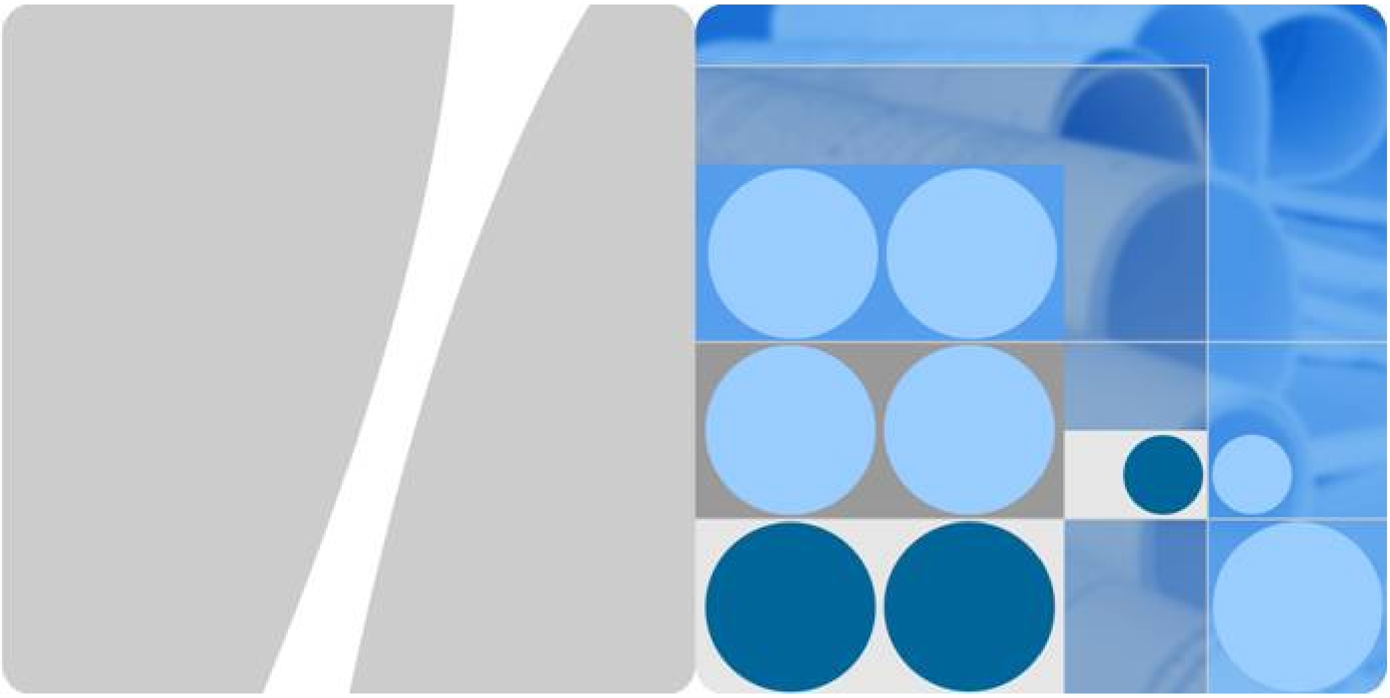


Graph Engine Service

User Guide

Date **2023-03-01**

Contents

1 GES Overview

Graph Engine Service (GES) facilitates query and analysis of multi-relational graph data structures. It is particularly well suited for scenarios requiring analysis of rich relationships, including social network analysis, marketing recommendations, social listening, information distribution, and fraud detection.

This document describes how to operate and analyze graph data on the GES management console.

2 Permissions Management

2.1 Creating a User

If you need to assign different permissions to employees in your enterprise to access GES resources, Identity and Access Management(IAM) is a good choice for fine-grained permissions management.

With IAM, you can:

- Create IAM users for different employees based on the organizational structure of your enterprise. Each IAM user will have their own login credentials for access to GES resources.
- Grant users only the permissions required to perform a given task.
- Entrust a cloud account or cloud service to perform professional and efficient O&M on your GES resources.

If your account does not need individual IAM users, then you may skip over this chapter.

Permission Type

Type

- Roles: A type of coarse-grained authorization mechanism that defines permissions related to user responsibilities. There are only a limited number of roles. When using roles to grant permissions, you need to also assign dependency roles. However, roles are not an ideal choice for fine-grained authorization and secure access control.
- Policies: A type of fine-grained authorization mechanism that defines permissions required to perform operations on specific cloud resources under certain conditions. Policies allow for more flexible permissions control than roles. They allow you to meet requirements for more secure access control. For example, you can grant GES users only the permissions for managing a certain type of cloud servers.

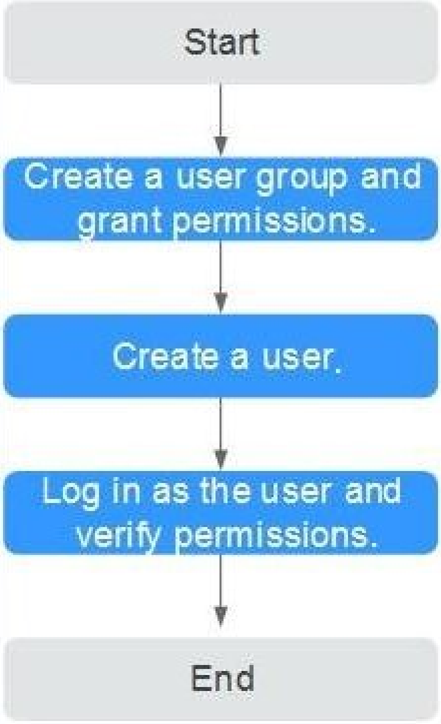
  NOTE

  GES ReadOnlyAccess is a policy.

Procedure

This section describes how to use a group to grant permissions to a user. **Figure 2-1** shows the process.

Figure 2-1 Granting GES permissions


Start
Create a user group and grant permissions.
Create a user.
Log in as the user and verify permissions.
End

1. Create a user group and assign permissions.

   Create a user group on the IAM console, and assign the **GES ReadOnlyAccess** policy to the group.

2. Create a user and add it to a user group.

   Create a user on the IAM console and add the user to the group created in step 1.

3. Log in.

   Log in to the management console using the user your created and verify the user permissions.

   - Choose **Service List** > **Graph Engine Service** to enter the GES management console, and click **Create Graph** in the upper right corner to create a graph. If you cannot create one, the **GES ReadOnlyAccess** policy has taken effect.
   - Choose any other service in **Service List**. If a message appears indicating that you have insufficient permissions to access the service, the **GES ReadOnlyAccess** policy has taken effect.

2.2 Policy Permissions

2.2.1 Policy

IAM supports both system-defined and custom policies.

System-defined Policies

System-defined policies cover various common actions of a cloud service. System-defined policies can be used to assign permissions to user groups, but they cannot be modified.

The system-defined policies for GES include **GES FullAccess**, **GES Development**, and **GES ReadOnlyAccess**. These policies are recommended as they can cover most of the role assignments your will need in most scenarios. For details, see **GES System-defined Policy**.

Custom Policies

If the supplied system policies are unable to meet your needs, you can create custom policies for more refined control. You can create custom policies in the visual editor or using a JSON editor. For details, see **GES Custom Policy**.

2.2.2 System-defined Policies

Table 2-1 GES system-defined policies

Policy Name	Description
GES FullAccess	Administrator permissions for GES. Users granted these permissions can perform all operations on GES, including creating, deleting, accessing, and updating graphs. **NOTE** ● Users with permissions of this policy must also be granted permissions of the **Tenant Guest**, **Server Administrator**, and **VPC Administrator** policies. ● To bind or unbind an EIP, you need the **Security Administrator** permission to create agencies. The **Security Administrator** role has fairly high-level permissions. You can use the following custom policies to replace this role: "iam:agencies:listAgencies","iam:permissions:listRolesForAgency","iam:permissions:listRolesForAgencyOnProject","iam:permissions:listRolesForAgencyOnDomain". ● To use resources stored on OBS for other services, you need the **OBS OperateAccess** permission. OBS is a global service. You can find the corresponding OBS policy in the **Global service project** scope. ● When granting **GES FullAccess** to an enterprise project, you need to configure the following permissions policies in IAM: ● ecs:availabilityZones:list ● ecs:cloudServerNics:update

Policy Name	Description
GES Development	Operator permissions for all operations except creating, deleting, resizing, and expanding graphs. **NOTE** ● To bind or unbind an EIP, you must have the **Security Administrator** permission to create agencies. The **Security Administrator** role can be replaced by the following custom policies: "iam:agencies:listAgencies","iam:permissions:listRolesForAgency","iam:permissions:listRolesForAgencyOnProject","iam:permissions:listRolesForAgencyOnDomain". ● To use resources stored on OBS for other services, you need the **OBS OperateAccess** permission. OBS is a global service. You can find the corresponding OBS policy in the **Global service project** scope.
GES ReadOnlyAccess	Read-only permissions for viewing resources, such as graphs, metadata, and backup data. **NOTE** To use resources stored on OBS for other services, you need the **OBS OperateAccess** permission. OBS is a global service. You can find the corresponding OBS policy in the **Global service project** scope.

NOTE

It takes about 13 minutes for an OBS role to take effect after being applied to a user or group. A policy takes about 5 minutes.

Table 2-2 Common operations supported by each system-defined policy

Operation	GES FullAccess	GES Development	GES ReadOnlyAccess	Resource
Querying the graph list	Yes	Yes	Yes	-
Querying graph details	Yes	Yes	Yes	graphName
Creating graphs	Yes	No	No	graphName
Accessing graphs	Yes	Yes	No	graphName
Stopping graphs	Yes	Yes	No	graphName
Starting graphs	Yes	Yes	No	graphName
Deleting graphs	Yes	No	No	graphName

Operation	GES FullAccess	GES Development	GES ReadOnlyAccess	Resource
Importing Incremental data to graphs	Yes	Yes	No	graphName
Exporting graphs	Yes	Yes	No	graphName
Clearing graphs	Yes	Yes	No	graphName
Upgrading graphs	Yes	Yes	No	graphName
Resizing a Graph	√	No	No	graphName
Expanding a Graph	√	No	No	graphName
Restarting a Graph	√	Yes	No	graphName
Binding EIPs	Yes	Yes	No	graphName
Unbinding an EIP	Yes	Yes	No	graphName
Querying backups of all graphs	Yes	Yes	Yes	-
Querying backups of a graph	Yes	Yes	Yes	-
Adding backups	Yes	Yes	No	backupName
Deleting a graph backup	Yes	Yes	No	backupName
Querying the metadata list	Yes	Yes	Yes	-
Querying metadata	Yes	Yes	Yes	metadataName
Verifying metadata	Yes	Yes	No	-
Adding metadata	Yes	Yes	No	metadataName
Deleting metadata	Yes	Yes	No	metadataName
Querying task statuses	Yes	Yes	Yes	-

Operation	GES FullAccess	GES Development	GES ReadOnlyAccess	Resource
Querying the task list	Yes	Yes	Yes	-

2.2.3 Custom Policies

In addition to the system-defined policies of GES, you can also create your own custom policies.

You can create custom policies using the visual editor or by editing a JSON file:

- Visual editor: Just select the relevant cloud services, actions, resources, and request conditions. You do not need to understand policy syntax.
- JSON: You can create a policy using a JSON file or edit the JSON file for an existing policy.

The following section contains examples of common GES custom policies.

Examples

- Example 1: Allowing users to query and operate graphs

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                     "ges:*:get*",
                     "ges:*:list*",
                     "ges:graph:operate"
            ]
        }
    ]
}
```

- Example 2: Preventing graph deletion

  A deny policy must be used in conjunction with other policies to take effect. If the policies assigned to a user contain both "Allow" and "Deny", the "Deny" permissions take precedence over the "Allow" permissions.

  If you need to assign the **GES FullAccess** policy to a user but also forbid that user from deleting graphs, you can create a custom policy that blocks graph deletion, and then assign both policies to the group the user belongs to. The user will be granted full access based on the system policy, but the custom policy will then override the permission allowing graph deletion. The following is an example of a deny policy:

```
{
      "Version": "1.1",
      "Statement": [
            {
          "Effect": "Deny",
                  "Action": [
                        "ges:graph:delete"
                  ]
            }
```

```
    ]
}
```

- Example 3: Authorizing users to perform operations on graphs whose name prefix is **ges_project** (**ges_project** names are case insensitive) and access the graph list

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ges:graph:create",
                "ges:graph:delete",
                "ges:graph:access",
                "ges:graph:getDetail"
            ],
            "Resource": [
                "ges:*:*:graphName:ges_project*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ges:graph:list"
            ]
        }
    ]
}
```

- Example 4: Authorizing users to operate only some graph resources, but allowing them to view all resources

  The policy consists of the following two parts:

  – Part 1: Authorizing users to perform operations on resources whose name prefix is **ges_project**. The resources include graphs, metadata, and backups.

  – Part 2: Authorizing users to query the graph, backups, tasks, and metadata lists; verify metadata files, and view job details

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Action": [
                "ges:backup:delete",
                "ges:graph:access",
                "ges:metadata:create",
                "ges:graph:operate",
                "ges:graph:delete",
                "ges:metadata:delete",
                "ges:graph:create",
                "ges:backup:create",
                "ges:metadata:getDetail",
                "ges:graph:getDetail"
            ],
            "Resource": [
                "ges:*:*:backupName:ges_project*",
                "ges:*:*:graphName:ges_project*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "ges:graph:list",
                "ges:backup:list",
                "ges:jobs:list",
                "ges:metadata:list",
                "ges:metadata:operate",
                "ges:jobs:getDetail"
            ],
            "Effect": "Allow"
        }
    ]
}
```

2.3 Role Permissions

Roles can be used for fairly coarse-grained permissions control. They grant service-level permissions based on user responsibilities. GES does not support custom roles. The following system roles are available.

Table 2-3 System roles

Role Name	Description
Tenant Guest	Regular tenant users ● Permissions: querying GES resources ● Scope: project-level service
GES Administrator	GES administrator ● Permissions: performing any operations on GES resources ● Scope: project-level service **NOTE** If you have the **Tenant Guest**, **Server Administrator**, and **VPC Administrator** permissions, you can perform any operations on GES resources. If you do not have the **Tenant Guest** or **Server Administrator** permission, you cannot use GES properly. ● If you need to bind or unbind an EIP, you need the **Security Administrator** permissions to create agencies. ● If GES needs to interact with OBS, for instance, when creating and importing data, OBS permissions are required. For details, see **Common GES operations supported by each OBS policy**.

Role Name	Description
GES Manager	GES manager - Permissions: performing any operations on GES resources other than creating, deleting graphs, resizing, and expanding graphs - Scope: project-level service **NOTE** If you have both the **Tenant Guest** and **Server Administrator** permissions, you can perform any operations on GES resources **except for creating and deleting graphs**. If you do not have the **Tenant Guest** permission, you cannot use GES properly. - If you need to bind or unbind an EIP, you need the **Security Administrator** and **Server Administrator** permissions. - If GES needs to interact with OBS, for instance, when importing data, OBS permissions are required. For details, see **Common GES operations supported by each OBS policy**.
GES Operator	Regular GES users - Permissions: viewing and accessing GES resources - Scope: project-level service **NOTE** If you have both the **GES Operator** and **Tenant Guest** permissions, you can view and access GES resources. If you do not have the **Tenant Guest** permissions, you cannot view resources or access graphs. To interact with OBS, for instance, to view the metadata, you need the OBS permissions. For details, see **Common GES operations supported by each OBS policy**.

Table 2-4 Common GES operations supported by each role

Operation	GES Administrator	GES Manager	GES Operator	Tenant Guest
Creating graphs	Yes	No	No	No
Deleting graphs	Yes	No	No	No
Querying graphs	Yes	Yes	Yes	Yes
Accessing graphs	Yes	Yes	Yes	No
Importing data	Yes	Yes	No	No

Operation	GES Administrator	GES Manager	GES Operator	Tenant Guest
Creating metadata	Yes	Yes	No	No
Viewing metadata	Yes	Yes	Yes	Yes
Copying metadata	Yes	Yes	No	No
Editing metadata	Yes	Yes	No	No
Deleting metadata	Yes	Yes	No	No
Clearing data	Yes	Yes	No	No
Backing up graphs	Yes	Yes	No	No
Restoring graphs from backups	Yes	Yes	No	No
Deleting backups	Yes	Yes	No	No
Querying backups	Yes	Yes	Yes	Yes
Starting graphs	Yes	Yes	No	No
Stopping graphs	Yes	Yes	No	No
Upgrading graphs	Yes	Yes	No	No
Exporting graphs	Yes	Yes	No	No
Binding EIPs	Yes	Yes	No	No
Unbinding an EIP	Yes	Yes	No	No
Viewing results in the task center	Yes	Yes	Yes	Yes
Resizing a Graph	√	No	No	×

Operation	GES Administrator	GES Manager	GES Operator	Tenant Guest
Expanding a Graph	√	No	No	×
Restarting a Graph	√	Yes	No	×

Table 2-5 Common GES operations supported by each OBS policy

GES Operation	Dependent OBS Permission
Viewing metadata	**OBS Viewer** policy or **OBS Buckets Viewer** role
Creating/Importing/Copying/Editing/Deleting metadata	**OBS Operator** policy or **Tenant Administrator** role
Creating, importing, and exporting graphs	**OBS Operator** policy or **Tenant Administrator** role

3 Importing Metadata

3.1 Static Graph

Before importing graph data, familiarize yourself with the graph data formats supported by GES.

- GES only supports the loading of raw graph data in the standard CSV format. If your raw data is not in this format, convert it to CSV.
- GES graph data consists of the vertex, edge, and metadata files.
  - Vertex files store vertex data.
  - Edge files store edge data.
  - Metadata is used to describe the formats of data in vertex and edge files.

Concept Description

Graph data is imported through a property graph model in GES, so you must learn the concept of the property graph.

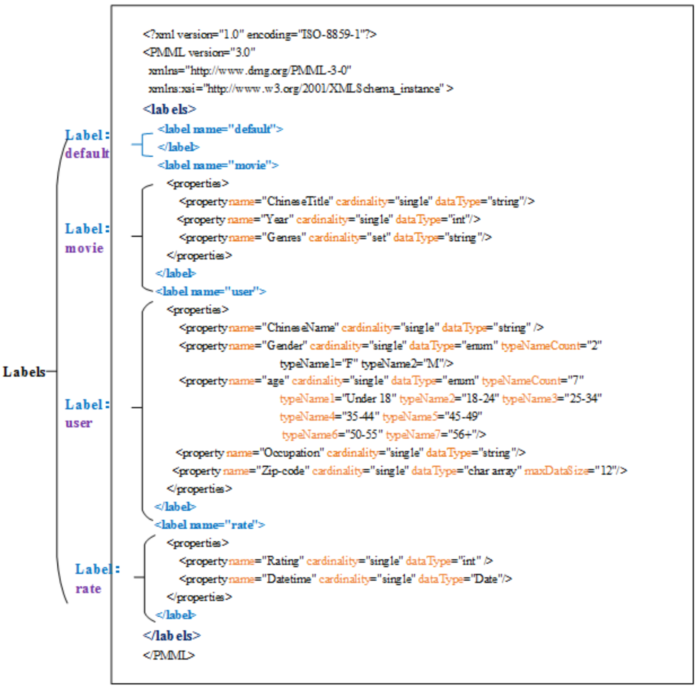
A property graph is a directed graph consisting of vertices, edges, labels, and properties.

- A vertex is also called a node, and an edge is also called a relationship. Nodes and relationships are the most important entities.
- Metadata describes vertex and edge properties. It contains multiple labels, and each label consists of one or more properties.
- Vertices with the same label belong to a group or a set.
- Each vertex or edge can have only one label.

Metadata

The following figure shows the metadata structure.

Figure 3-1 Metadata structure



```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<PMML version="3.0"
 xmlns="http://www.dmg.org/PMML-3-0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema_instance" >
<labels>
  <label name="default">
  </label>
  <label name="movie">
    <properties>
      <property name="ChineseTitle" cardinality="single" dataType="string"/>
      <property name="Year" cardinality="single" dataType="int"/>
      <property name="Genres" cardinality="set" dataType="string"/>
    </properties>
  </label>
  <label name="user">
    <properties>
      <property name="ChineseName" cardinality="single" dataType="string" />
      <property name="Gender" cardinality="single" dataType="enum" typeNameCount="2"
                           typeName1="F" typeName2="M"/>
      <property name="age" cardinality="single" dataType="enum" typeNameCount="7"
                           typeName1="Under 18" typeName2="18-24" typeName3="25-34"
                           typeName4="35-44" typeName5="45-49"
                           typeName6="50-55" typeName7="56+"/>
      <property name="Occupation" cardinality="single" dataType="string"/>
     <property name="Zip-code" cardinality="single" dataType="char array" maxDataSize="12"/>
    </properties>
  </label>
  <label name="rate">
    <properties>
      <property name="Rating" cardinality="single" dataType="int" />
      <property name="Datetime" cardinality="single" dataType="Date"/>
    </properties>
  </label>
</labels>
</PMML>
```

GES metadata is stored in an XML file and is used to define vertex and edge properties.

It contains labels and properties.

- **Label**

  A label is a collection of properties. It describes formats of property data contained within a vertex or an edge.

  NOTE

  If the same property **name** is defined in different labels, the **cardinality** and **dataType** of the properties in different labels must be the same.

- **Property**

  A property refers to the data format of a single property and contains three fields.

  – **name**: Indicates the name of a property. It contains 1 to 256 characters and cannot contain special characters such as angle brackets (<>) and ampersands (&).

    NOTE

    A label cannot contain two properties with the same name.

  – **cardinality**: Indicates the composite type of data. Possible values are **single**, **list**, and **set**.

    ■ **single** indicates that the data of this property has a single value, such as a digit or a character string.

      NOTE

      If **value1;value2** is of the **single** type, it is regarded as a single value.

- **list** and **set** indicate that data of this property consists of multiple values separated by semicolons (;).
  - **list**: The values are placed in sequence and can be repeated. For example, **1;1;1** contains three values.
  - **set**: The values are in random sequence and must be unique. Duplicate values will be overwritten. For example, **1;1;1** contains only one value (1).

  NOTE

  list and **set** do not support values of the **char array** data type.

- **dataType**: Indicates the data type of the property values. The following table lists the data types supported by GES.

Table 3-1 Supported data types

Object Type	Description
char	Character
char array	Fixed-length string. Set the maximum length using the **maxDataSize** parameter. **NOTE** ● You can set **maxDataSize** to limit the maximum length of the string. For details, see **Metadata structure**. ● Only **single** supports the data type. ● If the property data is a string, you are advised to set **dataType** to char array. If the data type is set to **string**, the import is slower.
float	Float type (32-bit float)
double	Double floating point type (64-bit float point)
bool	Boolean type. Available values are **0/1** and **true/false**.
long	Long integer (value range: **-2^63** to **2^63-1**)
int	Integer (value range: **-2^31** to **2^31-1**)
date	Date. Currently, the following formats are supported: ● YYYY-MM-DD HH:MM:SS ● YYYY-MM-DD **NOTE** The value of MM or DD must consist of two digits. If the day or month number contains only one digit, add 0 before it, for example, 05/01.
enum	Enumeration. Specify the number of the enumerated values and the name of each value. For details, see **Metadata structure**.

Object Type	Description
string	Variable-length string **NOTE** The data import efficiency can be very low if the string is too long. You are advised to use a char array instead. You can set the length of a char array as needed. It is recommended that the length be less than or equal to 32 characters.

The following figure shows a metadata example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<PMML version="3.0"
  xmlns="http://www.dmg.org/PMML-3-0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema_instance" >
  <labels>
    <label name="default">
    </label>
    <label name="movie">
        <properties>
            <property name="movieid" cardinality="single" dataType="int" />
            <property name="title" cardinality="single" dataType="string"/>
            <property name="genres" cardinality="single" dataType="string"/>
        </properties>
    </label>
    <label name="user">
        <properties>
            <property name="userid" cardinality="single" dataType="int" />
            <property name="gender" cardinality="single" dataType="string"/>
            <property name="age" cardinality="single" dataType="enum" typeNameCount="7"
            typeName1="Under 18" typeName2="18-24" typeName3="25-34" typeName4="35-44"
typeName5="45-49"
             typeName6="50-55" typeName7="56+"/>
            <property name="occupation" cardinality="single" dataType="enum" typeNameCount="21"
            typeName1="other or not specified" typeName2="academic/educator" typeName3="artist"
typeName4="clerical/admin" typeName5="college/grad student"
             typeName6="customer service" typeName7="doctor/health care" typeName8="executive/
managerial" typeName9="farmer" typeName10="homemaker"
              typeName11="K-12 student" typeName12="lawyer" typeName13="programmer"
typeName14="retired" typeName15="sales/marketing"
               typeName16="scientist" typeName17="self-employed" typeName18="technician/engineer"
typeName19="tradesman/craftsman" typeName20="unemployed"
                typeName21="writer"/>
            <property name="Zip-code" cardinality="single" dataType="char array" maxDataSize="12"/>
        </properties>
    </label>
    <label name="rate">
        <properties>
            <property name="Rating" cardinality="single" dataType="int" />
            <property name="Datetime" cardinality="single" dataType="string"/>
        </properties>
    </label>
</labels>
</PMML>
```

Vertex Files

A vertex file contains the data of each vertex. A vertex of data is generated for each behavior. The following is an example. **id** is the unique identifier of a set of vertex data.

```
id, label, property 1, property 2, property 3, ...
```

NOTE

- The vertex ID cannot contain hyphens (-).
- You do not need to set the data type of the vertex ID. It is of the string type by default.
- Do not add spaces before or after a label. Use commas (,) to separate information. If a space is identified as a part of a label, the label may fail to be identified. In this case, the system may display a message indicating that the label does not exist.

