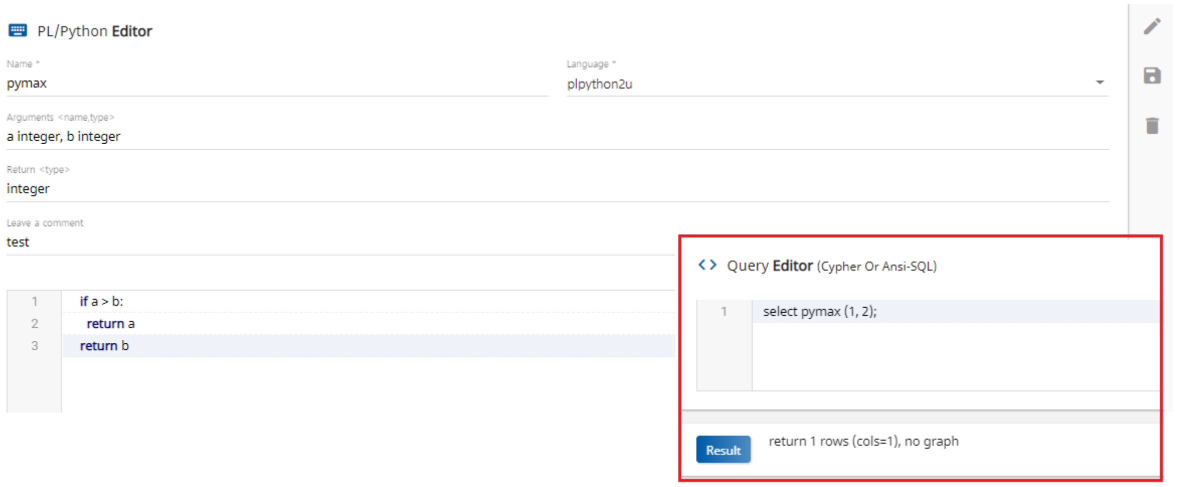
PL/Python **Functions List**

| Name | Arguments | Return |
|---|---|---|
| pystrip | x text | text |
| test_py_json | segments json | text |
| pytest02 | | text |
| pytest01 | hello text | text |
| pymax | a integer, b integer | integer |

1 selected / 5 total

SQL Messages
SUCCESS: UPDATE function 'pymax' affected

Example: This is the result of a created/executed PL/Python function. It can be executed from Graph tab.



PL/Python **Editor**

Name *
pymax

Language *
plpython2u

Arguments <name,type>
a integer, b integer

Return <type>
integer

Leave a comment
test

```
if a > b:
  return a
return b
```

Query **Editor** (Cypher Or Ansi-SQL)

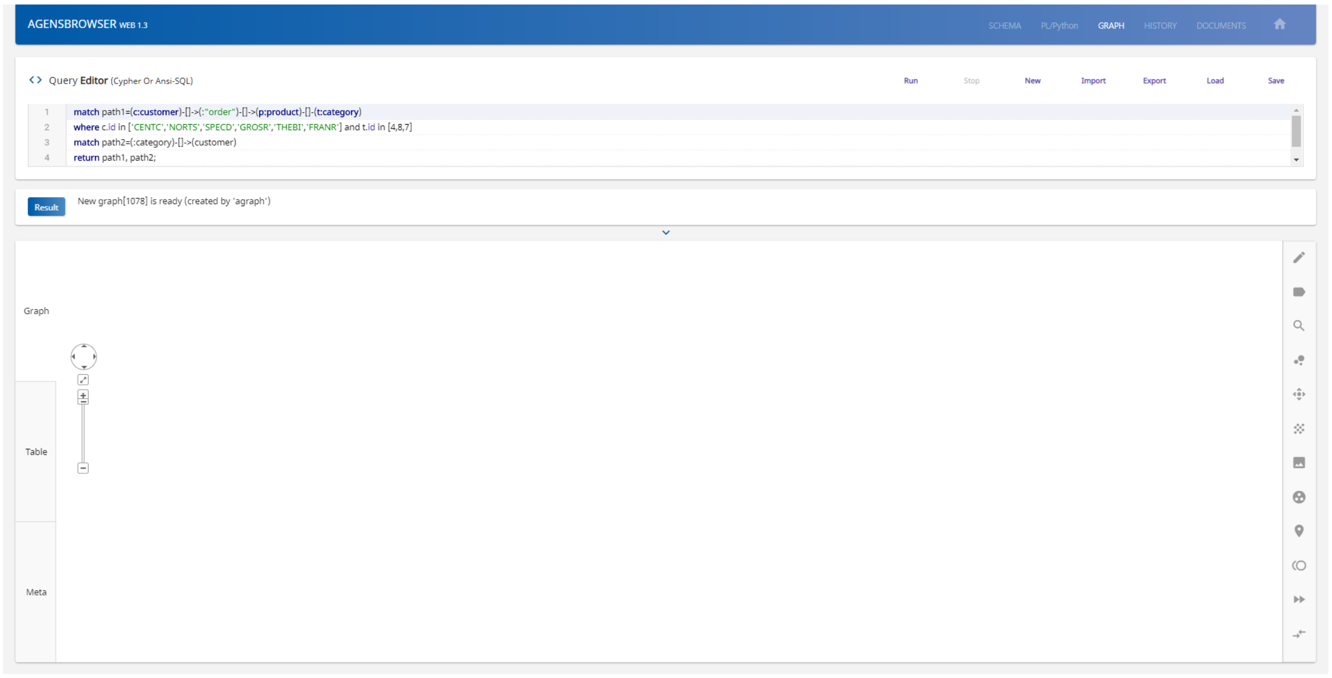
```
select pymax (1, 2);
```

return 1 rows (cols=1), no graph

## Graph Menu

Enables users to query AgensGraph database with SQL and Cypher (hybrid querying). The query results can be visualized as table or graph. It is possible to change the layout of the queried graph, apply a label style, and capture a screenshot and export it as an image file.

> Notice: If you switch pages while querying Graph DB, the previous tasks will not be saved.



```
match path1=(c:customer)-[]->(:"order")-[]->(p:product)-[]-(t:category)
where c.id in ['CENTC','NORTS','SPECD','GROSR','THEBI','FRANR'] and t.id in [4,8,7]
match path2=(:category)-[]->(customer)
return path1, path2;
```

New graph[1078] is ready (created by 'agraph')

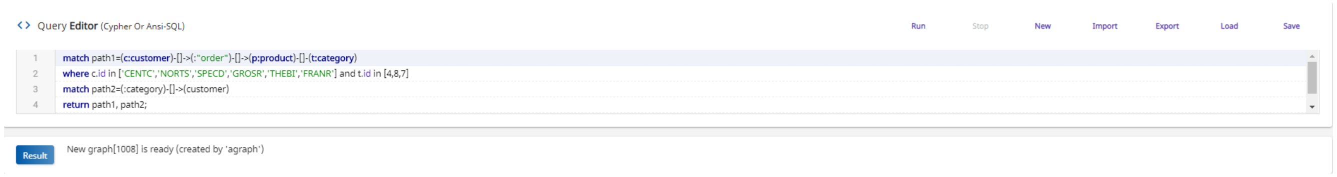
Query to Graph DB consists of:

- Query Editor
- Result Message
- Graph Tab
- Table Tab
- Meta Tab

### Query Editor

Enables users to query AgensGraph database with SQL and Cypher. Annotations can be written after -- (Ctrl+/). Query results are provided in three types (Graph, Meta, Table) and can be checked on each tab.

> Note: When accessing AgensBrowser, you are logged in through graph_path specified in yml. The command to change graph_path (set graph_path) does not work on the Query Editor .



```
match path1=(c:customer)-[]->(:"order")-[]->(p:product)-[]-(t:category)
where c.id in ['CENTC','NORTS','SPECD','GROSR','THEBI','FRANR'] and t.id in [4,8,7]
match path2=(:category)-[]->(customer)
return path1, path2;
```

New graph[1008] is ready (created by 'agraph')

The following is a description of the icons at the top.

- Run ( Run )

  Executes the written query.
  Click this button to run the query and see the query result (or an error message). You may execute only a certain statement that is selected(highlighted) by using mouse.

- Stop ( Stop )

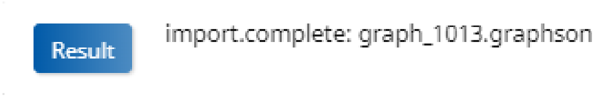
  Stops the running query.

- New ( New )

  Creates a new graph file.
  Pressing this button initializes the editor and result area.

- Import ( Import )

  Imports the Graphson type data. If the import is successful, the following message will be displayed.



import.complete: graph_1013.graphson

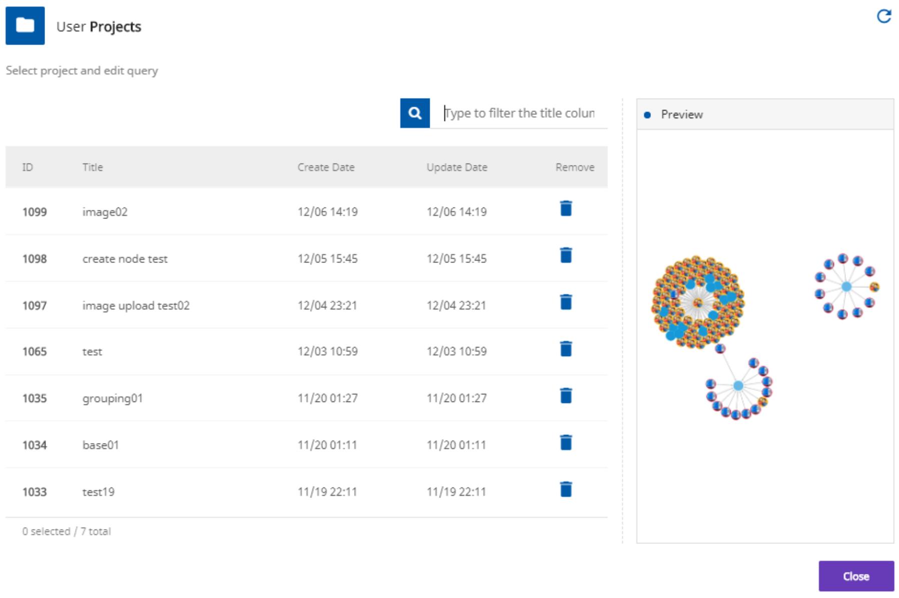
- Export ( Export )

  Exports the result of the performed query to Graphson type. The result is saved in the download folder as graph_xxxx.graphson.

```
> Tip: Graphson type is a JSON-based format for individual graph elements (vertices, edges). See [C
```

- Load ( Load )

  You can load a saved project. Click Load to see a list of saved projects as shown below.


