# **Graph Engine Service**

# **User Guide**

Issue 01

**Date** 2024-03-28





#### Copyright © Huawei Technologies Co., Ltd. 2024. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

#### **Trademarks and Permissions**

HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd. All other trademarks and trade names mentioned in this document are the property of their respective holders.

#### **Notice**

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

# Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base

Bantian, Longgang Shenzhen 518129

People's Republic of China

Website: <a href="https://www.huawei.com">https://www.huawei.com</a>

Email: <a href="mailto:support@huawei.com">support@huawei.com</a>

# **Security Declaration**

## **Vulnerability**

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process.* For details about this process, visit the following web page:

https://www.huawei.com/en/psirt/vul-response-process

For vulnerability information, enterprise customers can visit the following web page:

https://securitybulletin.huawei.com/enterprise/en/security-advisory

# **Contents**

1 GES Overview	1
2 Preparations	3
3 Permissions Management	4
3.1 Creating a User	4
3.2 Policy Permissions	5
3.2.1 Policy	6
3.2.2 System-Defined Policies	6
3.2.3 Custom Policies	
3.2.4 Request Conditions	11
3.2.5 GES Resources	12
3.3 Role Permissions	12
4 Metadata Operations	16
4.1 Graph Data Formats	16
4.1.1 Static Graph	16
4.1.2 Dynamic Graph	20
4.2 Importing a Metadata File	23
4.2.1 Preparing Metadata	23
4.2.2 Importing Data From a Local Path or OBS	24
4.3 Creating a Metadata File	25
4.4 Copying a Metadata File	27
4.5 Editing a Metadata File	28
4.6 Searching for a Metadata File	29
4.7 Deleting a Metadata File	29
5 Creating Graphs	30
5.1 Methods to Create a Graph	30
5.2 Creating a Graph Without Using a Template	30
5.3 Creating a Graph Using an Industry-Specific Template	37
5.4 Creating a Dynamic Graph	38
5.5 Starting a Graph	39
5.6 Stopping a Graph	40
5.7 Accessing Graphs	40
5.8 Importing Incremental Data	41

6 Managing Graphs	43
6.1 Graph Management Overview	43
6.2 Viewing a Failed Graph	44
6.3 Backing Up and Restoring Graphs	45
6.3.1 Backing Up a Graph	45
6.3.2 Restoring a Graph	46
6.3.3 Deleting a Backup	47
6.3.4 Exporting a Backup to OBS	48
6.3.5 Importing a Backup from OBS	49
6.4 Upgrading a Graph	50
6.5 Exporting a Graph	50
6.6 Restarting a Graph	51
6.7 Resizing a Graph	52
6.8 Expanding a Graph	53
6.9 Binding and Unbinding an EIP	54
6.10 Clearing Data	55
6.11 Deleting a Graph	56
6.12 Viewing Monitoring Metrics	57
6.13 Querying Schema	59
7 Accessing and Analyzing Graph Data	60
7.1 Graph Editor	60
7.2 Accessing the GES Graph Editor	69
7.3 Dynamic Graphs	69
7.3.1 Timeline	69
7.3.2 Community Evolution	71
7.3.3 Temporal BFS	73
7.3.4 Temporal Paths	75
7.4 Graph Exploration	77
7.5 Multi-Graph Management (Database Edition)	80
7.6 Adding Custom Operations	82
7.7 Editing Schema	83
7.8 Visual Query	89
7.9 Gremlin Query	95
7.10 Cypher Query	99
7.11 DSL Query	101
7.12 Analyzing Graphs Using Algorithms	103
7.13 Analyzing Graphs on the Canvas	105
7.14 Graph Display in 3D View	
7.15 Filter Criteria	112
7.16 Editing Properties	113
7.17 Statistics Display	114
7 18 View Running Records	115

7.19 Viewing Query Results	115
8 Viewing Graph Tasks	118
3.1 Graph Overview	118
8.2 Task Center	120
8.2.1 Management Plane Task Center	120
8.2.2 Service Plane Task Center	122
8.3 Viewing Monitoring Metrics	123
8.4 Managing Connections	132
9 Configuring Permissions	135
9.1 Configuring Granular Permissions	135
9.2 User Groups	137
9.3 User Details	138
10 O&M Monitoring and Alarm Reporting	140
10.1 Monitoring Metrics	
10.2 Graph Instance O&M Monitoring	
10.3 Monitoring	147
10.3.1 Nodes	147
10.3.2 Performance	151
10.3.3 Real-Time Queries	152
10.3.4 Historical Queries	152
10.4 Monitoring Clusters Using Cloud Eye	153
11 Algorithms	164
11.1 Algorithm List	164
11.2 PageRank	168
11.3 PersonalRank	169
11.4 K-core	171
11.5 K-hop	171
11.6 Shortest Path	
11.7 All Shortest Paths	175
11.8 Filtered Shortest Path	175
11.9 SSSP	176
11.10 Shortest Path of Vertex Sets	177
11.11 n-Paths	178
11.12 Closeness Centrality	179
11.13 Label Propagation	180
11.14 Louvain	182
11.15 Link Prediction	183
11.16 Node2vec	184
11.17 Real-time Recommendation	185
11.18 Common Neighbors	187
11.19 Connected Component	188

11.20 Degree Correlation	188
11.21 Triangle Count	189
11.22 Clustering Coefficient	190
11.23 Betweenness Centrality	190
11.24 Edge Betweenness Centrality	
11.25 Origin-Destination Betweenness Centrality	194
11.26 Circle Detection with a Single Vertex	
11.27 Common Neighbors of Vertex Sets	
11.28 All Shortest Paths of Vertex Sets	
11.29 Filtered Circle Detection	199
11.30 Subgraph Matching	200
11.31 Filtered All Pairs Shortest Paths	
11.32 Filtered All Shortest Paths	203
11.33 TopicRank	
11.34 Filtered n-Paths	
11.35 Temporal Paths	206

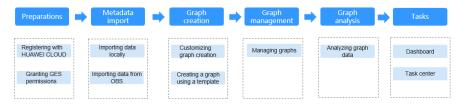
# **1** GES Overview

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.

The following figure shows the procedure to use GES.

Figure 1-1 Use procedure



**Table 1-1** Process description

Task	Sub Task	Description	Instructions
Preparatio ns	Creating a Huawei ID	Before using GES, create a Huawei ID.	Creating a Huawei ID and enabling Huawei Cloud services
	Granting GES permissions	Grant GES permissions to a user group and add a user to the user group.	Granting GES permissions
Metadata import	Importing metadata from a local path	Import the metadata file to GES for graph creation.	Importing a metadata file from a local path

Task	Sub Task	Description	Instructions
	Importing metadata from OBS	Upload the prepared metadata file to an OBS bucket.	Importing metadata from OBS
Graph creation	Creating a graph	Create a graph without using a template.	Creating a graph
Graph creation	Creating a dynamic graph	Create a graph with the dynamic graph template.	Creating a dynamic graph
Graph Managem ent	Managing graphs	Back up, restore, and upgrade a graph, expand an instance, and scale out a graph node.	Managing graphs
Graph analysis	Analyzing graph data	Use the Graph Editor to query and analyze graph data.	Analyzing graph data
Tasks	Dashboard	The overview page displays information about your resources, including basic graph information and billing details.	Dashboard
	Task center	The task center displays details about asynchronous tasks, such as creating, backing up, starting, and deleting graphs.	Task center

# **2** Preparations

Before using GES, create a Huawei ID.

## Creating a Huawei ID and Enabling Huawei Cloud Services

Skip this step if you have created one.

- **Step 1** Log in to the **Huawei Cloud** official website.
- **Step 2** Click **Register** in the upper right corner to access the registration page.
- **Step 3** Complete the registration as instructed. For details, see **Account Registration Process**.

----End

# 3 Permissions Management

# 3.1 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 Huawei Cloud account does not need individual IAM users, then you may skip over this chapter.

## **Permission Type**

Туре

- 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. For the API actions supported by GES, see
  Permissions Policies and Supported Actions.

∩ NOTE

GES ReadOnlyAccess is a policy.

#### Procedure

This section describes how to use a group to grant permissions to a user. **Figure 3-1** shows the process.

Create a user group and grant permissions.

Create a user.

Log in as the user and verify permissions.

Figure 3-1 Granting GES permissions

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 as the user you created and verify permissions.

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.

# 3.2 Policy Permissions

## **3.2.1 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**.

## 3.2.2 System-Defined Policies

Table 3-1 GES system-defined policies

Policy Name	Description		
Policy Name	Description		
GES FullAccess	Permissions for all operations on GES, including creating, deleting, accessing, and updating graphs.  NOTE		
	<ul> <li>Users with the permissions of this policy also need the following policy permissions granted: Tenant Guest, Server Administrator, and VPC Administrator.</li> </ul>		
	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".		
	<ul> <li>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.</li> </ul>		
	When granting GES FullAccess to an enterprise project, you need to configure the following permissions policies in IAM:		
	<ul> <li>ecs:availabilityZones:list. For details, see AZ Management.</li> </ul>		
	<ul> <li>ecs:cloudServerNics:update. For details, see NIC Management.</li> </ul>		

Policy Name	Description		
GES Development	Operator permissions for all operations except creating, deleting, resizing, and expanding graphs.		
	NOTE		
	<ul> <li>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".</li> </ul>		
	<ul> <li>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.</li> </ul>		
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 3-2 Common operations supported by each system-defined policy

Operation	GES FullAccess	GES Development	GES ReadOnlyAcc ess	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 ReadOnlyAcc ess	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	metadataNa me
Verifying metadata	Yes	Yes	No	-
Adding metadata	Yes	Yes	No	metadataNa me
Deleting metadata	Yes	Yes	No	metadataNa me
Querying task statuses	Yes	Yes	Yes	-

Operation	GES FullAccess	GES Development	GES ReadOnlyAcc ess	Resource
Querying the task list	Yes	Yes	Yes	-
Configuring fine- grained permissions	√	Yes	No	-
Configuring user groups	√	Yes	No	-
Importing IAM users	√	Yes	No	-
Viewing user details	√	Yes	Yes	-

## 3.2.3 Custom Policies

In addition to the system-defined policies of GES, you can also create your own custom policies. For the actions supported for custom policies, see **Permissions Policies and Supported Actions**.