Example:

```
Vivian, user, Vivian, F, 25-34, artist, 98133
Eric, user, Eric, M, 18-24, college/grad student, 40205
```

Edge Files

An edge file contains the data of each edge. An edge of data is generated for each behavior. The graph size in GES is defined by the quantity level of the edges, for example, one million edges. The following is an example. **id 1** and **id 2** are the IDs of the two endpoints (vertices) of an edge.

```
id 1, id 2, label, property 1, property 2, ...
```

Example:

```
Eric,Lethal Weapon,rate,4,2000-11-21 15:33:18
Vivian,Eric,friends
```

3.2 Importing a Metadata File

3.2.1 Preparing Metadata

Preparing Metadata on a Local PC

You need to prepare a metadata file on your PC and import the file to GES for subsequent use.

The metadata files you want to import must meet the following requirements:

1. A maximum of 50 metadata files can be imported.
2. The metadata files must be in XML format.

(Optional) Importing Metadata to OBS

You can upload a prepared metadata file to an OBS bucket to import it to GES.

The procedure is as follows:

1. Log in to the OBS console and create an OBS bucket. If you already have a bucket, ensure that the OBS bucket and GES are in the same region.
2. Upload the prepared file to the OBS bucket. The metadata file must be in XML format.

3.2.2 Importing Data From a Local Path or OBS

1. On the GES management console, click **Metadata Management** in the navigation tree on the left.
2. On the **Metadata Management** page, click **Import** in the upper left corner.
3. In the **Import** dialog box, select **Local** or **OBS** for **Type** to import a metadata file form a local path or OBS.
   - Import a metadata file from a local path.

     Select Local File: Click **Upload** to select the metadata file.

     NOTE

     The file must be in the XML format.

     Name: Enter a name for the metadata.

     Storage Path: Select an OBS path for storing the metadata file.
   - Import a metadata file from OBS.

     Select File Path: Select the metadata file from OBS.

     NOTE

     - The file must be in the XML format.
     - Ensure that you have uploaded the metadata file to your OBS bucket.

     Name: Enter a name for the metadata.
4. Click **OK** to import the metadata.

   If the import is successful, the metadata file is displayed on the **Metadata Management** page.

3.3 Creating a Metadata File

If you currently have no metadata file, you can create metadata files on GES.

Procedure

1. On the **Metadata Management** page, click **Create Metadata File** in the upper right corner.
2. Configure the following parameters on the displayed page:

   Name: Enter the metadata file name. The default file format is XML.

   Storage Path: Select an OBS path for storing the metadata file. If this is your first time creating a metadata file, you need to enable OBS. You are advised to obtain user authorization to automatically create OBS buckets for the metadata.

   Encrypt Metadata: Whether to encrypt the metadata. It is disabled by default.

   Key Source: Retain the default value **KMS**.

   KMS Key: Select the key as needed.

   NOTE

   Some functions will be affected if you disable or delete a KMS key.

Definition: Labels of vertices and edges in the metadata file. Multiple labels are allowed. Click **Add label** to add labels as required.

Specify the **Label Name** and click the down arrow to expand the label information and add properties for the label. Click **Add** to enter a property name and click **Up** and **Down** to change orders of properties. **Table 3-2** lists the property parameters. For details about other metadata information, see section **Static Graph**.

Table 3-2 Property parameters

Name	Description
Property Name	Name of a property. It contains 1 to 256 characters. Special characters such as angle brackets (<>) and ampersands (&) are not allowed.
Cardinality	Composite type of data ● **Single value**: indicates that the property has a single value, such as a digit or a string. ● **Multiple values**: indicates that the property has multiple values separated by semicolons (;). You can determine whether to allow repetitive values.
Data Type	Data type of the property values. Available values are **char**, **float**, **double**, **bool**, **long**, **int**, **date**, **enum**, **string**, and **char array**. For details, see **Static Graph**. NOTE Only the single-value property supports the **char array** type.
Operation	Click **Remove** to delete a property.

3. Click **OK**. The created metadata file will be displayed on the **Metadata Management** page.

   On the **Metadata Management** page, you can view the storage path, status, and modification time of the metadata.

3.4 Copying a Metadata File

If you edit a metadata file, the original metadata file will be overwritten. To avoid loss of the original metadata, you can sabe a copy of the file before editing it.

Procedure

1. GES provides two methods for you to copy a metadata file on the **Data Management** page.
   - Click the metadata file name. On the details page, click **Copy**.
   - Click **Copy** in the **Operation** column of the target metadata file.
2. Specify the metadata file name and storage path.

   Name: Enter the name of the copied metadata file. The default file format is XML.

Storage Path: Enter an OBS path for storing the metadata file.

Encrypt Metadata: Whether to encrypt the copied metadata. This function is disabled by default. **Key Source**: Retain the default value **KMS**. **KMS Key**: Select a key as needed.

3. Click **OK**.

   The copy of the metadata file will be displayed on the **Metadata Management** page.

3.5 Editing a Metadata File

If the metadata file you imported or created needs to be modified, you can directly modify its labels and properties.

NOTE

After the metadata file is edited, the original metadata file will be overwritten. To avoid data loss, you are advised to save a copy of the metadata file before editing it.

Procedure

1. GES provides two methods for you to edit a metadata file on the **Data Management** page.
   - Click the metadata file name. On the metadata details page, click **Edit**.
   - Click **Edit** in the **Operation** column of the target metadata file.
2. On the **Edit** page, you can add labels and properties, change the label names, and sort properties by clicking **Up** and **Down**.
3. After the modification is complete, click **OK**.

3.6 Searching for a Metadata File

On the **Metadata Management** page, enter the name of the metadata file you want to search.

3.7 Deleting a Metadata File

If a metadata file becomes invalid, locate it in the metadata file list on the **Metadata Management** page and click **Delete** in its **Operation** column.

NOTE

Deleted data cannot be recovered. Exercise caution when performing this operation.

4 Creating Graphs

4.1 Methods to Create a Graph

The following content describes how to create a graph on GES console.

Custom graph: This is a default graph creation method that fully meets your requirements.

4.2 Creating a Graph Without Using a Template

1. Log in to the GES console and click **Create Graph** in the upper right corner of the home page. The **Create Graph** page is displayed.
2. On the **Create Graph** page, set the following parameters:
   a. In the **Configure** step, set the graph name and software version.

Parameter	Description
Graph Name	You can set a name or use the default name. After a graph is created, its name cannot be changed. The graph name must: ● Contain 4 to 50 characters and start with a letter. ● Be case-insensitive. ● Contain only letters, digits, and underscores (_).
GES Software Version	The system uses the latest version by default, and only the default version is available.

   b. Specify the network information, including **VPC**, **Subnet**, **Security Group**, **Enterprise Project**, and **Public Network Access**.

Parameter	Description
VPC	A VPC is a secure, isolated, and logical network environment. Select the VPC for which you want to create the graph and click **View VPC** to view the name and ID of the VPC. **NOTE** If your account has VPCs, a VPC will be automatically selected. You can change it as needed. If no VPC is available, you need to create a VPC. After the VPC is created, it will be automatically selected.
Subnet	A subnet provides dedicated network resources that are logically isolated from other networks for network security. Select the subnet for which you want to create the graph to enter the VPC and view the name and ID of the subnet.
Security Group	A security group implements access control for ECSs that have the same security protection requirements in a VPC. ● Click **Learn how to configure a security group.** to get instructions. ● Click **View Security Group** to learn security group details.
Public Network Access	The public network access to the graph. Set this parameter as you need. **Do not use**: A graph instance without an elastic IP (EIP) cannot be accessed over the Internet. However, the graph instance can be accessed through ECSs deployed on a private network. **Buy now**: GES automatically allocates an EIP with exclusive bandwidth to the graph instance so that the graph instance can be accessed over the Internet using the EIP. In addition, GES uses the tenant permission to automatically create an agency with the prefix of **ges_agency_default** in the project to support EIP binding. **Specify**: Select an EIP to allow the graph instance to be accessed over the Internet. Click **Create EIP** to access the VPC management console and create an EIP.
Enterprise Project	Centrally manages cloud resources and members by project. Click **Create Enterprise Project** to go to the **Enterprise Project Management** page.

Parameter	Description
Tag	Tags for a resource. Enter a tag key and value, and click **Add** to add the tag. You can view the added tag in the graph details and search for graphs by tag on the **Graph Management** page. **NOTE** It is recommended that you use TMS's predefined tag function to add the same tag to different cloud resources.
Security Mode	If you enable the security mode, communications will be encrypted when you access a graph instance, and only HTTPS can be used when you call APIs. This function affects GES performance.
cryptographic algorithm	Available values are as follows: ● General cryptographic algorithms (SM series cryptographic algorithms not supported) are used by all components to store and transmit sensitive data. These algorithms that do not have special requirements. ● SM series commercial encryption algorithm (compatible with the international general algorithm) is supported. Sensitive data of all components is stored using this algorithm. The SM series commercial encryption algorithm and international algorithm can be used for data transmission.

c. Set graph parameters.

Parameter	Description
Cross-AZ HA	Whether to support cross-AZ cluster. If this function is enabled, graph instances are distributed in different AZs to enhance reliability.
Purpose	Purpose of the graph to be created. **Enterprise production**: High reliability and concurrency are supported, suitable for production and large-scale applications. **Developer learning**: A complete function experience is offered, suitable for developer learning.

Parameter	Description
Versions	GES editions. ● Memory edition: The capacity is limited and a maximum of 10 billion edges are supported. Storage and compute based on memory storage. This edition is preset with a variety of algorithms, and Gremlin and Cypher query languages are supported. ● Database edition: The storage capacity is unlimited. Storage and compute based on distributed key-value databases. This edition has higher performance and has unlimited capacity, but it supports only the Cypher queries.
Compute Resource	Type of compute resources. An elastic cloud server (ECS) is a computer system that has complete hardware, an operating system (OS), and network functions and runs in a secure, isolated environment.
CPU Architecture	Currently, GES supports **X86** and **Kunpeng**.
Graph Size (Edges)	Available options based on your resource quota. Different graph specifications are displayed for **Enterprise production** and **Developer learning**. ● **Development learning**: Currently, there is only 10-thousand-edge graphs are available for this purpose, regardless of the edition. ● **Enterprise production**: The specifications vary depending on the edition. – **Memory edition**: Available options are million-edge, 10-million-edge, 100-million-edge, billion-edge, billion-edge-pro, and 10-billion-edge. – **Database edition**: Available options are billion-edge, 10-billion-edge, and 100-billion-edge.

Parameter	Description
Vertex ID Type	Only fixed-length string and hash types are available for graphs of the database edition. ● Fixed-length string: Vertex IDs are used for internal storage and compute. Specify the length limit. If the IDs are too long, the query performance can be reduced. Specify the length limit based on your dataset vertex IDs. If you cannot determine the maximum length, set the ID type to Hash.", ● Hash: Vertex IDs are converted into hash code for storage and compute. There is no limit on the ID length. However, there is an extremely low probability, approximately 10^(-43), that the vertex IDs will conflict. **NOTE** If you cannot determine the maximum length of a vertex ID, set this parameter to **Hash**.

d. **Advanced Settings**: Set this parameter to **Default** or **Custom**.

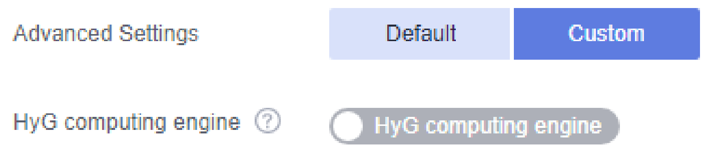
- **Default**: Use the default values.
- **Custom**:
  - If you choose the memory edition, the options include **Encrypt Instance**, **Operation Audit**, and **Fine-Grained Permission**.

Parameter	Description
Encrypted Instance	Whether to encrypt a graph instance. **Key Source** is default to **KMS**. **KMS Key**: Select the key as needed. **NOTE** Some functions will be affected if you disable or delete a KMS key.
Full-Text Index	Whether to enable full-text search, fuzzy search, prefix search, and regular expression match. **NOTE** ● Currently, this feature is available for the **Billion-edge-pro** specification only.
Fine-Grained Permission	Whether to enable fine-grained permission management. If this function is enabled, the traverse, read, and write permissions can be set for specific attributes each label.

Parameter	Description
Operation Audit	Whether to enable operation audit **LTS Log Group**: Select the corresponding log group. Click **View Log Group List** to view log information on the log management page. **NOTE** You will be billed for storing logs to LTS. For details, see the LTS billing standards.

- If you choose the database edition, you can enable or disable **HyG computing engine**.

Figure 4-1 Advanced settings for the database edition



Advanced Settings: Default | Custom

HyG computing engine: HyG computing engine

Parameter	Description
HyG computing engine	HyG is a high-performance distributed graph computing framework that supports many graph analysis algorithms. HyG engine is suitable for complex graph analysis.

3. Click **Confirm**. The **Confirm** page is displayed.
4. Confirm the information and click **Submit** to create the graph.
5. After the submission is successful, the **Finish** tab page is displayed. You can click **Back to Task Center** to view the status and running result of the created graph.

4.3 Starting a Graph

Scenario

You can start graphs in **Stopped** status in the graph list so that they can be accessed and analyzed again.

Graphs in **Running** status cannot be started.

Procedure

Step 1 Log in to the GES management console.

Step 2 In the navigation tree on the left, select **Graph Management**.

Step 3 Locate the target graph in the graph list and choose **More** > **Start** in the **Operation** column.

- If the graph to be started has backups, a dialog box is displayed indicating that you can select either of the following methods to start the graph:
  - **Restore Last Graph**: Restart the graph that stopped running.
  - **Start Backup**: Start the graph using the backup data.

  After selecting a startup method, click **Yes**. The graph status becomes **Preparing** and the progress is displayed.
- If the graph to be started does not have backups, the graph status changes to **Preparing** and the progress is displayed after you click **Start**.

Step 4 After the graph is started, the status changes from **Preparing** to **Starting**. Wait several minutes. When the startup is successful, the graph status is switched to **Running**.

----End

4.4 Stopping a Graph

Scenario

If you do not need to use a graph, you can stop it. After the graph is stopped, you cannot access it.

NOTE

Resources are not released after you stop the graph.

Procedure

Step 1 Log in to the GES management console.

Step 2 In the navigation tree on the left, select **Graph Management**.

Step 3 Locate the target graph in the graph list and choose **More** > **Stop** in the **Operation** column.

Step 4 The graph status changes to **Stopping**. Wait several minutes. When the graph is successfully stopped, the graph status is switched to **Stopped**.

----End

4.5 Accessing Graphs

Scenario

On the **Graph Management** page, you can click **Access** to query and analyze a created graph.

Procedure

On the **Graph Management** page, view all created graphs and click **Access** in the **Operation** column of a target graph.

4.6 Importing Incremental Data

Scenario

After you create a graph, you need to import graph data. If you need to add new graph data, you can import data to the graph.

NOTE

- To prevent failures in restoring the imported graph data during system restart, do not delete the data stored on OBS when the graph is in use.
- The default separator of data columns is comma (,). You cannot define a separator.

Procedure

Step 1 Log in to the GES management console.

Step 2 In the navigation tree on the left, select **Graph Management**.

Step 3 Locate the target graph in the graph list and select **Import** in the **Operation** column.

Step 4 In the **Import** dialog box that is displayed, set the following parameters:

- **Metadata**: Select an existing metadata file or create one. For details, see **Creating a Metadata File**.
- **Edge Data**: Select the corresponding edge data set.
- **Vertex Data**: Select the corresponding vertex data set. If you leave it blank, the vertices in the **Edge Data** set are used as the source of **Vertex Data**.
- **Log Storage Path**: Stores vertex and edge data sets that do not comply with the metadata definition, as well as detailed logs generated during graph import.
- **Edge Processing**: Includes **Allow repetitive edges**, **Ignore subsequent repetitive edges**, **Overwrite previous repetitive edges**, and **Ignore labels on repetitive edges**.

  Edge Processing: Repetitive edges have the same source and target vertices. When labels are considered, repetitive edges must have the same source and target vertices and the same labels.

  – **Allow repetitive edges**: Multiple edges may exist between a source vertex and a target vertex.

  – **Ignore subsequent repetitive edges**: If there are multiple edges between a source vertex and a target vertex, only the first edge read is retained.

  – **Overwrite previous repetitive edges**: If there are multiple edges between a source vertex and a target vertex, only the last edge read is retained.

  - **Ignore labels on repetitive edges**: If labels are ignored, edges with the same source vertex and target vertex are repetitive edges.
- **Import Type**: The value can be **Online import** or **Offline import**.

NOTE

- The edge and vertex data sets can only be stored in English paths and folders.
- Currently, you can import the edge and vertex data sets only from OBS. You need to store data files in an OBS bucket..
- The sequence of the properties and labels in the selected edge or vertex data set must be the same as the sequence in the selected metadata file. Otherwise, **The edge/vertex data file does not match the metadata file** is displayed in the upper right corner and the graph fails to be created. For details about the graph data format, refer to **Graph Data Formats**.
- You need to import the graph data (including the metadata file, and edge and vertex data sets) in the format specified in the template. The template contains a copy of movie information. You can click **Download** to download and import it.

Step 5 Click **OK**.

----End

5 Managing Graphs

5.1 Graph Management Overview

On the **Graph Management** page, you can view the name, running status, internal access address, external access address, and creation time of a graph.

NOTE

To view the **internal access address** is the floating IP address for accessing the graph instance. You can click the IP address to view the list of physical IP addresses of the graph instance. To prevent service interruption caused by floating IP address switchover, poll the physical IP addresses to access the graph instance.

Click ⌄ next to a graph name to view detailed information, including the graph ID, VPC, subnet, security group, graph size (edges), vertex data set, edge data set, and metadata, as well as graph version, cross-AZ HA, operation audit, creator, granular permissions, encryption, CPU architecture, .

5.2 Viewing a Failed Graph

Scenario

If the ECS quota is insufficient, graphs may fail to be created. You can view failed graphs on the **Graph Management** page.

Procedure

Step 1 In the navigation tree on the left, select **Graph Management**.

Step 2 In the upper left corner of the displayed page, view the number of graphs that fail to be created next to **Graph Management**.

Step 3 Click 🔥 to view the name, running status, and creation time of the graph that fails to be created. You can also delete the failed graph.

NOTE

Graphs that fail to be created will occupy quotas if they are not deleted.

Step 4 Click **View Details** in the **Operation** column to go to the **Task Center** page. View the start time, end time, failure cause, and job ID of the failed creation task.

NOTE

Asynchronous task details can be retained only for one month. You cannot view information about graphs created more than one month ago.

----End

5.3 Backing Up and Restoring Graphs

5.3.1 Backing Up a Graph

Scenario

To ensure data security, back up the graph data so that you can restore it when faults occur.

Procedure

You can perform the backup operation on the **Graph Management** page or the **Backup Management** page.

1. Graph management operations
   a. Log in to the GES management console. In the navigation tree on the left, select **Graph Management**.
   b. Locate the target graph in the graph list and select **Back Up** in the **Operation** column.
   c. In the displayed dialog box, click **Yes**.

      NOTE

      On the **Graph Management** page, the backup operation can be performed only on the selected graph. The associated graph cannot be changed.
   d. In the navigation tree on the left, click **Backup Management**. You can view the backup task in the backup list.