User **Projects**

Select project and edit query

Type to filter the title colun

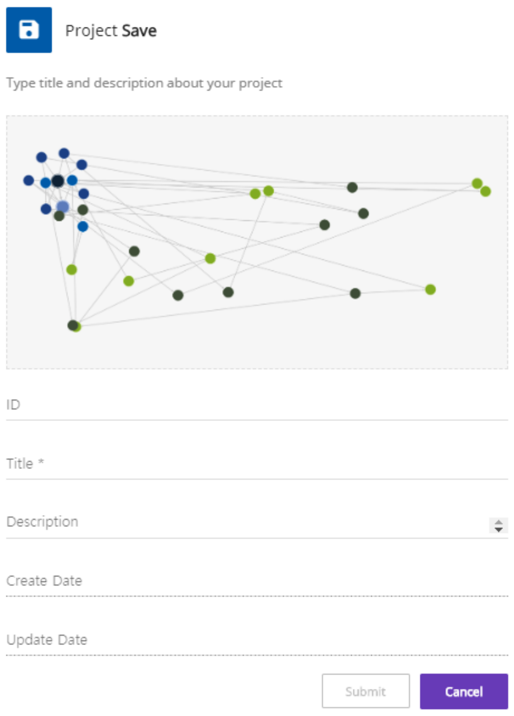
| ID | Title | Create Date | Update Date | Remove |
|---|---|---|---|---|
| **1099** | image02 | 12/06 14:19 | 12/06 14:19 | 🗑 |
| **1098** | create node test | 12/05 15:45 | 12/05 15:45 | 🗑 |
| **1097** | image upload test02 | 12/04 23:21 | 12/04 23:21 | 🗑 |
| **1065** | test | 12/03 10:59 | 12/03 10:59 | 🗑 |
| **1035** | grouping01 | 11/20 01:27 | 11/20 01:27 | 🗑 |
| **1034** | base01 | 11/20 01:11 | 11/20 01:11 | 🗑 |
| **1033** | test19 | 11/19 22:11 | 11/19 22:11 | 🗑 |

0 selected / 7 total

Select a project from the list to see the performed query and graph result for the saved project in the Graph area. You may preview the project before loading, or select Remove(🗑) to delete the saved project. You may also search for a specific project by title.

- Save ( Save )

  You may save the result of a query that has been performed. Press Save button to save the project as shown below. Title is a required field, and you may add a description of the project optionally in Description.


Project **Save**

Type title and description about your project

ID

Title *

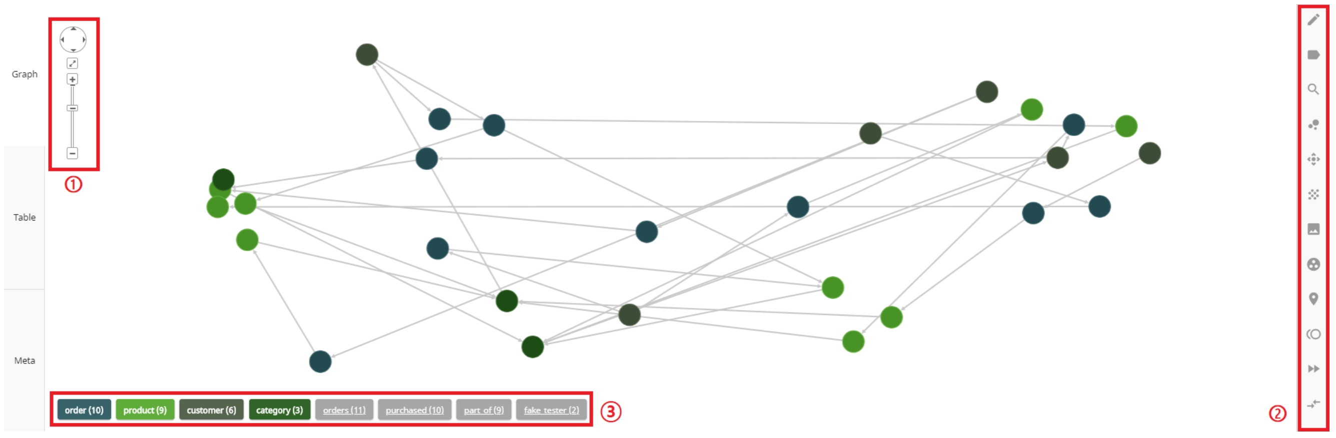
Description

Create Date

Update Date

## Graph Tab

Visualizes the result of the performed query in graph.

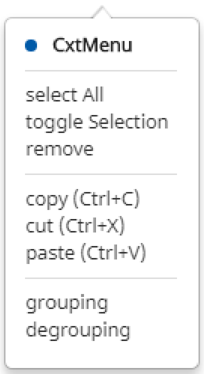


The following tasks can be done in the graph editor.

- Edit/modify graph
- Graph styling
- Transform graph
  - group, filter
- Navigate graph
  - Central point search
  - Title search
  - Shortest path search
  - Connection component
  - Timeline
  - Circular ring
  - Overlay (full/partial matching)

You can pan or zoom in/out the screen using Controller (①) located in the upper left corner of the graph window. You can also adjust the screen size with ⤢ button.
Here is a description on Edit/Modify Graph and change of Label Styles in the Graph area.

- Edit/Modify Graph
  Click the right mouse button in the Graph area to activate the following popup menu; it has select All, toggle Selection, remove, copy/cut/paste, and grouping/degrouping.



CxtMenu

select All
toggle Selection
remove

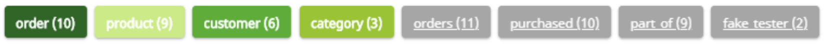
copy (Ctrl+C)
cut (Ctrl+X)
paste (Ctrl+V)

grouping
degrouping

<Shortcut>

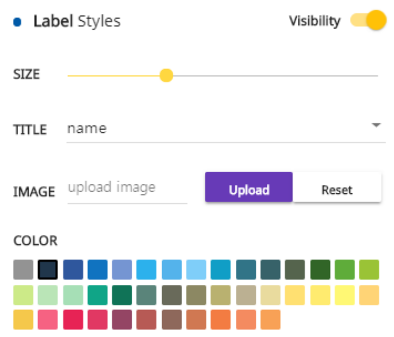
* Select All : Ctrl + a
* Toogle Selection : Ctrl + e
* Copy : Ctrl + c
* Cut : Ctrl + x
* Paste : Ctrl + v
* Undo : Ctrl + z
* Redo : Ctrl + y

- Label Styles
  Select a Label name.



order (10) product (9) customer (6) category (3) orders (11) purchased (10) part_of (9) fake_tester (2)

Select a Label that you want to change. In Label Styles, you can change the title, size, and color of the label, and insert a desired image. You can also choose Visibility of the label.



Label Styles Visibility

SIZE

TITLE name

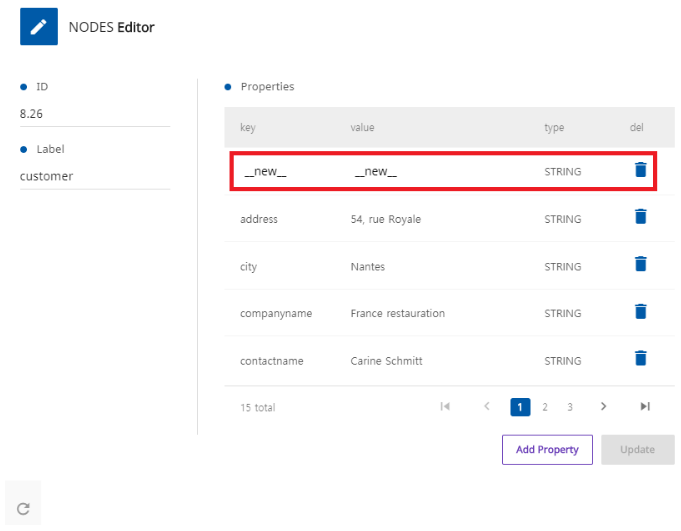
IMAGE upload image Upload Reset

COLOR

The following is a detailed description of the Graph Result menu (②).

- Edit Mode ( )

1) Add property
   Click Edit Mode and select a node or an edge to change its property. You may add a new property using Add Property. Double click the existing property if you want to edit. You can also delete the existing or erroneous property with del( ).


NODES Editor

● ID

8.26

● Label

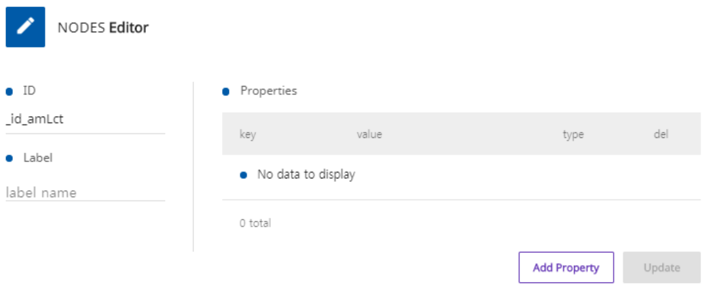
customer

● Properties

| key | value | type | del |
|---|---|---|---|
| __new__ | __new__ | STRING | |
| address | 54, rue Royale | STRING | |
| city | Nantes | STRING | |
| companyname | France restauration | STRING | |
| contactname | Carine Schmitt | STRING | |

15 total

2) Edit Node/Edge
   Click Edit Mode and then double-click the canvas to create a node or an edge. You can specify a label and add a property.


