

## AgensBrowser Guide

SKAI worldwide Inc.

October 08, 2021

---



## Copyright Notice

**Copyright © 2017-2025, SKAI worldwide Inc. All Rights Reserved.**

*Restricted Rights Legend*

*PostgreSQL is Copyright © 1996-2025 by the PostgreSQL Global Development Group.*

*Postgres95 is Copyright © 1994-5 by the Regents of the University of California.*

*AgensGraph is Copyright © 2017-2025 by SKAI worldwide Inc.*

*Permission to use, copy, modify, and distribute this software and its documentation for any purpose, without fee, and without a written agreement is hereby granted, provided that the above copyright notice and this paragraph and the following two paragraphs appear in all copies.*

*IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.*

*THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS-IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.*

## Trademarks

*AgensGraph® is a registered trademark of SKAI worldwide Inc. Other products, titles or services may be registered trademarks of their respective companies.*

## Open Source Software Notice

*Some modules or files of this product are subject to the terms of the following licenses. :  
OpenSSL, RSA Data Security, Inc., AgensGraph Commercial License, Jean-loup Gailly and Mark Adler, Paul Hsieh's hash.*

## Information of technical documentation

*Title : AgensBrowser Guide*

*Published date : 08 11, 2021*

## 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: SKAI worldwide
      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
                                (Rendering performance degrades
                                if set to True, False is recommended by
                                default)

    output:
        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,"description":"","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,"group":"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:

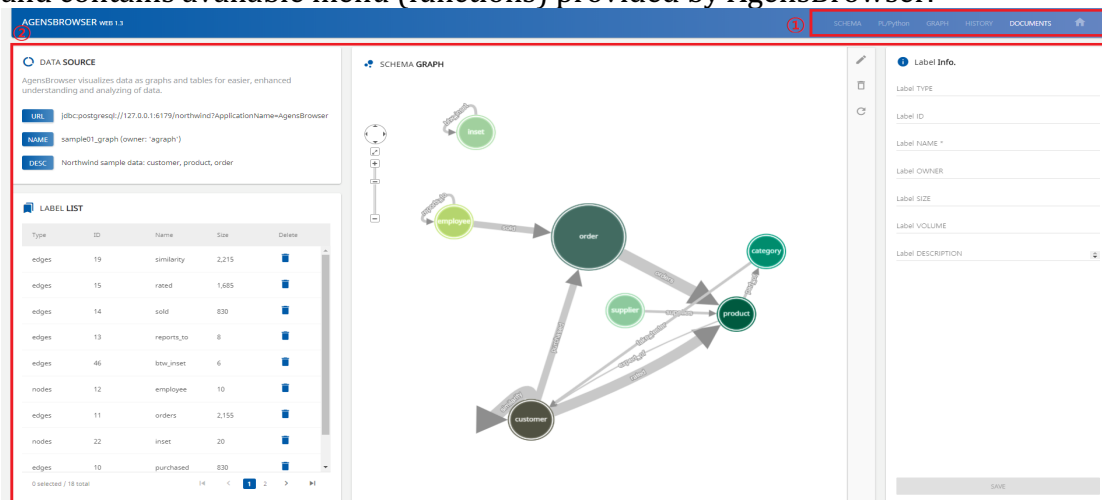
```
http://localhost:8085/index.html
```

## 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



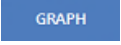


If the login is successful, it redirects you to the main screen as shown below. The main screen shows the information about database connection and meta graph(schema graph) and contains available menu (functions) provided by AgensBrowser.



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   |
|--|---|
|   | To check the connection and metagraph information.                        |
|   | To create PL/Python functions and check the list.                         |
|   | To run Query Editor, Query Output, Graph Visualization, Project Save etc. |
|   | To check the query logs.  |
|   | To refer to the AgensBrowser manual.                                      |
| <ul style="list-style-type: none"> <li>• Main Area (②)<br/>The work space that provides control/monitoring information of the tasks performed through the menu.</li> </ul> |   |

## Functions Detail

### Schema Menu

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

The screenshot displays the AgensGraph Schema Menu interface, which is divided into four main panels:


- DATA SOURCE:** Contains fields for URL, NAME, and DESC. The URL is set to `jdbc:postgresql://127.0.0.1:5432/northwind?ApplicationName=AgensBrowser`. The NAME is `sample01_graph (owner: 'agraph')`. The DESC is `Northwind sample data: customer, product, order`.
- LABEL LIST:** A table listing graph labels with columns for Type, ID, Name, Size, and Delete. The table contains 19 entries, including edges like 'similarity', 'reports\_to', 'orders', 'supplies', 'expert\_of' and nodes like 'product', 'order', 'supplier', 'sold'.
- SCHEMA GRAPH:** A visual representation of the graph schema showing nodes (customer, order, product, category, supplier, employee) and their relationships (edges) with labels like 'reports\_to', 'supplies', 'expert\_of', 'sold', 'likes', 'works\_for'.
- Label Info:** A panel for editing the 'Label Info' of a selected label. It includes fields for Label TYPE, Label ID, Label NAME, Label OWNER, Label SIZE, Label VOLUME, and a dropdown for Label DESCRIPTION. A SAVE button is at the bottom.