      If **Status** is **Backing up**, wait several minutes. When **Status** is switched to **Succeeded**, the backup is successful.
2. Backup management operations
   a. Log in to the GES management console. In the navigation tree on the left, select **Backup Management**.
   b. In the upper right corner of the **Backup Management** page, click **Create Backup**.
   c. In the **Create Backup** dialog box, select an **Associated Graph** (a graph created by the current user) and click **OK** to start backup.

NOTE

You can select an **Associated Graph** for the backup. However, if there is only one graph, you cannot change the value of **Associated Graph**.

d. In the backup list, you can view the data being backup up or newly backed up.

   If **Status** is **Backing up**, wait several minutes. When **Status** is switched to **Succeeded**, the backup is successful.

e. Go to the **Backup Management** page, view the backup name and type, name, status, and size of the associated graph, CPU architecture, creation time, end time, backup size, and backup duration.

5.3.2 Restoring a Graph

Scenario

If the graph data being edited is incorrect, you can load the backup data to restore the graph data for analysis.

NOTE

Ten-thousand-edge graphs cannot be automatically backed up. You need to back up a graph and restore data from the manul backup. For graphs of other sizes, you can restore data from an automatic backup or manual backup.

Procedure

Step 1 Log in to the GES management console.

Step 2 In the navigation tree on the left, select **Backup Management**.

Step 3 On the **Backup Management** page, locate the target backup and click **Restore** in the **Operation** column.

Step 4 In the **Restore** dialog box, select **This operation will overwrite the target graph. After the restoration starts, the graph will be restarted using the backup**. Click **Yes**.

Step 5 After a message is prompted indicating that the restoration is successful, you can access the target graph and obtain the restored data on the **Graph Management** page.

----End

5.3.3 Deleting a Backup

Scenario

If the backup data will not be used any more, you can delete it based on site requirements.

Procedure

Step 1 Log in to the GES management console.

Step 2 In the navigation tree on the left, select **Backup Management**.

Step 3 In the backup list, select the backup data to be deleted and click **Delete** in the **Operation** column.

Step 4 In the displayed dialog box, click **Yes** to delete the data.

NOTE

1. Deleted data cannot be recovered. Exercise caution when performing this operation.
2. You cannot delete the automatic backup data of a graph that has not been deleted.

----End

5.4 Upgrading a Graph

Scenario

Because the GES software is upgraded continuously, graphs of earlier versions can also be upgraded to the new version.

Procedure

Step 1 Log in to the GES management console.

Step 2 In the navigation tree on the left, select **Graph Management**.

Step 3 Locate the target graph in the graph list and choose **More** > **Upgrade** in the **Operation** column.

Step 4 In the displayed dialog box, select a version from the **Version List** and determine whether to select **Forcible Upgrade**.

NOTE

If **Forcible Upgrade** is selected, all in-progress tasks will be interrupted. Exercise caution when performing this operation.

Step 5 Click **OK**. The graph status changes to **Upgrading**. Wait several minutes, the status will become **Running** after the upgrade is successful.

NOTE

If the upgrade fails, the graph automatically rolls back to the source version.

----End

5.5 Exporting a Graph

Scenario

This topic describes how to export the graph data to a user-defined OBS directory.

Procedure

Step 1 Log in to the GES management console.

Step 2 In the navigation tree on the left, select **Graph Management**.

Step 3 Locate the target graph in the graph list and choose **More** > **Export** in the **Operation** column.

Step 4 In the lower part of the displayed page, select a storage path.

Step 5 Click **OK**. The graph status changes to **Exporting**. Wait several minutes, the status will become **Running** after the export is successful.

You can check whether the data is exported successfully in the selected OBS path.

----End

5.6 Restarting a Graph

Scenario

You need to restart a graph in the following cases:

1. If you access a graph in the **Running**, **Importing**, **Exporting**, or **Clearing** status and an unknown exception occurs, you can restart the graph.
2. You can restart a graph that is stuck in a state. For example, if a graph stuck in the **Exporting** status for a long time because the data to be exported is too much. You can restart the graph to stop exporting.

Procedure

Step 1 Logging In to the GES management console.

Step 2 In the navigation pane on the left, choose **Graph Management**. On the displayed page, locate the graph to be restarted and choose **More** > **Restart** in the **Operation** column.

Step 3 In the displayed dialog box, check the name of the graph to be restarted.

NOTE

Restarting a graph will forcibly terminate the running task. For an import task, only partial data can be imported.

Step 4 Click **OK**. The graph status changes to **Stopping**. After several minutes, the graph status changes to **Running**.

----End

5.7 Resizing a Graph

Scenario

If the storage capacity, computing capability, or service capability of a graph cannot meet service requirements, you can resize the graph.

NOTE

- Currently, 10,000-edge and 10-billion-edge graphs cannot be resized.
- After the graph is resized, you need to re-create all indexes.

Procedure

Step 1 Log in to the management console.

Step 2 In the navigation pane on the left, choose **Graph Management**. On the displayed page, locate the target graph and choose **More** > **Resize** in the **Operation** column.

Step 3 In the displayed dialog box, select the target specifications. You can only select higher specifications. For example, a graph with 1 million edges can be changed to 10 million, 100 million, 1 billion, or 10 billion edges.

Step 4 Click **OK**. The graph status changes to **Preparing for resize**. After several minutes, the graph status changes to **Resizing**. When the resize is complete, the graph status changes to **Running**.

----End

5.8 Expanding a Graph

Scenario

Graph expanding increases the maximum number of concurrent read-only requests that can be processed, without changing the graph size.

NOTE

- Currently, 10,000-edge and 10-billion-edge graphs cannot be expanded.
- Graphs cannot be resized after expansion. If you want to resize and expand the graph, resize the graph before you expand it.

Procedure

Step 1 Log in to the management console.

Step 2 In the navigation pane, choose **Graph Management**. On the displayed page, locate the target graph and choose **More** > **Expand** in the **Operation** column.

NOTE

Only a running graph can be expanded.

Step 3 In the displayed dialog box, set the number of nodes to be added.

Step 4 Click **OK**. The graph status changes to **Expanding**. Wait several minutes, the status will become **Running** after the expansion is successful.

----End

5.9 Binding and Unbinding an EIP

Binding an EIP

To access GES over the Internet, you can bind an Elastic IP Address (EIP) to your instance.

The procedure is as follows:

Step 1 Log in to the GES management console.

Step 2 In the navigation tree on the left, select **Graph Management**.

Step 3 Locate the target graph in the graph list and choose **More** > **Bind EIP** in the **Operation** column.

Step 4 On the displayed **Bind EIP** page, select an available EIP.

Step 5 Click **OK**.

----End

Unbinding an EIP

If you do not need to use the EIP, you can unbind the EIP to release network resources.

The procedure is as follows:

Step 1 Log in to the GES management console.

Step 2 In the navigation tree on the left, select **Graph Management**.

Step 3 Locate the target graph in the graph list and choose **More** > **Unbind EIP** in the **Operation** column.

Step 4 In the displayed dialog box, click **Yes**.

----End

5.10 Clearing Data

Scenario

If unnecessary data is imported or the imported data volume exceeds the graph size, you can clear the data.

In addition, if you delete data by mistake using Gremlin or Cypher commands, you can clear the broken data and import the correct data again.

NOTE

This operation will clear all vertex and edge data of the graph. Exercise caution when performing this operation.

Procedure

Step 1 Log in to the GES management console.

Step 2 In the navigation tree on the left, select **Graph Management**.

Step 3 Locate the target graph in the graph list and choose **More** > **Clear Data** in the **Operation** column.

Step 4 In the dialog box that is displayed, select or deselect **Clear the metadata in the graph**.

NOTE

Deleted metadata cannot be recovered. Exercise caution when performing this operation.

Step 5 Click **Yes**.

----End

5.11 Deleting a Graph

Scenario

If you have analyzed the graph data, you can delete the graph to release resources.

NOTE

Backups of a graph will be also deleted after the graph is deleted, and data cannot be recovered. Exercise caution when performing this operation.

Procedure

Step 1 Log in to the GES management console.

Step 2 In the navigation tree on the left, select **Graph Management**.

Step 3 Locate the target graph in the graph list and choose **More** > **Delete** in the **Operation** column.

Step 4 In the displayed dialog box, select **Release the EIP bound with the graph instance**. This option is available only for graphs have EIPs bound.Select **Keep one auto backup and two manual backups, which occupy the quota** as required.

Step 5 Click **Yes** to complete the deletion.

----End

5.12 Viewing Monitoring Metrics

Scenario

It takes a period of time for transmitting and displaying data. The GES status displayed in the Cloud Eye monitoring data is the status obtained 5 to 10 minutes before. You can view the monitored data of a newly created graph 5 to 10 minutes later.

Prerequisites

- The created graph is running properly.
- The graph has been properly running for at least 10 minutes. For a newly created graph, you need to wait for a while before viewing its metrics.
- Cloud Eye does not display the metrics of a faulty or stopped graph. You can view the monitoring metrics after the graph starts or recovers.

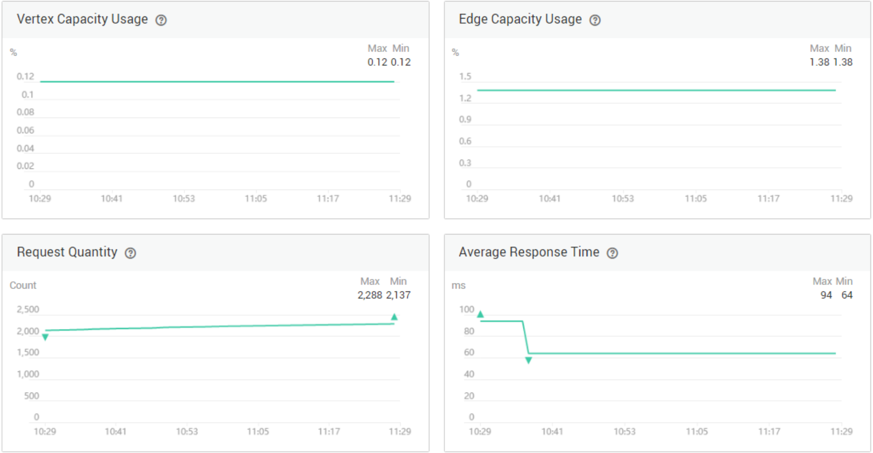
Procedure

Step 1 Log in to the management console.

Step 2 In the navigation pane, choose **Graph Management**. In the **Operation** column, choose **More** > **View Metrics**. The Cloud Eye management console is displayed.

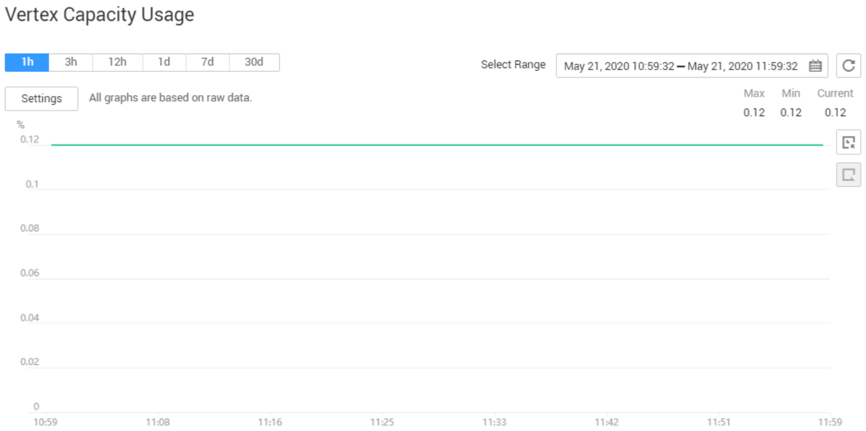
Step 3 On the monitoring page for GES, you can view the figures of all monitoring metrics.

Figure 5-1 Viewing monitoring metrics



Step 4 To view the monitoring curve in a longer time range, click **Full Image** to view a chart in a bigger view.

Figure 5-2 Zoomed in graph



All graphs are based on raw data.

Select Range May 21, 2020 10:59:32 – May 21, 2020 11:59:32

Step 5 The system allows you to select a fixed time range or use automatic refresh.

1. Fixed time ranges include **1h**, **3h**, **12h**, **1d**, **7d**, **30d**.

----End

5.13 Querying Schema

Scenario

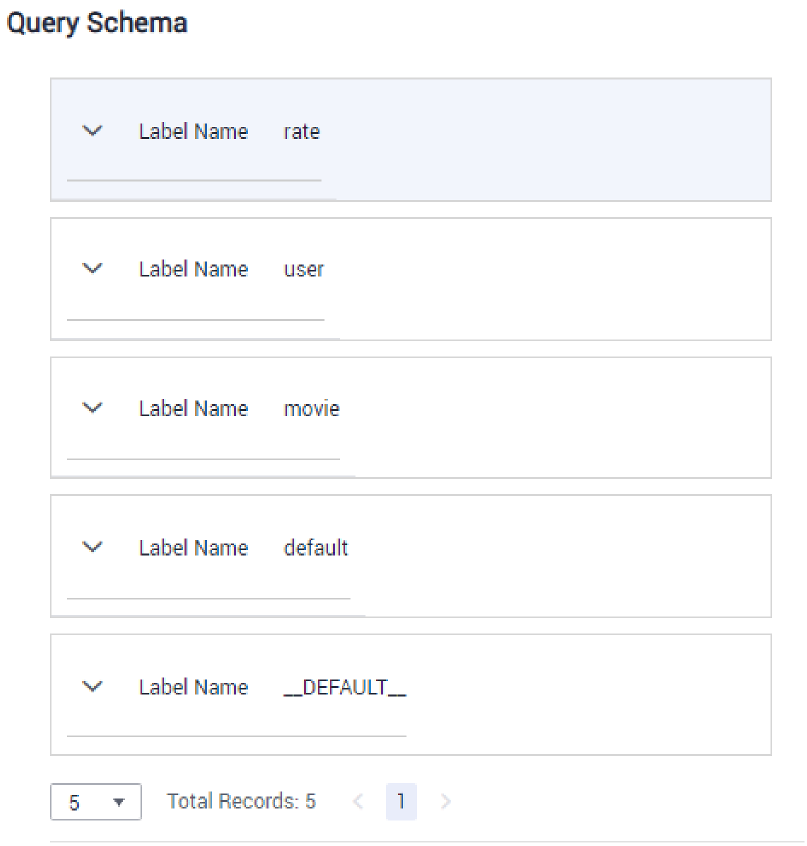
Query the metadata of a graph. The metadata contains labels and properties.

Procedure

Step 1 Log in to the management console.

Step 2 In the navigation pane, choose **Graph Management**. In the **Operation** column, choose **More > Query Schema**. A window is displayed, showing the labels contained in the metadata of the current graph.

Figure 5-3 Querying schema



Query Schema

Label Name rate

Label Name user

Label Name movie

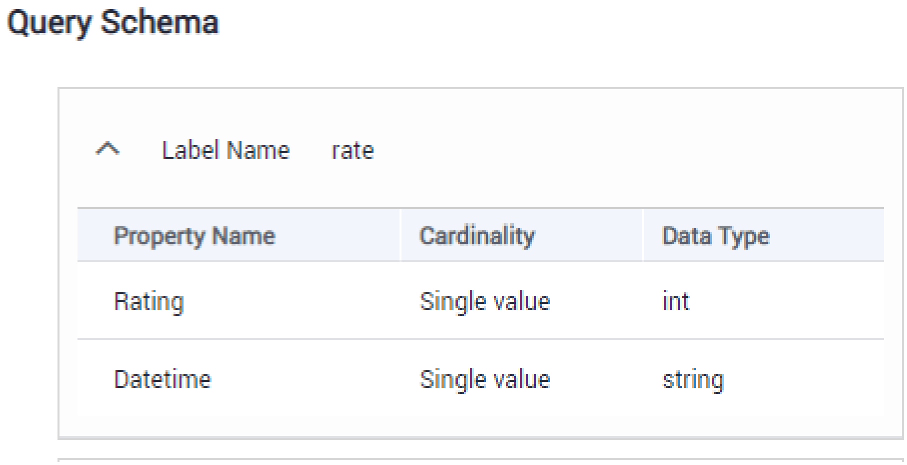
Label Name default

Label Name __DEFAULT__

5 Total Records: 5 1

Step 3 To view the properties contained in a label, click ∨ of each label.

Figure 5-4 Viewing properties in labels



Query Schema

Label Name rate

Property Name	Cardinality	Data Type
Rating	Single value	int
Datetime	Single value	string

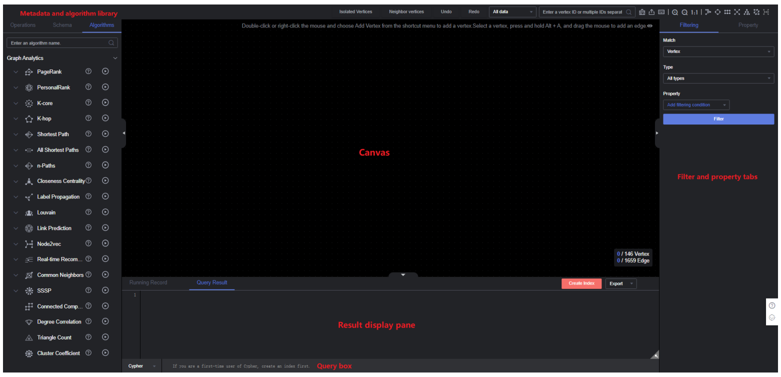
----End

6 Accessing and Analyzing Graph Data

6.1 Graph Editor

The graph editor consists of an algorithm library, metadata tab, operation tab, graph exploration, query text box, canvas, result display pane, and filtering and property tabs.

Figure 6-1 Editor page



Double-click or right-click the mouse and choose Add Vertex from the shortcut menu to add a vertex.Select a vertex, press and hold Alt + A, and drag the mouse to add an edge.

If you are a first-time user of Cypher, create an index first.

Table 6-1 Graph editor

Area	Description
Algorithms	Algorithms supported by GES. You can set the properties of each algorithm in this area. Table 6-2 describes the functions of the algorithm library. **NOTE** After you select an algorithm in the algorithm library and execute it, the canvas displays the sampling sub-graph that contains the key result. The execution result is incomplete. To obtain the complete returned result, call the corresponding API.
Schema	Metadata operations, such as adding, hiding, importing, and exporting data. For details, see **Editing Schema**.

Area	Description
Operations	Operations executed by API calls. For details, see **Adding Custom Operations**.
Exploration pane	Graph exploration tools, for example, path expansion. For details about the functions, see **Exploring Graphs**.
Query box	● Where you can enter Gremlin statements to query a graph. ● Where you can enter Cypher statements to query a graph.
Canvas	Graph structure of data. Shortcut operations are preset in the drawing area for you to easily analyze the graph data. **Table 6-3** describes the functions of the drawing area.
Result display pane	There are two tab pages: ● **Running Record** where you can **View Running Records**. ● **Query Result** where you can **Viewing Query Results**.
Filter and Property area	On the canvas, select a vertex and right-click it. Then, choose **View Property** from the shortcut menu to view the Filter and Property area. It contains the following three tabs: ● The **Filtering** tab page allows you to set properties and conditions to filter the data for analysis. For details, see **Filtering Conditions**. ● The **Property** tab page displays the property information about a vertex or an edge. ● The statistics tab displays the number of labels and vertex weights of the selected vertices and edges. For details, see **Statistics Display**.

Figure 6-2 Algorithm Library

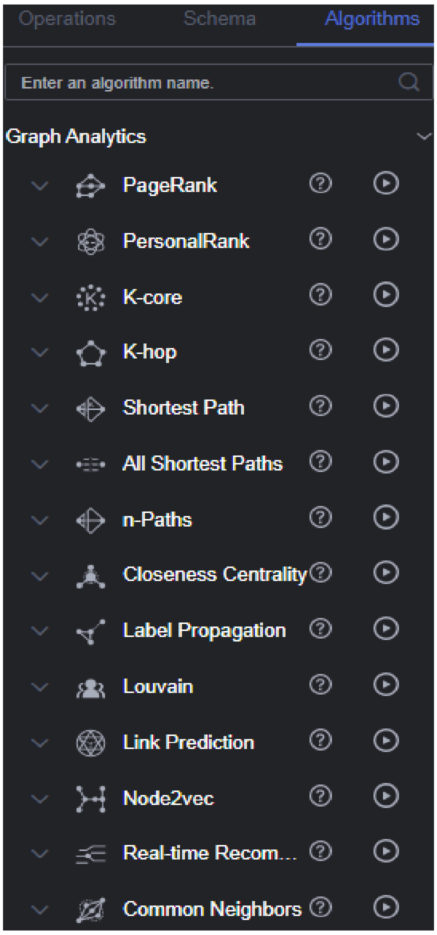


Table 6-2 Algorithm library description

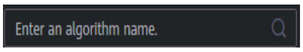
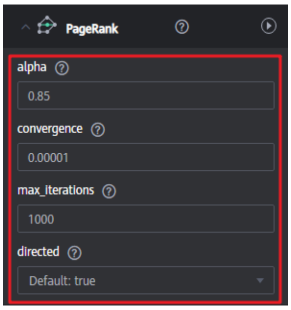
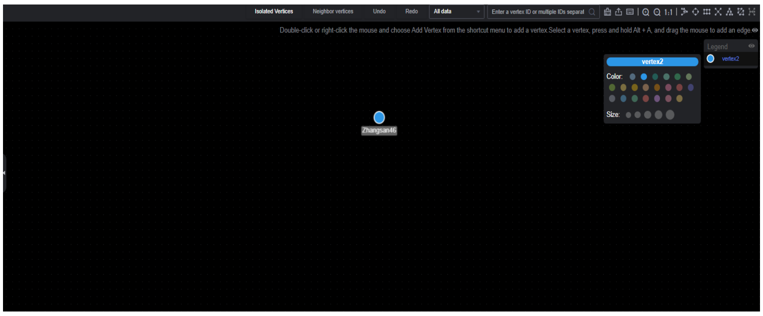
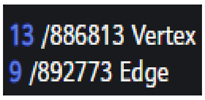
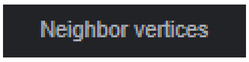
Interface Element	Description
 Enter an algorithm name.	Enter the algorithm name to quickly find it.
⌄	Expand the algorithm parameter configuration area.
⏵	Run the algorithm.
 PageRank alpha 0.85 convergence 0.00001 max_iterations 1000 directed Default: true	Set the properties of an algorithm. Different algorithms have different properties. For details, see **Algorithms**.