NODES Editor

● ID

_id_amLct

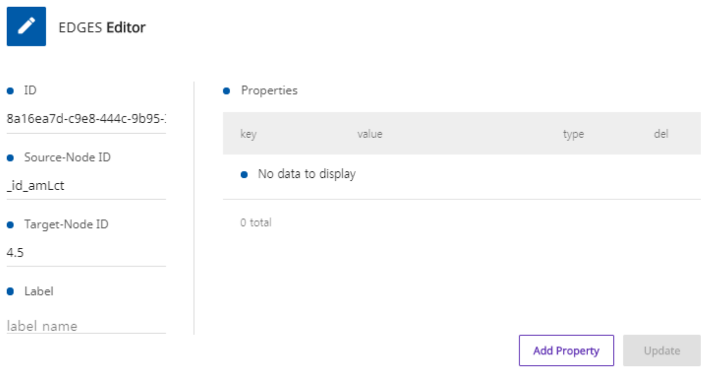
● Label

label name

● Properties

| key | value | type | del |
|---|---|---|---|
| ● No data to display | | | |

0 total


EDGES Editor

● ID

8a16ea7d-c9e8-444c-9b95-

● Source-Node ID

_id_amLct

● Target-Node ID

4.5

● Label

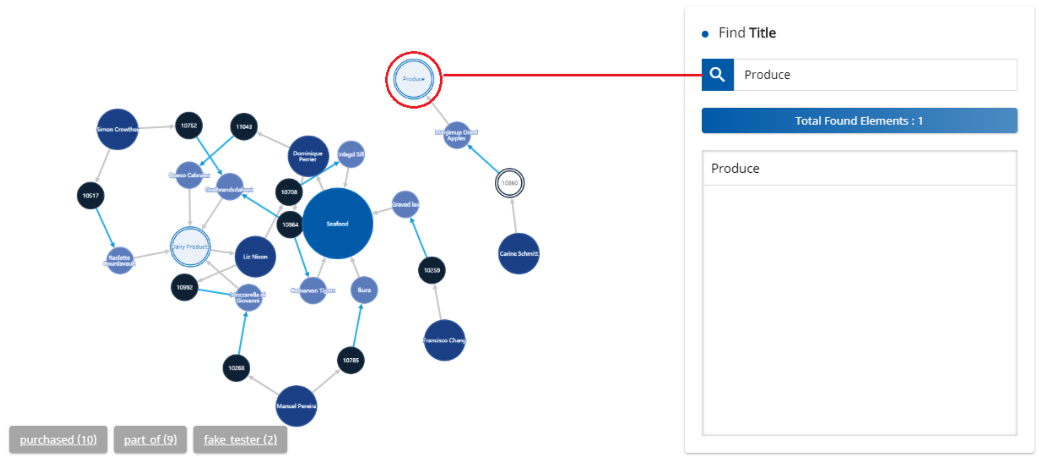
label name

● Properties

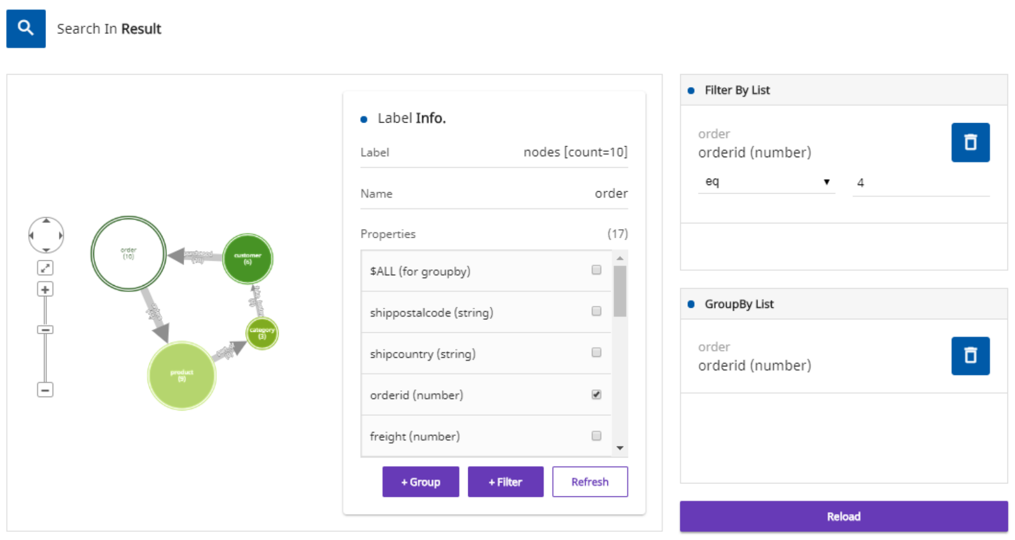
| key | value | type | del |
|---|---|---|---|
| ● No data to display | | | |

0 total

- Show/Hide Title ( )
  You can show or hide the title, and search for a node label you want by its title.



- Filtering & Grouping ( )
  Labels can be filtered or grouped by property.



| | |
|---|---|
| Label | nodes [count=10] |
| Name | order |
| Properties | (17) |

| | |
|---|---|
| $ALL (for groupby) | ☐ |
| shippostalcode (string) | ☐ |
| shipcountry (string) | ☐ |
| orderid (number) | ✓ |
| freight (number) | ☐ |

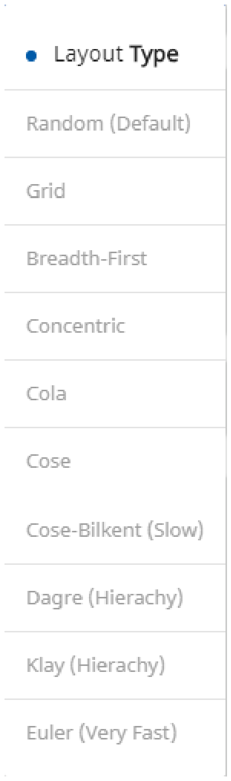
Filter By List: order, orderid (number), eq, 4

GroupBy List: order, orderid (number)

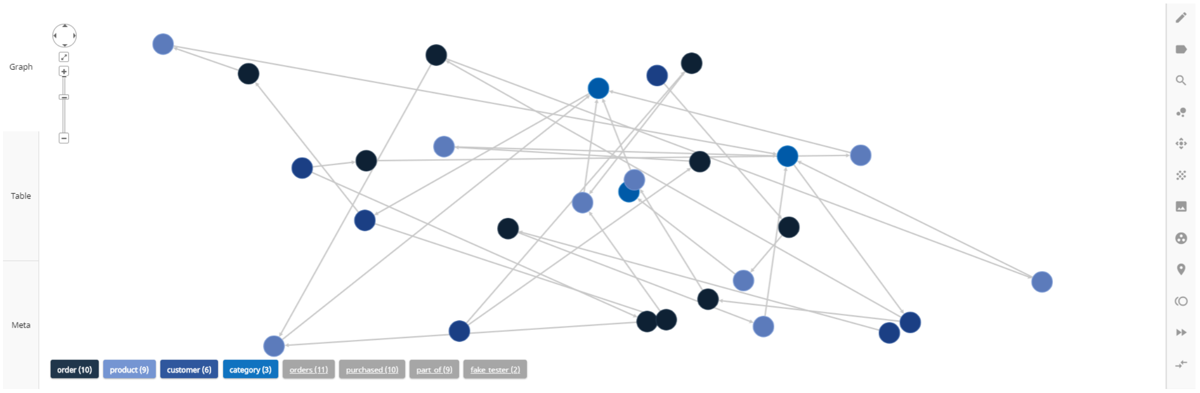
```
*   filterBy : The graph queried by user is filtered by the condition set from user input.

Example: Find a node containing "UK" in shipcountry, a property of the node `Order`.


\begin{center}\resizebox{400pt}{215pt}{\includegraphics[width=800pt,height=430pt]{newimg/filter_ex1} } 


\begin{center}\resizebox{400pt}{165pt}{\includegraphics[width=800pt,height=330pt]{newimg/filter_ex2} } 

* groupBy : The graph queried by user is grouped by the property value of a specific label.

Example: Grouped by (groupBy) shipcountry, a property of the node `Order`.


\begin{center}\resizebox{400pt}{215pt}{\includegraphics[width=800pt,height=430pt]{newimg/group_ex1} } \
```

\begin{center}\resizebox{400pt}{165pt}{\includegraphics[width=800pt,height=330pt]{newimg/group_ex2} } \

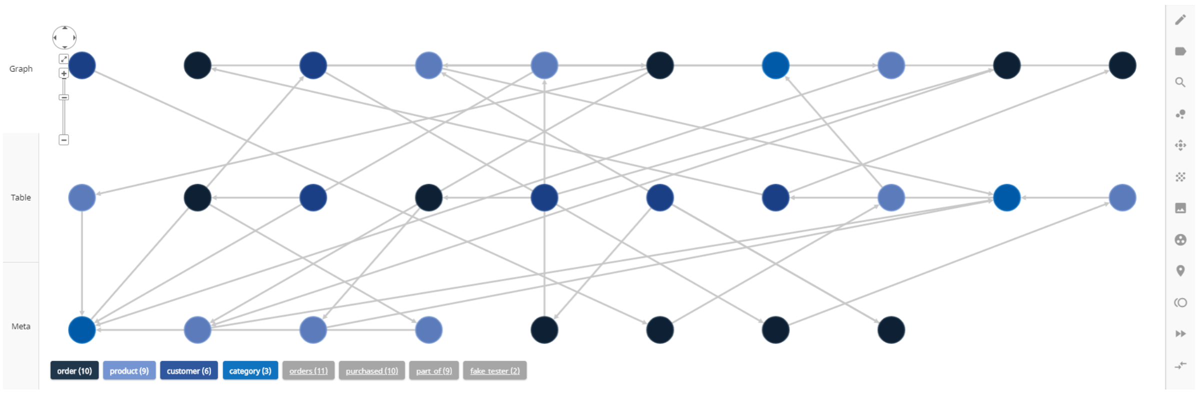
- Layout ( )
  You can change the layout of graph result by selecting a layout. Select some nodes/edges by holding down the Shift key and dragging the mouse, and apply the layout to them.



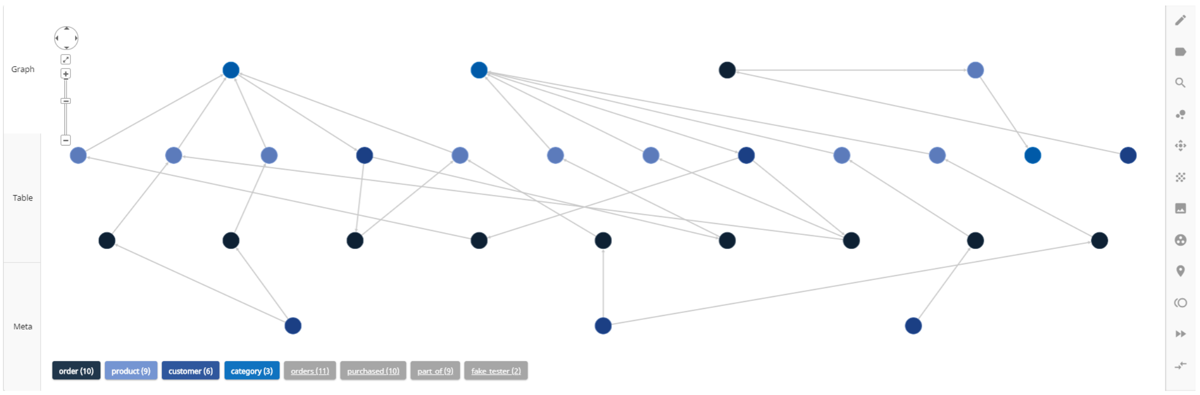
1) Random (Default): Arranges nodes randomly to have an unexpected shape.



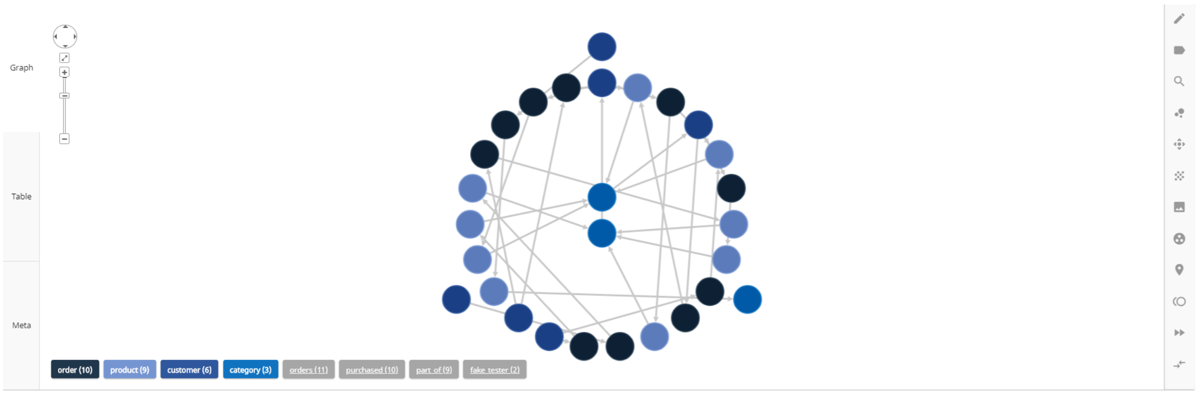
2) Grid : Arranges nodes in a grid structure.