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.

For details, see **Creating a Custom Policy**.

## **Examples**

• Example 1: Allowing users to guery and operate graphs

• 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:

 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"
      1,
      "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 and backups.
- Part 2: Authorizing users to query the graph, backups, tasks, and metadata lists, and view job details

```
],
    "Resource": [
        "ges:*:*:backupName:ges_project*",
        "ges:*:graphName:ges_project*"
],
    "Effect": "Allow"
},
{
    "Action": [
        "ges:graph:list",
        "ges:backup:list",
        "ges:jobs:list",
        "ges:jobs:list",
        "ges:jobs:getDetail"
],
    "Effect": "Allow"
}
```

# 3.2.4 Request Conditions

Request conditions are useful in determining when a custom policy takes effect. A request condition consists of a condition key and operator. Condition keys are either global or service-level and are used in the Condition element of a policy statement. **Global condition keys** (starting with **g:**) are available for operations of all services, while service-level condition keys (starting with a service name such as **ges**) are available only for operations of a specific service. An operator is used together with a condition key to form a complete condition statement.

GES has a group of predefined condition keys that can be used in IAM. For example, to define an allow permission, you can use the condition key **hw:SourceIp** to match requesters by IP address. The following table shows the request conditions that are used with GES.

Table	3-3	Request	conditions
-------	-----	---------	------------

Condition Key	Туре	Description
g:CurrentTime	Date and time	Time when an authentication request is received  NOTE  The time is in ISO 8601 format, for example, 2012-11-11T23:59:59Z.
g:MFAPresent	Boolean	Whether multi-factor authentication is used for user login
g:UserId	String	User ID used for current login
g:UserName	String	Username used for current login
g:ProjectName	String	Project of the current login
g:DomainName	String	Domain of the current login

## 3.2.5 GES Resources

A resource is an object that exists within a service. On GES, you can select these resources by specifying their paths.

**Table 3-4** GES resources and their paths

Resource	Resource Name	Path
graphName	GES graph name	graphName
backupNam e	GES backup name	backupName

# 3.3 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 3-5** System roles

Role Name	Description
Tenant Guest	Regular tenant users  • Permissions: querying GES resources  • Scope: project-level service
GES Administrator	<ul> <li>GES administrator</li> <li>Permissions: performing any operations on GES resources</li> <li>Scope: project-level service</li> <li>NOTE         <ul> <li>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.</li> <li>If you need to bind or unbind an EIP, you need the Security Administrator permissions to create agencies.</li> <li>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. When granting OBS permissions, specify the permission scope as global service resources.</li> </ul> </li> </ul>

Role Name	Description	
GES Manager	GES manager	
	<ul> <li>Permissions: performing any operations on GES resources other than creating, deleting graphs, resizing, and expanding graphs</li> </ul>	
	Scope: project-level service	
	If you have both 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. When granting OBS permissions, specify the permission scope as global service resources.	
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.	
	<ul> <li>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.</li> </ul>	

**Table 3-6** 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

Operation	GES Administrator	GES Manager	GES Operator	Tenant Guest
Importing data	Yes	Yes	No	No
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

Operation	GES Administrator	GES Manager	GES Operator	Tenant Guest
Resizing a graph	√	No	No	×
Expanding a graph	√	No	No	×
Restarting a graph	√	Yes	No	×
Configuring fine-grained permissions	√	Yes	No	×
Configuring user groups	√	Yes	No	×
Importing IAM users	√	Yes	No	×
Viewing user details	√	Yes	Yes	√

Table 3-7 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, and deleting metadata	OBS Operator policy or Tenant Administrator role
Creating, importing, and exporting graphs	OBS Operator policy or Tenant Administrator role

Table 3-8 Common GES operations supported by each IAM policy

GES Operation	Dependent IAM Permission
Importing IAM users	iam:users:listUsers (custom policy), IAM ReadOnlyAccess (system policy), or Server Administrator role
Creating or editing a user group	iam:users:listUsers (custom policy), IAM ReadOnlyAccess (system policy), or Server Administrator role

# 4 Metadata Operations

# 4.1 Graph Data Formats

# 4.1.1 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 4-1 Metadata structure

```
<?xm1 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">
                          <lab els>
                            <label name="default">
            Label
                            </laheb
           default
                             <label name="movie">
                              properties>
                                cproperty name="ChineseTitle" cardinality="single" dataType="string"/>
                                cproperty name="Year" cardinality="single" dataType="int"/>
           Label
                                 cproperty name="Genres" cardinality="set" data Type="string"/>
           movie
                              </properties>
                             </label>
                             <label nam
                              properties>
                                 cpropertyname="ChineseName" cardinality="single" dataType="string" />
                                 sproperty name="Gender" cardinality="single" dataType="enum" typeNameCount=
                                                   typeName1="F" typeName2="M"/>
Labels
                                 property name="age" cardinality="single" dataType="enum" typeNameCount="7"
                                                  typeName1="Under 18" typeName2="18-24" typeName3="25-34"
                                                  typeName4="35-44" typeName5="45-49"
           user
                                                   typeName6="50-55" typeName7="56+"/>
                                cproperty name="Occupation" cardinality="single" dataType="string"/>
                               cproperty name="Zip-code" cardinality="single" dataType="char array" maxDataSize="12"/>
                              </properties>
                             label name="rate"
                              properties>
                                 cpropertyname="Rating" cardinality="single" dataType="int" />
             Labe
                                 cproperty name="Datetime" cardinality="single" dataType="Date"/>
             rate
                              </properties>
                             </label>
                          </lab els>
                          </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. Starting from version 2.3.18, this restriction no longer exists, meaning that properties with the same name but different types under labels can be synchronized.

#### Property

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

- Property name: name of a custom property. The value can contain 1 to 256 characters and cannot contain special characters (<>&).

#### **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).

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 4-1 Supported data types

Туре	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 <b>single</b> supports the data type.
	<ul> <li>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.</li> </ul>
float	Float type (32-bit float)
double	Double floating point type (64-bit float point)
bool	Boolean type. Available values are <b>0/1</b> and <b>true/false</b> .
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.

Туре	Description
enum	Enumeration. Specify the number of the enumerated values and the name of each value. For details, see <b>Metadata structure</b> .
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">
    cproperties>
      cproperty name="movieid" cardinality="single" dataType="int" />
      cproperty name="title" cardinality="single" dataType="string"/>
      cardinality="single" dataType="string"/>
    </properties>
  </label>
  <label name="user">
    properties>
      cproperty name="userid" cardinality="single" dataType="int" />
      cproperty name="gender" cardinality="single" dataType="string"/>
      typeName1="Under 18" typeName2="18-24" typeName3="25-34" typeName4="35-44"
typeName5="45-49"
      typeName6="50-55" typeName7="56+"/>
      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"/>
      </properties>
  </label>
  <label name="rate">
    cproperties>
      property name="Rating" cardinality="single" dataType="int" />
      cproperty 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, ...

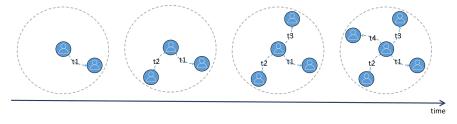
#### Example:

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

## 4.1.2 Dynamic Graph

In most real-life problems, entities and relationships change over time (such as disease transmission networks and transaction networks). The time sequence and changing information greatly affect the results. To predict these results, we use dynamic graphs to model, store, and analyze the dynamic data.

Figure 4-2 Dynamic graphs



#### ■ NOTE

This section mainly describes the data format of dynamic graphs. For details about operations related to these graphs, see **Creating Dynamic Graphs** and **Using Dynamic Graphs**.

#### Data Model

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

A dynamic graph evolves over time. From static to dynamic, there are spatio-temporal graphs (STGs), discrete-time dynamic graphs (DTDGs), and continuous-time dynamic graphs (CTDGs) as shown in dynamic graphs. CTDGs are dynamic graphs that store more details about the vertices and edges.

GES allows you to create CTDGs. The following is an example.

Assume that a graph has three vertices: **Vivian**, **P1**, and **P2**, and three edges: **(Vivian, P1)**, **(Vivian, P2)**, and **(Vivian, Vivian)**. There are two vertex types (lables): **Person** and **Place**, and relationship types (label): **Visited** and **Diagnosed**. The timestamp **[startTime, endTime]** indicates the duration of an event. For example, **(Vivian, P1)** indicates that Vivian visited P1 during 2021-11-21 12:05:21 to 2021-11-21 14:00:00, and **(Vivian, Vivian)** indicates that Vivian was infected with COVID-19 during 2021-11-25 23:00:00 to 2021-12-04 08:00:00. (Note: The vertex state changed, and the edge that starts and ends with this vertex represents this event.)

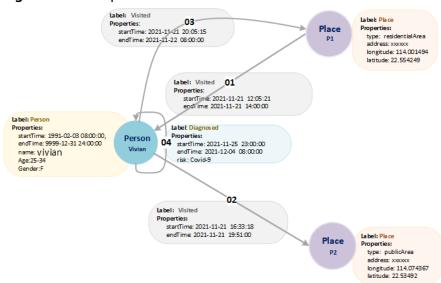


Figure 4-3 Example data model

## **Metadata of Dynamic Graphs**

Timestamps are important features of dynamic graphs. To describe dynamic graph data, you need to define timestamp-related properties such as **startTime** and **endTime** in metadata.

Note that **startTime** and **endTime** dynamic graph properties and are related to the life cycle of vertices and edges in the graph. The type must be **date** or **long**. The following is an example:

```
cproperty dataType="string" name="name" cardinality="single"/>
    cordinality="single"/>
    cproperty dataType="string" name="gender" cardinality="single"/>
   </properties>
  </label>
  <label name="Place">
   cproperties>
    cardinality="single"/>
    cproperty dataType="string" name="address" cardinality="single"/>
    cproperty dataType="float" name="longitude" cardinality="single"/>
    cproperty dataType="float" name="latitude" cardinality="single"/>
   </label>
  <label name="Visited">
   cproperties>
    - - cardinality="single"/>
    cproperty dataType="long" name="endTime" cardinality="single"/>
   </properties>
  </label>
  <label name="Dignosed">
   cproperties>
    property dataType="long" name="startTime" cardinality="single"/>
    cproperty dataType="long" name="endTime" cardinality="single"/>
    cproperty dataType="string" name="risk" cardinality="single"/>
   </properties>
  </label>
 </labels>
</PMML>
```

### **Vertices of Dynamic Graphs**

Dynamic vertex

For dynamic graphs, each line of the vertex file contains the data of a vertex. **id** uniquely identifies vertex data, **startTime** indicates the start time of the vertex lifecycle, and **endTime** indicates the end time of the vertex lifecycle.

id, label, start Time, end Time, property 1, property 2...

The following is an example:

Vivian, Person, 1991-02-03 08:00:00, 9999-12-31 24:00:00, Vivian, F, 25-34

Static vertex

A vertex without specified **startTime** and **endTime** is a static vertex.

id,label,property1,property2...

The following is an example for static vertex:

Vivian, Person, Vivian, F, 25-34

P1, Place, residential Area, xxxxxx, 114.001494, 22.554249

P2, Place, public Area, xxxxxxx, 114.074367, 22.53492

Note

If a vertex changes over time in its lifecycle, for example, the health status information of a person changes in a certain period, the changes can be modeled as an edge. The edge data is stored in a line of the edge file, representing status changes of the vertex.

id,id,label,startTime,endTime,property...

The following is an example:

Vivian, Vivian, Diagnosed, 2021-11-25 23:00:00, 2021-12-04 08:00:00, Covid-9

### **Edges of Dynamic Graphs**

Dynamic edge

The following example shows the data of an edge in a dynamic graph. Each line in the edge file contains the data of an edge. **id 1** and **id 2** indicate the IDs of the start and end vertices of an edge, respectively. **startTime** indicates the start time of the edge lifecycle, and **endTime** indicates the end time of the edge lifecycle.

id 1, id 2, label, startTime, endTime, property 1, property 2, ...

The following is an example:

Vivian,P1,Visited,2021-11-21 12:05:21,2021-11-21 14:00:00

Vivian,P2,Visited,2021-11-21 16:33:18,2021-11-21 19:51:00

• Static edge

An edge without the start time and end time is a static edge.

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

#### Vertex and Edge Data File

Vertex data file

Each line in the file indicates a dynamic/static vertex. You can use more than one vertex file.

Vivian, Person, Vivian, F, 25-34

P1,Place,residentialArea,xxxxxx,114.001494,22.554249

P2, Place, public Area, xxxxxxx, 114.074367, 22.53492

Edge data file

Each line in the file indicates a dynamic/static edge. You can use more than one edge file.

Vivian,P1,Visited,2021-11-21 12:05:21,2021-11-21 14:00:00

Vivian, P2, Visited, 2021-11-21 16:33:18, 2021-11-21 19:51:00

# 4.2 Importing a Metadata File

## 4.2.1 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:

- 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. For
  details about how to create a bucket and upload files, see Creating a Bucket
  and Uploading an Object.
- 2. Upload the prepared file to the OBS bucket by referring to **Uploading a File**. The metadata file must be in XML format.

## 4.2.2 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.

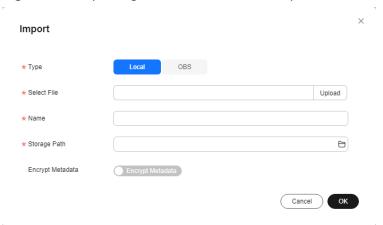
**◯** NOTE

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.

Figure 4-4 Importing metadata from a local path



- Import a metadata file from OBS.

Select File Path: Select the metadata file from OBS.

**M** 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.

Import

\* Type Local OBS

\* Select File Path

\* Name

Encrypt Metadata

Cancel OK

Figure 4-5 Importing metadata from OBS

4. Click **OK** to import the metadata.

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

# 4.3 Creating a Metadata File

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

□ NOTE

A maximum of 50 metadata files can be created.

#### **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 you create metadata for the first time, you need to enable OBS. (You are advised to obtain user authorization and automatically create OBS buckets for the metadata.)
  - Encrypt Metadata: whether to encrypt metadata. This function is disabled by default. Key Source is default to KMS. KMS Key: Select the key as needed.

**Ⅲ** NOTE

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

 Definition: Metadata models can be built manually or in a visualized manner.

**Manual**: Click **Add Label**. Define the label name and label type. Click **Add** under the label name to add a property. You can also click **Up** or **Down** to sort properties. **Table 4-2** lists the property parameters. For details about other metadata information, see **Graph Data Formats**.

#### ■ NOTE

- 1. Multiple labels are allowed. Click **Add label** to add labels as needed.
- 2. There are three types of labels: vertex, edge, and general-purpose (both vertex and edge).

Figure 4-6 Manual



#### Visual:

- Adding a vertex label: Drag a circle to the canvas to add a vertex.
   Click the vertex in the canvas to define its name, description, and properties.
- Adding an edge label: Click a connection point on a vertex and drag it to the connection point of another vertex to create an edge. Define its name, description, source vertex, target vertex, and properties. Table 4-2 lists the property parameters.

Figure 4-7 Creating a metadata file

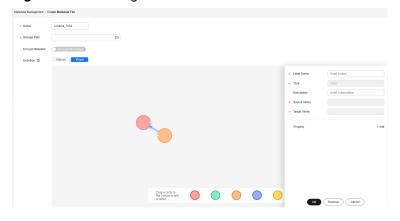


Table 4-2 Property parameters

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.

Name	Description
Cardinalit y	<ul><li>Composite type of data</li><li>Single value: indicates that the property has a single</li></ul>
	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 <b>char array</b> type.
Operation	Click <b>Remove</b> 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, encryption status, and modification time of the metadata.

# 4.4 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**

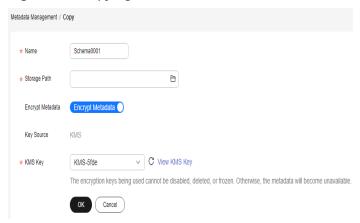
- 1. GES provides two methods for you to copy a metadata file on the **Data Management** page.
  - Click the metadata file name. On the details page, click Copy.
  - Click Copy in the Operation column of the target metadata file.
- 2. Specify the metadata file name and storage path.

**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.

Figure 4-8 Copying a metadata file



#### 3. Click OK.

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

# 4.5 Editing a Metadata File

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

#### **MOTE**

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.

Figure 4-9 Clicking Edit



- 2. On the editing page:
  - On the Manual tab, you can add labels and properties, change label names, and sort properties by clicking Up and Down.
  - On the Visual tab, you can drag a vertex to the canvas to add a label, or click a vertex or edge to modify the label information.
- 3. After the modification is complete, click **OK**.

# 4.6 Searching for a Metadata File

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

Figure 4-10 Searching metadata



# 4.7 Deleting a Metadata File

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

□ NOTE

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

Figure 4-11 Deleting a metadata file



# 5 Creating Graphs

## 5.1 Methods to Create a Graph

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

You can create a graph using an industry-specific template or without any template, or you can create a dynamic graph. No template is selected by default.

- Custom graph: This is a default graph creation method that fully meets your requirements.
- With an industry template: You can select a template you want, specify graph specifications, and add data to the template to create a graph.
- Dynamic graph: By default, the **dynamic graph analysis function** is enabled for graphs created in this mode.

You must create a dynamic graph to use the function. This function cannot be enabled for custom graphs and template-based graphs.