Figure 6-3 Canvas

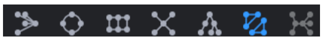
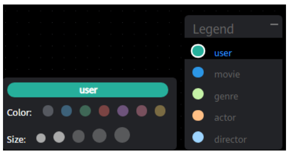
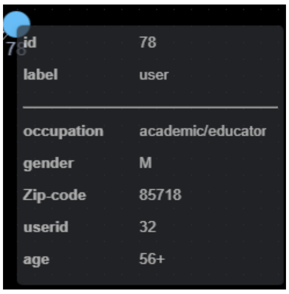
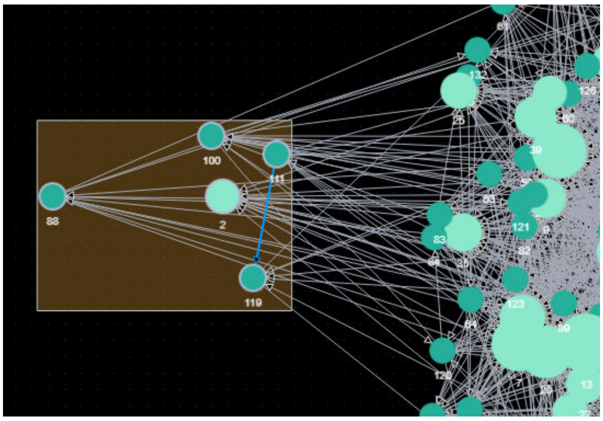


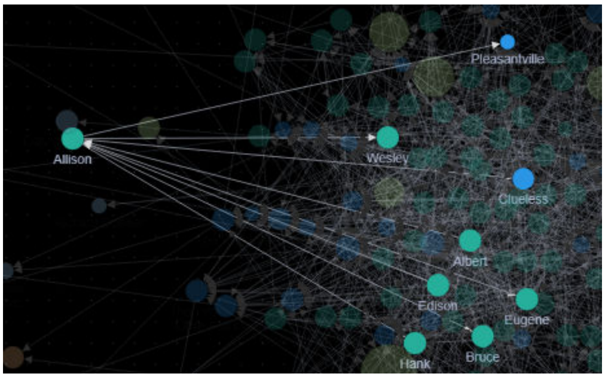
Double-click or right-click the mouse and choose Add Vertex from the shortcut menu to add a vertex.Select a vertex, press and hold Alt + A, and drag the mouse to add an edge.

Table 6-3 Canvas description

Interface Element	Description
 13 /886813 Vertex 9 /892773 Edge	Row 1: **13** indicates the number of vertices displayed on the current canvas and **886813** indicates the total number of vertices in the entire graph. Row 2: **9** indicates the number of edges displayed on the current canvas and **892773** indicates the total number of edges in the entire graph.
 Isolated Vertices	An isolated vertex is a vertex that is not an endpoint of any edge. ● To display isolated vertices in a selected area, press **Ctrl** and click and drag to select an area on the canvas, and then click **Isolated Vertices**. ● To display all isolated vertices in the canvas, click **Isolated Vertices**.
 Neighbor vertices	Select a vertex in the canvas and click neighbor vertices to view all vertices associated.
 Undo	Cancel the previous operation.
 Redo	Redo the canceled previous operation.
 All data	Select **All data** or **Current data**. ● **All data** indicates all data of a graph. ● **Current data** indicates the data rendered on the canvas.

Interface Element	Description
 Enter a vertex ID or multiple IDs separat	After you select **All data** or **Current data**, enter the node ID in the search box, for example, **2**. Press **Enter** or click the query icon to search for the corresponding vertex and render it to the canvas. **NOTE** ● Currently, only a single vertex ID can be entered. ● If you choose **Current data** from the drop-down list, vertices on the current canvas are highlighted.
🧹	Click **Clear** to clear all content on the canvas.
📤	Export the canvas content as a PNG or CSV file (snapshot or vertex and edge file of the current canvas).
⌨	Keyboard shortcuts ● Ctrl+E: Select an associated entity. ● Ctrl+'+': Zoom in. ● Ctrl+'-': Zoom out ● Ctrl+Z: Undo an operation. ● Ctrl+A: Select all. ● Ctrl+Delete: Clear the canvas. ● Delete: Hide vertices. ● Ctrl+Click: Select multiple vertices and edges.
⊕	Zoom in the graph. You can zoom in a graph to at most 600%.
⊖	Zoom out the graph. You can zoom out a graph to 5%.
1:1	Automatic screen adaptation When the displayed graph data is too large (cannot be completely displayed) or too small, you can click this button to quickly adjust it based on the screen size.
 Legend 👁	Whether to display legends

Interface Element	Description
	Quick layout switchover. From left to right: **Force directed**, **Circle**, **Grid**, **Radial-tree**, **Hierarchical**, **CoSE**, and **Double-core**. Figure **Force directed** shows how the graph looks on the canvas. **NOTE** The **Double-core** takes effect only when two nodes are selected.
 Legend user movie genre actor director user Color: Size:	Click a vertex to select the color and size, which is a good way to mark data.
id 78 label user occupation academic/educator gender M Zip-code 85718 userid 32 age 56+	Vertex details. Move the cursor to a non-virtualized vertex. The ID, label, and properties of this vertex are displayed. **NOTE** A maximum of six properties of a vertex can be displayed in the pop-up window. When the number of properties is greater than six, you can view all of them in the filter and property tab as shown in **Editor page**.
Shortcut operations in the drawing area	**Box-select: Shift + Left-click and drag** All vertices in the box are selected, as illustrated in the following figure. 

Interface Element	Description
	Multi-select: Ctrl + Left-click and drag All vertices in the box are selected and highlighted, as illustrated in the following figure.  Pleasantville Allison Wesley Clueless Albert Edison Eugene Hank Bruce
	Select/Deselect: Ctrl + Left-click Press **Ctrl** and left-click a vertex or an edge to select and highlight it. Press **Ctrl** and left-click the vertex or edge again to deselect it.
	Select all: Ctrl + A Select and highlight all vertices and edges.
	Select associated vertices and edges: Ctrl + E Select a vertex and press **Ctrl + E** to highlight all vertices and edges associated with it.
	Hide: Delete Quickly hide a vertex or an edge.
	Adaptation: Ctrl + F Automatically zoom in or out all vertices and edges based on the current screen width and height.
	Zoom out: - Press the **-** key on the keyboard to zoom out the graph.
	Zoom in: = (+) Press the **+** key on the keyboard to zoom in the graph.
	Deselect: Esc Deselect all selected and highlighted vertices and edges.

Interface Element	Description
	Zoom in and zoom out: Scroll the mouse wheel forwards and backwards. Scroll the mouse wheel to zoom in or out the graph.

Figure 6-4 Force directed

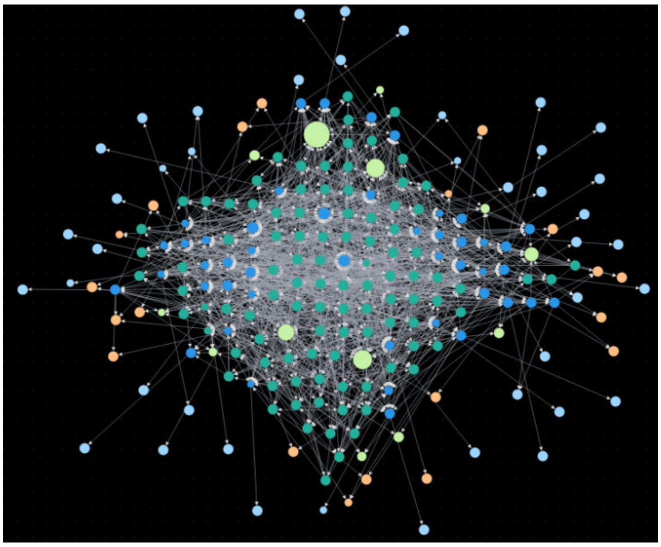


Figure 6-5 Circle

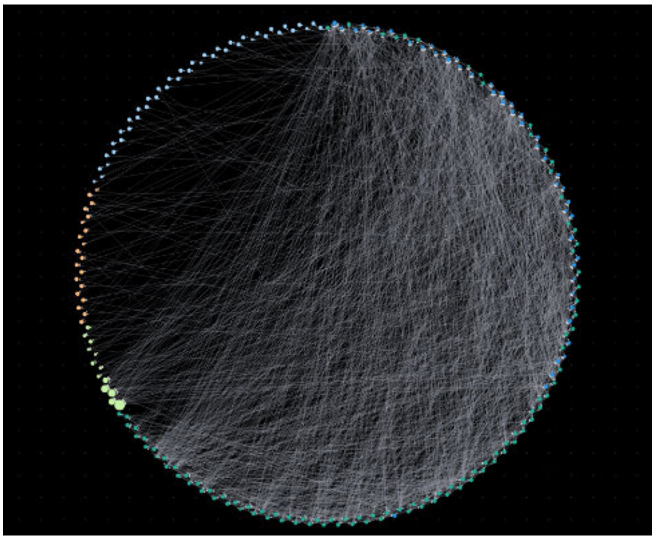


Figure 6-6 Grid

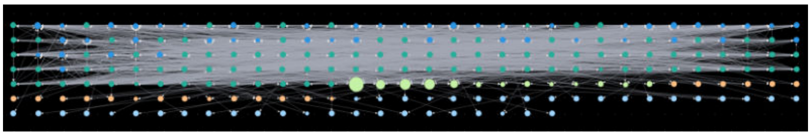


Figure 6-7 Radial-tree

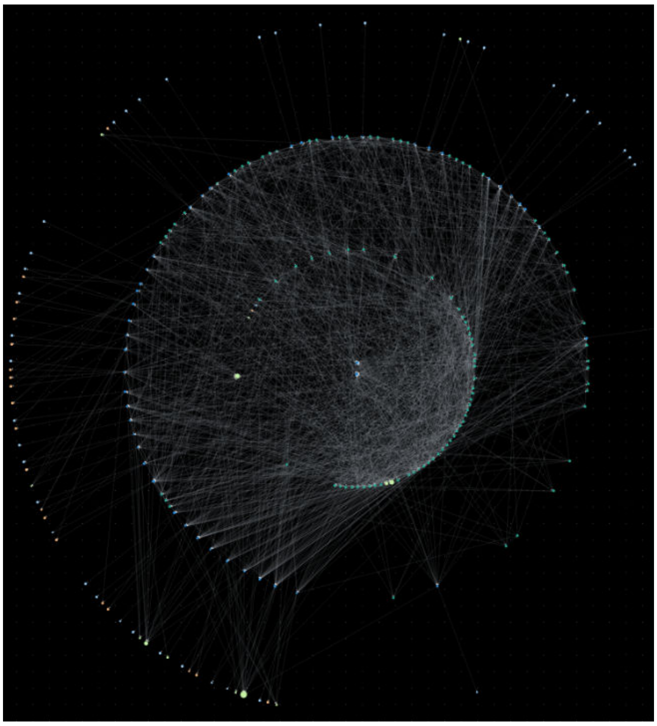


Figure 6-8 Hierarchical

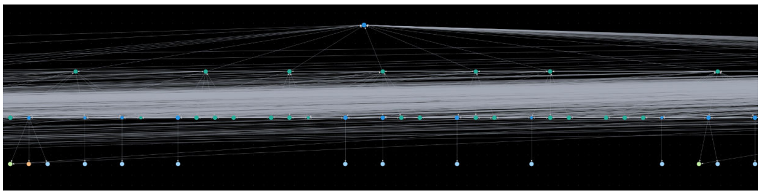


Figure 6-9 CoSE

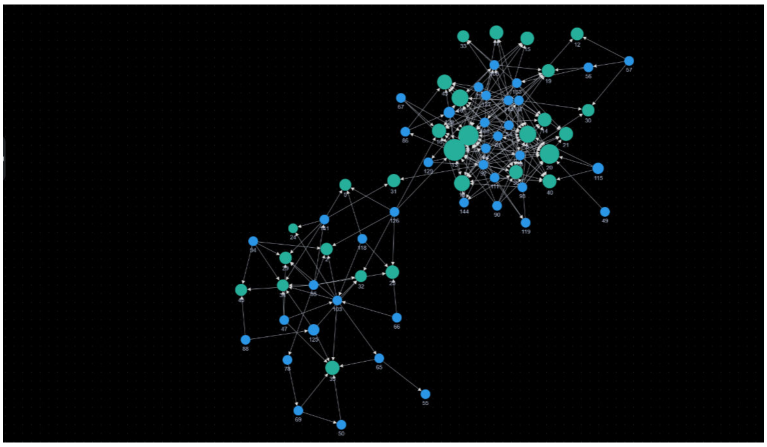
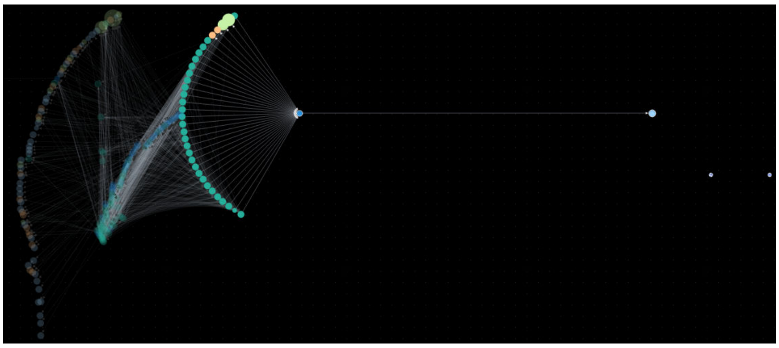


Figure 6-10 Double-core



6.2 Accessing the GES Graph Editor

Scenario

You can use the graph editor to query and analyze graphs. It has extensive built-in algorithms for customers to use in different scenarios of different fields. In addition, it is compatible with the Gremlin and Cypher query languages and supports open APIs. GES is easy to use even for zero-based users.

Procedure

Step 1 Log in to the GES management console and choose **Graph Management** from the navigation pane on the left.

Step 2 On the **Graph Management** page, select the graph to be accessed and click **Access** in the **Operation** column.

You can analyze the graph data on the graph editor. For details, see **Graph Editor**.

----End

6.3 Exploring Graphs

Handful graph exploration tools facilitate your analysis.

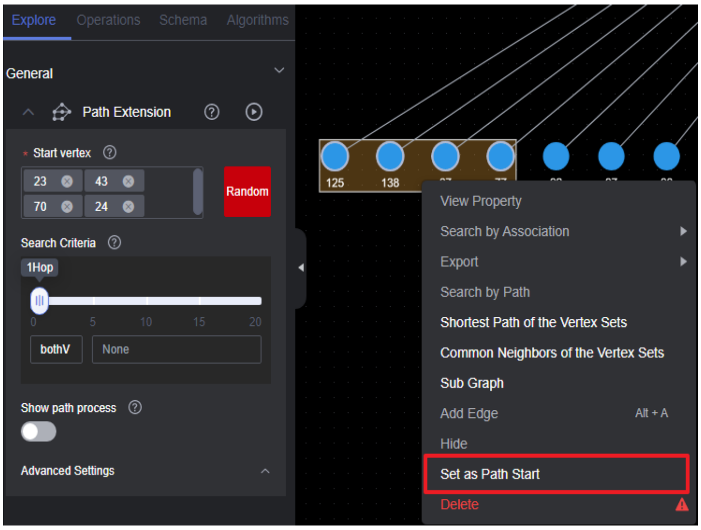
Path Extension

Filters are added to query APIs to search for the desired k-hop vertices or edges.

In the **Path Extension** area on the left of the GES graph editor, set the following parameters:

- **Start Vertex**: IDs of start vertices. You can use any of the following methods to query the vertices:
  a. Press and hold **Shift** and drag a rectangle using the left mouse button to select desired vertices, right-click a vertex, and choose **Set as Path Start** from the shortcut menu. The **Path Extension** will be displayed. The IDs of the selected vertices are automatically filled in the **Start Vertex** box. In this box, you can add or delete vertex IDs. After you finish selecting, click ▶. The query result is displayed on the canvas.

Figure 6-11 Selecting start vertices



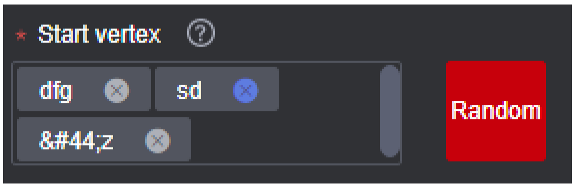
b. Random selection: Click **Random** next to the start vertex box. The system automatically selects vertices in the graph and enters vertex IDs. You can add or delete vertex IDs in the box. After you finish selecting, click . The query result is displayed on the canvas.

c. Specifying one start vertex: Enter the ID of a vertex in the text box and press **Enter**.

d. Specifying a batch of start vertices: Enter IDs of desired vertices in the text box and separate them with commas (,). Then, press **Enter**. A window is displayed when you enter many vertex IDs so you can view them clearly.

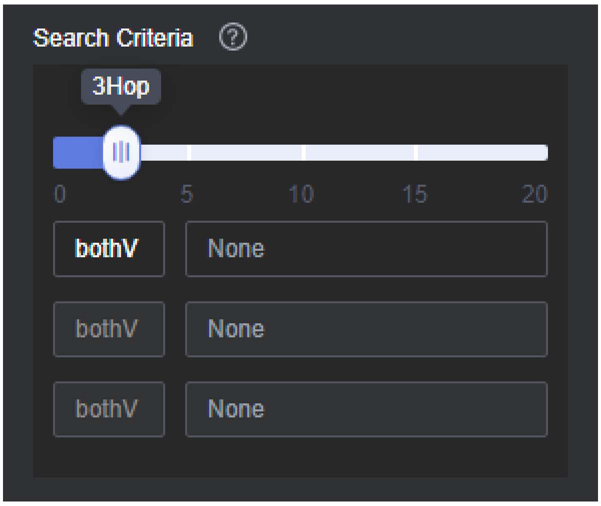
NOTE

Do not enter the same vertex ID repeatedly or an empty value. If the entered vertex ID name contains commas (,), replace the commas with "&#44;".



- **Search Criteria**: Each row in the list corresponds to a query type and criterion of each hop. If there are more hops than criteria, the criteria will be repeated.

Figure 6-12 Search criteria



Refer to the following description to set the search criteria:

- Hop count: Number of search criteria.
- Search criterion: Each hop has a search criterion. Click a search statement text box. The **Search Settings** window is displayed. Enter a search statement.

  The following search criteria operators are available:

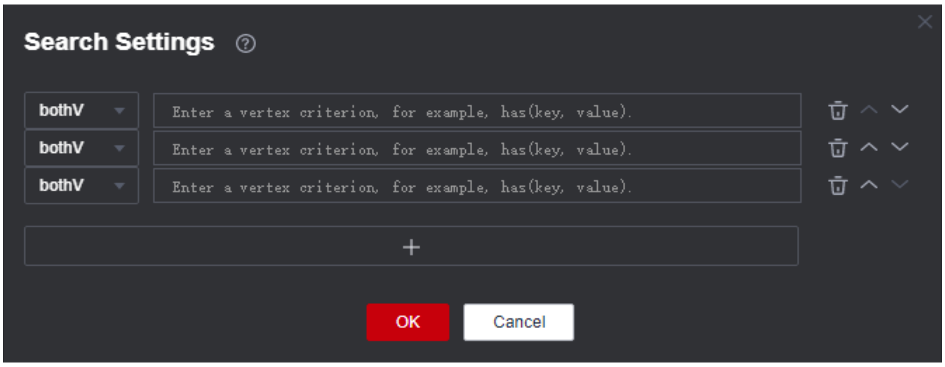
  has: A property key or the value of a property key must be contained.

  hasLabel: The label value must be one of the specified values.

  and: Conditions A and B (can be nested) must be met.

  or: Either condition A or B (can be nested) must be met.

Figure 6-13 Search settings


Search Settings
bothV
Enter a vertex criterion, for example, has(key, value).
bothV
Enter a vertex criterion, for example, has(key, value).
bothV
Enter a vertex criterion, for example, has(key, value).
+
OK
Cancel

NOTE

1. To view a sample criterion, double-click a blank text box. Regular search statements are as follows:

   has(PropertyName): Search for a vertex that has **PropertyName**.

   has(PropertyName, PropertyValue): Search for a vertex that has a property whose name is **PropertyValue**.

   hasLabel(LabelName1,LabelName2): Search for a vertex that has a label whose value is **LabelName1** or **LabelName2**

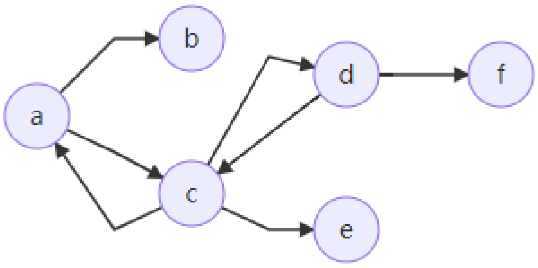
   or(has('name', 'peter'), has('age', '30')): Search for a vertex whose **name** is **Peter** or **age** is **30**.

   and(has('person'),or(has('name','peter'),has('age','30')): Search for a vertex whose **name** is **peter** and **age** is **30**.
2. If there is only one search criterion, the delete, up, and down buttons are grayed out. The first criterion cannot be upshifted, and the last criterion cannot be downshifted. The maximum number of search criteria is 20 (that is, the maximum number of hops).

- **Show path process**: Whether the vertices that are not on the final path will be displayed. This is disabled by default.
- **Advanced Settings**: You can set the expansion strategy here.

  Currently the following traversal methods are available for graph expansion:
  - **ShortestPath**: This method traverses all the shortest paths from the start vertex to every vertex in the graph. This effectively suppresses the exponential growth of the query volume in multi-hop queries.
  - **Walk**: Duplicate vertices are not filtered during traversal.

  NOTE



As shown in the figure, the third-hop neighbor of vertex **a** is queried.

If you use the walk method, the paths are: **a->c->a->b**, **a->c->d->f**, **a->c->d->c**, and **a->c->a->c**.

Vertices **a** and **c** appear repeatedly in the paths such as **a->c->a->b** and **a->c->d->c**. Using **ShortestPath** can reduce duplicate paths, speed up the query process, and reduce the number of queries in this process.

For **ShortestPath**, the query process generates the **a->c->d->f** path only.

6.4 Adding Custom Operations

Scenario

You can add custom operations executed by calling APIs. You can create shortcut operation sets.