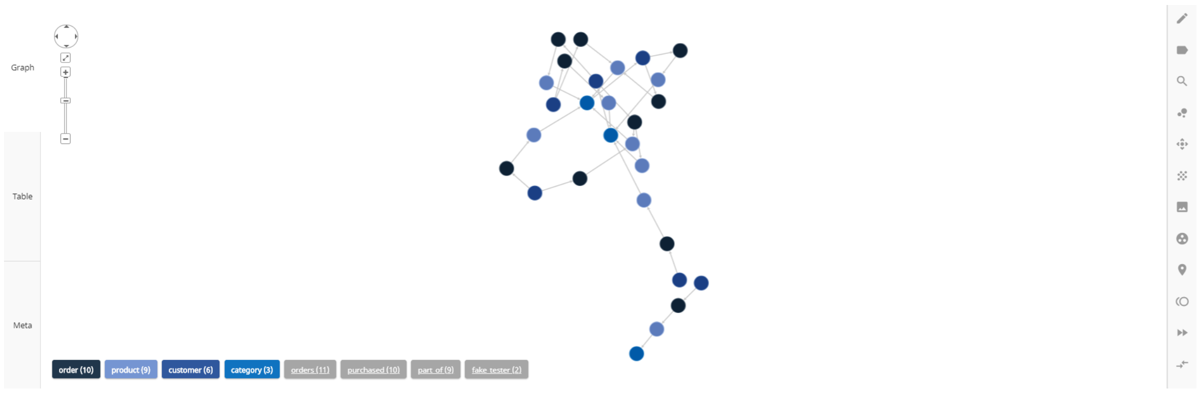
3) Breadth-First : The Breadth-First-method that arranges nodes with many edges at the top and othe nodes to them at the bottom.



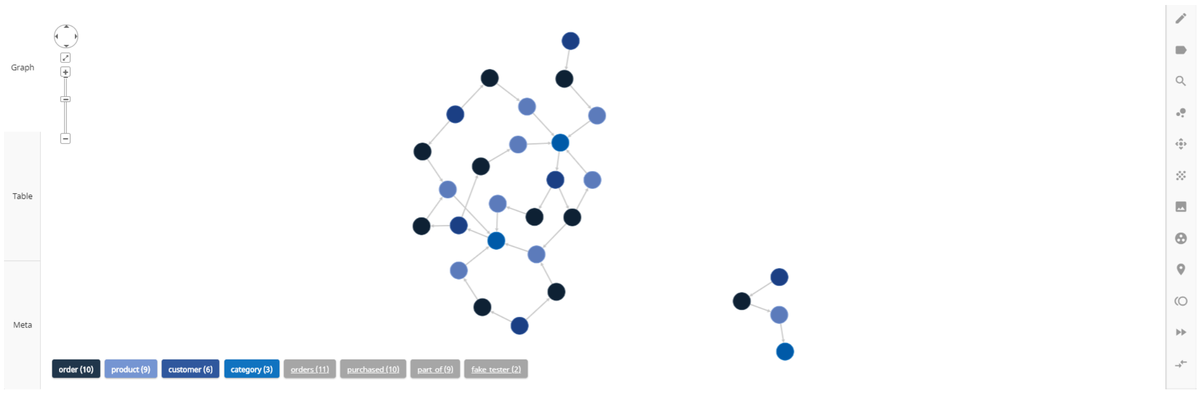
4) Concentric : Arranges the nodes with many edges at the center and other nodes connected to them at the outer.



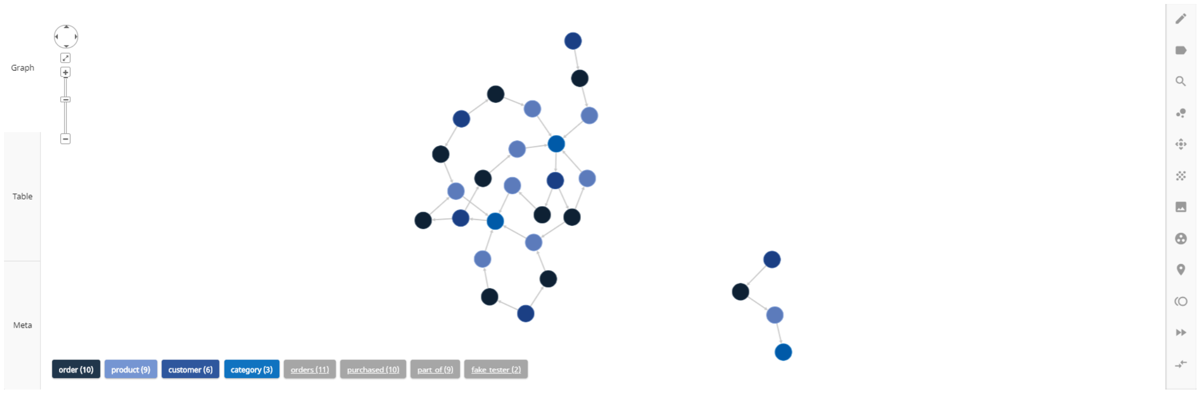
5) Cola : Places a node cluster with many connected nodes at the top.



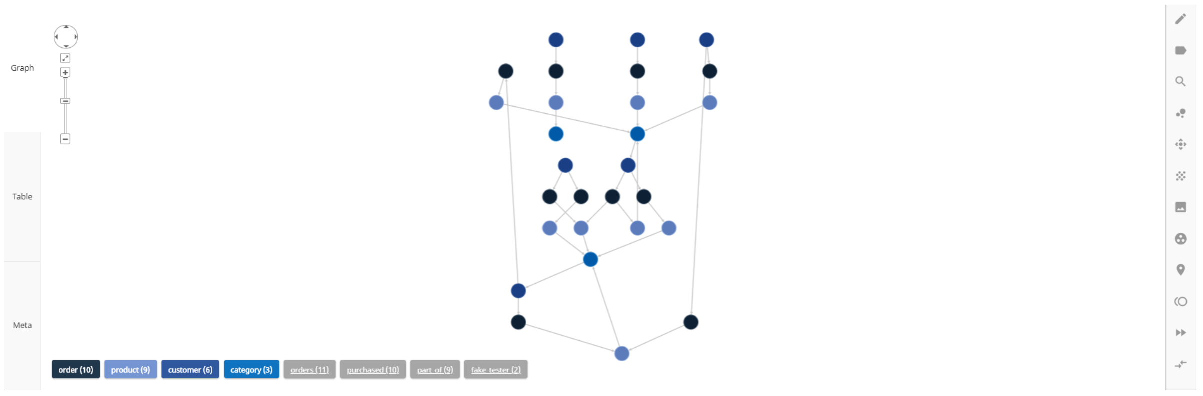
6) Cose : Keep a certain distance among the connected node clusters so that the cluster distribution can be easily identified.



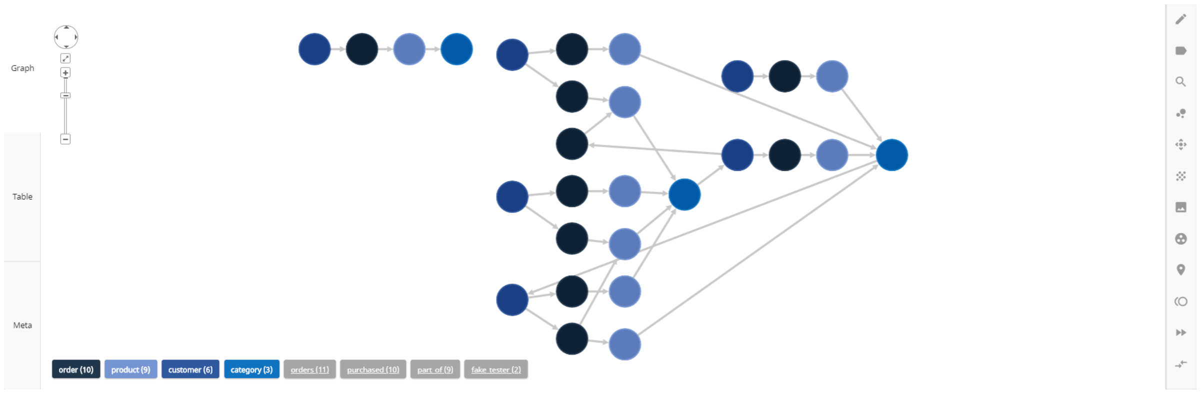
7) Cose-Bilkent (Slow) : A modified version of the Cose layout.



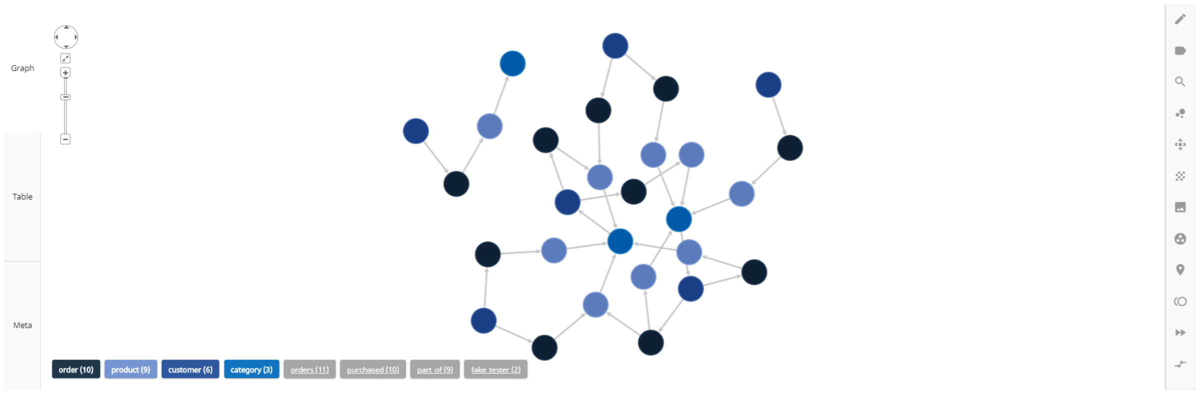
8) Dagre (Hierarchy) : Arranges the connected node clusters horizontally.



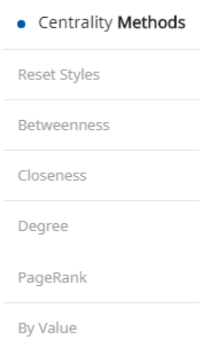
9) Klay (Hierarchy) : Forms a hierarchical layout from left to right horizontally.



10) Euler (Very Fast) : A force-directed optimization using quad-tree data structure allowing fast arrangement of a big graph.