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

 < 1 2 > 

- 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

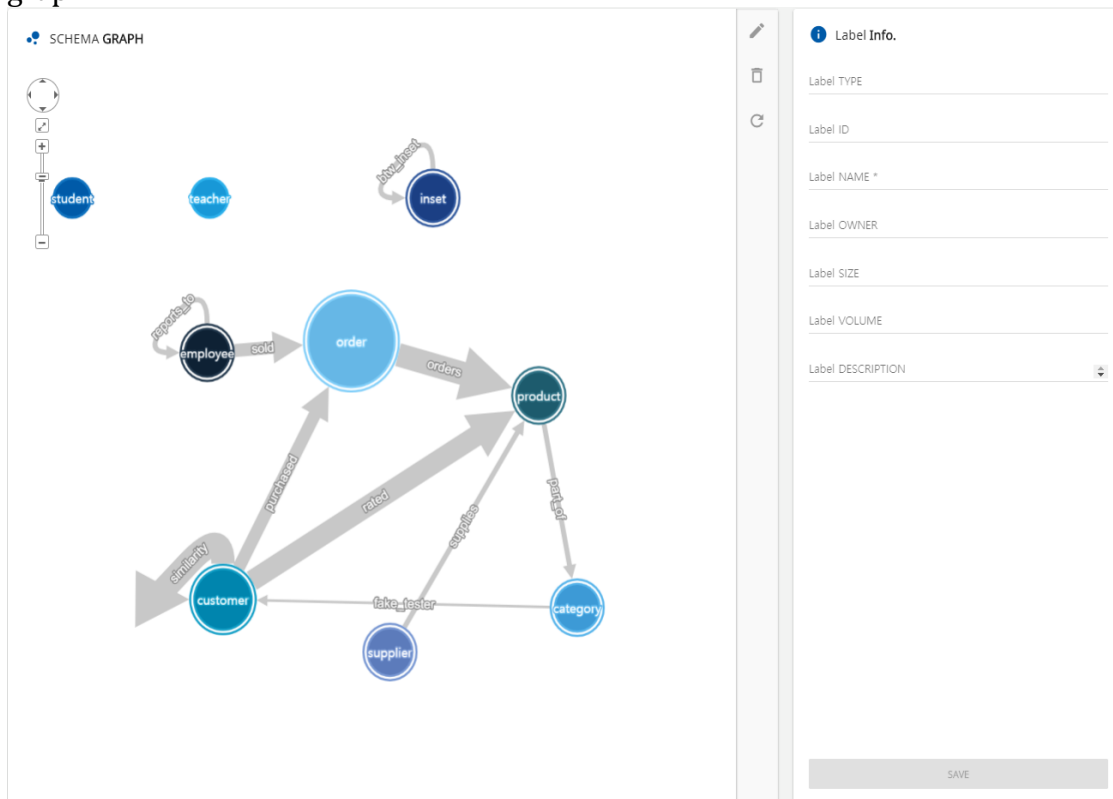
ID  
3

NAME  
product

SIZE  
77

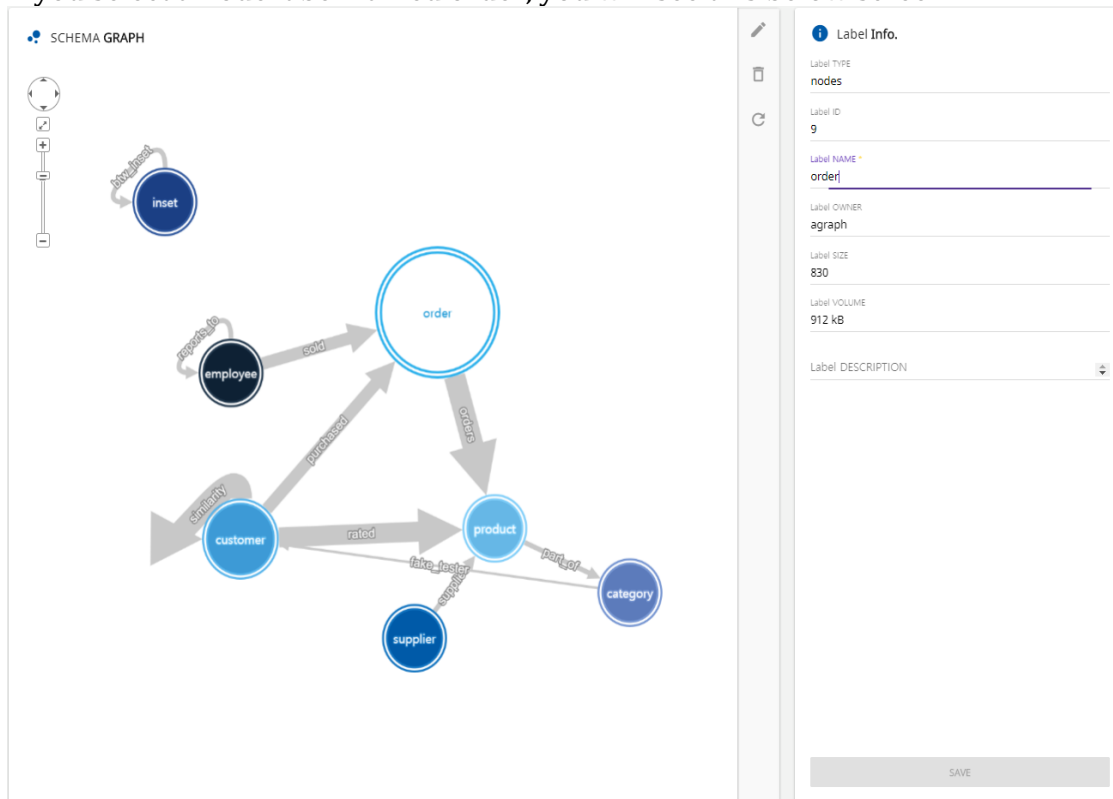
### Schema Graph & Label Info

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:



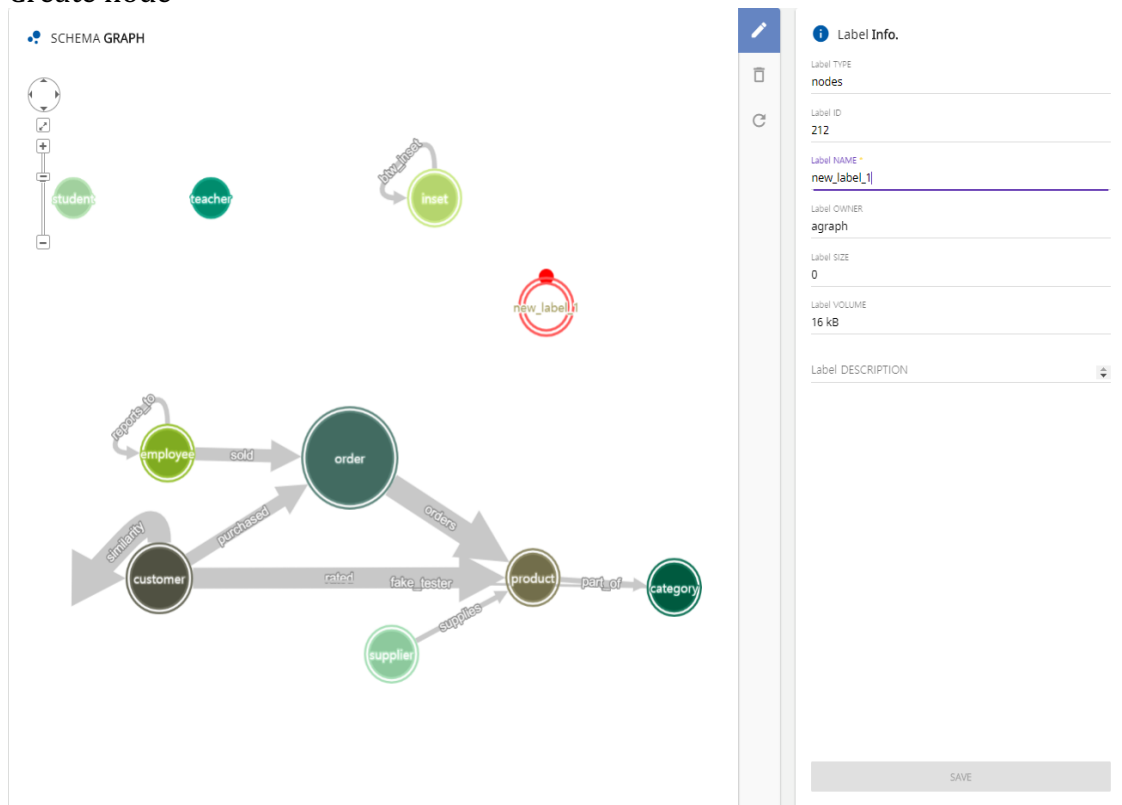
- Edit a Label

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.

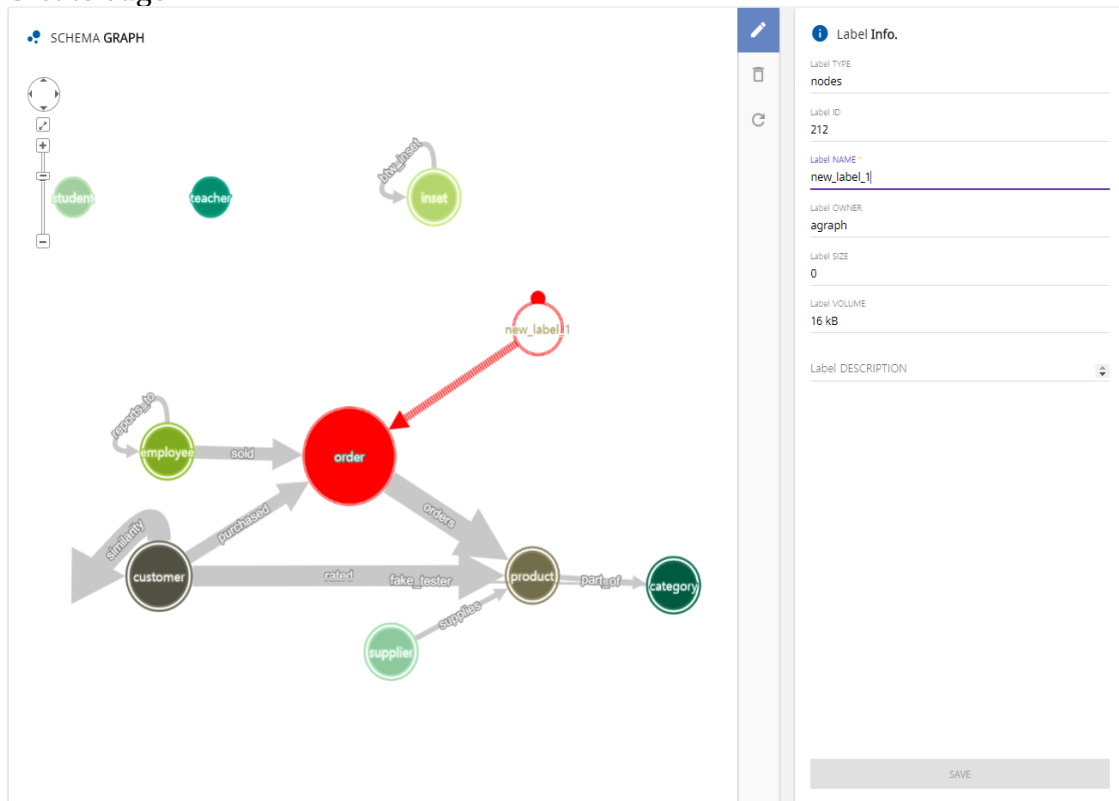
- Create a Label

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




- Create edge



- 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 AgensGraph database and create functions.

The screenshot displays the AgensBrowser interface for PL/Python. On the left, the 'PL/Python Functions List' panel shows a table of functions:

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

Below the table, it indicates '0 selected / 4 total'. At the bottom left, a status bar shows 'SQL Messages: functions.count = 4'.

On the right, the 'PL/Python Editor' panel is shown. It includes fields for 'Name \*' and 'Language \*', and labels for 'Arguments <name,type>', 'Return <type>', and 'Leave a comment'. A large text area for the function source is visible, with a line number '1' on the left. A red error message at the bottom states: 'Error: 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.

 PL/Python Functions List

| 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(  ).