## 5.2 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. Select the **Region** where the cluster works from the drop-down list in the upper left corner of the page.
- 3. On the **Create Graph** page, click the **Customize Graph** tab and set the following parameters:
  - a. In the **Configure** step, set the graph name and software version.

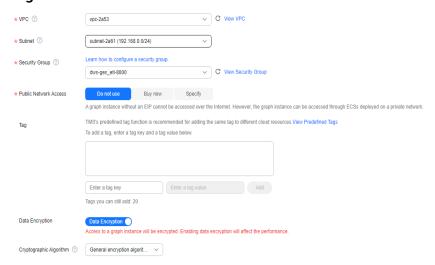
Figure 5-1 Graph name and software version



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.

Figure 5-2 Network information



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 <b>View VPC</b> 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 <b>Learn how to configure a security group.</b> 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.	
	<b>Do not use</b> : 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.	
	<b>Buy now</b> : 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 <b>ges_agency_default</b> in the project to support EIP binding.	
	<b>Specify</b> : Select an EIP to allow the graph instance to be accessed over the Internet.	
	Click <b>Create EIP</b> 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 <b>Add</b> to add the tag.	
	You can view the added tag in the graph details and search for graphs by tag on the <b>Graph Management</b> page.	
	Cough Management © States that (Collisions) was Commission of the party of collisions (Collisions) was Collisions (Collisions) was Collisions (Collisions) was Collisions (Collisions) was Collisions) was Collisions (Collisions) was Collisions) (Collisions) was Collisions) (Collisions) (Colli	
	NOTE  It is recommended that you use TMS's predefined tag function to add the same tag to different cloud resources.	
	Procedured Stigs (a)  The contract of the stigs (b) the contract of the contra	
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.

**Figure 5-3** 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.	
Versions	<ul> <li>GES editions.</li> <li>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.</li> <li>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.</li> </ul>	
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.	

Parameter	Description	
Graph Size	Available options based on your resource quota.	
(Edges)	Different graph specifications are displayed for <b>Enterprise production</b> and <b>Developer learning</b> .	
	<ul> <li>Development learning: Currently, there is only 10-thousand-edge graphs are available for this purpose, regardless of the edition.</li> </ul>	
	• <b>Enterprise production</b> : The specifications vary depending on the edition.	
	<ul> <li>Memory edition: Available options are million- edge, 10-million-edge, 100-million-edge, billion-edge, billion-edge-pro, and 10-billion- edge.</li> </ul>	
	<ul> <li>Database edition: Available options are billion-edge, 10-billion-edge, and 100-billion- edge.</li> </ul>	
	NOTE Graph size, which is based on the number of edges. The value is not accurate. If there are a large number of vertices and properties, you are advised to apply for graphs with a larger size.	
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.	
	<ul> <li>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.</li> </ul>	
	NOTE  If you cannot determine the maximum length of a vertex ID, set this parameter to <b>Hash</b> .	

- 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.

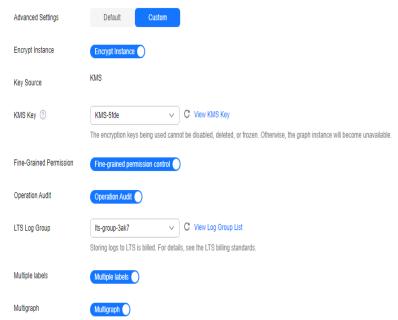


Figure 5-4 Advanced settings for the memory edition

Parameter	Description	
Encrypted Instance	Whether to encrypt a graph instance. <b>Key Source</b> is default to <b>KMS</b> . <b>KMS Key</b> : Select the key as needed. <b>NOTE</b> Some functions will be affected if you disable or delete a KMS key.	
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.	
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.	
Multiple labels	After this option is enabled, multiple labels can be set for the same vertex in the graph.	
	1. Only the memory edition supports this function.	
	<ol> <li>Each label corresponds to a unique property. When the API for querying vertex details is called, information about all labels and corresponding properties on the vertex is returned. Property filtering queries filter different labels on the vertex.</li> </ol>	

Parameter	Description
Multigraph	Once this option is enabled, multigraph clusters are supported by default. Such a cluster can have multiple graph instances, each allocated with different data. This allows you to analyze multiple graphs simultaneously.

If you choose the database edition, you can enable or disable
 HyG computing engine and Fine-Grained Permission.

**Figure 5-5** Advanced settings for the database edition



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.	
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 properties of a label.	

- 4. Click **Next**. The **Confirm** page is displayed.
- 5. Confirm the information and click **Submit** to create the graph.
- 6. 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.

## 5.3 Creating a Graph Using an Industry-Specific Template

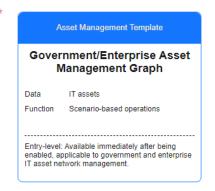
- 1. Log in to the GES console and click **Create Graph** in the upper right corner of the home page. The **Create Graph** page is displayed.
- 2. Select the **Region** where a graph works from the drop-down list in the upper left corner of the page.

3. On the **Create Graph** page, click the **Use Industry-Specific Graph Template** tab and configure the following parameters:

In the **Configure** step, select a template and configure network and graph information:

a. Select the desired template. Currently, **Asset Management Graph Template** and **Power Distribution Management Template** are available.

Figure 5-6 Selecting the template





- b. Set network information. Configure related parameters by referring to **Creating a Graph Without Using a Template**.
- 4. Click **Next**. On the **Confirm** page, confirm the specifications and click **Submit**. The system automatically creates the graph of the selected specifications and inserts the selected template data (schema and sample data).
- 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.

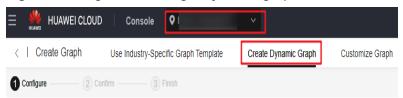
#### ∩ NOTE

- You do not need to set the name for a graph created using a template. By default, the name of the template is used as the prefix of the created graph, for example, assets management.
- After the graph is created, the name of the created graph is in assets\_management\_XXXX format, where XXXX is the unique identifier automatically generated by the system and cannot be modified.

## 5.4 Creating a Dynamic Graph

- Log in to the GES console. Select the region where the cluster is deployed from the region drop-down list. Click Create Graph in the upper right corner of the home page. The Create Graph page is displayed.
- On the displayed page, click the Create Dynamic Graph tab. The page for creating a dynamic graph is displayed.

Figure 5-7 Page for creating a dynamic graph



 Set required parameters by referring to Creating a Graph Without Using a Template.

By default, the **Dynamic graph analysis capability** is enabled for dynamic graphs.



- 4. Click **next**. On the **Confirm** page that is displayed, 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.
- 6. For details about how to use dynamic graphs, see **Dynamic Graphs**.

## 5.5 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**.

#### □ NOTE

If the startup fails, try again later. If the failure persists, fill in and submit a service ticket to contact the technical support.

----End

## 5.6 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.

Figure 5-8 Stopping a graph



**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

## 5.7 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.

Figure 5-9 Accessing a graph



## 5.8 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.

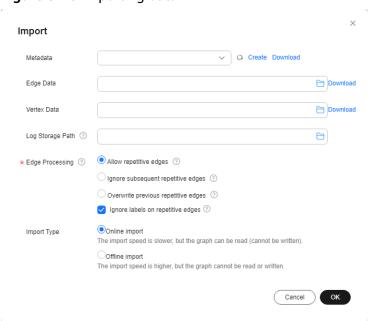
#### □ NOTE

- Currently, only graphs of version 1.1.8 and later support this function.
- 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 pane on the left, choose **Graph Management**.
- **Step 3** In the graph list, locate the target graph, click **More** in the **Operation** column, and select **Import**.

Figure 5-10 Importing data



**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. Storage on OBS may incur fees, so delete the data in time.
- 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.

#### 

- Graphs of the database edition support multi-graph management, and you need to select a graph name. Edge Processing and Import Type are not supported.
- 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

## **6** Managing Graphs

## 6.1 Graph Management Overview

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

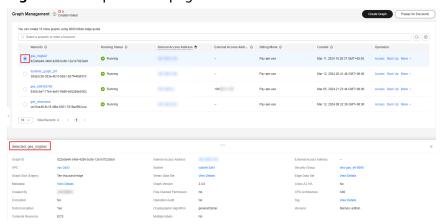
#### □ NOTE

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.