Procedure

1. In the **Operations** tab on the left of the graph editor, click **Edit** . The **Add Operation** button is displayed.
2. Click **Add Operation** and set the following parameters in the displayed dialog box:
   - **Name**: Enter a name for the custom operation.
   - **API Type**: **cypher**, **gremlin**, **algorithm**, and **path_query** are supported.
   - **Request Body**: Enter the request body for the calling the API.
   - **Description**: Add a description for the operation.
3. Click **OK**. These parameters cannot be changed after the operation is added.
4. The new custom operation is displayed in the **Operations** tab. You can click the run button to execute the operation and view the results on the canvas.

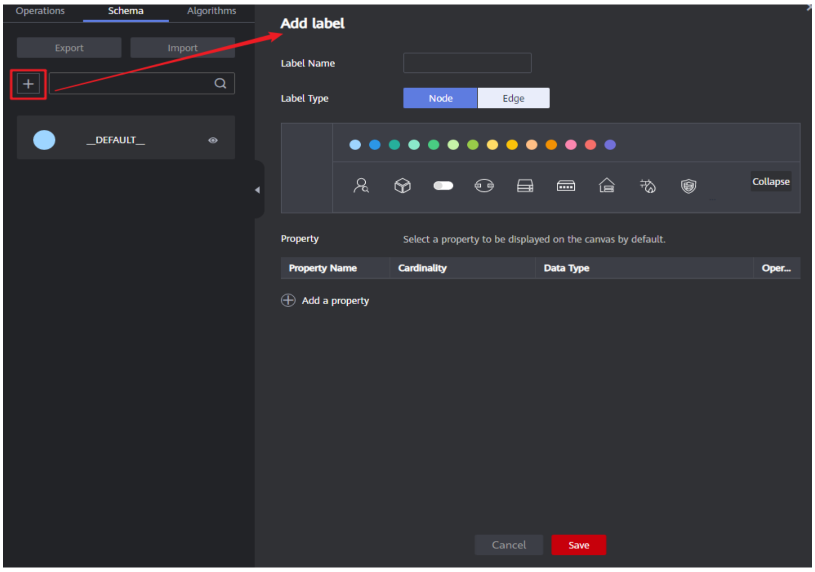
6.5 Editing Schema

Adding a Label

In the metadata list on the left of the graph editor, click + to add a label.

- Set **Label Name** to the name of the label to be added.
- Set **Label Type** to **Node** or **Edge**, and select color and label to distinguish vertices.
- Add properties. By default, only the first added property is displayed on the canvas. You can manually adjust the property to be displayed. The canvas will respond in real time.

Figure 6-14 Adding a label



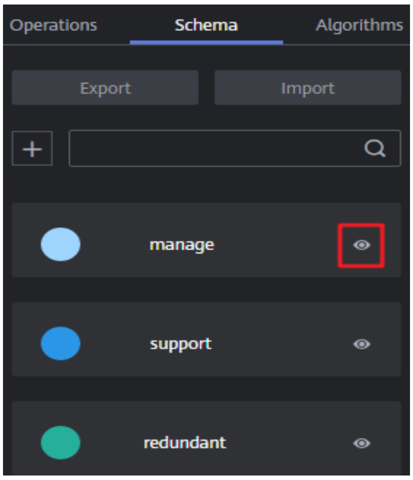
Select a property to be displayed on the canvas by default.

Hiding a Label

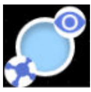
- Hide all vertices and edges of a label.

  In the metadata list on the left of the graph editor, click the eye button next to metadata to hide all vertices and edges of the metadata in the analysis result.

  Figure 6-15 Hiding a label

- Hide the vertices and edges of a selected label

  On the canvas, click any vertex in the graph. The selected vertex is displayed as .

  – is a label-based hide button. You can click this button next to a label to hide the vertices and edges of the selected label. That is, these vertices and edges are not displayed or dimmed on the canvas.

  – is a label-based display button. You can click the button to display the vertices and properties of the label.

Importing and Exporting Labels

You can import the metadata, edge data, and vertex data of a graph to or export them from an OBS bucket.

- Import: Click **Import** in the metadata list. In the dialog box that is displayed, set **Metadata**, **Edge Data**, **Vertex Data**, **Log Storage Path**, **Edge Processing**, and **Import Type**, and click **OK** to import the data from the OBS bucket to a graph.

  – **Log Storage Path**: Stores vertex and edge data sets that do not comply with the metadata definition, as well as detailed logs generated during graph import.

  – **Edge Processing**: Includes **Allow repetitive edges**, **Ignore subsequent repetitive edges**, **Overwrite previous repetitive edges**, and **Ignore labels on repetitive edges**. Repetitive edges have the same source vertex and target vertex. When labels are considered, repetitive edges must have the same source and target vertices and the same labels.

- Export: Click **Export** in the metadata list. In the dialog box that is displayed, set **Metadata Name**, **Vertex Data Set**, **Edge Data Set**, and **Export Path**, and click **OK** to export the data to the OBS bucket.

6.6 Gremlin Query

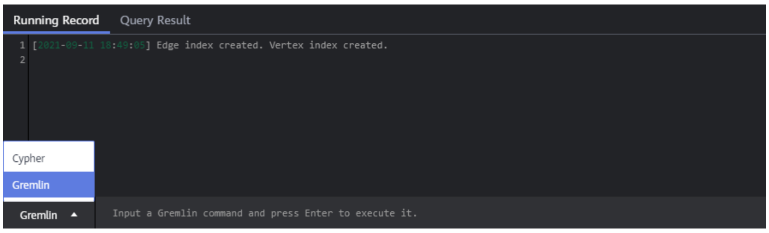
Scenario

Gremlin is a graph traversal language in the open source graph calculation framework of Apache TinkerPop. You can use Gremlin to query, modify, and traverse graph data as well as filter properties.

Procedure

1. Log in to the GES graph editor. For details, see **Accessing the GES Graph Editor**.
2. In the graph data query area, click the drop-up button to choose to Gremlin query. Enter a query statement and press **Enter** to run the statement.

Figure 6-16 Switching to Gremlin query



```
[2021-09-11 18:49:05] Edge index created. Vertex index created.
```

```
Input a Gremlin command and press Enter to execute it.
```

Gremlin Statement

Typical query commands are as follows:

- Querying vertices

  g.V().limit(100): This command is used to query all vertices and return only 100 vertices. You can also use the **range (x, y)** operator to obtain vertices within the specified quantity.

  g.V().hasLabel('movie'): This command is used to query vertices whose label value is **movie**.

  g.V('11'): This command is used to query the vertex whose ID is **11**.

  NOTE

  1. The **g.V ()** is not recommended because the query result cannot be completely displayed if the vertex scale is large.
  2. To prevent query timeout due to a large data volume, add the **limit** parameter and set it less than **1,000**.

- Querying edges

  g.E(): This command is used to query all edges. You are not advised using this command without filter criteria or limit to the returned results.

  g.E('55-81-5'): This command queries the edge whose ID is **55-81-5**.

g.E().hasLabel('rate'): This command queries edges whose label value is **rate**.

g.V('46').outE('rate'): This command queries the edge whose ID is **46** and all its labels are **rate**.

- Querying properties

  g.V().limit(3).valueMap(): This command is used to query all properties of a vertex. (You can specify a parameter to query only one vertex. All properties of the vertex will be displayed in one row.)

  g.V().limit(1).label(): This command is used to query the label of a vertex.

  g.V().limit(10).values('userid'): This command is used to query the **name** property of a vertex. (You can leave the parameter blank to query all properties. Each property value is displayed in one row, without the key).

- Adding a vertex

  g.addV('user').property(id,'600').property('age','18-24'): This command adds a vertex whose label is **user**, ID is **600**, and age ranges from **18** to **24**.

- Deleting a vertex

  g.V('600').drop(): This command deletes the vertex whose ID is **600**.

- Adding an edge

  g.addV('user').property(id,'501').property('age','18-24')

  g.addV('movie').property(id,'502').property('title','love')

  g.addE('rate').property('Rating', '4').from(V('501')).to(V('502'))

  The preceding commands add two vertices and an edge. The two vertex IDs are 501 and 502.

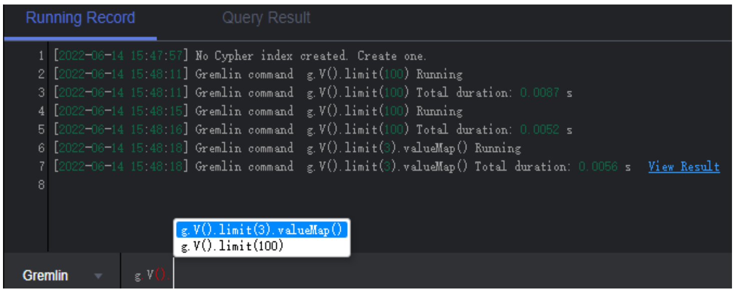
- Deleting an edge

  g.E('501-502-0').drop(): This command deletes the edge whose ID is **501-502-0**.

NOTE

1. You can press the up and down arrow keys in the text box to view historical query commands.
2. When you enter a syntax keyword, the system automatically displays historical statements with the same keyword.

Figure 6-17 Historical queries



```
[2022-06-14 15:47:57] No Cypher index created. Create one.
[2022-06-14 15:48:11] Gremlin command  g.V().limit(100) Running
[2022-06-14 15:48:11] Gremlin command  g.V().limit(100) Total duration: 0.0087 s
[2022-06-14 15:48:15] Gremlin command  g.V().limit(100) Running
[2022-06-14 15:48:16] Gremlin command  g.V().limit(100) Total duration: 0.0052 s
[2022-06-14 15:48:18] Gremlin command  g.V().limit(3).valueMap() Running
[2022-06-14 15:48:18] Gremlin command  g.V().limit(3).valueMap() Total duration: 0.0056 s  View Result
```

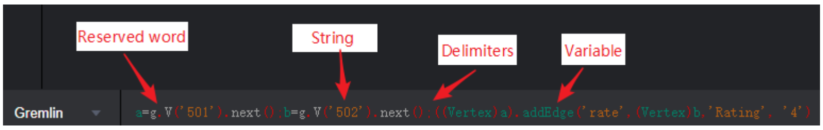
```
g.V().limit(3).valueMap()
g.V().limit(100)
```

```
g.V().
```

3. Keywords in the text box are displayed in different colors.
   - Reserved words in gray

     Note: A reserved word is predefined in the syntax system of a programming language. Reserved words vary depending on programming languages.
   - String values in orange
   - Delimiters in red. Regular delimiters including square brackets **[]**, curly brackets **{}**, parenthesis **()**, commas (**,**), and semicolons (**;**).
   - Variables in green

Figure 6-18 Gremlin keywords



```
a=g.V('501').next();b=g.V('502').next();((Vertex)a).addEdge('rate',(Vertex)b,'Rating', '4')
```

Gremlin Syntax Optimization

GES integrates the OLTP function of Gremlin, enhances some features, and optimizes the strategy.

- **Enhanced Text Predicate**

  g.V().has('name', Text.textSubString('xx'))

Predicate	Description
textSubString	Substring
textCISubString	Substring that ignores cases
textFuzzy	Fuzzy match
textPrefix	Prefix query
textRegex	Regular expression match

NOTE

When you specify a schema, do not name the attributes **id**, **label**, **property**, or **properties**.

When you do Gremlin queries with many steps, the results will be converted into a map. Two identical keys are not allowed in a map structure. If multiple identical keys are inserted into a map, the key value will be overwritten or this operation is canceled. If you set an attribute name to **id**, **label**, **property**, or **properties**, the returned results will be incomplete because in many queries the graph ID is returned together with the attribute ID.

Reference

Table 6-4 shows how Gremlin in GES differs from open source Gremlin.

Table 6-4 GES Gremlin differences

Difference	Description
Vertex and Edge IDs	An edge ID consists of the source vertex ID, target vertex ID, and index that distinguishes duplicate edges. The three parts are connected by hyphens (-), for example, sid-tid-index. Edge and vertex IDs must be the string type.
User Supplied IDs	Users can only provide vertex IDs without hyphens (-).
Vertex Property IDs	Both edge and vertex properties do not have IDs. The returned IDs are vertex IDs.
Vertex and Edge Property	Vertex and edge properties are defined by metadata files in GES. Therefore, you cannot add or delete properties, but you can use **property()** and **remove()** to modify property values. The value set by **property()** is determined by the corresponding parameter. **remove()** converts string properties into empty strings, digital properties into 0, and list properties into empty lists.
Variables	The GES graph structure does not support the **variables** feature.
Cardinality	GES supports the single and list cardinality. The value type of a vertex property is defined by the metadata file. Therefore, no new property is added when you set the property value.
Transactions	During GES Gremlin implementation, transactions are not explicitly used.

You can use the **feature** function to view the supported Gremlin features. If **false** is displayed, GES does not support the feature. If **true** is displayed, GES supports the feature.

```
gremlin> graph.features()
==>FEATURES
```

NOTE

Currently, the following step commands are not supported:

- tryNext()
- explain()
- tree()

6.7 Cypher Query

Scenario

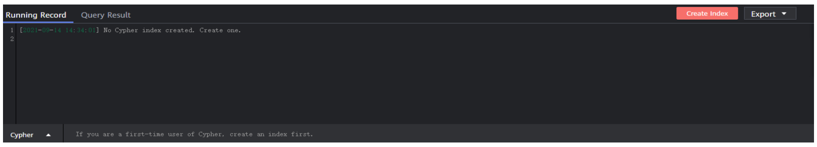
Cypher is a declarative graph query language. You can use Cypher statements to obtain query result and modify data in GES.

Procedure

1. Access the GES graph editor. For details, see **Accessing the GES Graph Editor**.
2. Use label-based vertex and edge indexes during Cypher query.

   If this is your first time using Cypher, click **Create Index** in the upper right corner of the result display area. You do not need to perform this operation in subsequent operations.

   Figure 6-19 Result display area

```
[2021-09-14 14:34:01] No Cypher index created. Create one.
```

   If you are a first-time user of Cypher, create an index first.

3. In the graph data query area, enter the query statement and press **Enter**.

Cypher Statements

The following are typical query statements.

- Querying a vertex

  match (n:movie) return n: Query the vertex whose label is **movie**.

  match (n) return n limit 100: Query details about 100 vertices.

  match (n{Occupation:'artist'}) return id(n), n.Gender limit 100: Query the first 100 vertices whose **Occupation** is **artist**, and return their IDs and genders.

  match (n) where id(n)='Vivian' return n: Query the vertex whose ID is **Vivian**.

  match (n) return n skip 50 limit 100: Query all vertices of a graph. Skip the first 50 vertices, and return a total of 100 vertices.
- Querying an edge

  match (n)-[r]->(m) return r, n, m: Query all edges. Return the edges and vertices at both ends.

match (n)-[r:rate]->(m) return r, n, m: Query the edges whose label is **rate**.

match (n)-[r:rate|:friends]-(m) where id(n)='Vivian' return n,r,m: Query all edges whose start vertex is **Vivian** and edge label is **rate** or **friends**.

- Searching by path

  match p=(n:user)--(m1:user)--(m2:movie) return p limit 100: Query the paths whose start vertex is **user**, first-hop end vertex is **user**, and second-hop end vertex is **movie**. Returns the first 100 paths.

- Aggregating and deduplicating based on groups

  match (n) return count(*): Query the number of all vertices in a graph.

  match (n:user) return n.Gender, count(n): Collect statistics on the number of **user** vertices in every gender.

  match (n:user) return distinct n.Occupation: Return deduplicated occupations of all **user** vertices.

- Sorting

  match (n:user) return id(n) as name order by name: Change IDs of all **user** vertices to **name**, and sort the vertices by **name**.

- Creating a vertex

  create(n:movie{_ID_:'The Captain', Year:2019})return n: Create a vertex whose ID is **The Captain**, label is **movie**, and **Year** is **2019**. Return the vertex.

  create(n:movie{_ID_:'The Captain', Year:2019})-[r:rate]->(m:movie{_ID_:'The Climbers',Title: 'The Climbers', Year:2019}) return r: Create two vertices and their associated edges.

- Creating an edge

  match (n),(m) where id(n)= 'The Captain' and id(m)= 'Lethal Weapon' create (n)-[r:rate]->(m) return r : Create an edge whose label is **rate** between two vertices with specified IDs. (You are advised to use this query in 2.2.21 and later versions.)

- Modifying properties

  match (n) where id(n)= 'The Captain' set n.Title= 'The Captain' return n: Search for the vertex whose ID is **The Captain** and change the attribute **Title** to **Ji Zhang**.

- Deleting a vertex

  match (n) where id(n)=' The Captain' delete n: Search and delete the vertex whose ID is **The Captain**.

  match (n) where id(n)=' "detach delete n": Search for the vertex whose ID is **The Captain**. Delete the vertex and its edges.

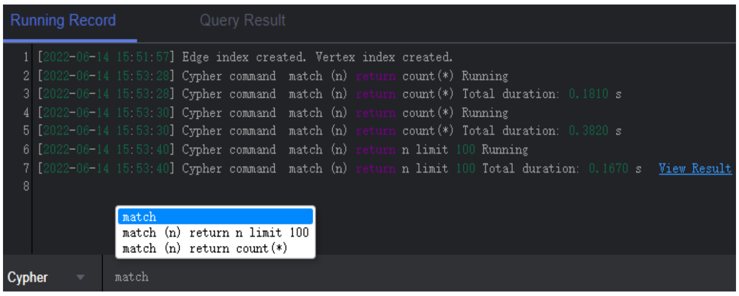
- Querying a schema

  If you call **db.schema()** independently, only the schema metadata of the vertices is returned. Multiple isolated vertices are displayed on the canvas.

NOTE

1. You can press the up and down arrow keys in the text box to view historical query commands.
2. When you enter a syntax keyword, the system automatically displays historical statements with the same keyword.

Figure 6-20 Historical queries



```
[2022-06-14 15:51:57] Edge index created. Vertex index created.
[2022-06-14 15:53:28] Cypher command  match (n) return count(*) Running
[2022-06-14 15:53:28] Cypher command  match (n) return count(*) Total duration: 0.1810 s
[2022-06-14 15:53:30] Cypher command  match (n) return count(*) Running
[2022-06-14 15:53:30] Cypher command  match (n) return count(*) Total duration: 0.3820 s
[2022-06-14 15:53:40] Cypher command  match (n) return n limit 100 Running
[2022-06-14 15:53:40] Cypher command  match (n) return n limit 100 Total duration: 0.1670 s  View Result
```

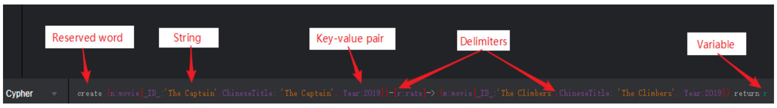
```
match
match (n) return n limit 100
match (n) return count(*)
```

```
match
```

3. Keywords in the text box are displayed in different colors.
   - Reserved words in gray

     Note: A reserved word is predefined in the syntax system of a programming language. Reserved words vary depending on programming languages.
   - String values in orange
   - Key-value pairs in purple. They are of the non-string type in the *key*:*value* format.
   - Delimiters in red. Regular delimiters including square brackets **[]**, curly brackets **{}**, parenthesis **()**, commas (**,**), and semicolons (**;**).
   - Variables in green

Figure 6-21 Cypher keywords



```
create (n:movie{_ID_:'The Captain',ChineseTitle: 'The Captain', Year:2019})-[r:rate]-> (m:movie{_ID_:'The Climbers',ChineseTitle: 'The Climbers', Year:2019}) return r
```

6.8 Analyzing Graphs Using Algorithms

Scenario

In scenarios such as social networking and e-commerce recommendation, graph algorithms can be used for relationship analysis and social community discovery. For example, you can use the PageRank algorithm to analyze key roles in social networks, use the Shortest Path algorithm to find relationship paths and recommend friends among the roles, and use the K-core algorithm to discover small circles.

Procedure

Step 1 Log in to the GES graph editor. For details, see **Accessing the GES Graph Editor**.

Step 2 In the algorithm library area, you can select an algorithm and set its parameters.

Algorithm List shows the algorithms supported by GES and **Algorithms** describes the algorithm details.

Step 3 Run the algorithm. You can view the query result after the analysis is complete.

> NOTE
>
> Only the results of 500 vertices are displayed due to the size of the result display area. If you want to view the complete query results of global iterative algorithms, such as the PageRank algorithm, you can call the algorithm APIs.

Step 4 Adjust the parameters, and run the algorithm again. PageRank value is different this time, but the top ranking does not change.

Step 5 Perform association prediction to obtain the association degree.

----End

6.9 Analyzing Graphs on the Canvas

Scenario

The canvas intuitively displays the graph data. You can also edit and analyze data in this area.

For details about the shortcut keys and interface elements on the canvas, see **Table 6-3**.

Procedure

Step 1 Log in to the GES graph editor. For details, see **Accessing the GES Graph Editor**.

Step 2 On the canvas, right-click a vertex or an edge, and perform the following operations:

- **View Property**

  Select **View Property** to view the property information about the selected vertex or edge on the **Property** tab page.

- **Search by Association**

  You can select **OUT**, **IN**, and **ALL** to expand vertices related to the current vertex.

  - **OUT**: Query the vertices using this vertex as the source vertex.
  - **IN**: Query the vertices using this vertex as the target vertex.
  - **ALL**: Query all vertices of **OUT** and **IN**.

- **Export**

  Export the graph displayed on the canvas.

- **Search by Path**

  Query paths between two vertices. All possible paths are listed.

  Procedure: Hold down **Ctrl** and click two vertices. The first is the source vertex and the second is the target vertex. Then, Right-click and choose **Search by Path** from the shortcut menu.

NOTE

This option is valid only when two vertices are selected. Otherwise, it is dimmed.

After this function is executed, the canvas is cleared, and then the queried vertex and edge data is returned and rendered in the canvas. A path is formed based on the selected two vertices.

- **Shortest Path of the Vertex Sets**
  a. Hold down **Shift** and box-select a group of vertices (a single vertex or multiple vertices).
  b. Hold down **Shift** and box-select another group of vertices (a single vertex or multiple vertices).
  c. Right-click in the selection box and choose **Shortest Path of the Vertex Sets** from the shortcut menu.
  d. In the dialog box that is displayed, you can edit the selected two sets of vertices and click **+** to quickly add vertices.
  e. Click **Run**. The shortest paths between two vertex sets are returned.
- **Common Neighbors of Vertex Sets**
  – Function

    By box-selecting the common neighbors of two vertex sets, you can intuitively discover the objects associated with the two sets.