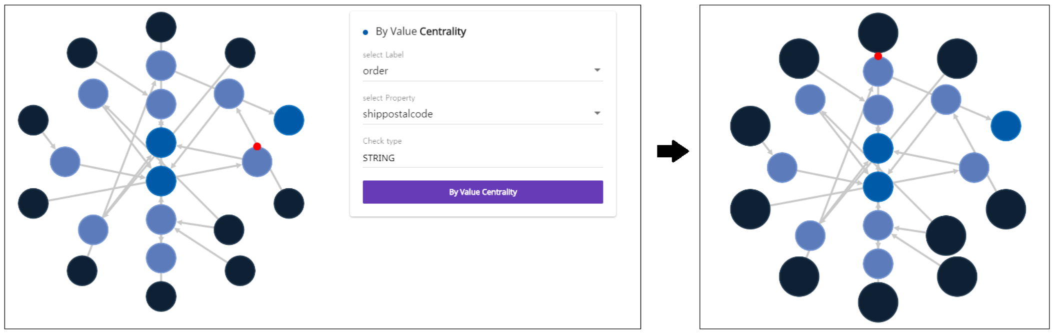
- Centrality ( )
  Provides 'byValue' that changes the size of node by Betweenness, Closeness, Degree, PageRank, and the property value of graph. Select Reset Styles to reset the applied style.



The below is a brief description on Betweenness, Closeness, Degree, and PageRank:

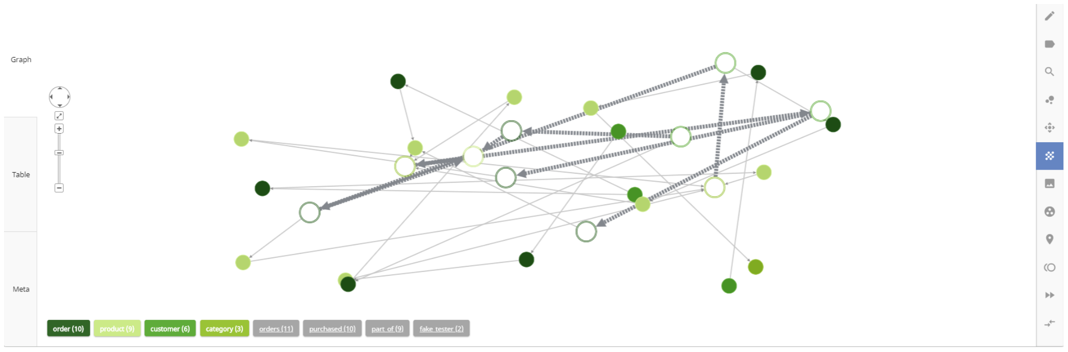
- Betweenness : Centrality considering whether the shortest path between all node pairs on the network passes through a specific node.
- Closeness : Centrality considering how close it is to every node on the network.
- Degree : Centrality considering only the weights of edges directly connected to each node.
- PageRank : Measures relative importance of objects on the network by assigning weight to the relationship between the referencing node(Link Analysis algorithm) and the referenced node.
- ByValue : Centrality considering the size of the node based on the attributes of the graph data.

Example: Result after changing the node size based on the attributes of the label.



select Label: order
select Property: shippostalcode
Check type: STRING

• Highlight Neighbors ( )
Toggle/Toggle off the highlighting on neighboring vertices. With this function, you may select the adjacent nodes (up to depth 3) whose edge labels are not redundant as neighbors.



- Image Export ( )
  You can save the result as an image file (PNG); the water-mark is available and, the image is saved with a transparent background so it can be used for presentation materials.

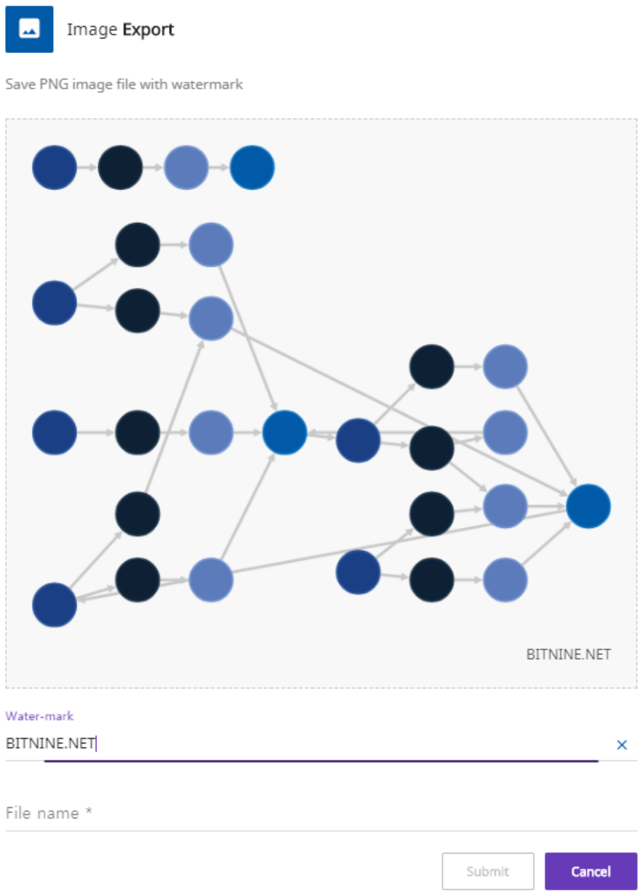


Image Export

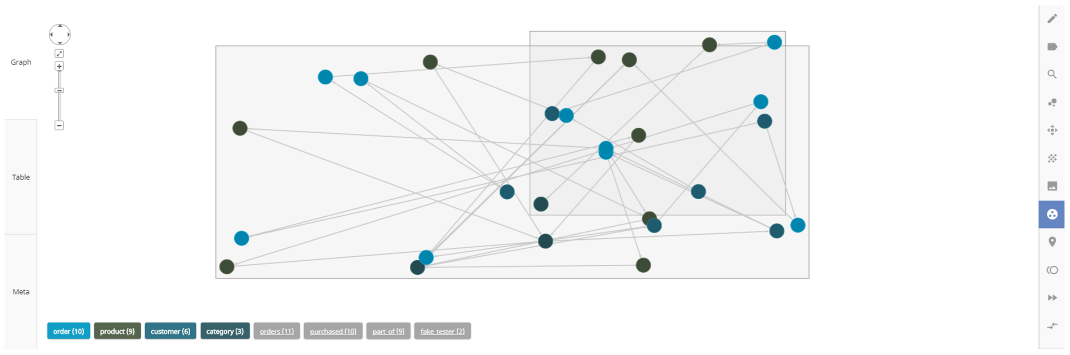
Save PNG image file with watermark

BITNINE.NET

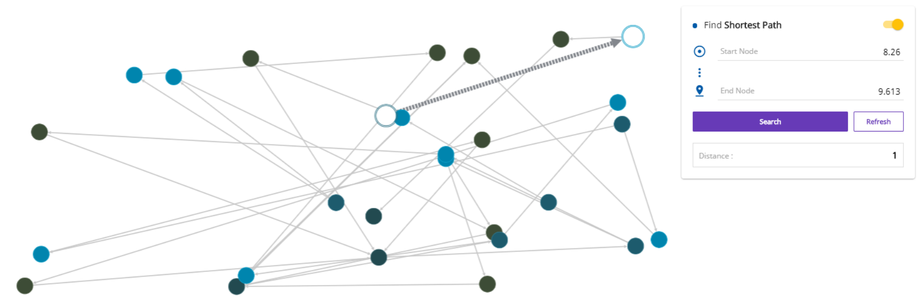
Water-mark
BITNINE.NET

File name *

- Find Connected Group ( )
  You may see the connected components.



- Find Shortes Path ( )
  You may check the shortest path. Select Start Node and End Node, and then search for the shortest path. The shortest path between the two nodes will be displayed. Select Directed Option( )to see bidirectional results.

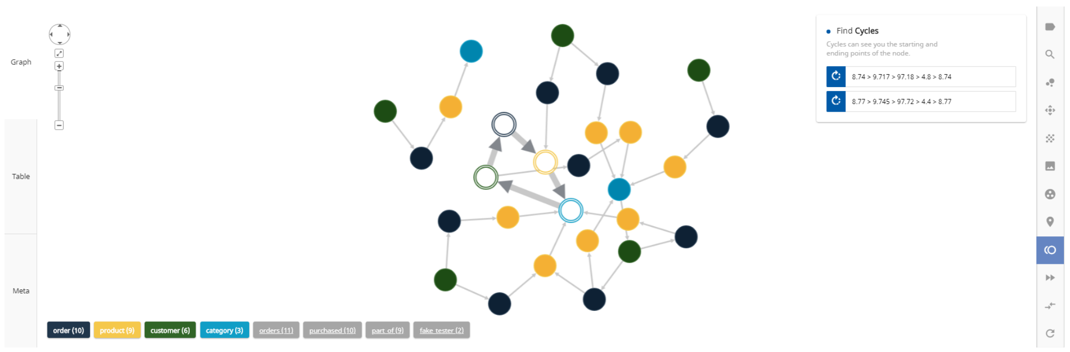


Start Node 8.26

End Node 9.613

Distance : 1

- Find Cycles ( )
  Cycle detection is provided.

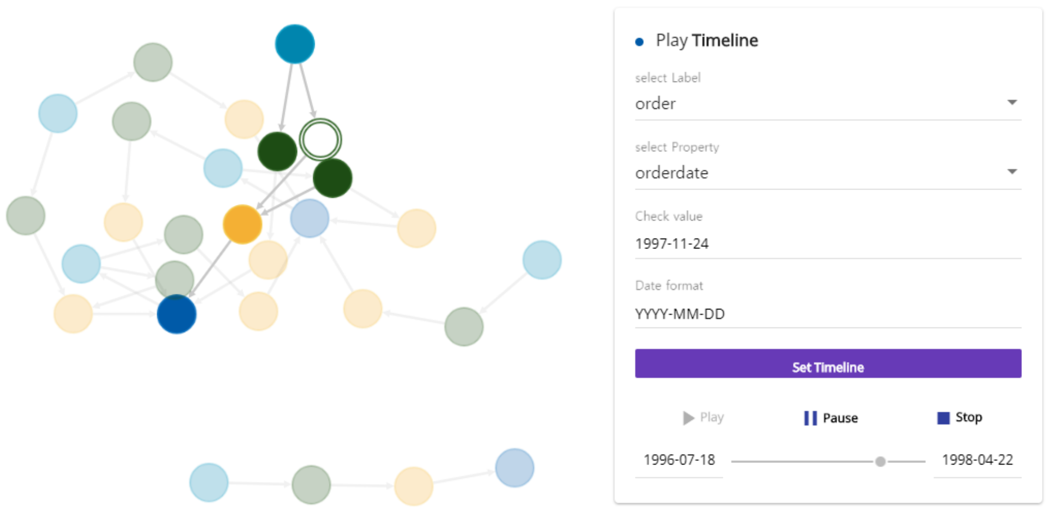


Cycles can see you the starting and ending points of the node.