PL/Python Editor

Name \*

pymax

Language \*

plpython2u

Arguments <name,type>

a int, b int

Return <type>

integer

Leave a comment

pl/python test

1

if a > b:

2

return a

3


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

1

if a > b:

2

return a

3

return b

<> Query Editor (Cypher Or Ansi-SQL)

1

select pymax (1, 2);

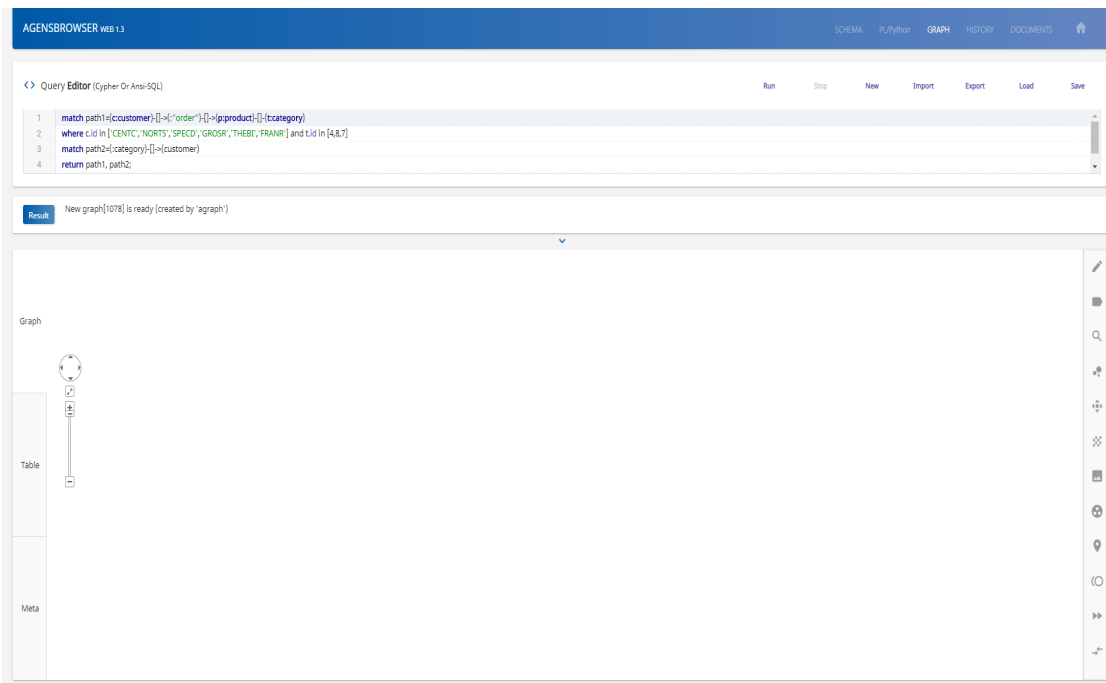
Result

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.



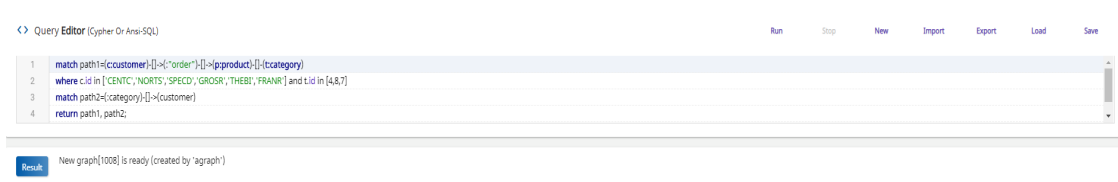
Query to Graph DB consists of:

- 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 .



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.  

The screenshot shows a 'Result' button followed by the text '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 [GraphSON-Format](<https://github.com/thinkaurelius/faunus/wiki/GraphSON-Format>) for more information.

- 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

Q

Type to filter the title column

| 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

Preview

Close

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](#) )

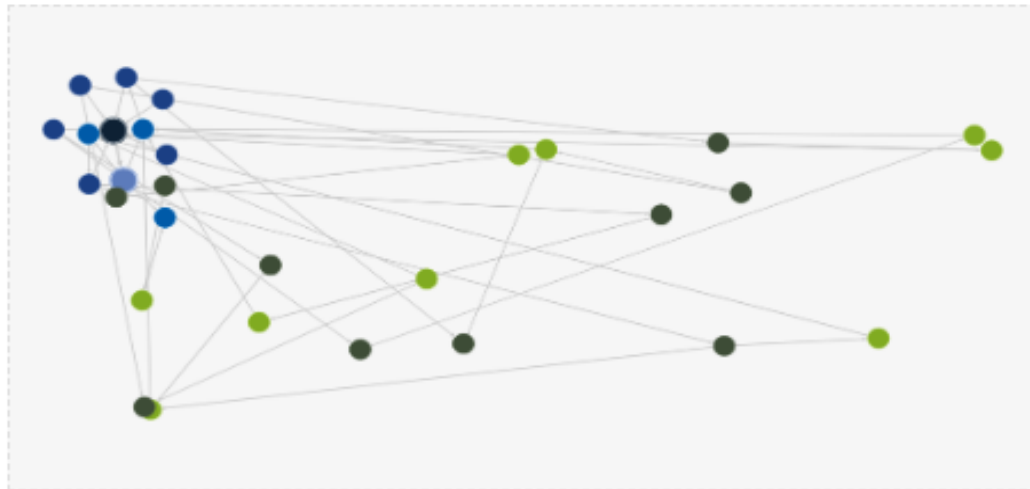
You may save the result of a query that has been performed. Press Save button to save the project as shown below. Title is a required field, and you may add a

description of the project optionally in Description.



Project **Save**

Type title and description about your project



ID

Title \*

Description

Create Date

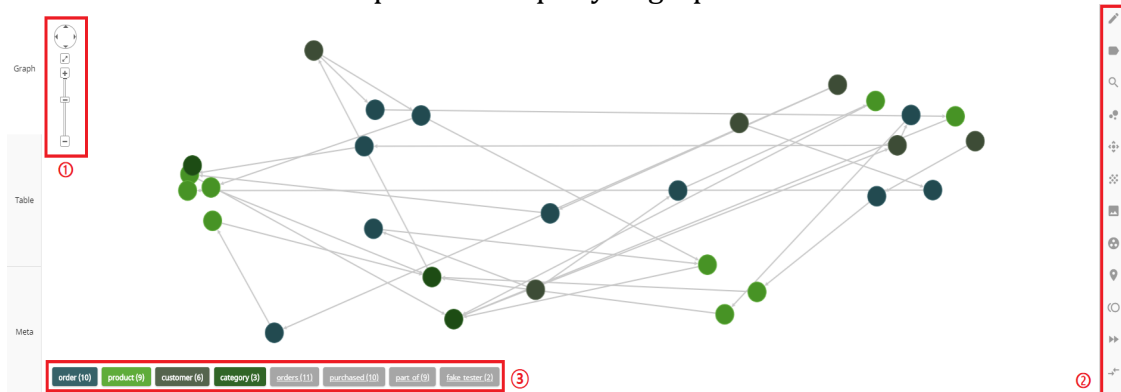
Update Date

Submit

Cancel


## Graph Tab

Visualizes the result of the performed query in graph.



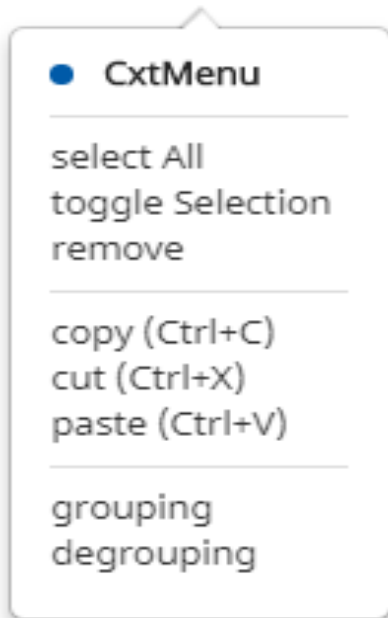
The following tasks can be done in the graph editor.

- 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.



#### <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.



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


IMAGE

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

1

2


3

Add Property
Update



## 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

Add Property
Update



EDGES Editor

• ID

8a16ea7d-c9e8-444c-9b95-

• Source-Node ID

\_ld\_amLct

• Target-Node ID

4,5

• Label

label name

• Properties

| key | value | type | del |
|-----|-------|------|-----|
|-----|-------|------|-----|

• No data to display

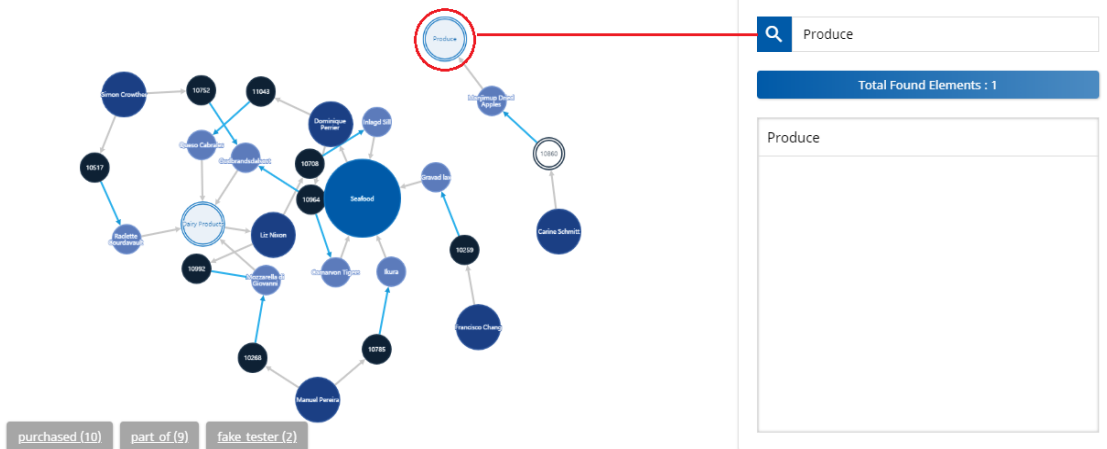
0 total

Add Property

Update

• 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.