  – Procedure
    i. Hold down **Shift** and box-select two vertex sets.
    ii. Right-click a vertex set and choose **Common Neighbors of Vertex Sets** from the shortcut menu.
    iii. In the displayed dialog box, confirm the vertices in the vertex set. You can add or delete vertices and determine whether to carry additional parameters. Then, click **Run**.

      NOTE

      The **Carrying additional constraints** option allows you to limit the result set:

      ○ If this option is not selected, the found common neighbors are the intersection of the neighbors corresponding to the source vertex set and target vertex set.

      ○ If this option is selected, the found common neighbors are not only the intersection of the neighbors corresponding to the source vertex set and target vertex set, but each vertex in the common neighbor set has at least two neighboring vertices in the source vertex set and target vertex set.

    iv. Display the result.
- **Sub Graph**: Press and hold **Ctrl** and select some vertices. The edges between those vertices and the selected vertices form a new graph.
- **Add Edge**: You can add an edge using either of the following methods:
  a. Hold down **Ctrl**, select any two vertices on the canvas, right-click the selected vertices, and choose **Add Edge** from the shortcut menu to add an edge between the vertices. By default, the vertex selected first is the source vertex, and that selected later is the target vertex. After the edge is added, you can select the label of the edge and set the edge properties.

b. Select a vertex, press **Alt+A**, drag the cursor to the target vertex, and left-click to add an edge.

- **Hide**: Hide the selected vertex.
- **Delete**: Delete a vertex, an edge, multiple vertices, and multiple edges, or delete edges and vertices in batches.
  - To delete a vertex /edge, select the vertex/edge and delete it.
  - To delete multiple vertices/edges, press **Ctrl** to select the vertices/edges and delete them.
  - To delete vertices and edges in batches, hold down **Shift** and drag the left key of the mouse to select multiple vertices and edges and delete them.

  After you click **Delete**, a confirmation dialog box is displayed. Confirm information about the vertices and edges you want to delete and click **OK**.

  NOTE

  The vertices and edges will be permanently deleted and cannot be restored. Exercise caution when performing this operation.

Step 3 View the details about a vertex.

Move the cursor to a non-virtualized vertex. The ID, label, and properties of this vertex are displayed.

NOTE

A maximum of six properties of a vertex can be displayed in the pop-up window. When the number of properties is greater than six, you can view all of them in the filter and property tab as shown in **Editor page**.

----End

6.10 Graph Display in 3D View

Scenario

The 3D view of a graph provides you intuitive analysis experience.

NOTE

Constraints:

1. The 3D view is available for 1-billion-edge graphs only.
2. Currently, only PagePank and PersonalRank algorithms are available in the 3D view. You can still use Cypher queries and Gremlin queries. For other algorithms or functions, switch to the 2D view.

Procedure

The following example shows how to view results of the PagePank algorithm in the 3D view graph:

1. In the algorithm area on the left of the graph editor, select the PagePank algorithm and set required parameters.

2. Run the algorithm. After the analysis is complete, you can view the result in the canvas.

3. Click 2D / 3D in the upper left corner of the canvas to switch to the 3D view.

6.11 Filtering Conditions

Scenario

To facilitate graph data analysis, you can set filtering conditions to further filter and analyze the graph data.

Procedure

Step 1 Log in to the GES graph editor. For details, see **Accessing the GES Graph Editor**.

Step 2 Click ◀ on the right of the canvas, or select a vertex on the canvas, right-click it , and choose **View Property**, to display the **Filtering and Property** page.

Step 3 In the **Filtering** area, set the filtering conditions and click **Filter**.

- **Match**: **Vertex** is selected by default. Possible values are **Vertex** and **Edge**.
- **Type**: **All types** is selected by default. You can select the vertex or edge type from the drop-down list. The type is defined by the metadata file you upload.
- **Add filtering condition**: Click **Add filtering condition** to select a property and choose a condition (**Less than**, **Greater than**, **Equal to**, **Not equal to**, **In range**, **Existent**, **Non-existent**, **Greater than or equal to**, or **Less than or equal to**). Properties are defined by the metadata file you upload. You can add multiple filtering conditions or click **Delete** to delete set conditions.

Step 4 After the execution is complete, the filtering result is displayed in the drawing area and result area.

----End

6.12 Editing Properties

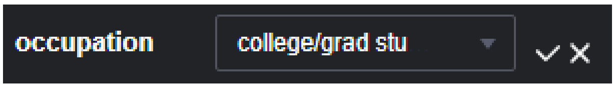
Scenario

The **Property** tab displays information about the properties of the selected vertices and edges. You can edit the properties of a single vertex or edge.

Procedure

Step 1 Right-click a vertex/edge in the canvas and choose **View Property** from the shortcut menu. The **Property** tab is displayed on the right, showing the properties of the selected vertex/edge.

Step 2 Click ✎ next to the property to edit it.



occupation college/grad stu ✓ ✕

Click **Edit All** at the bottom of the property area to edit all the displayed properties. Click **Save All**.

Step 3 Click ✓ after you finish editing.

NOTE

In the **Property** tab, only the properties of a single vertex or edge can be edited. In the **Schema** tab of the metadata area, you can add or delete all properties of a tag, as described in section **Editing Schema**.

----End

6.13 Statistics Display

Scenario

To view the number of tags and vertex weights of specified vertices and edges, you can select the vertices and edges in the canvas. For details about the concepts of vertices and edges, see **Graph Data Formats**.

Procedure

Step 1 Log in to the GES graph editor. For details, see **Accessing the GES Graph Editor**.

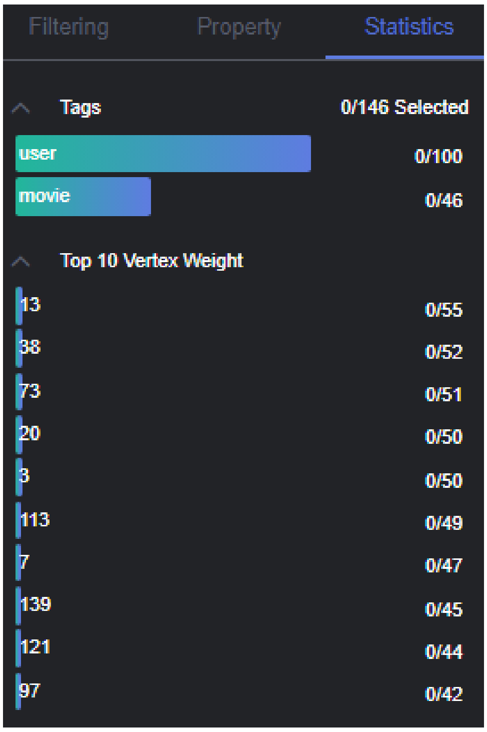
Step 2 Click ◂ on the right side of the canvas. The **Filter**, **Property**, and **Statistics** tabs are displayed. Click the **Statistics** tab.

- **Tags**: Statistics on all tags, and the number of vertices and edges of each tag on the current canvas
- **Top 10 Vertex Weight**: Top 10 vertices with the largest number of edges in the current graph

In the following example, there are two tags: **user** and **movie**. There are 100 vertices tagged with **user** and 46 vertices tagged with **movie**.

In the example graph, the vertex whose ID is 13 has the largest weight. There are 55 edges in total. The vertex ranked at 10 is vertex 97. There are 42 edges in total.

Figure 6-22 Tag statistics



Tags	0/146 Selected
user	0/100
movie	0/46

Top 10 Vertex Weight	
13	0/55
38	0/52
73	0/51
20	0/50
3	0/50
113	0/49
7	0/47
139	0/45
121	0/44
97	0/42

Step 3 Press **Shift** and drag the left key of the mouse to select vertices and edges in the graph. The tags of the selected vertices and edges are displayed along with the top 10 vertices with the highest weights among the selected verities.

----End

6.14 View Running Records

Scenario

The system records your operations so that you can learn the execution progress and completion time when analyzing data.

Procedure

Step 1 Log in to the GES graph editor. For details, see **Accessing the GES Graph Editor**.

Step 2 Execute a Gremlin or Cypher command for querying or an algorithm for analysis. Then the **Running Record** tab page displays the operation statistics.

- After a Gremlin command is executed, the local time, and the command name and content are displayed in the **Running Record** tab page.
- After a Cypher command is executed, the local time, and the command name and content are displayed in the **Running Record** tab page.
- After an algorithm is executed, the local time, algorithm name, and running status are displayed on the **Running Record** tab page. The running statuses include **waiting to run**, **running**, and **ran successfully**.

----End

6.15 Viewing Query Results

Scenario

After data analysis is complete, you can directly view the result on the canvas or on the **Query Result** tab page.

Procedure

Step 1 Log in to the GES graph editor. For details, see **Accessing the GES Graph Editor**.

Step 2 Execute a Gremlin or Cypher command for querying or an algorithm for analysis. Then the **Query Result** tab page displays the query results.

If the result is too large to be completely displayed on the canvas and result area, click the **Export** button in the upper right corner to download the analysis result.

- Run a Gremlin command. The command output is quickly displayed. For example, if you run the **g.V().limit(100)** command, the result is as follows:
- Run a Cypher command. The command output is quickly displayed. For example, if you run the **match (n) return n limit 100** command, the result is as follows:
- Run an algorithm. The running time and result are displayed. For example, if you run PageRank, the result is as follows:

----End

7 Viewing Graph Tasks

7.1 Graph Overview

The **Overview** page displays resource information, including **Graph Status**, **Graph Size**, and **Graph Backup**, enabling you to quickly learn the information about existing graphs.

Graph Status

The **Graph Status** pane displays the number of graphs in different statuses. Currently, the system supports the following statuses.

Table 7-1 Graph statuses

Status	Description
Running	Indicates running graphs. Graphs in this status can be accessed.
Preparing	Indicates graphs whose ECSs are being created or started.
Starting	Indicates graphs being started.
Stopping	Indicates graphs being stopped.
Upgrading	Indicates graphs being upgraded.
Importing	Indicates graphs being imported.
Exporting	Indicates graphs being exported.
Rolling back	Indicates graphs being rolled back.
Clearing	Indicates graphs being cleared.
Preparing for resize	Indicates graphs preparing for resize.
Resizing	Indicates graphs being resized.

Status	Description
Rolling back resize	Indicates graphs where resize is being rolled back.
Preparing for expansion	Indicates graphs preparing for expansion.
Expanding	Indicates graphs being expanded.
Stopped	Indicates stopped graphs. Graphs in this status cannot be accessed, but can be restarted.
Abnormal	Indicates abnormal graphs. Graphs in this status cannot be accessed.
Failed	Indicates graphs failed to be created.

Graph Size

The **Graph Size** pane displays the number of graphs in different sizes. Currently, the system supports the sizes described in the following table.

Table 7-2 Graph sizes

Size	Description
10 thousand	Indicates that the number of edges of a graph cannot exceed 10 thousand.
1 million	Indicates that the number of edges of a graph cannot exceed 1 million.
10 million	Indicates that the number of edges of a graph cannot exceed 10 million.
100 million	Indicates that the number of edges of a graph cannot exceed 100 million.
1 billion	Indicates that the number of edges of a graph cannot exceed 1 billion.
1 billion pro	Indicates that the number of edges of a graph cannot exceed 2 billion.
10 billion	Indicates that the number of edges of a graph cannot exceed 10 billion.
100 billion	Indicates that the number of edges of a graph cannot exceed 100 billion.

Graph Backup

You can back up graphs to prevent data loss. The **Graph Backup** pane displays the numbers of graphs with and without backups.

Table 7-3 Backup statuses

Backup Status	Description
Backed up	Indicates the number of graphs that are backed up.
Non-backed up	Indicates the number of graphs that are not backed up.

7.2 Task Center

7.2.1 Management Plane Task Center

Scenario

If you want to view details about creating, backing up, starting, backing up, importing, exporting, and upgrading tasks, you can go to the **Task Center** page.

Procedure

1. In the navigation pane on the left, click **Task Center**.
2. On the **Task Center** page, view the task type, name, associated graph, start time, end time, status, and running result of the graph.
3. In the **Running Result** column, click **View Details** to view the detailed information. You can also click **Cause of Failure** or **Job ID**.

   If the status of a data import task is **Partially successful**, you can click **View Details** to view information such as the type of data that fails to be imported and the number of rows that fail to be imported. To view the cause of failure, check the log path (optional) specified when you import the graph because failure logs are uploaded to the path.
4. On the **Task Center** page, search for a task in any of the following ways:
   a. Setting the time
   b. Selecting the task type
   c. Selecting the task status
   d. Entering an associated graph
   e. Entering a task ID
   f. Searching a task name in the task list

7.2.2 Service Plane Task Center

Scenario

The task center allows you to view details about the historical tasks and asynchronous tasks that are being executed.

Procedure

1. In the navigation pane, choose **Graph Management**. On the displayed page, locate the target graph and choose **More** > **Task Center** in the **Operation** column.

   NOTE

   - The query task center is available for graphs of version 2.2.23 and later.
   - You can access the task center when graphs are in the running, importing, exporting, or clearing states only.
2. In the upper left corner of the **Task Center** page, select a node from the drop-down list to view details about the asynchronous tasks that are being executed or have been executed. The following task information is displayed:
   - **Job ID**: Job ID of an asynchronous task
   - **Task Type**: Type of the asynchronous task, including **ImportGraph** and **VertexQuery**
   - **Original Request**: Original request body sent by the user
   - **Status**: Task status, which can be **Suspended**, **Running**, **Succeeded**, or **Failed**
   - **Progress**: Progress of the task
   - **Start Time**: Time when the task starts. If the task does not start, the start time is empty.
   - **End Time**: Time when the task ends. If the task does not end, the end time is empty.
   - **Operation**: You can suspend the task.
   - **Running Result**: You can view the task details. If the task fails, you can view the failure cause.
3. To view details about an asynchronous task, search the task by its job ID using the search box in the upper right corner of the page.

7.3 Viewing Monitoring Metrics

Scenario

It takes a period of time for transmitting and displaying data. The GES status displayed in the Cloud Eye monitoring data is the status obtained 5 to 10 minutes before. You can view the monitored data of a graph 5 to 10 minutes after it is created.

Prerequisites

- The created graph is running properly.
- The graph has been properly running for at least 10 minutes. For a newly created graph, you need to wait for a while before viewing its metrics.
- Cloud Eye does not display the metrics of a faulty or stopped graph. You can view the monitoring metrics after the graph starts or recovers.

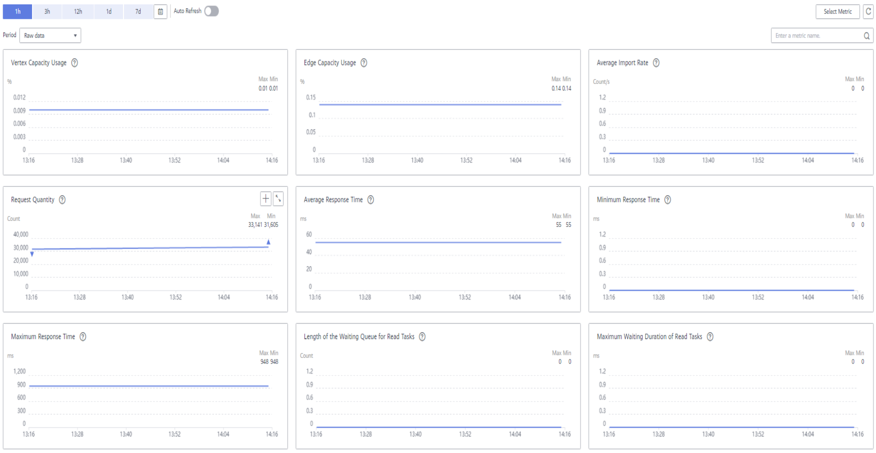
Procedure

Step 1 Log in to the management console.

Step 2 In the navigation pane, choose **Graph Management**. In the **Operation** column, choose **More** > **View Metrics**. The Cloud Eye management console is displayed.

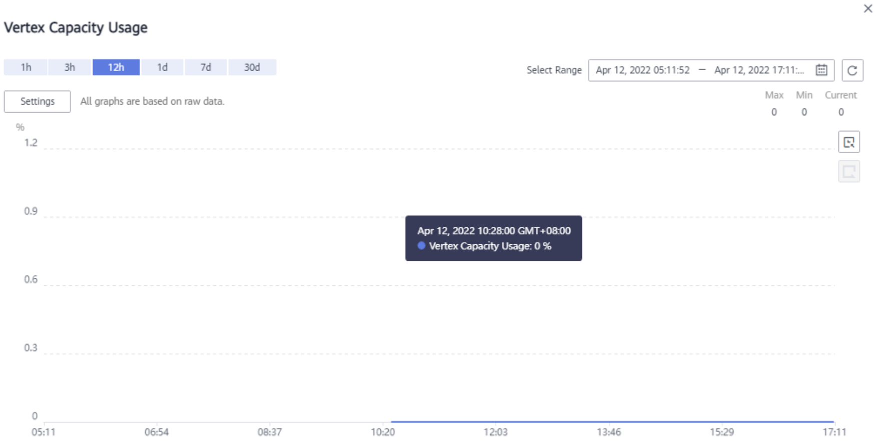
Step 3 On the monitoring page for GES, you can view the figures of all monitoring metrics.

Figure 7-1 Viewing monitoring metrics



Step 4 To view the monitored data of a longer time range, click .

Figure 7-2 Viewing data in a bigger view



All graphs are based on raw data.

Select Range Apr 12, 2022 05:11:52 – Apr 12, 2022 17:11:...

Apr 12, 2022 10:28:00 GMT+08:00
Vertex Capacity Usage: 0 %

Step 5 The system allows you to select a fixed time range or use automatic refresh.

1. Fixed time ranges include **1h**, **3h**, **12h**.

----End

Metrics

This chapter describes metrics reported by GES to the cloud monitoring service as well as their namespaces, lists, and dimensions. You can use the management console and APIs provided by the cloud monitoring service to query the metric and alarm information generated for GES.

NOTE

The namespace of the metrics reported by GES to the cloud monitoring service is SYS.GES.

Table 7-4 GES metrics

Metric ID	Metric	Description	Value Range	Monitored Object	Monitoring Period (Original Metric)
ges001_vertex_util	Vertex Capacity Usage	Capacity usage of vertices in a graph instance. The value is the ratio of the number of used vertices to the total vertex capacity. Unit: %	0-100 Value type: Float	GES instance	1 minute
ges002_edge_util	Edge Capacity Usage	Capacity usage of edges in a graph instance. The value is the ratio of the number of used edges to the total edge capacity. Unit: %	0-100 Value type: Float	GES instance	1 minute
ges003_average_import_rate	Average Import Rate	Average rate of importing vertices or edges to a graph instance Unit: count/s	0-400000 Value type: Float	GES instance	1 minute

Metric ID	Metric	Description	Value Range	Monitored Object	Monitoring Period (Original Metric)
ges004_request_count	Request Quantity	Number of requests received by a graph instance Unit: count	≥ 0 Value type: Int	GES instance	1 minute
ges005_average_response_time	Average Response Time	Average response time of requests received by a graph instance Unit: ms	≥ 0 Value type: Int	GES instance	1 minute
ges006_min_response_time	Minimum Response Time	Minimum response time of requests received by a graph instance Unit: ms	≥ 0 Value type: Int	GES instance	1 minute
ges007_max_response_time	Maximum Response Time	Maximum response time of requests received by a graph instance Unit: ms	≥ 0 Value type: Int	GES instance	1 minute
ges008_read_task_pending_queue_size	Length of the Waiting Queue for Read Tasks	Length of the waiting queue for read requests received by a graph instance. This metric is used to view the number of read requests waiting in the queue. Unit: count	≥ 0 Value type: Int	GES instance	1 minute

Metric ID	Metric	Description	Value Range	Monitored Object	Monitoring Period (Original Metric)
ges009_read_task_pending_max_time	Maximum Waiting Duration of Read Tasks	Maximum waiting duration of read requests received by a graph instance Unit: ms	≥ 0 Value type: Int	GES instance	1 minute
ges010_pending_max_time_ read_task_type	Type of the Read Task That Waits the Longest	Type of the read request that waits the longest in a graph instance. Refer to **Table 7-6** to find the corresponding task name.	≥ 1 Value type: Int	GES instance	1 minute
ges011_read_task_running_queue_size	Length of the Running Queue for Read Tasks	Length of the running queue for read requests received by a graph instance. This metric is used to view the number of running read requests. Unit: count	≥ 0 Value type: Int	GES instance	1 minute
ges012_read_task_running_max_time	Maximum Running Duration of Read Tasks	Maximum running duration of read requests received by a graph instance Unit: ms	≥ 0 Value type: Int	GES instance	1 minute

Metric ID	Metric	Description	Value Range	Monitored Object	Monitoring Period (Original Metric)
ges013_running_max_time_read_task_type	Type of the Read Task That Runs the Longest	Type of the read request that runs the longest in a graph instance. You can find the corresponding task name in GES documentation.	≥ 1 Value type: Int	GES instance	1 minute
ges014_write_task_pending_queue_size	Length of the Waiting Queue for Write Tasks	Length of the waiting queue for write requests received by a graph instance. This metric is used to view the number of write requests waiting in the queue. Unit: count	≥ 0 Value type: Int	GES instance	1 minute
ges015_write_task_pending_max_time	Maximum Waiting Duration of Write Tasks	Maximum waiting duration of write requests received by a graph instance Unit: ms	≥ 0 Value type: Int	GES instance	1 minute
ges016_pending_max_time_write_task_type	Type of the Write Task That Waits the Longest	Type of the write request that waits the longest in a graph instance. Refer to **Table 7-6** to find the corresponding task name.	≥ 1 Value type: Int	GES instance	1 minute

Metric ID	Metric	Description	Value Range	Monitored Object	Monitoring Period (Original Metric)
ges017_write_task_running_queue_size	Length of the Running Queue for Write Tasks	Length of the running queue for write requests received by a graph instance. This metric is used to view the number of running write requests. Unit: count	≥ 0 Value type: Int	GES instance	1 minute
ges018_write_task_running_max_time	Maximum Running Duration of Write Tasks	Maximum running duration of write requests received by a graph instance Unit: ms	≥ 0 Value type: Int	GES instance	1 minute
ges019_running_max_time_write_task_type	Type of the Write Task That Runs the Longest	Type of the write request that runs the longest in a graph instance. You can find the corresponding task name in GES documentation.	≥ 1 Value type: Int	GES instance	1 minute
ges020_computer_resource_usage	Computing Resource Usage	Computing resource usage of each graph instance Unit: %	0-100 Value type: Float	GES instance	1 minute
ges021_memory_usage	Memory Usage	Memory usage of each graph instance Unit: %	0-100 Value type: Float	GES instance	1 minute

Metric ID	Metric	Description	Value Range	Monitored Object	Monitoring Period (Original Metric)
ges022_iops	IOPS	Number of I/O requests processed by each graph instance per second Unit: count/s	≥ 0 Value type: Int	GES instance	1 minute
ges023_bytes_in	Network Input Throughput	Data input to each graph instance per second over the network Unit: byte/s	≥ 0 Value type: Float	GES instance	1 minute
ges024_bytes_out	Network Output Throughput	Data sent to the network per second from each graph instance Unit: byte/s	≥ 0 Value type: Float	GES instance	1 minute
ges025_disk_usage	Disk Usage	Disk usage of each graph instance Unit: %	0-100 Value type: Float	GES instance	1 minute
ges026_disk_total_size	Total Disk Size	Total data disk space of each graph instance Unit: GB	≥ 0 Value type: Float	GES instance	1 minute
ges027_disk_used_size	Disk Space Used	Used data disk space of each graph instance Unit: GB	≥ 0 Value type: Float	GES instance	1 minute
ges028_disk_read_throughput	Disk Read Throughput	Data volume read from the disk in a graph instance per second Unit: byte/s	≥ 0 Value type: Float	GES instance	1 minute

Metric ID	Metric	Description	Value Range	Monitored Object	Monitoring Period (Original Metric)
ges029_disk_write_throughput	Disk Write Throughput	Data volume written to the disk in a graph instance per second Unit: byte/s	≥ 0 Value type: Float	GES instance	1 minute
ges030_avg_disk_sec_per_read	Average Time per Disk Read	Average time used each time when the disk of a graph instance reads data Unit: second	≥ 0 Value type: Float	GES instance	1 minute
ges031_avg_disk_sec_per_write	Average Time per Disk Write	Average time used each time when data is written to the disk of a graph instance Unit: second	GES instance	GES instance	1 minute
ges032_avg_disk_queue_length	Average Disk Queue Length	Average I/O queue length of the disk in a graph instance Unit: count	≥ 0 Value type: Int	GES instance	1 minute

Dimensions

Table 7-5 Dimensions

Key	Value
instance_id	GES instance

Mapping Between Task Types and Names

Table 7-6 Mapping table

Type	Name
100	Querying a vertex
101	Creating a vertex
102	Deleting a vertex
103	Modifying a vertex property
104	Adding a vertex label
105	Deleting a vertex label
200	Querying an edge
201	Creating an edge
202	Deleting an edge
203	Modifying an edge property
300	Querying schema details
301	Adding a Label
302	Modifying a Label
303	Querying a Label
304	Modifying a property
400	Querying graph details
401	Clearing graphs
402	Incrementally importing graph data online
403	Creating graphs
405	Deleting graphs
406	Exporting graphs
407	filtered_khop
408	Querying path details
409	Incrementally importing graph data offline
500	Creating a graph backup
501	Restoring a graph from a backup
601	Creating an index.