Method 1: Click next to a graph name to view the graph information, including Graph ID, VPC, Subnet, Security Group, Graph Size (Edges), Vertex Data Set, Edge Data Set, Metadata, Graph Version, Cross-AZ HA, Full-Text Indexing, Created By, Enterprise Project, CPU Architecture, Encrypted, Operation Audit, and Multiple labels.

Figure 6-1 Graph details page



Method 2: Click a graph name to access the details page and check its details.
 In the upper right corner of the page, you can click Access, Back Up, or More to manage the graph.

Figure 6-2 Graph details page



## 6.2 Viewing a Failed Graph

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

#### **Procedure**

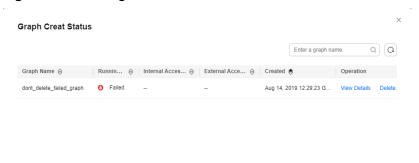
- **Step 1** In the navigation tree on the left, select **Graph Management**.
- **Step 2** In the upper left corner of the displayed page, view the number of graphs that fail to be created next to **Graph Management**.

Figure 6-3 Number of failed graphs



**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.

Figure 6-4 Viewing the creation status



□ 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.

Figure 6-5 Task details



**MOTE** 

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

----End

## 6.3 Backing Up and Restoring Graphs

## 6.3.1 Backing Up a Graph

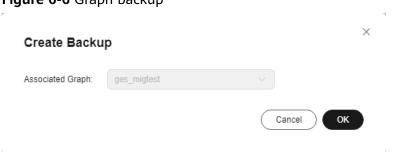
To ensure data security, back up the graph data so that you can restore it when faults occur.

#### **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.
  - In the dialog box displayed, click OK.

Figure 6-6 Graph backup



**◯** 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.

Figure 6-7 Backup management



- 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, set **Associated Graph** (a graph created by the current user) and click **OK** to start the backup.

Figure 6-8 Creating a backup



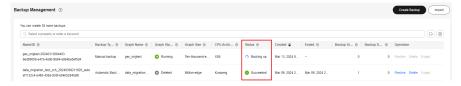
#### 

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.

Figure 6-9 Backup management



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.

### 6.3.2 Restoring a Graph

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

#### 

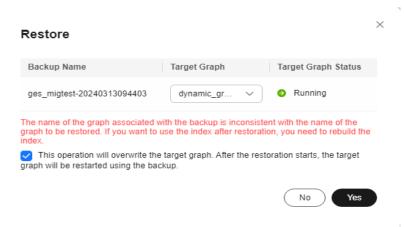
Ten-thousand-edge graphs and graphs of the database edition cannot be automatically backed up. You need to back up a graph and restore data from the manul backup. For graphs of other sizes, you can restore data from an automatic backup or manual backup.

The procedure is as follows:

- **Step 1** Log in to the GES management console and choose **Backup Management** from the navigation pane on the left.
- **Step 2** On the **Backup Management** page displayed, locate the row containing your desired backup and click **Restore** in the **Operation** column.
- Step 3 In the Restore dialog box, select This operation will overwrite the target graph.

  After the restoration starts, the target graph will be restarted using the backup. Then, click Yes.

Figure 6-10 Restoring data



**Step 4** 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

## 6.3.3 Deleting a Backup

If backup data is no longer used, you can delete it as needed.

- **Step 1** Log in to the GES management console and choose **Backup Management** from the navigation pane on the left.
- **Step 2** In the backup list, select your desired backup and click **Delete** in the **Operation** column.
- **Step 3** In the displayed dialog box, click **Yes** to delete the data.

#### 

- 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

### 6.3.4 Exporting a Backup to OBS

To migrate GES data across regions, you can export backup files to OBS.

#### **◯** NOTE

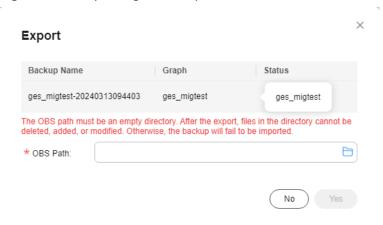
- Graphs of the database edition do not support this function.
- Only graphs of memory edition 2.3.16 or later support this function. To export graphs of
  an earlier version, you need to upgrade the graphs by referring to Upgrading a Graph,
  and then export the graphs.
- You need to back up the graph on the **Graph Management** page so that the graph can be displayed on the **Backup Management** page. For details, see **Backing Up a Graph**.
- On the Backup Management page, only graphs whose Graph Status is Running and Status is Successful can be exported to OBS. Otherwise, the Export button is unavailable.



- **Step 1** Log in to the GES management console and choose **Backup Management** from the navigation pane on the left.
- **Step 2** In the backup list, select the backup to be exported and click **Export** in the **Operation** column.
- **Step 3** In the dialog box that is displayed, verify that the backup information is correct and select an OBS path.

Note that the OBS export path can only be an empty directory, and after the export, the graph data files under that directory cannot be deleted, added, or modified. Otherwise, the backup will fail when importing from OBS to the graph.

Figure 6-11 Exporting a backup to OBS



**Step 4** Click **OK** to back up the graph.

Storing backup files in OBS will incur charges. For details, see OBS Billing.

**Step 5** After the task is delivered, you can view its execution status on the **Task Center** page.

----End

## 6.3.5 Importing a Backup from OBS

You can import a backup file exported to OBS to a graph. After the import is successful, you can use the backup to restore the graph instance.

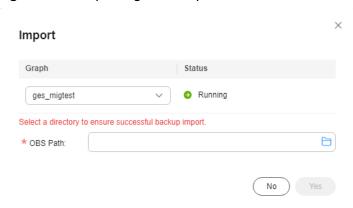
#### **Ⅲ** NOTE

- Graphs of the database edition do not support this function.
- Only graphs of memory edition 2.3.16 or later support this function. To export graphs of an earlier version, you need to upgrade the graphs by referring to **Upgrading a Graph**, and then import the graphs.

The procedure is as follows:

- **Step 1** Log in to the GES management console and choose **Backup Management** from the navigation pane on the left.
- **Step 2** In the upper right corner of the page displayed, click **Import**.
- **Step 3** In the dialog box that is displayed, select the graph to be imported and the OBS path where the backup is stored, and click **OK** to import the backup.

Figure 6-12 Importing a backup



**Ⅲ** NOTE

Select a directory (folder) to ensure successful backup import.

**Step 4** After the task is delivered, you can view its execution status on the **Task Center** page.

----End

## 6.4 Upgrading a Graph

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

□ NOTE

Currently, only graphs of version 1.0.3 and later can be upgraded.

The procedure is as follows:

- **Step 1** Log in to the GES management console and choose **Graph Management** from the navigation pane on the left.
- **Step 2** Locate the target graph in the graph list and choose **More** > **Upgrade** in the **Operation** column.
- **Step 3** 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 4** Click **OK**. The graph status changes to **Upgrading**. Wait several minutes, the status will become **Running** after the upgrade is successful.

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

----End

## 6.5 Exporting a Graph

You can export graph data to a custom OBS directory.

- Graph data of memory edition 1.0.3 or later can be exported.
- Graph data of database edition 2.3.14 or later can be exported.

- **Step 1** Log in to the GES management console and choose **Graph Management** from the navigation pane on the left.
- **Step 2** Locate the target graph in the graph list and choose **More** > **Export** in the **Operation** column.

Figure 6-13 Exporting a graph

- **Step 3** In the lower part of the page that is displayed, select a storage path. (For a graph of the database edition, you also need to select the graph name.)
- **Step 4** 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.

#### 

If you choose to export CSV files to your local host, the files are opened using the spreadsheet software by default. You are advised to open the files in a text editor. If the data contains special characters such as plus signs (+), minus signs (-), equal signs (=), and at signs (@), the data will be parsed into formulas by the software. To ensure system security, pay attention to the following when opening such files:

- 1. Do not select Enable Dynamic Data Exchange Server Launch (not recommended).
- 2. Do not select **Enable** or **Yes** if a dialog box indicating a security issue is displayed.

----End

## 6.6 Restarting a Graph

You need to restart a graph in the following cases:

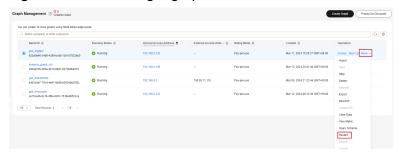
- 1. If you access a graph in the **Running**, **Importing**, **Exporting**, or **Clearing** status and an unknown exception occurs, you can restart the graph.
- 2. You can restart a graph that is stuck in a state. For example, if a graph stuck in the **Exporting** status for a long time because the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

The procedure is as follows:

**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.

Figure 6-14 Restarting a graph



**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

## 6.7 Resizing a Graph

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 including composite indexes and full-text indexes.

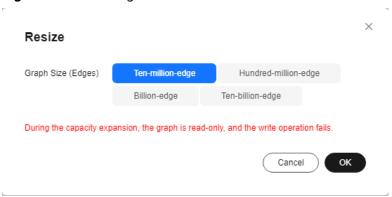
- **Step 1** Log in to the GES management console. In the navigation pane on the left, choose **Graph Management**.
- **Step 2** In the graph list, locate the row containing the graph you want to resize, click **More** in the **Operation** column, and select **Resize**.

| Compt | Management | Managem

Figure 6-15 Selecting Resize

**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.

Figure 6-16 Resizing



**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

## 6.8 Expanding a Graph

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

#### **Ⅲ** 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.

- **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.

| Value of Control Con

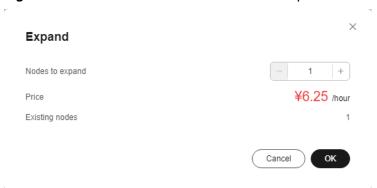
Figure 6-17 Expanding a graph

□ NOTE

Only a running graph can be expanded.

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

Figure 6-18 Select the number of nodes to expand



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

----End

## 6.9 Binding and Unbinding an EIP

#### Binding an EIP

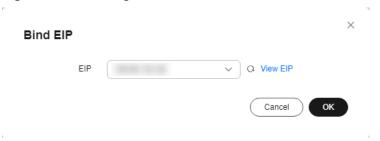
To access GES over the Internet, you can bind an Elastic IP Address (EIP) to your instance.

- **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.

If no EIP is available, click **Create EIP** to create one. Then, click to refresh the list and select the created EIP.

Figure 6-19 Binding an 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

## 6.10 Clearing Data

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.

The procedure is as follows:

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

**Step 2** In the graph list, locate your desired graph, click **More** in the **Operation** column, and select **Clear Data**.

Figure 6-20 Selecting Clear Data



**Step 3** In the dialog box that is displayed, select or deselect **Clear the metadata in the graph**. (For a database edition graph, you need to select the graph name first.)

#### ■ NOTE

- If you clear graph metadata, the graph will be reset, and all data and running tasks will be cleared.
- Deleted metadata cannot be recovered. Exercise caution when performing this operation.

#### Step 4 Click Yes.

----End

## 6.11 Deleting a Graph

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

#### **Ⅲ** 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.

- **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 **Delete Graph** dialog box displayed, determine:
  - Whether to delete the EIPs bound to the graph instance. If no EIPs are bound, this option is unavailable. EIPs that are not selected will continue to incur fees. If you do not select the EIPs, the EIPs are retained by default.
  - Whether to delete graph backups. By default, one automated backup and two manual backups are retained, occupying the backup quota. If you do not select the backups, the backups are retained by default.

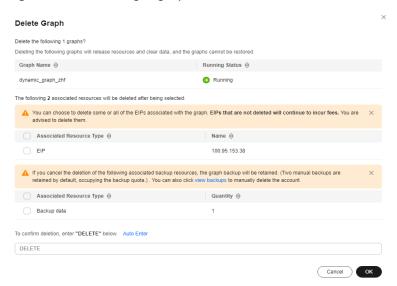


Figure 6-21 Deleting a graph

Step 5 Click OK.

----End

## **6.12 Viewing Monitoring Metrics**

Cloud Eye monitors the running status of GES. You can view the monitoring metrics of GES on the Cloud Eye management console.

It takes a period of time for transmitting and displaying monitoring data. The GES status displayed in the Cloud Eye monitoring data is the status obtained 5 to 10 minutes before. You can view the monitoring 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 monitoring metrics.
- You can view monitoring data of graphs in the running, importing, exporting, and clearing states.

#### **Viewing Monitoring Metrics**

- **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.

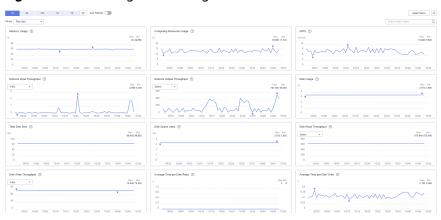


Figure 6-22 Viewing monitoring metrics

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



Figure 6-23 Enabling the full image mode

Figure 6-24 Zoomed in monitoring graph



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

- 1. Fixed time ranges include 1h, 3h, 12h.
- 2. The automatic refresh interval is 60s, which is used as the user monitoring period.

----End

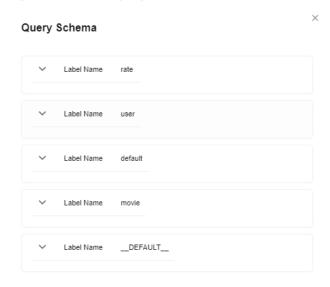
## 6.13 Querying Schema

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

The procedure is as follows:

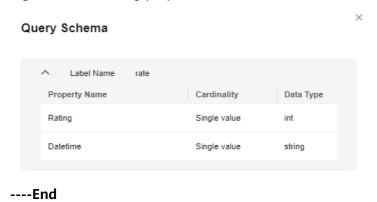
- **Step 1** Log in to the management console.
- **Step 2** In the navigation pane, choose **Graph Management**. In the **Operation** column, choose **More > Query Schema**. A window is displayed, showing the labels contained in the metadata of the current graph.

Figure 6-25 Querying schema



**Step 3** To view the properties contained in a label, click  $\checkmark$  of each label.

Figure 6-26 Viewing properties in labels



## Accessing and Analyzing Graph Data

## 7.1 Graph Editor

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

Figure 7-1 Editor page

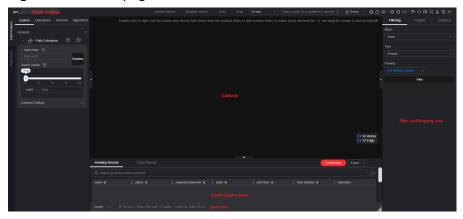


Table 7-1 Graph editor

Area	Description	
Exploratio n pane	Graph exploration tools, for example, path expansion. For details about the functions, see <b>Exploring Graphs</b> .	
Operation s	Operations executed by API calls. For details, see <b>Adding Custom Operations</b> .	
Schema	Metadata operations, such as adding, hiding, importing, and exporting data. For details, see <b>Editing Schema</b> .	

Area	Description	
Algorithm s	Algorithms supported by GES. You can set the properties of each algorithm in this area. Table 7-2 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.	
Canvas	Graph structure of data. Shortcut operations are preset in the drawing area for you to easily analyze the graph data.  Table 7-3 describes the functions of the drawing area.	
Query box	<ol> <li>Gremlin query statements</li> <li>Cypher query statements</li> <li>DSL query statements</li> </ol>	
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 <b>View Property</b> from the shortcut menu to view the Filter and Property area.	
	It contains the following three tabs:	
	<ul> <li>The Filtering tab page allows you to set properties and conditions to filter the data for analysis. For details, see Filter Criteria.</li> </ul>	
	The <b>Property</b> tab page displays the property information about a vertex or an edge.	
	<ul> <li>The statistics tab displays the number of labels and vertex weights of the selected vertices and edges. For details, see Statistics Display.</li> </ul>	

Figure 7-2 Algorithm Library

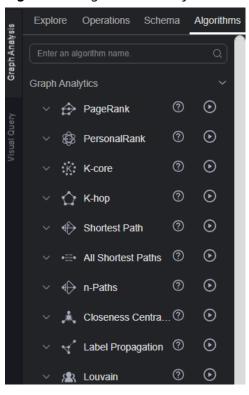


Table 7-2 Algorithm library description

Interface Element	Description
Enter an algorithm name.	Enter the algorithm name to quickly find it.
~	Expand the algorithm parameter configuration area.
<b>•</b>	Run the algorithm.
A PageRank ② ●  alpha ②  0.85  convergence ②  0.00001  max_iterations ③  1000  directed ②  true	Set the properties of an algorithm. Different algorithms have different properties. For details, see <b>Algorithms</b> .

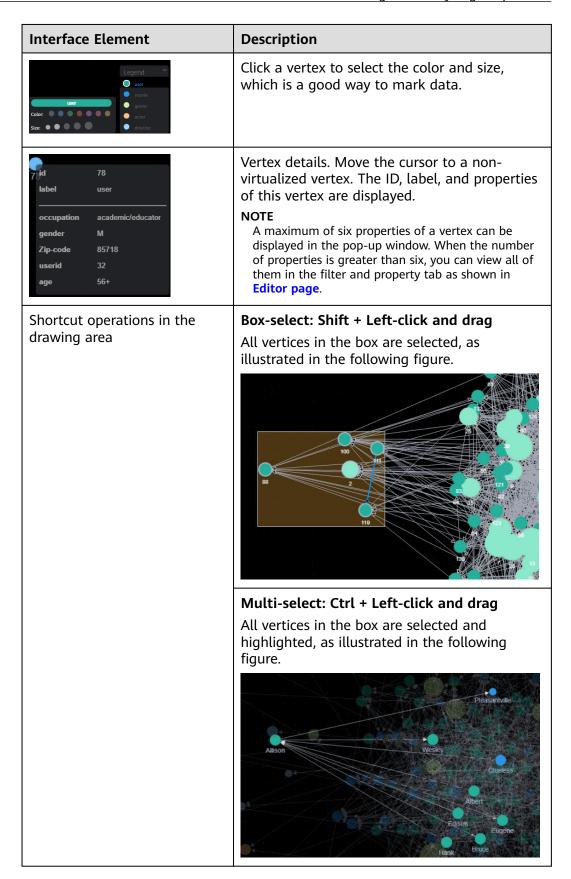
Figure 7-3 Canvas



Table 7-3 Canvas description

Interface Element	Description
13 /886813 Vertex 9 /892773 Edge	Row 1: <b>13</b> indicates the number of vertices displayed on the current canvas and <b>886813</b> indicates the total number of vertices in the entire graph.  Row 2: <b>9</b> indicates the number of edges
	displayed on the current canvas and <b>892773</b> 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 <b>Ctrl</b> and click and drag to select an area on the canvas, and then click <b>Isolated Vertices</b> .
	<ul> <li>To display all isolated vertices in the canvas, click Isolated Vertices.</li> </ul>
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.
Ail data	All data indicates all data of a graph.
	Current data indicates the data rendered on the canvas.
ি☆ Theme ∨	You can change the theme of the GES editor. Three themes are supported: light, dark, and system.

Interface Element	Description
Enter a vertex ID or multiple IDs separate Q	After you select <b>All data</b> or <b>Current data</b> , enter the node ID in the search box, for example, <b>2</b> . Press <b>Enter</b> or click the query icon to search for the corresponding vertex and render it to the canvas. <b>NOTE</b> • Currently, only a single vertex ID can be entered. • If you choose <b>Current data</b> from the drop-down list, vertices on the current canvas are highlighted.
台	Click <b>Clear</b> to clear all content on the canvas.
企	Export the canvas content as a PNG or CSV file (snapshot or vertex and edge file of the current canvas).