Search In Result

• Label Info.

Label nodes [count=10]

Name order

Properties (17)

- \$ALL (for groupby)
- shippostalcode (string)
- shipcountry (string)
- orderid (number) ☒
- freight (number)

+ Group + Filter Refresh

• Filter By List

order  
orderid (number)  
eq 4

• GroupBy List

order  
orderid (number)

Reload

- 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.

Search In Result

• Label Info.

Label nodes [count=10]

Name order

Properties (17)

- \$ALL (for groupby)
- shippostalcode (string)
- shipcountry (string) ☒
- orderid (number)
- freight (number)
- shipaddress (string)

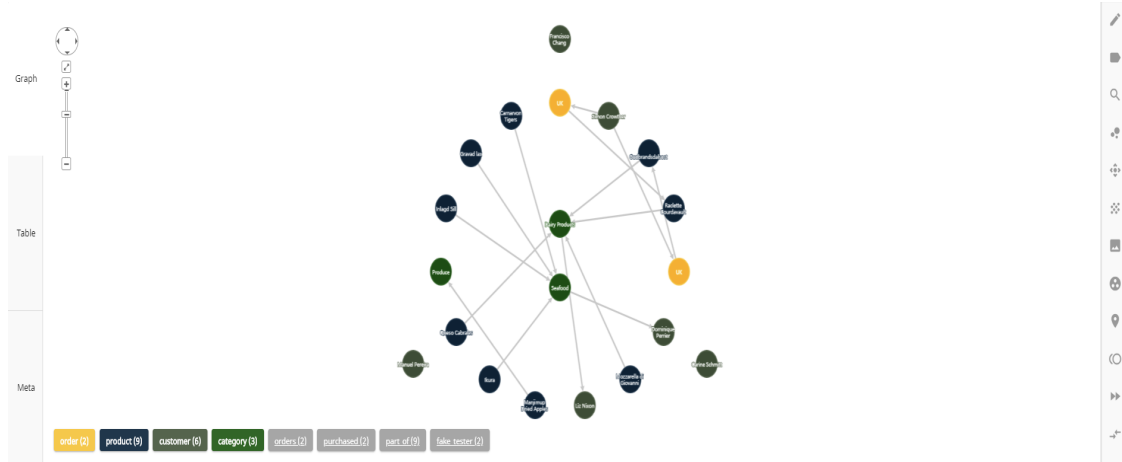
+ Group + Filter Refresh

• Filter By List

order  
shipcountry (string)  
contains UK

• GroupBy List

Reload



- **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.

Search In Result

**Label Info.**

Label: nodes [count=10]

Name: order

Properties: (17)

- \$ALL (for groupby) ☐
- shippostalcode (string) ☐
- shipcountry (string) ☒
- orderid (number) ☐
- freight (number) ☐
- shipaddress (string) ☐

+ Group   + Filter   Refresh

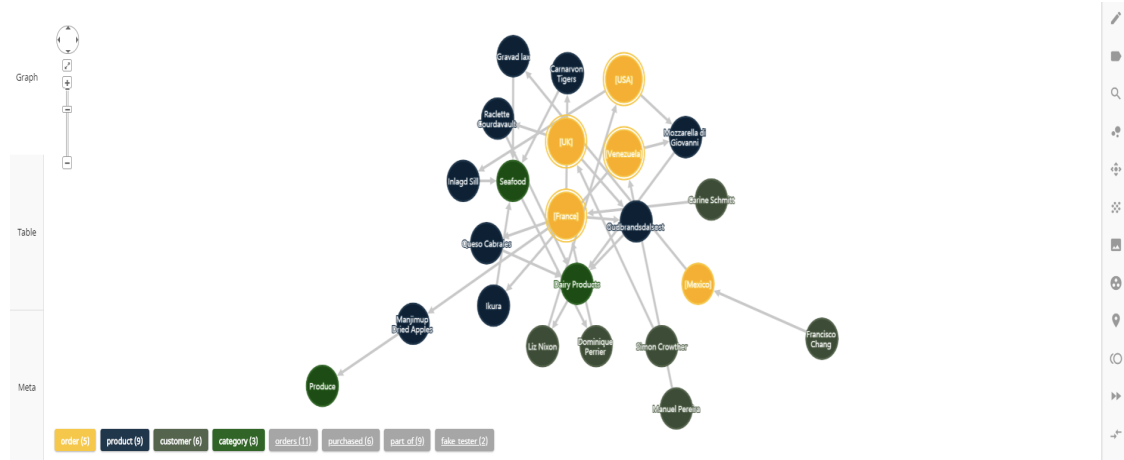
**Filter By List**


**GroupBy List**

order

shipcountry (string)

Reload



- 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.

- **Layout Type**

Random (Default)

Grid

Breadth-First

Concentric

Cola

Cose

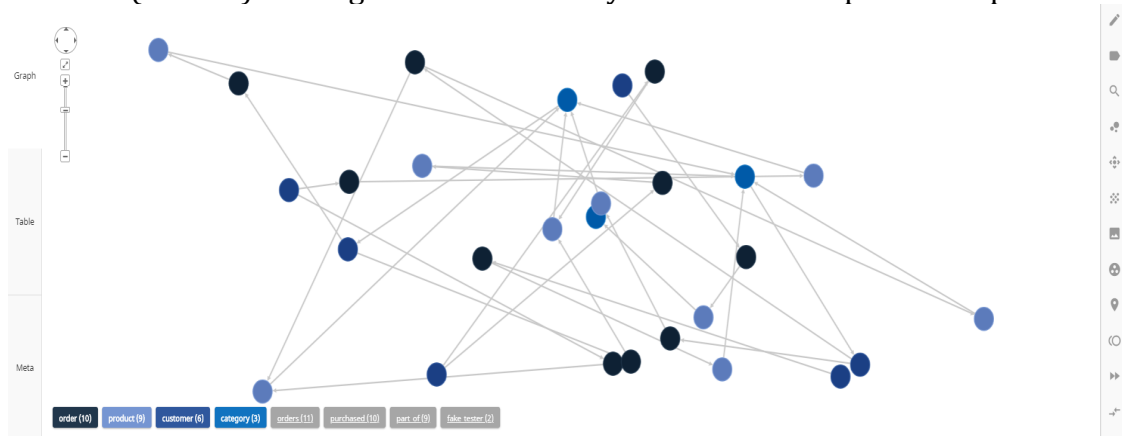
Cose-Bilkent (Slow)

Dagre (Hierachy)

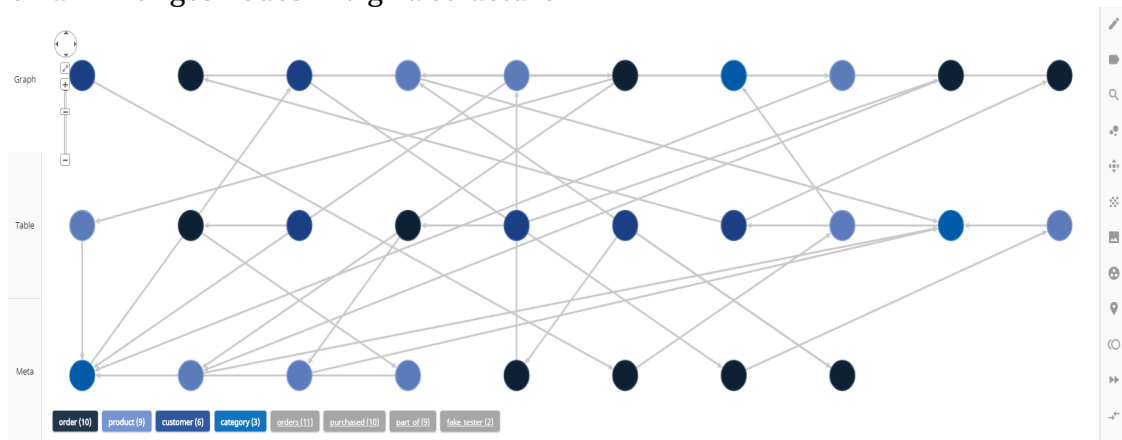
Klay (Hierachy)

Euler (Very Fast)