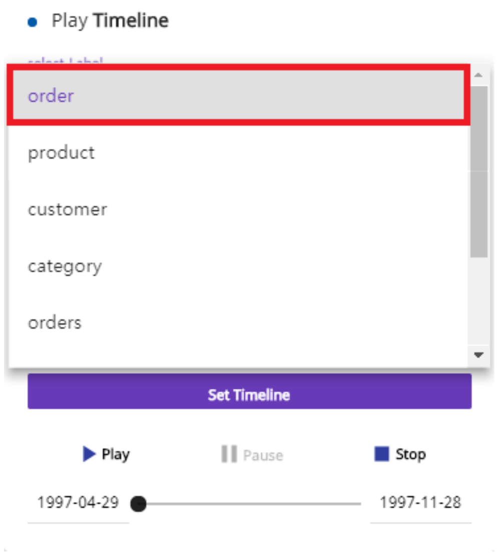
8.74 > 9.717 > 97.18 > 4.8 > 8.74

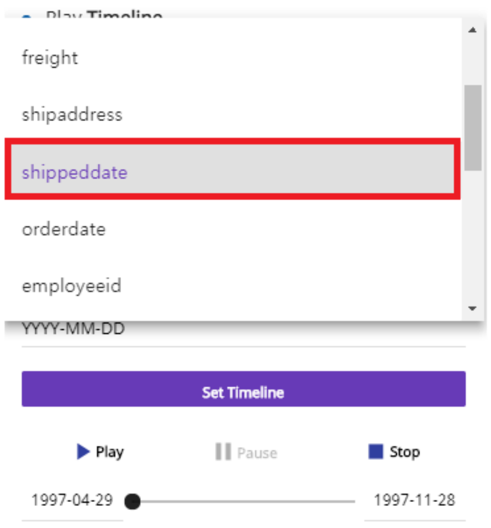
8.77 > 9.745 > 97.72 > 4.4 > 8.77

- Play Timeline ( )
  This shows the flow of graph change by Date. Enter the Date or Time format of the selected property. (default is `YYYY-MM-DD`)

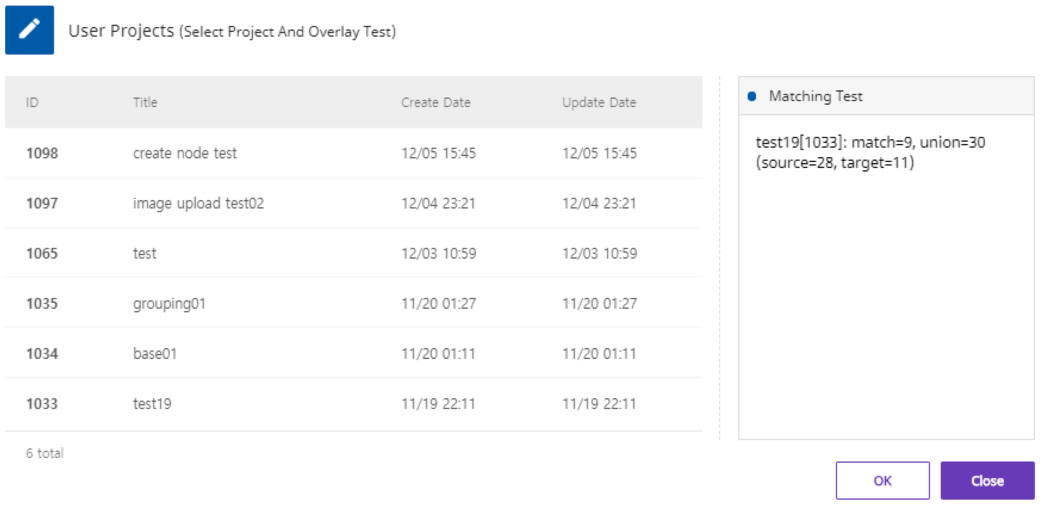


This is an example of using Play Timeline. Choose a label in select Label, and select a property whose type is date in select Property. Press `Set Timeline` button to insert data. Click `Play` to activate the timeline; Operate at 3 seconds intervals (interval not adjustable) from the minimum date to the maximum date based on the Date column.





- Overlay Graph ( )
  You can import a saved project to merge the graphs into full matching (identical node ID matching) or partial matching (identical Label/Direction matching for neighbor nodes connected with the same ID). Check the result by selecting the desired project from the list below; check the Matching Test of the project on the right side of the screen.



User Projects (Select Project And Overlay Test)

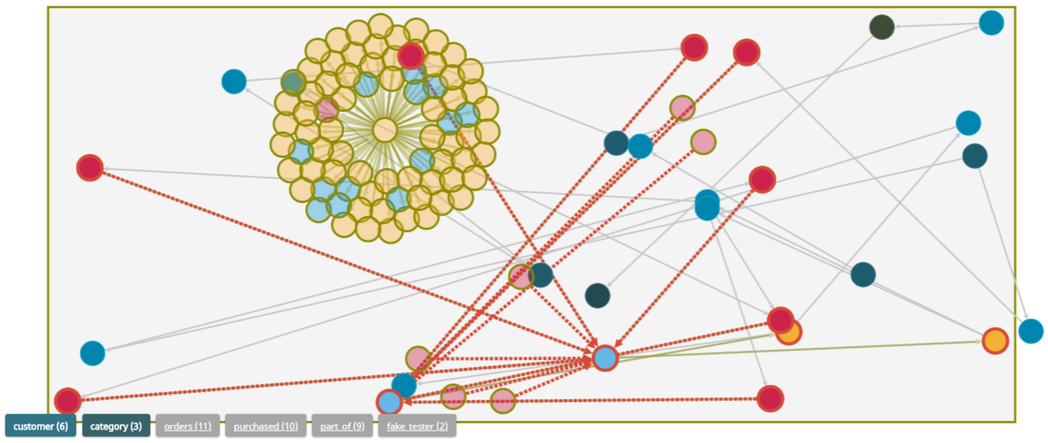
| ID | Title | Create Date | Update Date |
|---|---|---|---|
| 1098 | create node test | 12/05 15:45 | 12/05 15:45 |
| 1097 | image upload test02 | 12/04 23:21 | 12/04 23:21 |
| 1065 | test | 12/03 10:59 | 12/03 10:59 |
| 1035 | grouping01 | 11/20 01:27 | 11/20 01:27 |
| 1034 | base01 | 11/20 01:11 | 11/20 01:11 |
| 1033 | test19 | 11/19 22:11 | 11/19 22:11 |

6 total

Matching Test

test19[1033]: match=9, union=30 (source=28, target=11)

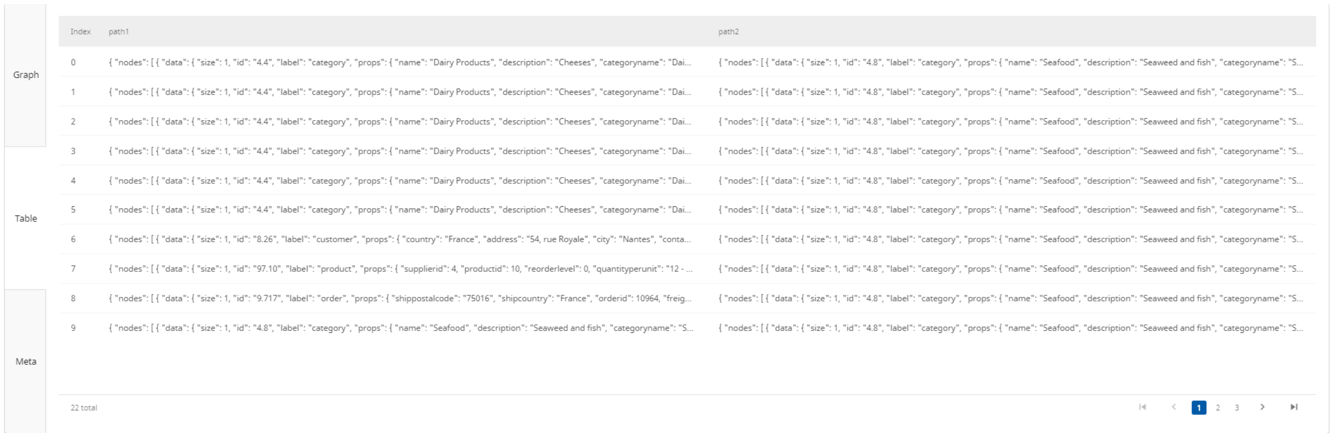
The following is the result of such graph merging. Full matching is indicated by red solid lines, while partial matching is represented by dotted lines.



- Reload Graph ( )
  You can reload the original graph after checking the result.

### Table Tab

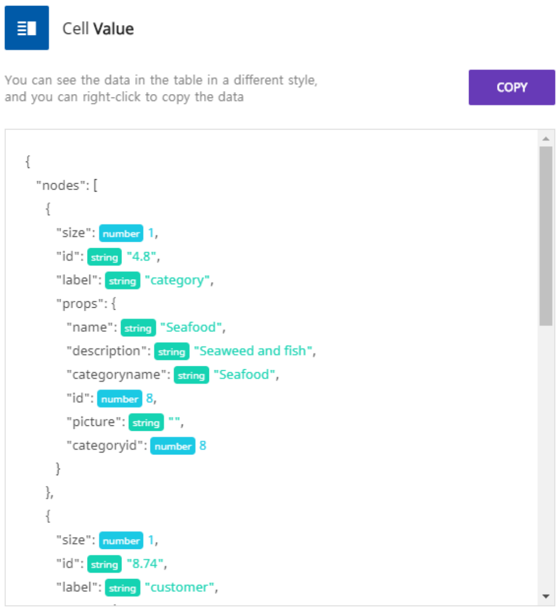
Displays the query results in a table.



| Index | path1 | path2 |
|---|---|---|
| 0 | { "nodes": [ { "data": { "size": 1, "id": "4.4", "label": "category", "props": { "name": "Dairy Products", "description": "Cheeses", "categoryname": "Dai... | { "nodes": [ { "data": { "size": 1, "id": "4.8", "label": "category", "props": { "name": "Seafood", "description": "Seaweed and fish", "categoryname": "S... |
| 1 | { "nodes": [ { "data": { "size": 1, "id": "4.4", "label": "category", "props": { "name": "Dairy Products", "description": "Cheeses", "categoryname": "Dai... | { "nodes": [ { "data": { "size": 1, "id": "4.8", "label": "category", "props": { "name": "Seafood", "description": "Seaweed and fish", "categoryname": "S... |
| 2 | { "nodes": [ { "data": { "size": 1, "id": "4.4", "label": "category", "props": { "name": "Dairy Products", "description": "Cheeses", "categoryname": "Dai... | { "nodes": [ { "data": { "size": 1, "id": "4.8", "label": "category", "props": { "name": "Seafood", "description": "Seaweed and fish", "categoryname": "S... |
| 3 | { "nodes": [ { "data": { "size": 1, "id": "4.4", "label": "category", "props": { "name": "Dairy Products", "description": "Cheeses", "categoryname": "Dai... | { "nodes": [ { "data": { "size": 1, "id": "4.8", "label": "category", "props": { "name": "Seafood", "description": "Seaweed and fish", "categoryname": "S... |
| 4 | { "nodes": [ { "data": { "size": 1, "id": "4.4", "label": "category", "props": { "name": "Dairy Products", "description": "Cheeses", "categoryname": "Dai... | { "nodes": [ { "data": { "size": 1, "id": "4.8", "label": "category", "props": { "name": "Seafood", "description": "Seaweed and fish", "categoryname": "S... |
| 5 | { "nodes": [ { "data": { "size": 1, "id": "4.4", "label": "category", "props": { "name": "Dairy Products", "description": "Cheeses", "categoryname": "Dai... | { "nodes": [ { "data": { "size": 1, "id": "4.8", "label": "category", "props": { "name": "Seafood", "description": "Seaweed and fish", "categoryname": "S... |
| 6 | { "nodes": [ { "data": { "size": 1, "id": "8.26", "label": "customer", "props": { "country": "France", "address": "54, rue Royale", "city": "Nantes", "conta... | { "nodes": [ { "data": { "size": 1, "id": "4.8", "label": "category", "props": { "name": "Seafood", "description": "Seaweed and fish", "categoryname": "S... |
| 7 | { "nodes": [ { "data": { "size": 1, "id": "97.10", "label": "product", "props": { "supplierid": 4, "productid": 10, "reorderlevel": 0, "quantityperunit": "12 - ... | { "nodes": [ { "data": { "size": 1, "id": "4.8", "label": "category", "props": { "name": "Seafood", "description": "Seaweed and fish", "categoryname": "S... |
| 8 | { "nodes": [ { "data": { "size": 1, "id": "9.717", "label": "order", "props": { "shippostalcode": "75016", "shipcountry": "France", "orderid": 10964, "freig... | { "nodes": [ { "data": { "size": 1, "id": "4.8", "label": "category", "props": { "name": "Seafood", "description": "Seaweed and fish", "categoryname": "S... |
| 9 | { "nodes": [ { "data": { "size": 1, "id": "4.8", "label": "category", "props": { "name": "Seafood", "description": "Seaweed and fish", "categoryname": "S... | { "nodes": [ { "data": { "size": 1, "id": "4.8", "label": "category", "props": { "name": "Seafood", "description": "Seaweed and fish", "categoryname": "S... |