	<ul> <li>Keyboard shortcuts</li> <li>Ctrl+E: Select an associated entity.</li> <li>Ctrl+'+': Zoom in.</li> <li>Ctrl+'-': Zoom out</li> <li>Ctrl+Z: Undo an operation.</li> <li>Ctrl+A: Select all.</li> <li>Ctrl+Delete: Clear the canvas.</li> <li>Delete: Hide vertices.</li> <li>Ctrl+Click: Select multiple vertices and edges.</li> </ul>
Q	Zoom in the graph. You can zoom in a graph to at most 600%.
Q	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.
> ○ m × A Z K	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.



Interface Element	Description
	Select/Deselect: Ctrl + Left-click
	Press <b>Ctrl</b> and left-click a vertex or an edge to select and highlight it. Press <b>Ctrl</b> 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 <b>Ctrl + E</b> 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.
	Zoom in and zoom out: Scroll the mouse wheel forwards and backwards.
	Scroll the mouse wheel to zoom in or out the graph.

Figure 7-4 Force directed

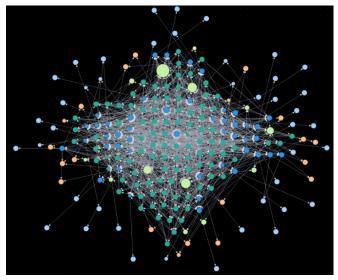


Figure 7-5 Circle

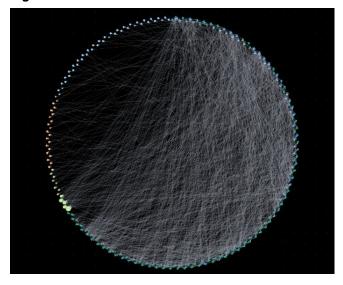


Figure 7-6 Grid

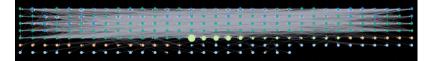


Figure 7-7 Radial-tree

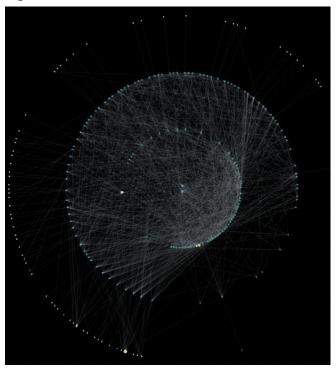


Figure 7-8 Hierarchical

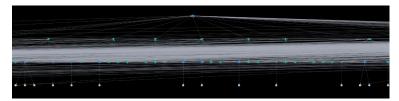
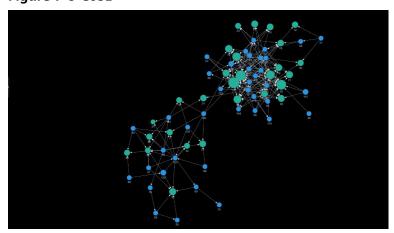


Figure 7-9 CoSE



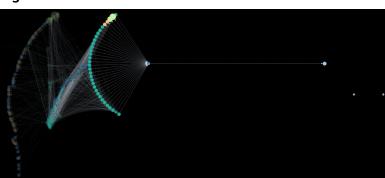


Figure 7-10 Double-core

# 7.2 Accessing the GES Graph Editor

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.

The procedure is as follows:

- 1. Log in to the GES management console and choose **Graph Management** from the navigation pane on the left.
- 2. On the **Graph Management** page, select the graph to be accessed and click **Access** in the **Operation** column.

**Figure 7-11** shows the graph editor page. You can analyze the graph data on the graph editor. For details, see **Graph Editor**.



Figure 7-11 Graph editor

# 7.3 Dynamic Graphs

# 7.3.1 Timeline

If you want to view vertex and edge changes over time, a timeline is required to convert a static graph into a dynamic graph. This also allows you to get dynamic analysis result.

### ■ NOTE

To use this function, you need to create a dynamic graph. For details, see **Creating a Dynamic Graph**.

## **Setting a Timeline**

- Log in to the GES console and choose Graph Management from the navigation pane on the left. On the displayed page, locate the dynamic graph and click Access in the Operation column.
- 2. On the displayed graph editor page, set the following parameters in the **Timeline Settings** dialog box:

#### **○** NOTE

The parameters set here will be synchronized to those in **Community Evolution** and **Temporal BFS**.

- Start Time Property: Name of the start time property that is a property
  of the imported or created metadata. The default value is startTime. The
  name must be of the date, long, or int type.
- **Start**: Start time of the dynamic graph. The start time must be earlier than or equal to the end time.
- End Time Property: Name of the end time property that is a property of the imported or created metadata. The default value is endTime. The name must be of the date, long, or int type.
- End: End time of the dynamic graph.
- Advanced Settings: Use Default settings or Custom settings.
  - Default: Use the default settings.
  - Custom: Set the display duration of vertices and edges in the graph and the display priority of labels.
    - Vertex/Edge Display: How long the vertices and edges in an algorithm result will be displayed on the canvas. This function is supported for Temporal BFS only. The value must be a timestamp in seconds. The default value is 604800 (7 days).

This function is used to the returned vertex and edge data that contains the start time only.

 Label Display Priority: This parameter is available only for temporal graphs with multiple labels. For details about how to create such a graph, see Creating a Graph. You can select multiple labels. When two vertex labels have identical start and end times, the label on the left will be displayed first.

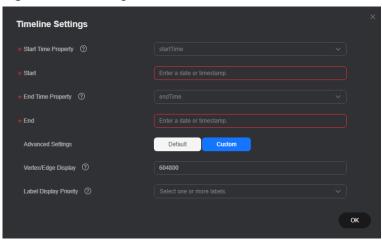


Figure 7-12 Setting a timeline

3. Click OK.

■ NOTE

If you want to modify the timeline parameters, click in the lower left corner of the canvas.

# 7.3.2 Community Evolution

The community evolution algorithm generates a dynamic graph that shows structure changes of a community over time. The procedure to use this algorithm is as follows:

- 1. Set parameters in the **Community Evolution** drop-down list in the **Temporal** tab of the **Graph Analysis** area on the left of the graph editor page.
  - Set the start time, end time, and their properties. For details see Setting
     a Timeline. To modify the parameters, click in the lower left corner of the canvas.
  - Vertices: IDs of vertices in the community. You can enter a maximum of 100,000 vertex IDs. Use commas (,) to separate them.

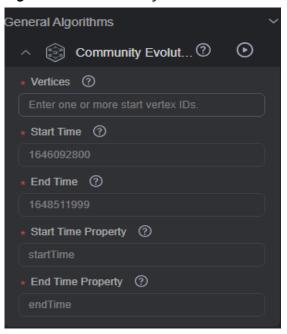
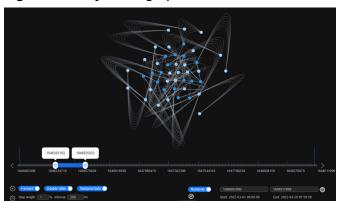


Figure 7-13 Community evolution

2. Click on the right of **Community Evolution**. The running result is displayed on the canvas.

Figure 7-14 Dynamic graph



UI Element	Description
•	Start playback.
Forward	Playback direction of the dynamic graph. If you toggle on this switch, the playback will be forward. If you toggle off this switch, the playback will be backward.

UI Element	Description
Double slider	Whether the playback uses the double slider
	Toggled on (by default): Two sliders are used for playback. The start and end sliders move forward or backward at the same time, and the length of the time window represented by the distance between the sliders remains unchanged.
	<ul> <li>Toggled off: Only the one slider is used for playback.</li> </ul>
	<ul> <li>If the playback is forward, the start slider is fixed and end slider moves froward on the timeline.</li> </ul>
	<ul> <li>If the playback is backward, the end slider is fixed and start slider moves backwards on the time line.</li> </ul>
All data	Whether data displayed on the canvas contains static data. If you toggle on this switch, only dynamic data is displayed.
	Static data refers to the data that does not change over time.
Numerals	<ul> <li>Whether the timeline uses dates or timestamps.</li> <li>By default, this switch is toggled on, which means that you need to enter timestamps to specify the duration.</li> </ul>
	<ul> <li>If you toggle this switch off, you enter dates and time to specify the duration.</li> </ul>
Start End	Steet time and end time of the duration you want to view graph data changes
<b>©</b>	Timeline settings. For details about how to set the parameters, see <b>Setting a Timeline</b> .
1646092800 1648511999 Start: 2022-03-01 08:00:00 End: 2022-03-29 07:	length: Length of each step that the slider
	Interval: Interval between two steps
NAME	NAMES DECISION DECISI

# 7.3.3 Temporal BFS

Temporal breadth-first search (BFS) algorithm searches for associated vertices based on temporal message passing and temporal BFS algorithms, and outputs

the visit time of each vertex and the distance from the vertex to the source start vertex. The procedure to use this algorithm is as follows:

- In the Temporal tab of the Graph Analysis area on the left of the graph editor page, click Temporal BFS, and set the parameters in the drop-down list.
  - Set the start time, end time, and their properties. For details see Setting
     a Timeline. To modify the parameters, click in the lower left corner of the canvas.
  - **Start Vertex**: ID of the start vertex
  - k: Traversal depth, indicating the maximum number of vertices in a traversal. The value ranges from 1 to 100. The default value is 3.
  - Direction: Whether the traversal is performed along the directions of edges in the graph. The value can be true (default value) or false.
    - true: Traversal is performed along edge directions.
    - **false**: Edge directions will not be considered in the traversal.

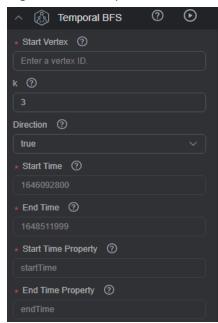


Figure 7-15 Temporal BFS

2. Click on the right of **Temporal BFS**. The running result is displayed on the canvas. In this algorithm, a single slider is used for playback. As shown in **Figure 7-16** and **Figure 7-17**, the vertices in the dynamic graph are increases over time.

Figure 7-16 Execution result 1

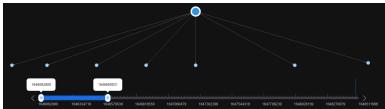
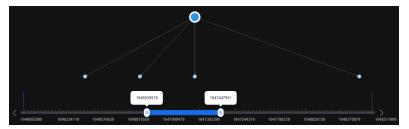


Figure 7-17 Execution result 2



# 7.3.4 Temporal Paths

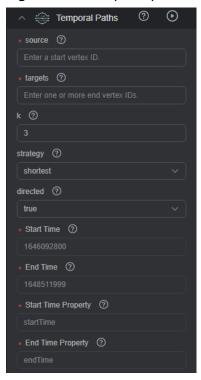
Temporal paths that start from a vertex to the target node show the trend of increment (or non-decrement) of vertices and edges over time on the canvas. The paths follow the order of information transmission on dynamic graphs, the passing time of an edge on a path must be later than or the same as that of the previous edge.