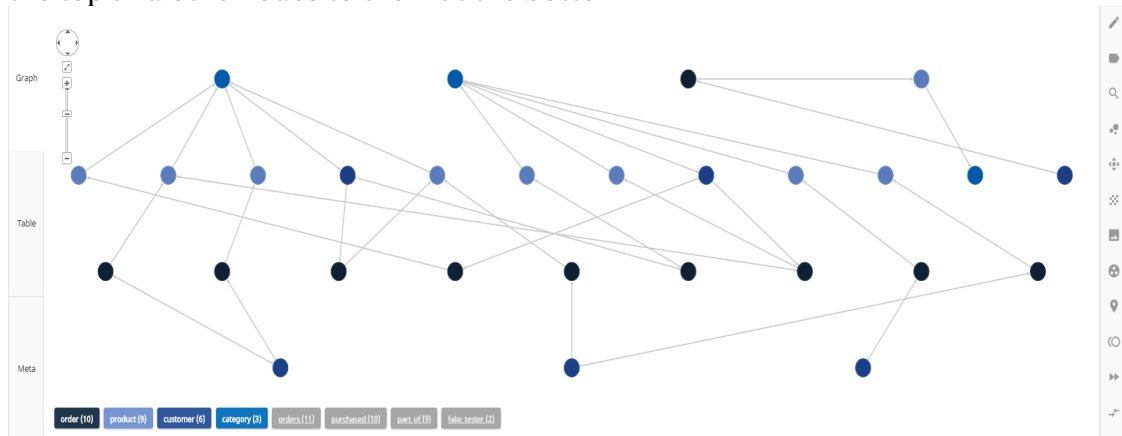
1) Random (Default): Arranges nodes randomly to have an unexpected shape.



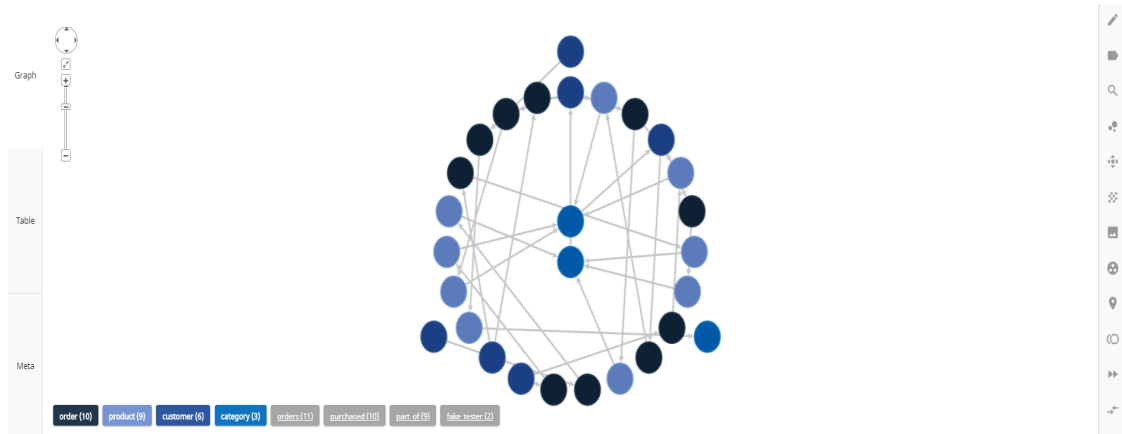
2) Grid : Arranges nodes in a grid structure.



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



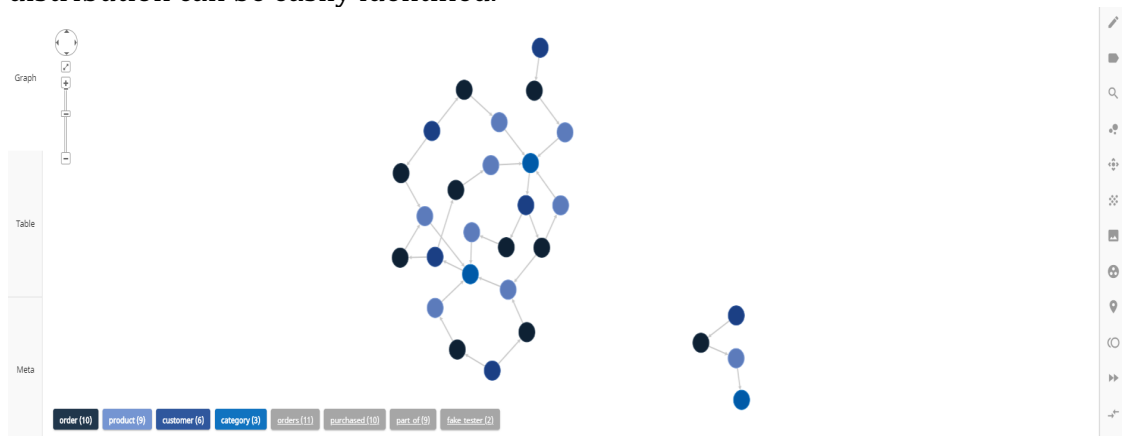
- 4) Concentric : Arranges the nodes with many edges at the center and other nodes connected to them at the outer.



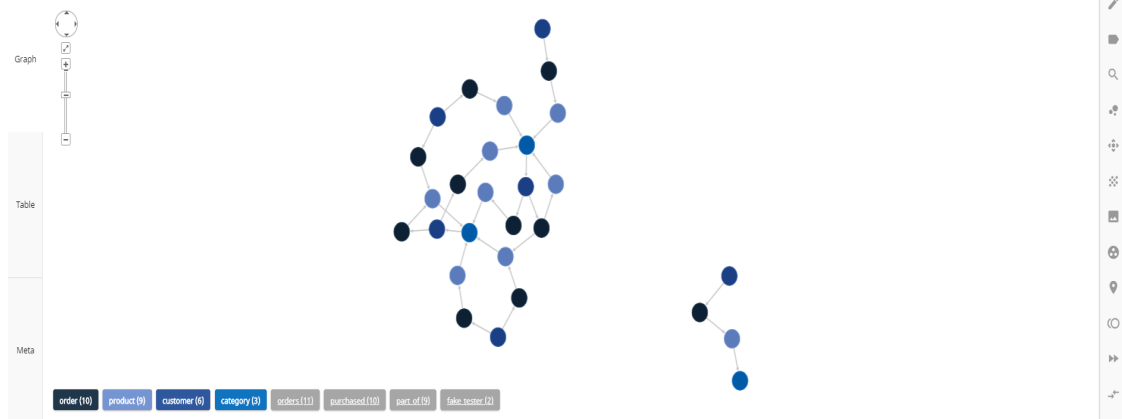
- 5) Cola : Places a node cluster with many connected nodes at the top.



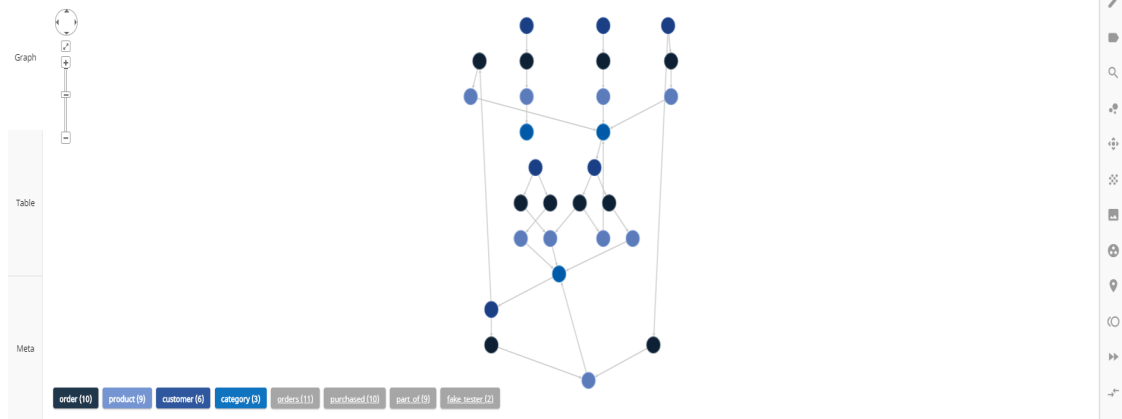
- 6) Cose : Keep a certain distance among the connected node clusters so that the cluster distribution can be easily identified.