22 total

Click a cell in the table to check all its values in the dialog box. Press Copy button in the dialog box to copy all values to clipboard.

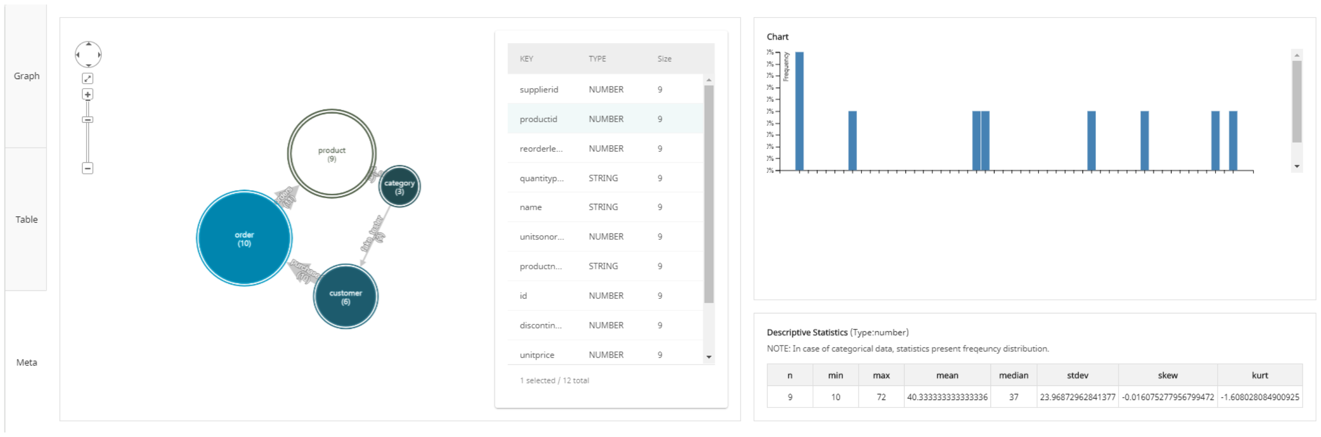


You can see the data in the table in a different style, and you can right-click to copy the data

```
{
  "nodes": [
    {
      "size": number 1,
      "id": string "4.8",
      "label": string "category",
      "props": {
        "name": string "Seafood",
        "description": string "Seaweed and fish",
        "categoryname": string "Seafood",
        "id": number 8,
        "picture": string "",
        "categoryid": number 8
      }
    },
    {
      "size": number 1,
      "id": string "8.74",
      "label": string "customer",
```

## Meta Tab

Provides a meta- graph for the result graph and attribute information of each label. Meta tab offers a frequency bar chart and technical statistics based on the result data when the property is clicked. The chart and statistics results show only provides numeric types and character types.



| KEY | TYPE | Size |
|---|---|---|
| supplierid | NUMBER | 9 |
| productid | NUMBER | 9 |
| reorderle... | NUMBER | 9 |
| quantityp... | STRING | 9 |
| name | STRING | 9 |
| unitsonor... | NUMBER | 9 |
| productn... | STRING | 9 |
| id | NUMBER | 9 |
| discontin... | NUMBER | 9 |
| unitprice | NUMBER | 9 |

1 selected / 12 total

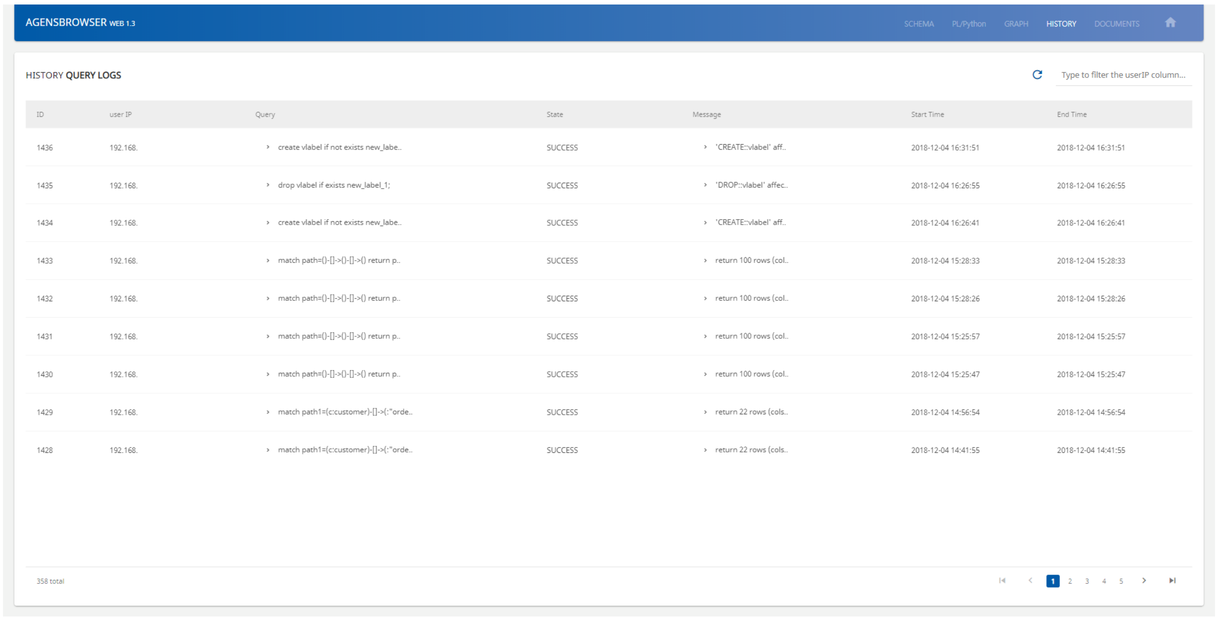
**Descriptive Statistics** (Type:number)

NOTE: In case of categorical data, statistics present freqeuncy distribution.

| n | min | max | mean | median | stdev | skew | kurt |
|---|---|---|---|---|---|---|---|
| 9 | 10 | 72 | 40.333333333333336 | 37 | 23.96872962841377 | -0.016075277956799472 | -1.608028084900925 |

## History Menu

You can retrieve the history of the performed queries.



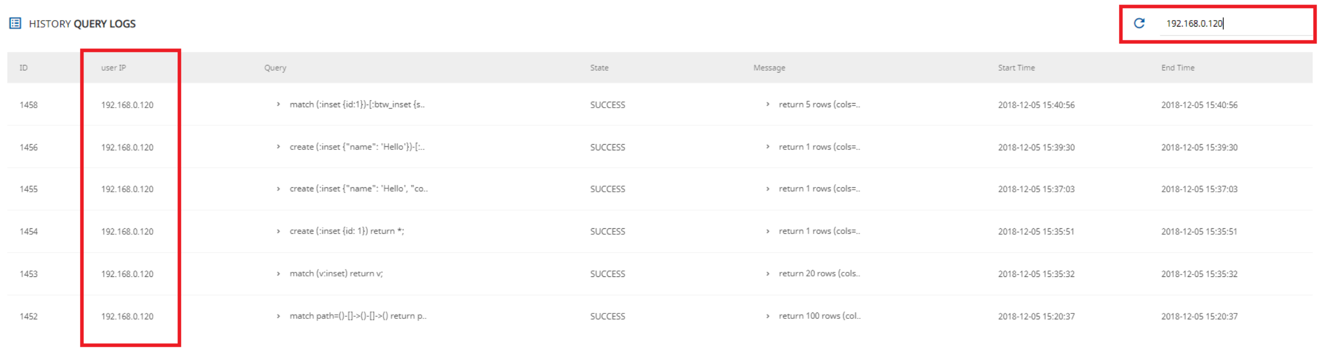
HISTORY **QUERY LOGS**

Type to filter the userIP column...

| ID | user IP | Query | State | Message | Start Time | End Time |
|---|---|---|---|---|---|---|
| 1436 | 192.168. | › create vlabel if not exists new_labe.. | SUCCESS | › 'CREATE::vlabel' aff.. | 2018-12-04 16:31:51 | 2018-12-04 16:31:51 |
| 1435 | 192.168. | › drop vlabel if exists new_label_1; | SUCCESS | › 'DROP::vlabel' affec.. | 2018-12-04 16:26:55 | 2018-12-04 16:26:55 |
| 1434 | 192.168. | › create vlabel if not exists new_labe.. | SUCCESS | › 'CREATE::vlabel' aff.. | 2018-12-04 16:26:41 | 2018-12-04 16:26:41 |
| 1433 | 192.168. | › match path=()-[]->()-[]->() return p.. | SUCCESS | › return 100 rows (col.. | 2018-12-04 15:28:33 | 2018-12-04 15:28:33 |
| 1432 | 192.168. | › match path=()-[]->()-[]->() return p.. | SUCCESS | › return 100 rows (col.. | 2018-12-04 15:28:26 | 2018-12-04 15:28:26 |
| 1431 | 192.168. | › match path=()-[]->()-[]->() return p.. | SUCCESS | › return 100 rows (col.. | 2018-12-04 15:25:57 | 2018-12-04 15:25:57 |
| 1430 | 192.168. | › match path=()-[]->()-[]->() return p.. | SUCCESS | › return 100 rows (col.. | 2018-12-04 15:25:47 | 2018-12-04 15:25:47 |
| 1429 | 192.168. | › match path1=(c:customer)-[]->(:"orde.. | SUCCESS | › return 22 rows (cols.. | 2018-12-04 14:56:54 | 2018-12-04 14:56:54 |
| 1428 | 192.168. | › match path1=(c:customer)-[]->(:"orde.. | SUCCESS | › return 22 rows (cols.. | 2018-12-04 14:41:55 | 2018-12-04 14:41:55 |

358 total

Performed queries can be searched by User IP.