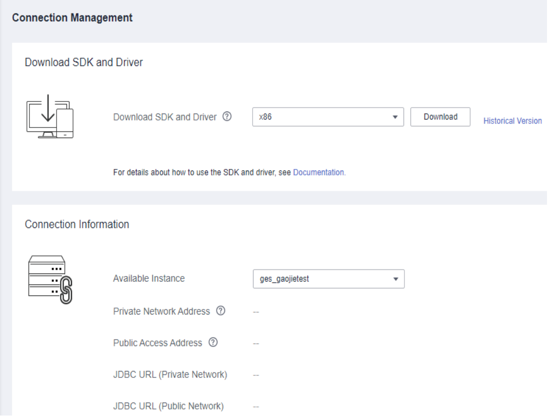
Type	Name
602	Querying an index
603	Updating an index
604	Deleting an index
700	Running the algorithm
800	Querying an asynchronous task

7.4 Managing Connections

After you create a graph instance, you can download the required SDK and driver and view the connection information of the graph.

In the navigation pane on the left, click **Connection Management**. The **Connection Management** page is displayed.

Figure 7-3 Connection management page



Connection Management

Download SDK and Driver

Download SDK and Driver ⓘ x86 ▾ Download Historical Version

For details about how to use the SDK and driver, see Documentation.

Connection Information

Available Instance ges_gaojietest ▾

Private Network Address ⓘ --

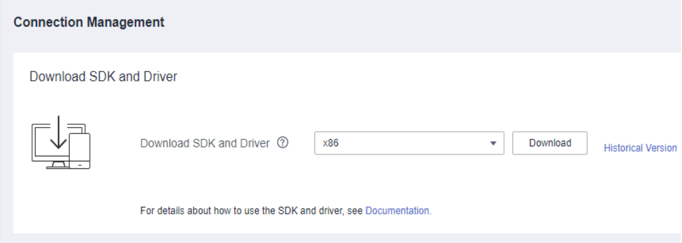
Public Access Address ⓘ --

JDBC URL (Private Network) --

JDBC URL (Public Network) --

Downloading SDK and Driver

Figure 7-4 SDK and driver



Connection Management

Download SDK and Driver

Download SDK and Driver ⓘ x86 ▾ Download Historical Version

For details about how to use the SDK and driver, see Documentation.

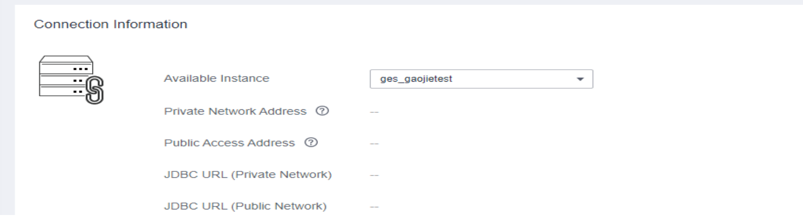
- Download an SDK and driver
  - The SDK encapsulates the service plane APIs. You are advised to use the SDK to access graph instances.

  - You need to download the Cypher-JDBC driver for Cypher API access. For details, see "Using the Cypher JDBC Driver to Access GES".

- Select the CPU architecture supported by the cluster. Currently, **x86** and **Arm** are available. Click **Download** to download the SDK.
- Click **Historical Version** to view historical SDK and driver versions and CPU architecture of the driver. You can click **Download** in the **Operation** column to download the historical driver.

Connection Information

Figure 7-5 Instance information



Connection Information

Available Instance: ges_gaojietest

Private Network Address: --

Public Access Address: --

JDBC URL (Private Network): --

JDBC URL (Public Network): --

Select the name of a created graph instance to view the following information:

- **Private Network Address**: ECSs in the same private network can connect to the graph instance using the private network address.
- **Public Access Address**: You can use the public access address (EIP) to access the graph instance through the Internet. You can bind an EIP to or unbind one from a graph instance.
- **JDBC URL (Private Network)**: Configure this parameter when the JDBC driver executor and the graph instance are in the same private network.
- **JDBC URL (Public Network)**: Configure this parameter when the JDBC driver executor can access the graph instance (with an EIP bound) through the Internet.

8 Configuring Permissions

8.1 Configuring Granular Permissions

Scenario

GES graph instances support granular permission control. You can set the traverse, read, and write permissions for specific properties of specific labels. You are allowed to manage these permissions of a specific label or property of a graph and grant them to a user group.

NOTE

- This function allows you to set granular permissions for graphs of version 2.2.21 or later. You can **upgrade a graph** of an earlier version to 2.2.21 or a later version and then set granular permissions.
- To configure granular permissions, you need to use the main account or assign the Tenant Administrator and Security Administrator permissions to the subaccount.

Procedure

1. Before setting granular permissions, configure the user group first. For details, see **Configuring a User Group**.
2. In the navigation pane, choose **Granular Permissions** > **Permission Configuration**.
3. On the **Permission Configuration** page, you can view the graph name, permission status, enabling time, and operations that can be performed on a graph in the **Running** status.

   NOTE

   1. Only graphs in the **Running** status are displayed on this page.
   2. You can set permissions only when its status is **Disabled**.
   3. You can search for graphs by their names in the upper right corner of the page.
4. Select the graph for which you want to set permission and click **Set** in the **Operation** column. The **Set Permission** page is displayed. You can create metadata permissions and granular permissions on this page.

5. Click **Create** under **Metadata Write Permission** to create permission. After the metadata write permission is created, all labels of the metadata can be modified.
6. Click **Create Policy** under **Granular Permission Policy** to set granular permissions for a graph. You can set label- and property-level graph permissions and grant them to user groups.
   - **Policy Name**: You can set a name or use the default name.
   - **View**: You can configure permissions in form or code view.
   - **Permissions**: You can select labels whose traversal permission will be granted to a certain group of users. You can set read and write permissions of the label properties.
7. Click **Save**. The **Set Permission** page is displayed. You can view the created permission policy in the **Granular Permission Policy** pane.

   NOTE

   In this case, the **Associate User Group** in the **Operation** column is unavailable. You need to enable granular permissions before associating the policy with a user group.
8. Return to the **Permission Configuration** page, locate the graph for which the granular permission has been set, and click **Enable** in the **Operation** column. The permission status changes to **Enabled**.
9. Click **Set** in the **Operation** column to associate the created granular permission with a user group.
10. Click **OK**. On the **Granular Permission Policy** pane, you can view the number of users who have been granted the permission.

8.2 User Groups

Scenario

You can create and manage user groups, and check whether a user group has been associated with permissions.

Procedure

1. On the **User Groups** page, click **Create User Group** in the upper right corner. The **Create User Group** page is displayed.
2. Set the user group name and add group members.
   - **Name**: Set a name for the user group or use the default name.
   - **Members**: All IAM users created under your account are displayed in this area. Select members you want to add to the user group. The selected members are displayed on the right.
     - Click ⌄ on the left of User/ID to view all group members at a time or clear all selected group members.
     - Click □ on the left of User/ID to select all users on the current page.
3. Click **Save** in the lower right corner. The user group is created. The created user group is displayed on the **User Groups** page. You can edit or delete the user group.

NOTE

You are not allowed to delete user groups that have been associated with granular permissions.

8.3 User Permissions

Scenario

You can view the granular permissions of all IAM users created base on your account.

Procedure

1. On the **User Permissions** page, click ⌄ next to the target username to view it granular permissions.
2. Click the permission name to view the details.

9 Algorithms

9.1 Algorithm List

To meet the requirements of various scenarios, GES provides extensive basic graph algorithms, graph analytics algorithms, and graph metrics algorithms. The following table lists the algorithms:

Table 9-1 Algorithm List

Algorithm	Description
PageRank	PageRank, also known as web page ranking, is a hyperlink analysis algorithm used to rank web pages (nodes) based on their search engine results. PageRank is a way of measuring the relevance and importance of web pages (nodes).
PersonalRank	PersonalRank is also called Personalized PageRank. It inherits the idea of the classic PageRank algorithm and uses the graph link structure to recursively calculate the importance of each node. However, unlike the PageRank algorithm, to ensure that the access probability of each node in the random walk can reflect user preferences, the PersonalRank algorithm returns each hop to the source node at a **(1-alpha)** probability during random walk. Therefore, the relevance and importance of network nodes can be calculated based on the source node (the higher the PersonalRank value, the higher the correlation/ importance of the source node).
K-core	K-core is a classic graph algorithm used to calculate the number of cores of each node. The calculation result is one of the most commonly used reference values for determining the importance of a node so that the propagation capability of the node can be better understood.

Algorithm	Description
K-hop	K-hop is an algorithm used to search all nodes in the k layer that are associated with the source node through breadth-first search (BFS). The found sub-graph is the source node's ego-net. The K-hop algorithm returns the number of nodes in the ego-net.
Shortest Path	The Shortest Path algorithm is used to find the shortest path between two nodes in a graph.
All Shortest Paths	The All Shortest Paths algorithm is used to find all shortest paths between two nodes in a graph.
SSSP	The SSSP algorithm finds the shortest paths from a specified node (source node) to all other nodes.
Shortest Path of Vertex Sets	The Shortest Path of Vertex Sets algorithm finds the shortest path between two vertex sets. It can be used to analyze the relationships between blocks in scenarios such as Internet social networking, financial risk control, road network transportation, and logistics delivery.
n-Paths	The n-Paths algorithm is used to find the *n* paths between two vertices on the k layer of a graph. It applies to scenarios such as relationship analysis, path design, and network planning.
Closeness Centrality	Closeness centrality is the average distance from a node to all other reachable nodes. It can be used to measure the time for transmitting information from this node to other nodes. A small **Closeness Centrality** within a node corresponds to a central location of the node.
Label Propagation	The Label Propagation algorithm is a graph-based semi-supervised learning method. Its basic principle is to predict the label information about unlabeled nodes using that of the labeled nodes. This algorithm can create graphs based on the relationships between samples. Nodes include labeled data and unlabeled data, and the edge indicates the similarity between two nodes. Node labels are transferred to other nodes based on the similarity. Labeled data is like a source used to label unlabeled data. Greater node similarity corresponds to an easier label propagation.
Louvain	Louvain is a modularity-based community detection algorithm with high efficiency and effect. It detects hierarchical community structures and aims to maximize the modularity of the entire community network.
Link Prediction	The Link Prediction algorithm is used to calculate the similarity between two nodes and predict their relationship based on the Jaccard measurement method.

Algorithm	Description
Node2vec	By invoking the Word2vec algorithm, the Node2vec algorithm maps nodes in the network to the Euclidean space, and uses vectors to represent the node characteristics. The Node2vec algorithm generates random steps from each node using the rollback parameter **P** and forward parameter **Q**. It combines BFS and DFS. The rollback probability is proportional to 1/P, and the forward probability is proportional to 1/Q. Multiple random steps are generated to reflect the network structures.
Real-time Recommendation	The Real-time Recommendation algorithm is based on the random walk model and is used to recommend nodes that are similar (have similar relationships or preferences) to the input node. This algorithm can be used to recommend similar products based on historical browsing data or recommend potential friends with similar preferences.
Common Neighbors	Common Neighbors is a basic graph analysis algorithm that obtains the neighboring nodes shared by two nodes and further speculate the potential relationship and similarity between the two nodes. For example, it can intuitively discover shared friends in social occasions or commodities that interest both nodes in the consumption field.
Connected Component	A connected component stands for a sub-graph, in which all nodes are connected with each other. Path directions are involved in the strongly connected components and are not considered in the weakly connected components. **NOTE** This algorithm generates weakly connected components.
Degree Correlation	The Degree Correlation algorithm calculates the Pearson correlation coefficient between the source vertex degree and the target vertex degree of each edge. It is used to indicate whether the high-degree nodes are connected to other high-degree nodes in a graph.
Triangle Count	The Triangle Count algorithm counts the number of triangles in a graph without considering the edge directions. More triangles mean higher node association degrees and closer organization relationships.
Clustering Coefficient	The clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together. Evidence suggests that in most real-world networks, and in particular social networks, nodes tend to create tightly knit groups characterized by a relatively high density of ties.
Betweenness Centrality	Betweenness centrality is a measure of centrality in a graph based on shortest paths. The Betweenness Centrality algorithm calculates shortest paths that pass through a vertex.

Algorithm	Description
Edge Betweenness Centrality	The Edge Betweenness Centrality algorithm calculates shortest paths that pass through an edge.
Origin-Destination Betweenness Centrality	The Origin-Destination Betweenness Centrality algorithm calculates shortest paths that pass through a (an) vertex/edge, with the origin and destination specified.
Circle Detection with a Single Vertex	This algorithm solves a classic graph problem: detecting loops in a graph. Vertices on looped paths reflect the importance of the vertices. This algorithm is suitable for transportation analysis and financial risk control.
Common Neighbors of Vertex Sets	This algorithm obtains vertex set neighbors, that are, the intersection of two vertex sets (groups). They are objects that are associated with both sets, for example, common friends, common products of interest, and persons contacting with both communities. You can use neighbors to further speculate potential relationships and the degree of the connection between two vertices.
All Shortest Paths of Vertex Sets	This algorithm is used to discover all shortest paths between two vertex sets. It can be used to analyze the relationships between blocks in scenarios such as social networking, financial risk control, road networks and transportation, and logistics delivery.
Subgraph Matching	This algorithm is used to find all subgraphs of a given small graph that is isomorphic to a given large graph. This is a basic graph query operation and is intended to explore important substructures of a graph.
Filtered All Pairs Shortest Paths	This algorithm is used to search for the shortest path between any two vertices in the graph that meets the condition. In a specific application scenario, you need to set a start vertex set (**sources**) and end vertex set (**targets**) as input for this algorithm. This algorithm returns the required shortest paths between the start and the end vertex sets.
Filtered All Shortest Paths	This algorithm allows you to search query results of the Shortest Path algorithm for the paths that meet the conditions between two vertices in a graph.
TopicRank	The TopicRank algorithm is one of commonly used algorithms for ranking topics by multiple dimensions. For example, this algorithm is applicable to rank complaint topics obtained through a government hotline.

Algorithm	Description
Filtered n-Paths (2.2.22)	The filtered n-Paths algorithm is used to find no more than n k-hop loop-free paths between the source and target vertices. The start vertex (source), end vertex (target), number of hops (k), number of paths (n), and filter criteria (filters) are the parameters for the algorithm.

9.2 PageRank

Overview

PageRank, also known as web page ranking, is a hyperlink analysis algorithm used to rank web pages (nodes) based on their search engine results. PageRank is a way of measuring the relevance and importance of web pages (nodes).

- If a web page is linked to many other web pages, the web page is of great importance. That is, the PageRank value is relatively high.
- If a web page with a high PageRank value is linked to another web page, the PageRank value of the linked web page increases accordingly.

Application Scenarios

This algorithm applies to scenarios such as web page sorting and key role discovery in social networking.

Parameter Description

Table 9-2 PageRank algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
alpha	No	Weight coefficient (also called damping coefficient)	Double	A real number between 0 and 1 (excluding 0 and 1)	0.85
convergence	No	Convergence	Double	A real number between 0 and 1 (excluding 0 and 1)	0.00001
max_iterations	No	Maximum iterations	Int	1-2,000	1,000

Parameter	Mandatory	Description	Type	Value Range	Default Value
directed	No	Whether to consider the edge direction	Bool	**true** or **false**	true

NOTE

- **alpha** determines the jump probability coefficient, also called damping coefficient, which is a computing control variable in the algorithm.
- **convergence** indicates the upper limit of the sum of each absolute vertex change between an iteration and the last iteration. If the sum is less than the value of this parameter, the computing is considered converged and the algorithm stops.
- When the convergence is set to a large value, the iteration will stop quickly.

Precautions

When the convergence is set to a large value, the iteration will stop quickly.

Example

Set parameters **alpha** to **0.85**, **coverage** to **0.00001**, **max_iterations** to **1,000**, and **directed** to **true**. The sub-graph formed by top nodes in the calculation result is displayed on the canvas. The size of a node varies with the PageRank values. The JSON result is displayed in the query result area.

9.3 PersonalRank

Overview

PersonalRank is also called Personalized PageRank. It inherits the idea of the classic PageRank algorithm and uses the graph link structure to recursively calculate the importance of each node. However, unlike the PageRank algorithm, to ensure that the access probability of each node in the random walk can reflect user preferences, the PersonalRank algorithm returns each hop to the source node at a **(1-alpha)** probability during random walk. Therefore, the relevance and importance of network nodes can be calculated based on the source node. (The higher the PersonalRank value, the higher the correlation/importance of the source node.)

Application Scenarios

This algorithm applies to fields such as commodity, friend, and web page recommendations.

Parameter Description

Table 9-3 PersonalRank algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
source	Yes	Node ID	String	-	-
alpha	No	Weight coefficient	Double	A real number between 0 and 1 (excluding 0 and 1)	0.85
convergence	No	Convergence	Double	A real number between 0 and 1 (excluding 0 and 1)	0.00001
max_iterations	No	Maximum iterations	Int	1-2,000	1,000
directed	No	Whether to consider the edge direction	Bool	**true** or **false**	true

NOTE

- **alpha** determines the jump probability coefficient, also called damping coefficient, which is a computing control variable in the algorithm.
- **convergence** defines the sum and upper limit of absolute values of each vertex in each iteration compared with the last iteration. If the sum is less than the value, the computing is considered to be converged and the algorithm stops.

Precautions

When the convergence is set to a large value, the iteration will stop quickly.

Example

Set parameters **source** to **Lee**, **alpha** to **0.85**, **convergence** to **0.00001**, **max_iterations** to **1,000**, and **directed** to **true**. The sub-graph formed by top nodes in the calculation result is displayed on the canvas. The size of a node varies with the PersonalRank values. The JSON result is displayed in the query result area.

9.4 K-core

Overview

K-core is a classic graph algorithm used to calculate the number of cores of each node. The calculation result is one of the most commonly used reference values for determining the importance of a node so that the propagation capability of the node can be better understood.

Application Scenarios

This algorithm applies to scenarios such as community discovery and finance risk control.

Parameter Description

Table 9-4 K-core algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
k	Yes	Number of cores The algorithm returns nodes whose number of cores is greater than or equal to k.	Int	Greater than or equal to 0	-

Precautions

None

Example

Set parameter **k** to **10**. The sub-graph formed by nodes whose number of cores is greater than or equal to 10 in the calculation result is displayed on the canvas. The color of a node varies with the number of cores. The JSON result is displayed in the query result area.

9.5 K-hop

Overview

K-hop is an algorithm used to search all nodes in the k layer that are associated with the source node through breadth-first search (BFS). The found sub-graph is the source node's **ego-net**. The K-hop algorithm returns the number of nodes in the ego-net.

Application Scenarios

This algorithm applies to scenarios such as relationship discovery, influence prediction, and friend recommendation.

Parameter Description

Table 9-5 K-hop algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
k	Yes	Number of hops	Integer	1-100	-
source	Yes	Node ID	String	-	-
mode	No	Direction: ● OUT: Hop from the outgoing edges. ● IN: Hop from the incoming edges. ● All: Hop from edges in both directions.	String	OUT, IN, ALL	OUT

Precautions

- A larger k value indicates a wider node coverage area.
- According to the six degrees of separation theory, all people in social networks will be covered after six hops.
- BFS searches information based on edges.

Example

Calculate the sub-graph formed by the three hops starting from the Lee node.

Set parameters **k** to **3**, **source** to **Lee**, and **mode** to **OUT**. The sub-graph is displayed on the canvas, and the JSON result is displayed in the query result area.