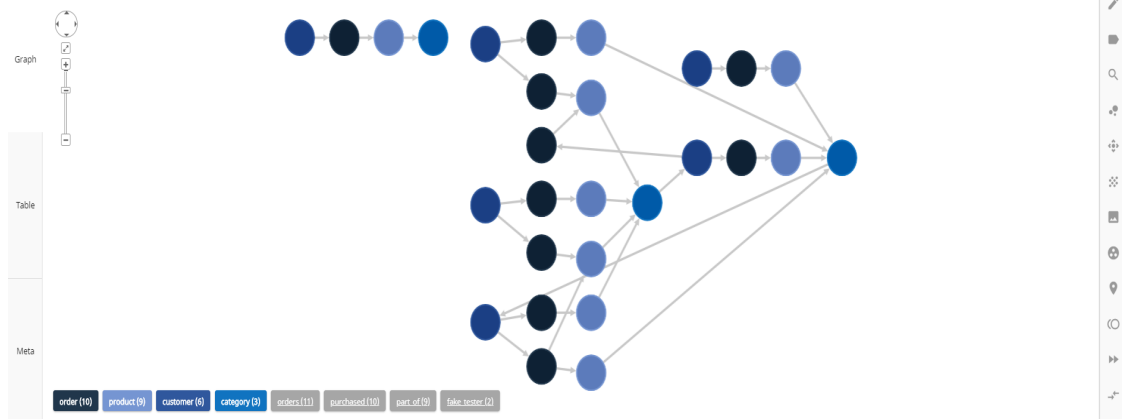
7) Cose-Bilkent (Slow) : A modified version of the Cose layout.



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.

● Centrality Methods

---

Reset Styles

---

Betweenness

---

Closeness

---

Degree

---

PageRank

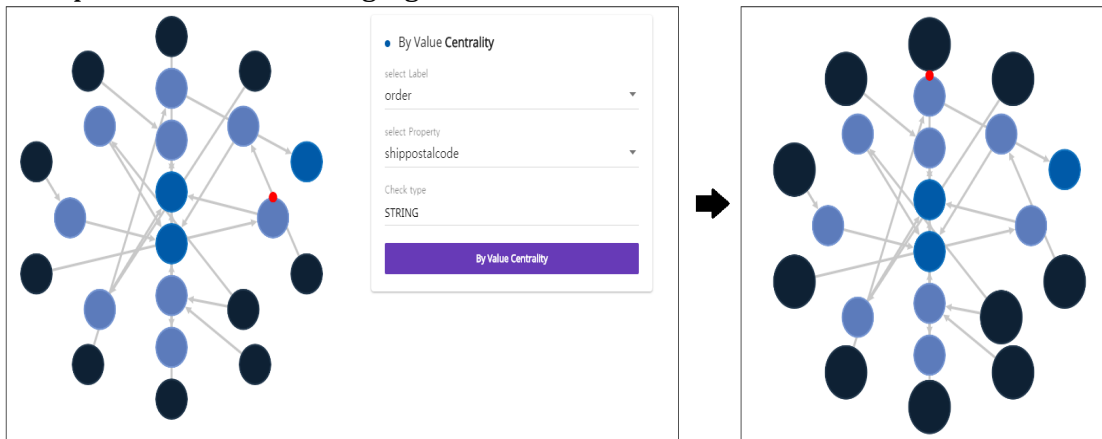
---


By Value

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

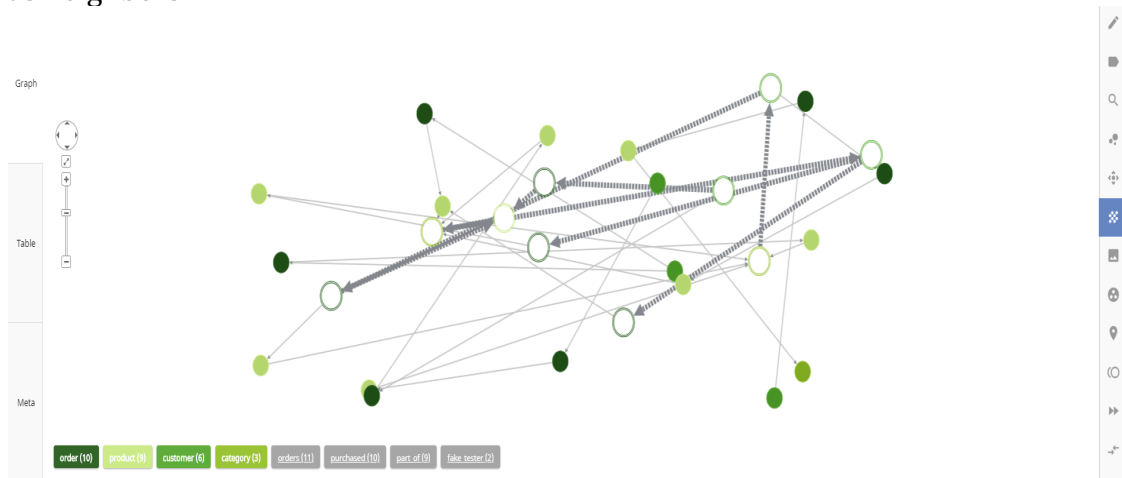
- Betweenness : Centrality considering whether the shortest path between all node pairs on the network passes through a specific node.
- Closeness : Centrality considering how close it is to every node on the network.
- Degree : Centrality considering only the weights of edges directly connected to each 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.



- 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.



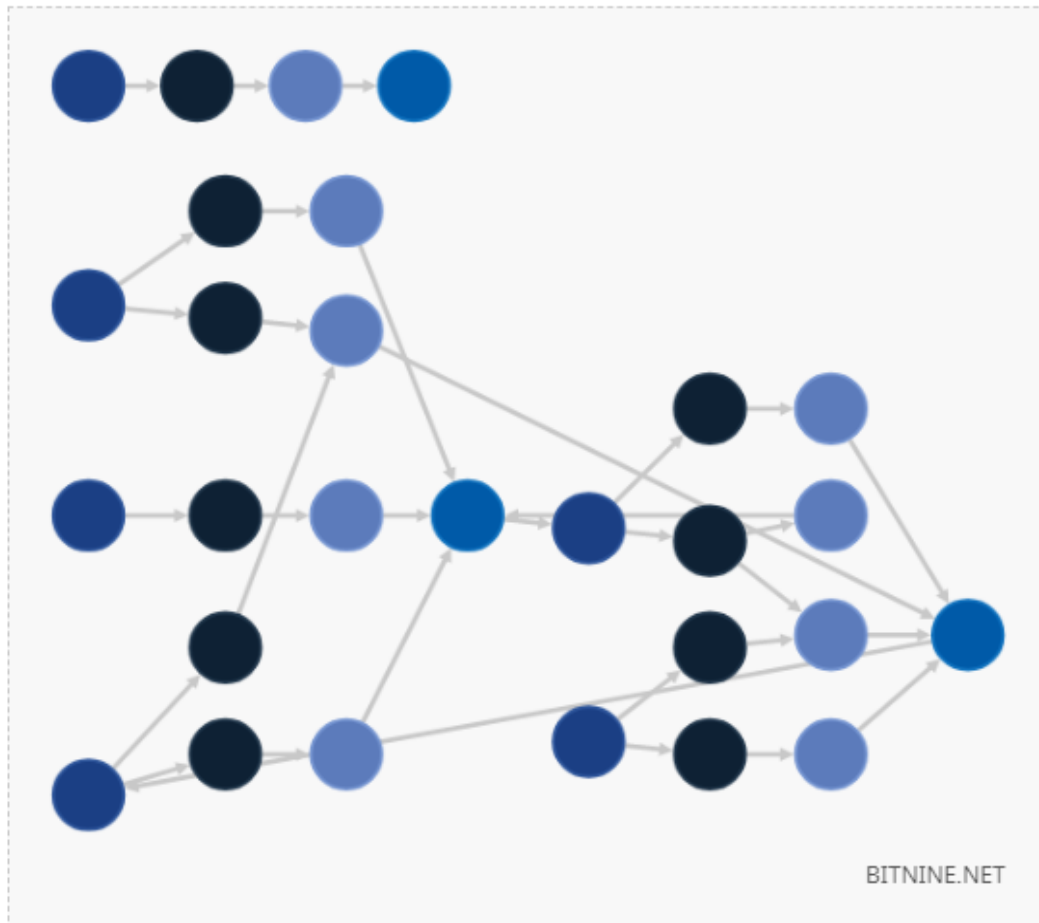
- Image Export (  )  
You can save the result as an image file (PNG); the water-mark is available and, the image is saved with a transparent background so it can be used for presentation

materials.