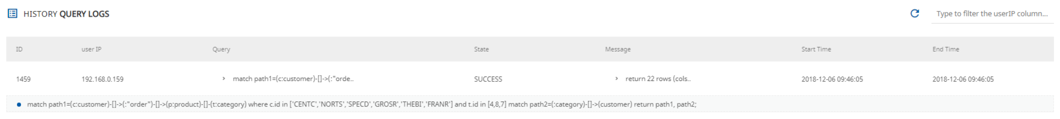


HISTORY **QUERY LOGS**

192.168.0.120

| ID | user IP | Query | State | Message | Start Time | End Time |
|---|---|---|---|---|---|---|
| 1458 | 192.168.0.120 | › match (:inset {id:1})-[:btw_inset {s.. | SUCCESS | › return 5 rows (cols=.. | 2018-12-05 15:40:56 | 2018-12-05 15:40:56 |
| 1456 | 192.168.0.120 | › create (:inset {"name": 'Hello'})-[:.. | SUCCESS | › return 1 rows (cols=.. | 2018-12-05 15:39:30 | 2018-12-05 15:39:30 |
| 1455 | 192.168.0.120 | › create (:inset {"name": 'Hello', "co.. | SUCCESS | › return 1 rows (cols=.. | 2018-12-05 15:37:03 | 2018-12-05 15:37:03 |
| 1454 | 192.168.0.120 | › create (:inset {id: 1}) return *; | SUCCESS | › return 1 rows (cols=.. | 2018-12-05 15:35:51 | 2018-12-05 15:35:51 |
| 1453 | 192.168.0.120 | › match (v:inset) return v; | SUCCESS | › return 20 rows (cols.. | 2018-12-05 15:35:32 | 2018-12-05 15:35:32 |
| 1452 | 192.168.0.120 | › match path=()-[]->()-[]->() return p.. | SUCCESS | › return 100 rows (col.. | 2018-12-05 15:20:37 | 2018-12-05 15:20:37 |

To see the details of a query, click Expand/Collapse Row( › ) button.



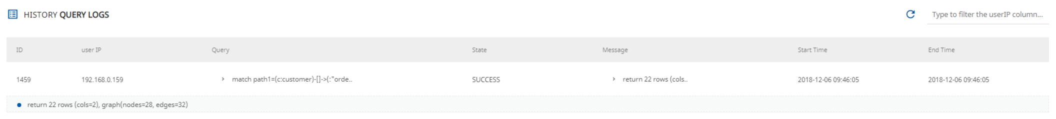
HISTORY **QUERY LOGS**

Type to filter the userIP column...

| ID | user IP | Query | State | Message | Start Time | End Time |
|---|---|---|---|---|---|---|
| 1459 | 192.168.0.159 | › match path1=(c:customer)-[]->(:"orde.. | SUCCESS | › return 22 rows (cols.. | 2018-12-06 09:46:05 | 2018-12-06 09:46:05 |

```
● match path1=(c:customer)-[]->(:"order")-[]->(p:product)-[]-(t:category) where c.id in ['CENTC','NORTS','SPECD','GROSR','THEBI','FRANR'] and t.id in [4,8,7] match path2=(:category)-[]->(customer) return path1, path2;
```

To see the details of the query result, click Expand/Collapse Row( › ) button.



HISTORY **QUERY LOGS**

Type to filter the userIP column...

| ID | user IP | Query | State | Message | Start Time | End Time |
|---|---|---|---|---|---|---|
| 1459 | 192.168.0.159 | › match path1=(c:customer)-[]->(:"orde.. | SUCCESS | › return 22 rows (cols.. | 2018-12-06 09:46:05 | 2018-12-06 09:46:05 |

```
● return 22 rows (cols=2), graph(nodes=28, edges=32)
```

## Documents Menu

For more information on how to use AgensBrowser, please refer to the AgensBrowser documents available from the Bitnine's website.

## LiveShare For Report

Provides functions like graph rendering, simple graph editing, label list output, and attribute data output.

### Report Function

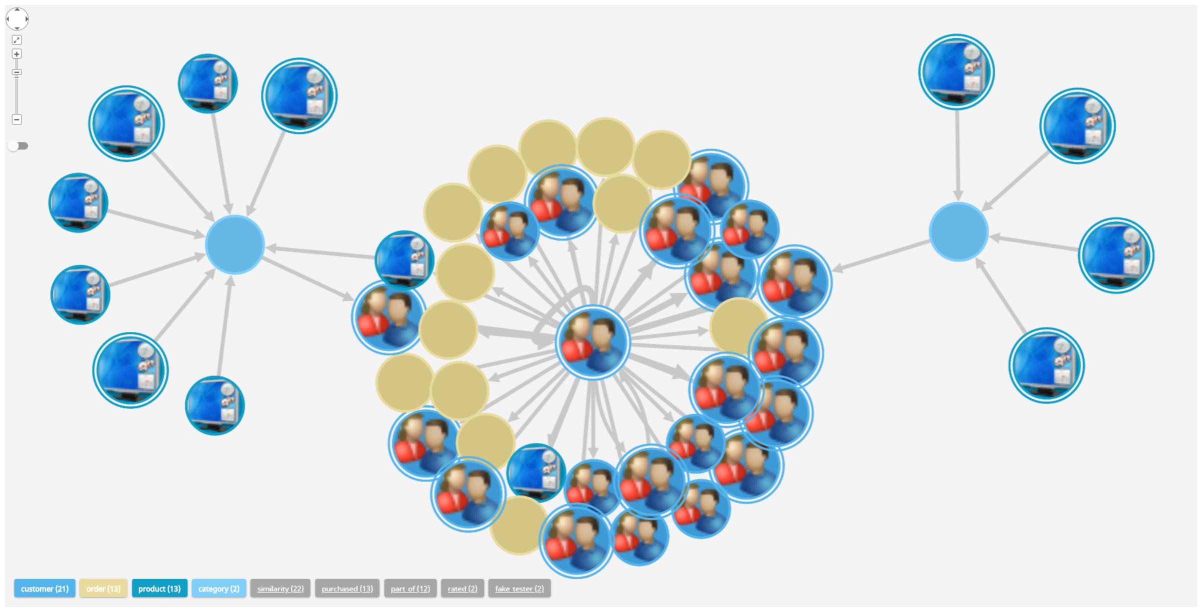
The projects saved by users for reporting can be checked here. See config.yml for guest-key, and you may get project-id from ID of the Load menu in the Graph tab.

```
http://DB_SERVER_IP_ADDRESS:WEB_SERVER_PORT/#/report/GUEST-KEY/PROJECT-ID
```

The following is an example of getting a project report.

```
http://localhost:8085/#/report/agens/1130
```

Here is the result of a Project Report. Right-clicking in the Graph area activates a pop-up menu. For more information on the menu, see Edit/Modify Graph.



## Using HTML Embed in Report

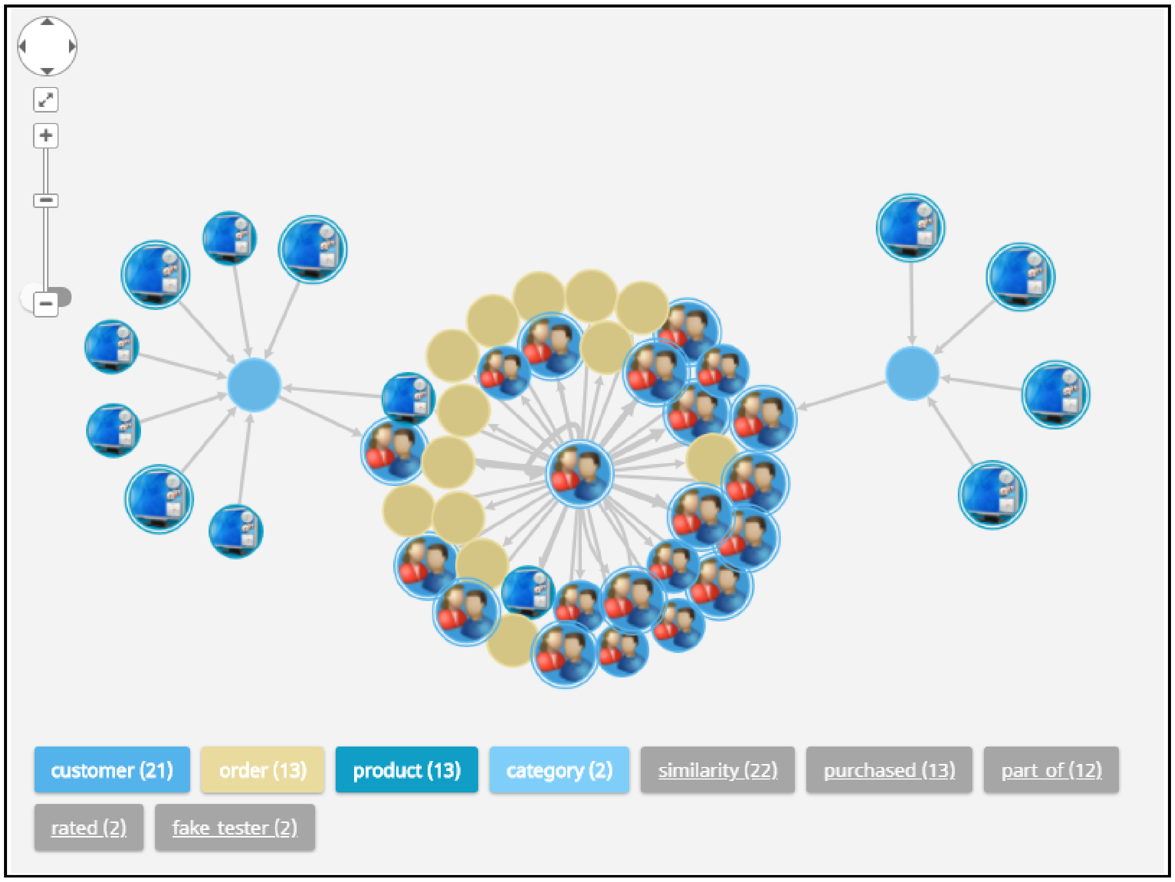
Provides functions to embed user-saved graph visualization without a dedicated visualization tool into other web pages using an embed tag.

```html
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>AgensWeb Embed</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="assets/icons/favicon-bn.ico">
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
<style>
  .border-styles {
    margin: 2px 0;
    padding: 1px 3px;
    border-width: 2px;
    border-color: '#000';
    border-style: solid;
  }
</style>

</head>
<body>
<h2> Agens Browser 2.1 </h2>
<h4> is comming soon. ^^</h4>
<div>
    <embed type="text/html"  class="border-styles"
        src="http://IP:PORT/#/report/agens/1130"  --embed      .
        width="800"
        height="600" />
</div>

</body>
</html>
```

The below is the result of a report using HTML Embed.

## Agens Browser 2.1

**is comming soon. ^^**


customer (21)
order (13)
product (13)
category (2)
similarity (22)
purchased (13)
part_of (12)
rated (2)
fake_tester (2)