9.6 Shortest Path

Overview

The Shortest Path algorithm is used to find the shortest path between two nodes in a graph.

Application Scenarios

This algorithm applies to scenarios such as path design and network planning.

Parameter Description

Table 9-6 Shortest Paths algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
source	Yes	Enter the source ID of a path.	String	-	-
target	Yes	Enter the target ID of a path.	String	-	-
directed	No	Whether to consider the edge direction	Bool	**true** or **false**	false
weight	No	Weight of an edge	String	Empty or null character string ● Empty: The default weight and distance are **1**. ● Character string: The attribute of the corresponding edge is the weight. When the edge does not have corresponding attribute, the weight is **1** by default. **NOTE** The weight of an edge must be greater than **0**.	-
timeWindow	No	Time window used for time filtering	Json	For details, see **Table 9-7**. **NOTE** **timeWindow** does not support the shortest path with weight. That is, parameters **timeWindow** and **weight** cannot be both specified.	-

Table 9-7 timeWindow parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
filterName	Yes	Name of the time attribute used for time filtering	String	Character string: The attribute on the corresponding vertex/edge is used as the time.	-
filterType	No	Filtering by vertex or edge	String	V: Filtering by vertex E: Filtering by edge BOTH: Filtering by vertex and edge	BOTH
startTime	No	Start time	String	Date character string or timestamp	-
endTime	No	End time	String	Date character string or timestamp	-

Precautions

This algorithm only returns one shortest path.

Example

Calculate the shortest path from the Lee node to the Alice node.

Set parameters **source** to **Lee**, **target** to **Alice**, **weight** to **weights**, and **directed** to **false**. The shortest path is displayed on the canvas, and the JSON result is displayed in the result area.

9.7 All Shortest Paths

Overview

The All Shortest Paths algorithm is used to find all shortest paths between two nodes in a graph.

Application Scenarios

This algorithm applies to scenarios such as path design and network planning.

Parameter Description

Table 9-8 All Shortest Paths algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
source	Yes	Enter the source ID of a path.	String	-	-
target	Yes	Enter the target ID of a path.	String	-	-
directed	No	Whether to consider the edge direction	Bool	**true** or **false**	false

Precautions

None

Example

Set parameters **source** to **Lee**, **target** to **Alice**, and **directed** to **false**. The calculation result is displayed on the canvas and the JSON result is displayed in the query result area.

9.8 SSSP

Overview

The SSSP algorithm finds the shortest paths from a specified node (source node) to all other nodes.

Application Scenarios

This algorithm applies to scenarios such as path design and network planning.

Parameter Description

Table 9-9 SSSP algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
source	Yes	Node ID	String	-	-

Parameter	Mandatory	Description	Type	Value Range	Default Value
directed	No	Whether to consider the edge direction	Bool	**true** or **false**	true

Example

Calculate the shortest paths from the Lee node to other nodes.

Set parameters **source** to **Lee** and **directed** to **true**.

9.9 Shortest Path of Vertex Sets

Overview

The Shortest Path of Vertex Sets algorithm finds the shortest path between two vertex sets.

Application Scenarios

This algorithm applies to block relationship analysis in Internet social networking, financial risk control, road network transportation, and logistics delivery scenarios.

Parameter Description

Table 9-10 Shortest Path of Vertex Sets algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
sources	Yes	Source vertex ID set	String	The value is in the standard CSV format. IDs are separated by commas (,), for example, **Alice, Nana**. The maximum ID number is 100,000.	-
targets	Yes	Target vertex ID set	String	The value is in the standard CSV format. IDs are separated by commas (,), for example, **Alice, Nana**. The maximum ID number is 100,000.	-

Parameter	Mandatory	Description	Type	Value Range	Default Value
directed	No	Whether to consider the edge direction	Bool	**true** or **false**	false
timeWindow	No	Time window used for time filtering	Json	For details, see **Table 9-11**.	-

Table 9-11 timeWindow parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
filterName	No	Name of the time attribute used for time filtering	String	Character string: The attribute on the corresponding vertex/ edge is used as the time.	-
filterType	No	Filtering by vertex or edge	String	V: Filtering by vertex E: Filtering by edge BOTH: Filtering by vertex and edge	BOTH
startTime	No	Start time	String	Date character string or timestamp	-
endTime	No	End time	String	Date character string or timestamp	-

NOTE

If a vertex ID contains commas (,), add double quotation marks to it. For example, when **Paris, je taime** and **Alice** IDs are used as sources, the ID set is **"Paris, je taime",Alice"**.

Example

Set parameters **directed** to **true**, **sources** to **"Alice,Nana"**, and **targets** to **"Lily,Amy"**. The JSON result is displayed in the query result area.

9.10 n-Paths

Overview

The n-Paths algorithm is used to find the *n* paths between two nodes within the layers of relationships in a graph.

Application Scenarios

This algorithm applies to scenarios such as relationship analysis, path design, and network planning.

Parameter Description

Table 9-12 n-Paths algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
source	Yes	Enter the source ID of a path.	String	-	-
target	Yes	Enter the target ID of a path.	String	-	-
directed	No	Whether to consider the edge direction	Bool	**true** or **false**	false
n	No	Number of paths	Int	1-100	10
k	No	Number of hops	Int	1-10	5

Example

Set parameters **source** to **Lee**, **target** to **Alice**, **n** to **10**, **k** to **5**, and **directed** to **false**. The calculation result is displayed on the canvas and the JSON result is displayed in the query result area.

9.11 Closeness Centrality

Overview

Closeness centrality of a node is a measure of centrality in a network, calculated as the reciprocal of the sum of the length of the shortest paths between the node and all other reachable nodes in a graph. It can be used to measure the time for transmitting information from this node to other nodes. The bigger the node's **Closeness Centrality** is, the more central the location of the node will be.

Application Scenarios

This algorithm is used in key node mining in social networking.

Parameter Description

Table 9-13 Closeness Centrality algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
source	Yes	Enter the ID of the node to be calculated.	String	-	-

Example

Set parameter **source** to **Lee** to calculate the closeness centrality of the Lee node. The JSON result is displayed in the query result area.

9.12 Label Propagation

Overview

The Label Propagation algorithm is a graph-based semi-supervised learning method. Its basic principle is to predict the label information about unlabeled nodes using that of the labeled nodes. This algorithm can create graphs based on the relationships between samples. Nodes include labeled data and unlabeled data, and the edge indicates the similarity between two nodes. Node labels are transferred to other nodes based on the similarity. Labeled data is like a source used to label unlabeled data. The greater the node similarity is, the easier the label propagation will be.

Application Scenarios

This algorithm applies to scenarios such as information propagation, advertisement recommendation, and community discovery.

Parameter Description

Table 9-14 Label Propagation algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
convergence	No	Convergence	Double	A real number between 0 and 1 (excluding 0 and 1)	0.00001

Parameter	Mandatory	Description	Type	Value Range	Default Value
max_iterations	No	Maximum iterations	Int	1-2,000	1,000

Parameter	Mandatory	Description	Type	Value Range	Default Value
initial	No	Name of the property used as the initialization label on a vertex	String	Null or character string ● Null: Each vertex is allocated with a unique initialization label. This method is applicable to scenarios where no vertex label information exists. ● Character string: The value of the property field corresponding to each vertex is used as the initialization label (the type is string, and the initialization label field is set to null for a vertex with unknown labels). This method is applicable to scenarios where some vertex labels are marked to predict unknown vertex labels.	-

Parameter	Mandatory	Description	Type	Value Range	Default Value
				NOTE If the value of **initial** is not null, the number of vertices with initialization labels must be greater than 0 and less than the total number of vertices.	

Precautions

Label Propagation uses IDs as labels by default.

Example

Set parameters **coverage** to **0.00001** and **max_iterations** to **1,000**, the sub-graphs with different labels are displayed on the canvas. The color of a node varies with labels. The JSON result is displayed in the query result area.

9.13 Louvain

Overview

Louvain is a modularity-based community detection algorithm with high efficiency and effect. It detects hierarchical community structures and aims to maximize the modularity of the entire community network.

Application Scenarios

This algorithm applies to scenarios such as community mining and hierarchical clustering.

Parameter Description

Table 9-15 Louvain algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
convergence	No	Convergence	Double	A real number between 0 and 1 (excluding 0 and 1)	0.00001

Parameter	Mandatory	Description	Type	Value Range	Default Value
max_iterations	No	Maximum iterations	Int	1-2,000	100
weight	No	Weight of an edge	String	Empty or null character string ● Empty: The default weight and distance are **1**. ● Character string: The attribute of the corresponding edge is the weight. When the edge does not have corresponding attribute, the weight is **1** by default. **NOTE** The weight of an edge must be greater than **0**.	weight

Precautions

This algorithm generates only the final community result and does not save the hierarchical results.

Example

Set parameters **coverage** to **0.00001** and **max_iterations** to **100**, the sub-graphs of different communities are displayed on the canvas. The color of a node varies with communities. The JSON result is displayed in the query result area.

9.14 Link Prediction

Overview

The Link Prediction algorithm is used to calculate the similarity between two nodes and predict their relationship based on the Jaccard measurement method.

Application Scenarios

This algorithm applies to scenarios such as friend recommendation and relationship prediction in social networks.

Parameter Description

Table 9-16 Link Prediction algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
source	Yes	Enter the source ID.	String	-	-
target	Yes	Enter the target ID.	String	-	-

Example

Set parameters **source** to **Lee** and **target** to **Alice** to calculate the association between two nodes. The JSON result is displayed in the query result area.

9.15 Node2vec

Overview

By invoking the Word2vec algorithm, the Node2vec algorithm maps nodes in the network to the Euclidean space, and uses vectors to represent the node characteristics.

The Node2vec algorithm generates random steps from each node using the rollback parameter **P** and forward parameter **Q**. It combines BFS and DFS. The rollback probability is proportional to 1/P, and the forward probability is proportional to 1/Q. Multiple random steps are generated to reflect the network structures.

Application Scenarios

This algorithm applies to scenarios such as node function similarity comparison, structural similarity comparison, and community clustering.

Parameter Description

Table 9-17 Node2vec algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
P	No	Rollback parameter	Double	-	1
Q	No	Forward parameter	Double	-	1
dim	No	Mapping dimension	Int	1 to 200, including 1 and 200	50
walkLength	No	Random walk length	Int	1 to 100, including 1 and 100	40
walkNumber	No	Number of random walk steps of each node.	Int	1 to 100, including 1 and 100	10
iterations	No	Number of iterations	Int	1 to 100, including 1 and 100	10

Precautions

None

Example

Set parameters **P** to **1**, **Q** to **0.3**, **dim** to **3**, **walkLength** to **20**, **walkNumber** to **10**, and **iterations** to **40** to obtain the three-dimensional vector display of each node.

9.16 Real-time Recommendation

Overview

The Real-time Recommendation algorithm is based on the random walk model and is used to recommend nodes that are similar (have similar relationships or preferences) to the input node.

Application Scenarios

This algorithm can be used to recommend similar products based on historical browsing data or recommend potential friends with similar preferences.

It is applicable to scenarios such as e-commerce and social networking.

Parameter Description

Table 9-18 Real-time Recommendation algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
sources	Yes	Node ID. Multiple node IDs separated by commas (,) are supported (standard CSV input format).	String	The number of source nodes cannot exceed 30.	-
alpha	No	Weight coefficient. A larger value indicates a longer step.	Double	A real number between 0 and 1 (excluding 0 and 1)	0.85
N	No	Total number of walk steps	Int	1-200,000	10,000
nv	No	Parameter indicating that the walk process ends ahead of schedule: minimum number of access times of a potential recommended node **NOTE** If a node is accessed during random walk and the number of access times reaches **nv**, the node will be recorded as the potential recommended node.	Int	1-10	5
np	No	Parameter indicating that the walk process ends ahead of schedule: number of potential recommended nodes **NOTE** If the number of potential recommended nodes of a source node reaches **np**, the random walk for the source node ends ahead of schedule.	Int	1-2,000	1,000

Parameter	Mandatory	Description	Type	Value Range	Default Value
label	No	Expected type of the vertex to be output. **NOTE** ● Expected type of the vertex to be output. If the value is null, the original calculation result of the algorithm is output without considering the vertex type. ● If the value is not null, vertices with the **label** are filtered from the calculation result.	String	Node label	-
directed	No	Whether to consider the edge direction	Bool	**true** or **false**	true

NOTE

alpha determines the jump probability coefficient, also called damping coefficient, which is a computing control variable in the algorithm.

Precautions

In the end conditions, the smaller the values of **nv** and **np**, the faster the algorithm ends.

Example

Set parameters **sources** to **Lee**, **alpha** to **0.85**, **N** to **10,000**, **nv** to **5**, **np** to **1,000**, **directed** to **true**, and **label** to null.

The sub-graph formed by top nodes in the calculation result is displayed on the canvas. The size of a node varies with the final scores. The JSON result is displayed in the query result area.

9.17 Common Neighbors

Overview

Common Neighbors is a basic graph analysis algorithm that obtains the neighboring nodes shared by two nodes and further speculate the potential relationship and similarity between the two nodes. For example, it can intuitively discover shared friends in social occasions or commodities that interest both nodes in the consumption field.

Application Scenarios

This algorithm applies to scenarios such as e-commerce and social networking.

Parameter Description

Table 9-19 Common Neighbors algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
source	Yes	Enter the source ID.	String	-	-
target	Yes	Enter the target ID.	String	-	-

Precautions

None

Example

Set parameters **source** to **Lee** and **target** to **Alice**. The calculation result is displayed on the canvas and the JSON result is displayed in the query result area.

9.18 Connected Component

Overview

A connected component stands for a sub-graph, in which all nodes are connected with each other. Path directions are involved in the strongly connected components and are not considered in the weakly connected components. This algorithm generates weakly connected components.

Parameter Description

None

Example

Run the algorithm to calculate the connected component to which each node belongs. The JSON result is displayed in the query result area.

9.19 Degree Correlation

Overview

The Degree Correlation algorithm calculates the Pearson correlation coefficient between the source vertex degree and the target vertex degree of each edge. It is used to indicate whether the high-degree nodes are connected to other high-degree nodes in a graph.

Application Scenarios

This algorithm is often used to measure the structure features of a graph.

Parameter Description

None

Example

Run the algorithm to calculate the degree correlation of a graph. The JSON result is displayed in the query result area.

9.20 Triangle Count

Overview

The Triangle Count algorithm counts the number of triangles in a graph. More triangles mean higher node association degrees and closer organization relationships.

Application Scenarios

This algorithm is often used to measure the structure features of a graph.

Parameter Description

Parameter	Mandatory	Description	Type	Value Range
statistics	No	Whether to export only the total statistical result. ● **true**: Export only the statistical result. ● **false**: Export the number of triangles corresponding to each vertex.	Boolean	**true** or **false**. The default value is **true**.

Instructions

The edge direction and multi-edge situation are not considered.

Example

Enter **statistics = true**. The JSON result is displayed in the query result area.

9.21 Clustering Coefficient

Overview

The clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together. Evidence suggests that in most real-world networks, and in particular social networks, nodes tend to create tightly knit groups characterized by a relatively high density of ties. This algorithm is used to calculate the aggregation degree of nodes in a graph.

Application Scenarios

This algorithm is often used to measure the structure features of a graph.

Parameter Description

None

Instructions

The multi-edge situation is not considered.

Example

Run the algorithm to calculate the clustering coefficient of a graph. The JSON result is displayed in the query result area.

9.22 Common Neighbors of Vertex Sets

Overview

The Common Neighbors of Vertex Sets algorithm can find common neighbors of two vertex sets, and intuitively discover an object jointly associated with both sets, for example, a common friend in a social occasion, a commodity that is of common interest, a person who has been contacted by community groups. In this way, the algorithm infers the potential relationship and degree of association between the vertex sets.

Application Scenarios

This algorithm applies to graph analysis such as relationship mining and product/ friend recommendations.

Parameter Description

Table 9-20 Common Neighbors of Vertex Sets algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
sources	Yes	Source vertex ID set	String	The value is in the standard CSV format. IDs are separated by commas (,), for example, **Alice, Nana**. The maximum ID number is 100,000.	-
targets	Yes	Target vertex ID set	String	The value is in the standard CSV format. IDs are separated by commas (,), for example, **Alice, Nana**. The maximum ID number is 100,000.	-

Precautions

None

Example

Enter **sources=Alice,Nana** and **targets=Mike,Amy**. The calculation result is displayed on the canvas and the JSON result is displayed in the query result area.

9.23 Subgraph Matching

Overview

The subgraph matching algorithm is used to find all subgraphs of a given small graph that is isomorphic to a given large graph. This is a basic graph query operation and is intended to explore important substructures of a graph.

Application Scenarios

This algorithm is applicable to fields such as social network analysis, bioinformatics, transportation, crowd discovery, and anomaly detection.

Parameter Description

Table 9-21 Subgraph matching parameters

Name	Mandatory	Description	Type	Value Range
edges	Yes	Edge set of the subgraph to be matched. The vertex ID must be a non-negative integer.	String	The value is in standard CSV format. The start and end vertices of an edge are separated by a comma (,), and edges are separated by a newline character (\n). For example, **1,2\n2,3**.
vertices	Yes	Label of each vertex on the subgraph to be matched.	String	The value is in standard CSV format. Vertices and their labels are separated by commas (,), and labels are separated by newline characters (\n). For example, **1,BP\n2,FBP\n3,CP**.
directed	No	Whether the graph is directed	Bool	The value can be **true** or **false**. The default value is **true**.
n	No	Maximum number of subgraphs to be searched for	Int	The value range is [1,100000]. The default value is **100**.
batch_number	No	Number of queries processed in batches each time	Int	The value range is [1,1000000]. The default value is **10000**.
statistics	No	Whether to display the number of all subgraphs that meet the conditions	Bool	The value can be **true** or **false**. The default value is **false**.

9.24 Filtered All Pairs Shortest Paths

Overview

The Filtered All Pairs Shortest Paths algorithm is used to search for the shortest path between any two vertices in the graph that meets the condition. In a specific application scenario, you need to set a start vertex set (**sources**) and end vertex set (**targets**) as input for this algorithm. This algorithm returns the required shortest paths between the start and the end vertex sets.

Application Scenarios

This algorithm applies to relationship mining, path planning, and network planning.

Parameter Description

Table 9-22 Parameters

Name	Mandatory	Description	Type	Value Range	Default
sources	Yes	Set of start vertex IDs. The value is in the standard CSV input format, that is, multiple vertex IDs are separated by commas (,).	String	The number of source vertices cannot exceed 10,000. -	-
targets	Yes	Set of end vertex IDs. The value is in the standard CSV input format, that is, multiple vertex IDs are separated by commas (,).	String	The number of target vertices cannot exceed 10,000. -	-
directed	No	Whether the edges are directed	Bool	The value can be **true** or **false**.	-
cutoff	No	Maximum length	Int	1-100	6

Name	Mandatory	Description	Type	Value Range	Default
path_limit	No	Maximum number of paths	Int	● For synchronous tasks: The value ranges from 1 to 100000. The default value is **100000**. ● For asynchronous tasks: The value ranges from 1 to 1000000. The default value is **1000000**. 1000000	100000/1000000

Example

Configure the parameters as follows: **directed=true**, **sources="Alice,Vivian"**, **targets="Jay,Bonnie"**, and set the edge search condition **labelName=friends**. The shortest paths between each pair of start and end vertices are returned in JSON format.

9.25 Filtered All Shortest Paths

Overview

The Filtered All Shortest Paths algorithm allows you to search query results of the Shortest Path algorithm for the paths that meet the conditions between two vertices in a graph.

Application Scenarios

This algorithm applies to scenarios such as relationship mining, path planing, and network planning.

Parameter Description

Table 9-23 Parameters

Name	Mandatory	Description	Type	Value Range	Default
source	Yes	Source vertex ID	String	-	-

Name	Mandatory	Description	Type	Value Range	Default
target	Yes	Target vertex ID	String	-	-
directed	No	Whether the edges are directed	Bool	The value can be **true** or **false**.	false

Example

Configure the parameters as follows: **directed=true**, **source="Alice"**, **target="Jay"**, and set the search condition to **labelName=friends**. The results are returned in JSON format.

9.26 TopicRank

Overview

TopicRank algorithm is one of commonly used algorithms for ranking topics by multiple dimensions.

Application Scenarios

This algorithm is applicable to rank hot topics. For example, it can be used to rank complaint topics obtained through a government hotline.

Parameter Description

Table 9-24 TopicRank parameters

Name	Mandatory	Description	Type	Value Range	Default
sources	Yes	Vertex ID. You can specify multiple IDs in CSV format and separate them with commas (,).	String	Currently, a maximum of 100000 IDs are allowed.	-
actived_p	No	Initial weight of the source vertices	Double	The value ranges from 0 to 100000.	1

Name	Mandatory	Description	Type	Value Range	Default
default_p	No	Initial weight of a non-source vertices	Double	The value ranges from 0 to 100000.	1
filtered	No	Whether to filter results	Boolean	The value can be **true** or **false**.	false
only_neighbors	No	Whether to display only the neighboring vertices of the sources	Boolean	The value can be **true** or **false**.	false
alpha	No	Weight coefficient	Real number	A real number between 0 and 1	0.85
convergence	No	Convergence	Real number	A real number between 0 and 1	0.00001
max_iterations	No	Maximum iterations	Positive integer	The value ranges from 1 to 2000.	1000
directed	No	Whether the edges are directed	Boolean	The value can be **true** or **false**.	true
num_thread	No	Number of threads	Positive integer	1-40	4

Example

Specify **sources="20190110004349,20190129023326,20190107003294,20190129023391"**, **filtered = true**, **only_neighbors=true, alpha=0.85, converage=0.00001**, **max_iterations=1000**, **directed=true**, and **label="Topic"** to obtain the topic ranking result.

9.27 Filtered n-Paths

Overview

The filtered n-Paths algorithm is used to find no more than n k-hop loop-free paths between the source and target vertices. The start vertex (source), end vertex (target), number of hops (k), number of paths (n), and filter criteria (filters) are the parameters for the algorithm.

Application Scenarios

Any network

Parameter Description

Table 9-25 filtered_n_paths parameters

Name	Mandatory	Description	Object Type	Value Range	Default
source	Yes	Source vertex	String	Internal vertices	None
target	Yes	Target vertex	String	Internal vertices	None
k	Yes	Number of hops	Int	[2,6]	2
n	Yes	Number of paths	Int	[1,1000]	1