Image **Export**

Save PNG image file with watermark



Water-mark

BITNINE.NET|

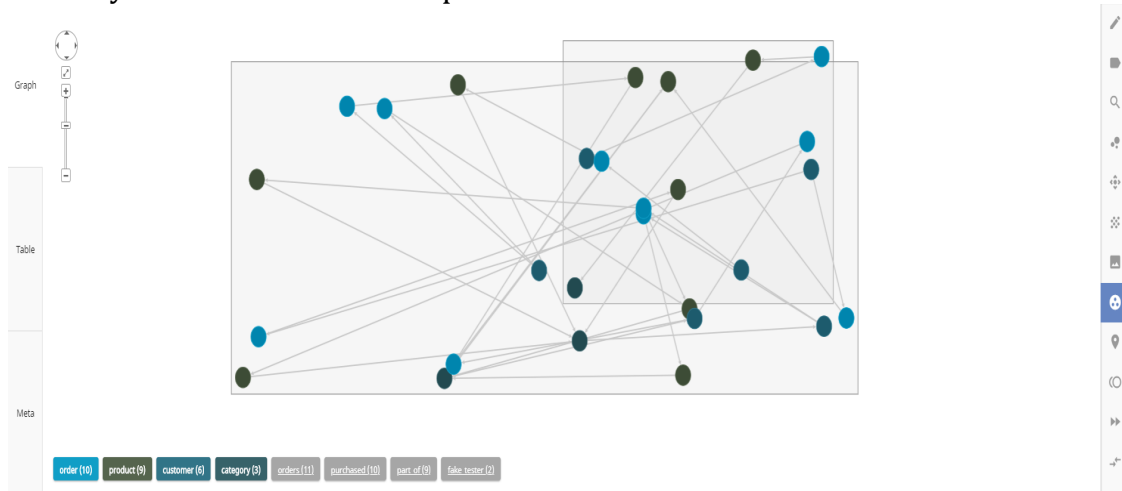




File name \*

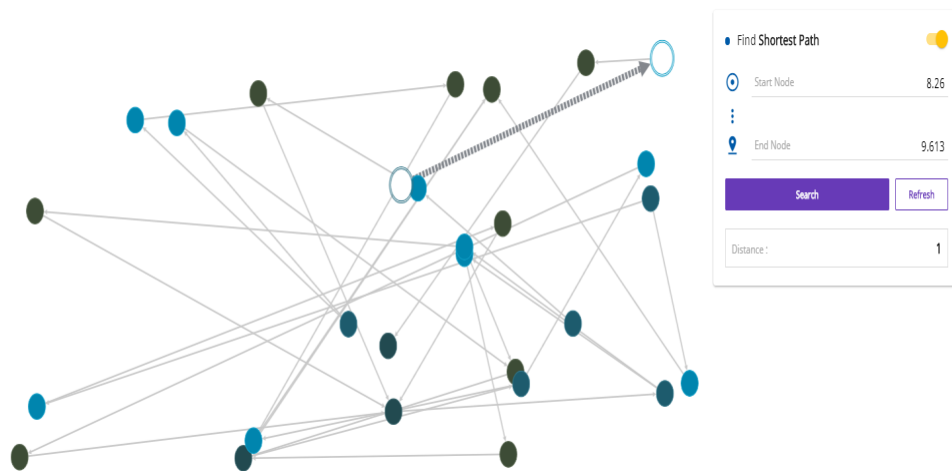
Submit

Cancel

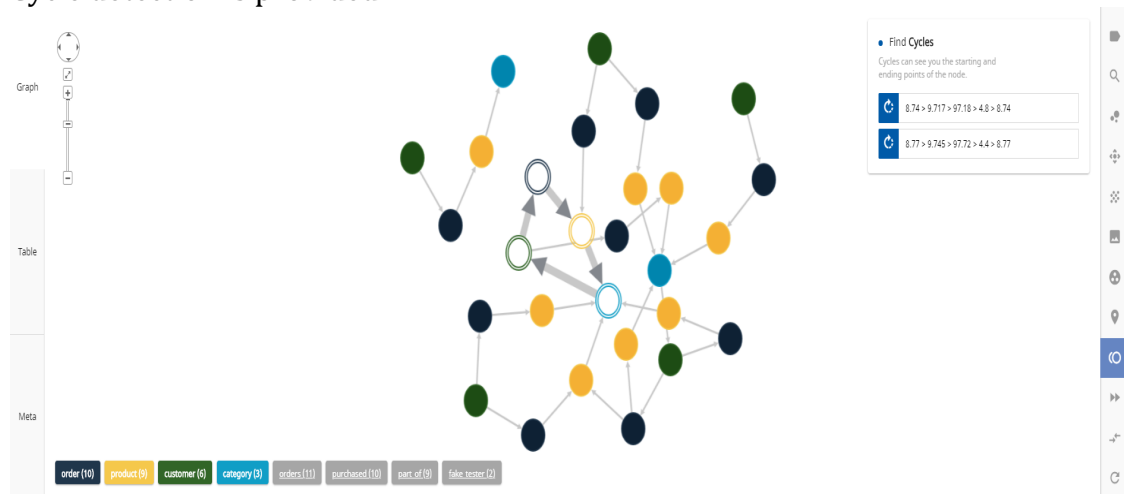
- Find Connected Group (  )  
You may see the connected components.



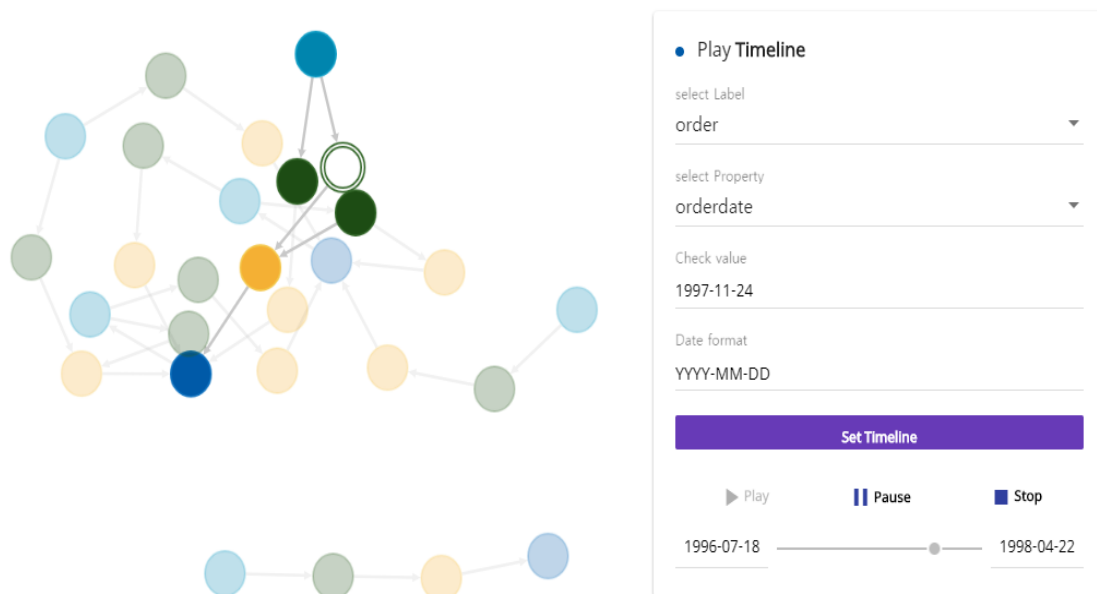
- Find Shortes Path (  )  
You may check the shortest path. Select Start Node and End Node, and then search for the shortest path. The shortest path between the two nodes will be displayed.  
Select Directed Option(  ) to see bidirectional results.



- Find Cycles (🔄)  
Cycle detection is provided.

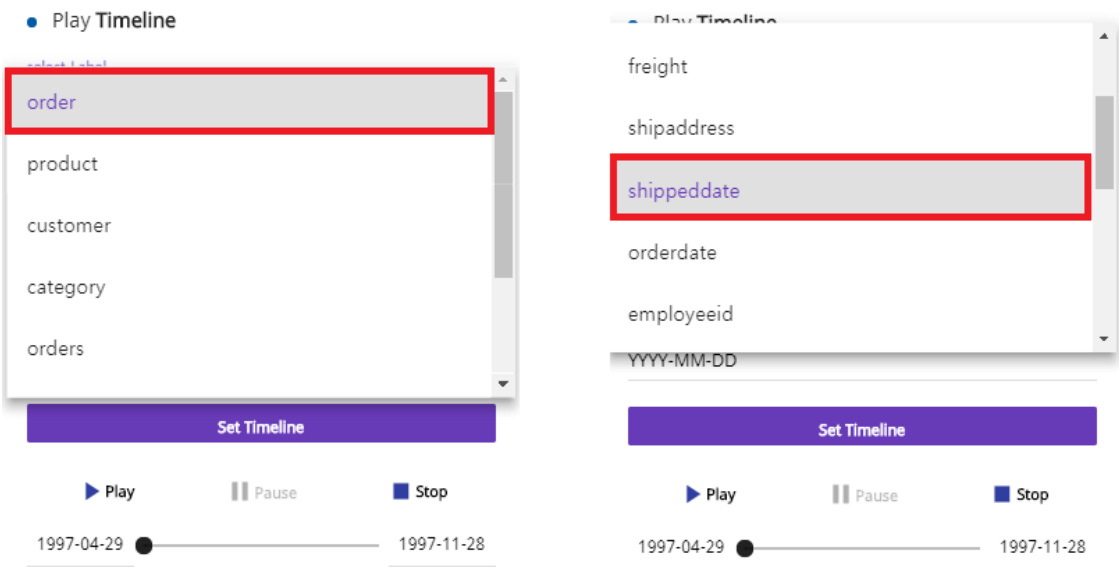


- Play Timeline (▶▶)  
This shows the flow of graph change by Date. Enter the Date or Time format of the selected property. (default is YYYY-MM-DD)




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

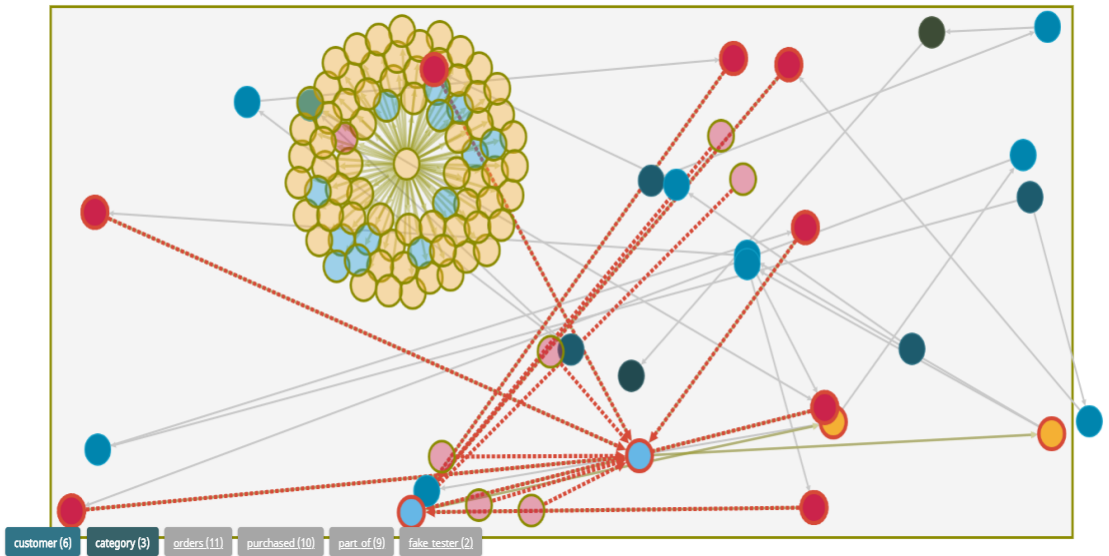
Matching Test

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

OK

Close

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  |
|----------|-------|---|--|
| Graph    | 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... |
| Table    | 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... |
| Meta     | 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 |       | 14 < 1 2 3 > H  |  |

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.



Cell Value

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

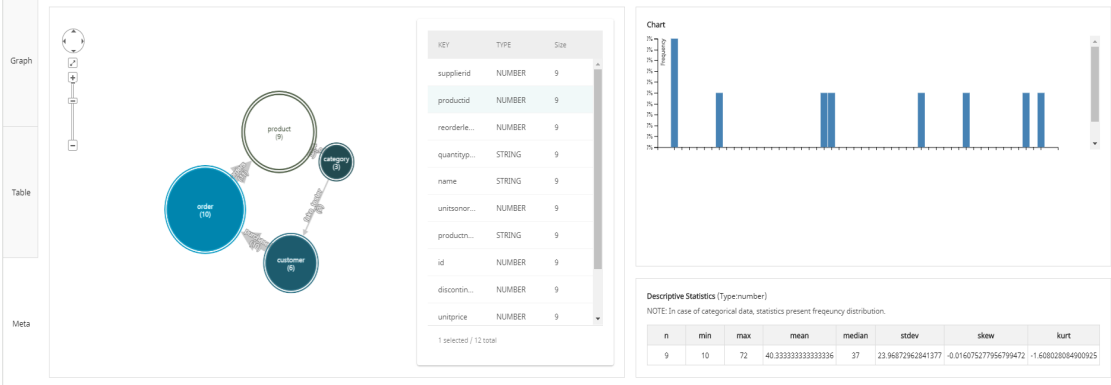
COPY

```
{
  "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.



## History Menu

You can retrieve the history of the performed queries.

| AGENSBROWSER WEB 1.3 |          |  |         |                           |                     |                     |
|----------------------|----------|--|---------|---------------------------|---------------------|---------------------|
|                      |          |  |         | SCHEMA                    | R_Python            | GRAPH               |
|                      |          |  |         | HISTORY                   | DOCUMENTS           |                     |
| HISTORY QUERY LOGS   |          |  |         |                           |                     |                     |
| ID                   | user IP  | Query                                      | Status  | Message                   | Start Time          | End Time            |
| 1436                 | 192.168. | > create vlabel if not exists new_label... | 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...       | 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_label... | 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 |               |   |         |                           |                     |                     |
|--------------------|---------------|---|---------|---------------------------|---------------------|---------------------|
| ID                 | user IP       | Query                                       | Status  | Message                   | Start Time          | End Time            |
| 1458               | 192.168.0.120 | > match (c[inset {id 1}]) {>new_inset {s... | SUCCESS | > return 5 rows (colou... | 2018-12-05 15:40:56 | 2018-12-05 15:40:56 |
| 1456               | 192.168.0.120 | > create (c[inset {"name": "Hello"}]);      | SUCCESS | > return 1 rows (colou... | 2018-12-05 15:39:30 | 2018-12-05 15:39:30 |
| 1455               | 192.168.0.120 | > create (c[inset {"name": "Hello"}, "co... | SUCCESS | > return 1 rows (colou... | 2018-12-05 15:37:03 | 2018-12-05 15:37:03 |
| 1454               | 192.168.0.120 | > create (c[inset {id 1}]) return *;        | SUCCESS | > return 1 rows (colou... | 2018-12-05 15:36:51 | 2018-12-05 15:36:51 |
| 1453               | 192.168.0.120 | > match (c[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   |               |   |         |                           |                     |                     |
|--|---------------|---|---------|---------------------------|---------------------|---------------------|
| ID   | user IP       | Query                                     | Status  | 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)-[])->("orde...") []->() product()-[] (category) where cid in [{"CENTIC","NORTS","SPEED","GRODR","THEBE","FRANR"}] and cid in [{"A,B,T"}] match path2=(category)-[]->("customer") return path1, path2; |               |   |         |                           |                     |                     |

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

| HISTORY QUERY LOGS                                   |               |   |         |                           |                     |                     |
|--|---------------|---|---------|---------------------------|---------------------|---------------------|
| ID   | user IP       | Query                                     | Status  | 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=92) |               |   |         |                           |                     |                     |

## Documents Menu

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

## LiveShare For Report

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

### Report Function

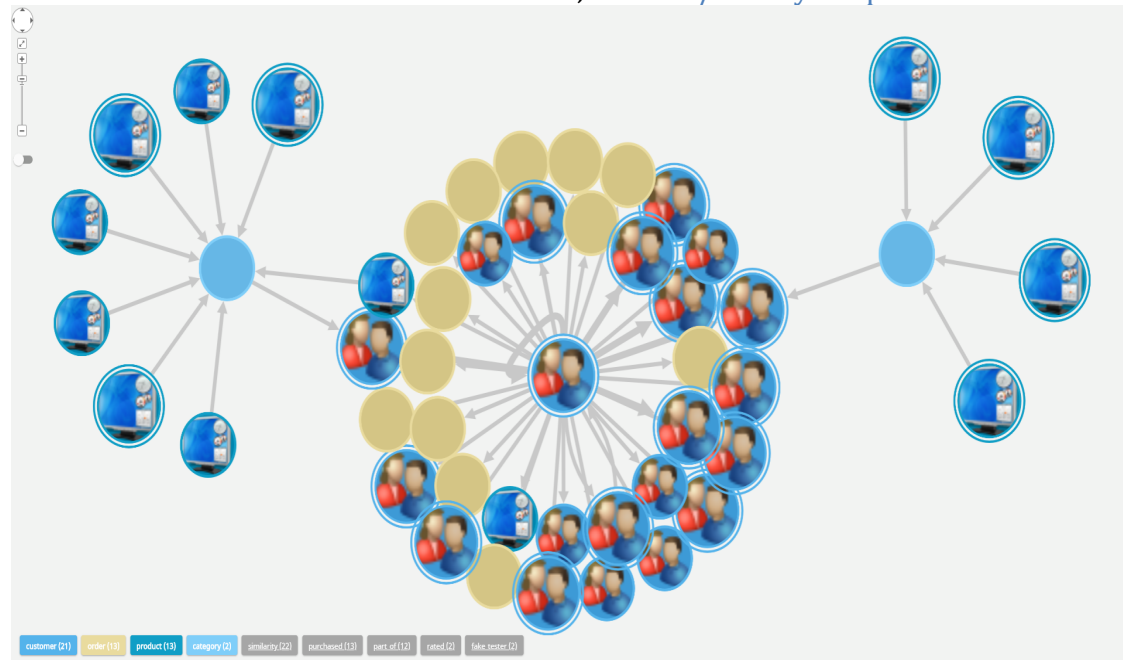
The projects saved by users for reporting can be checked here. See config.yml for guest-key, and you may get project-id from ID of the Load menu in the Graph tab.

```
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/agents/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.

```
<!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 태그를 기입한다.
    width="800"
    height="600" />
</div>

</body>
</html>
```

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

## Agens Browser 2.1

is coming soon. ^^