For this feature, you can use the **strategy** parameter to adjust whether the temporal path with the shortest distance or the temporal path that reaches the target node as early as possible is searched for. The procedure is as follows:

- In the **Temporal Paths** tab of the **Graph Analysis** area on the left of the graph editor page, click **Temporal BFS**, and set the parameters in the dropdown list.
  - Set the start time, end time, and their properties. For details see **Setting** a Timeline. To modify the parameters, click in the lower left corner of the canvas.
  - source: ID of the start node
  - targets: set of end node IDs. Multiple end node IDs can be configured.
  - k: Traversal depth, indicating the maximum number of vertices in a traversal. The value ranges from 1 to 100. The default value is 3.
  - **strategy**: execution strategy of the algorithm. The value can be **shortest** or foremost.
    - **shortest**: the temporal path with the shortest distance is returned
    - **foremost**: the temporal path that reaches the target node as early as possible is returned
  - directed: Whether the traversal is performed along the directions of edges in the graph. The value can be true (default) or false.

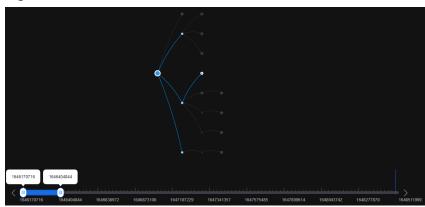
- true: Traversal is performed along edge directions.
- false: Edge directions will not be considered in the traversal.

Figure 7-18 Temporal paths



 Click on the right of Temporal Paths. The execution results are displayed on the canvas. As shown in Figure 7-19 and Figure 7-20, the vertices in the dynamic graph change over time.

Figure 7-19 Execution result 1



1646592145 1546028275 1647107016 164710729 164721307 1647275485 1647809514 164802742 164827427 164851109

Figure 7-20 Execution result 2

# 7.4 Graph Exploration

Handful graph exploration tools facilitate your analysis.

**◯** NOTE

Multi-label graphs do not support graph exploration.

### **Path Extension**

Filters are added to query APIs to search for the desired k-hop vertices or edges. For details about APIs for filtered queries, see **Filtered Query V2**.

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.

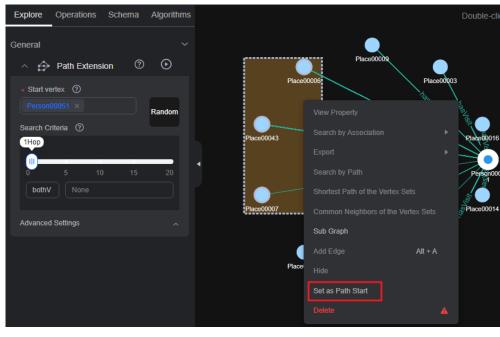
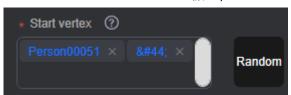


Figure 7-21 Selecting start vertices

- 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.

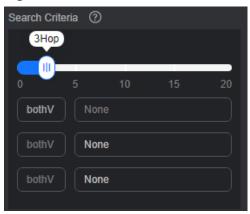
### 

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



• **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 7-22 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:

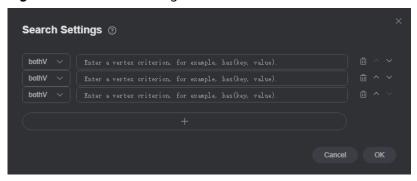
has: A property key or the value of a property key must be contained.

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 7-23 Search settings



### **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.

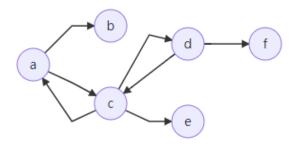
and(has('person'),or(has('name','peter'),has('age','30')): Search for a vertex whose name is peter and age is 30.

- 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.

### **Ⅲ** 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.

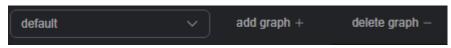
For **ShortestPath**, the query process generates the **a->c->d->f** path only.

# 7.5 Multi-Graph Management (Database Edition)

When you create a database graph, it is automatically upgraded to a multi-graph cluster. This cluster can have multiple graph instances, each allocated with different data. This allows you to analyze multiple graphs simultaneously.

In the graph engine editor, you can manage the graph instances in the graph cluster by clicking the dropdown menu next to the cluster name in the upper left corner of the page to switch between graph instances.

Figure 7-24 Multi-graph management



□ NOTE

Only graphs of the database edition support this function.

## Adding or Deleting a Graph

- 1. After the database graph cluster is created, the graph engine editor page is displayed. For details, see **Accessing the GES Graph Editor**.
- In the upper left corner of the page, click Add Graph. In the dialog box displayed, enter the graph name and select the vertex ID type.
   Currently, two vertex ID types are supported: fixed-length string and hash.
  - 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

- 1. If you cannot determine the maximum length of a vertex ID, set this parameter to **Hash**
- 2. If you select the fixed-length string (fixedLengthString), you also need to enter the length of the vertex ID.

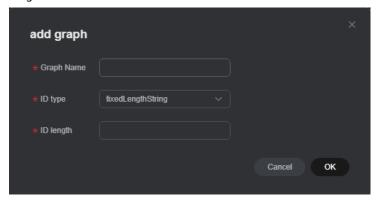
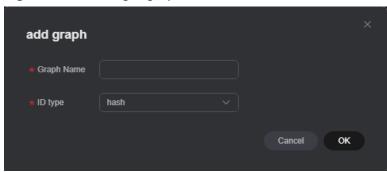


Figure 7-25 Adding a graph



- 3. After setting the parameters, click **OK**.
- 4. To delete a graph instance, click **Delete Graph**.

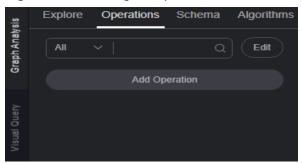
# 7.6 Adding Custom Operations

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

## **Procedure**

 In the Operations tab on the left of the graph editor, click Edit. The Add Operation button is displayed.

Figure 7-26 Adding an operation



- 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.

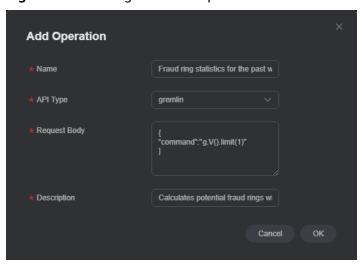
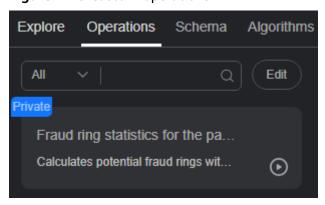


Figure 7-27 Adding a custom operation

- 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.

Figure 7-28 Custom operations



5. To delete the operation, click **Edit**. Then click displayed in the upper right corner of the operation.

# 7.7 Editing Schema

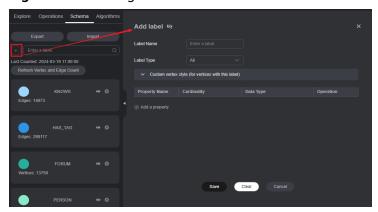
# Adding a Label

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

- Label Name: name of the label to be added.
- **Type**: You can select a label type (vertex, edge, or general-purpose). General-purpose indicates that a label can represent either a vertex or an edge.
- **Custom vertex style**: You can define the color and mark of a 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 7-29 Adding a label



# **Counting Vertices and Edges**

On the **Schema** tab of the GES editor, click **Refresh Vertex and Edge Count**. The system counts the total number of vertices and edges in the current graph. You can also view the last count time.

Export Import

Export Import

H Enter a label.

Last Counted: 2024-03-19 11:00:00

Refresh Vertex and Edge Count

KNOWS

Edges: 14073

HAS\_TAG

Edges: 290117

FORUM

Vertices: 13750

PERSON

©

DEFAULT

Vertices: 76750

Edges: 1

Figure 7-30 Counting vertices and edges

## Modifying a Label

## **Ⅲ** NOTE

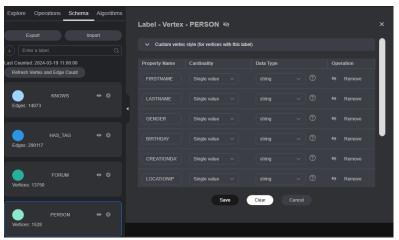
This function is only available on graph version 2.3.18 or later.

In the metadata file list, click the metadata file for which you want to modify the label. The metadata label details page is displayed.

- You can modify the label's property name, cardinality, and data type.
- To hide or delete a property, click the hide or remove button in the Operation column.
- If you accidentally deleted or incorrectly modified a property, simply click the reset button to revert back to the last saved data.

Confirm the modification and click Save.

Figure 7-31 Modifying a label



# 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.

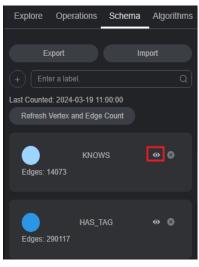


Figure 7-32 Hiding a label

- 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.

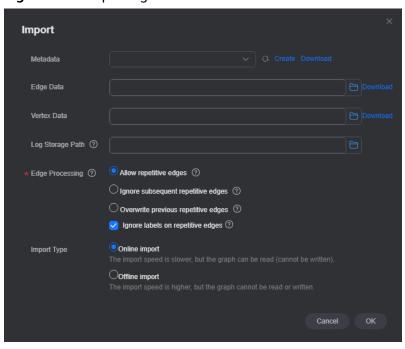
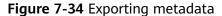
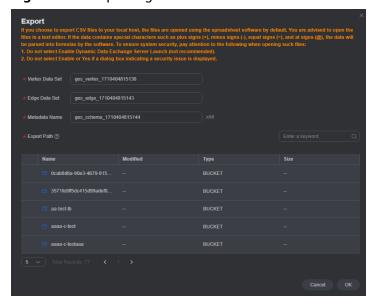


Figure 7-33 Importing metadata

• 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.





## **Deleting a Label**

## □ NOTE

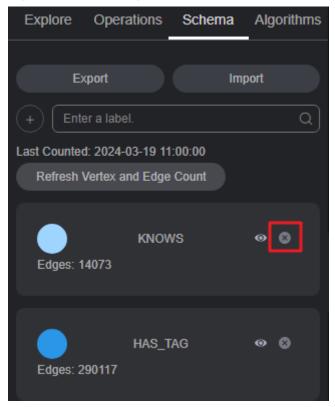
- 1. After this API is called, all data associated with the label will be deleted. Exercise caution when performing this operation.
- 2. If the graph version is earlier than 2.2.18, schema labels cannot be deleted.
- 3. Schema labels cannot be deleted from graphs of the database edition.
- 4. The default label \_DEFAULT\_ cannot be deleted.



To delete a label, do the following:

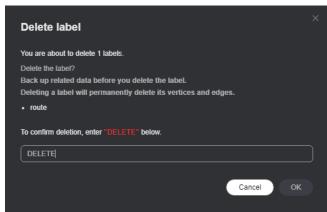
1. To delete a label, click the deletion icon next to the schema on the **Schema** tab on the left of the graph engine editor.

Figure 7-35 Deleting a label



2. In the dialog box that is displayed, read the message carefully, confirm the name of the label to be deleted, enter **DELETE** in the text box, and click **OK**.

Figure 7-36 Confirming the deletion



3. During the deletion, the result of deleting the label algorithm is displayed in the result display pane below the canvas.

Figure 7-37 Results display



During the deletion, the **Filter** button on the **Filtering** tab is grayed out and becomes unavailable.

# 7.8 Visual Query

In the graph editor, you can create graph query statements by dragging and dropping vertices and edges, and preview the query results without writing any code.

## **Procedure**

1. In the left pane of the graph editor, click the **Visual Query** tab.



Figure 7-38 Visual query

- 2. Add a vertex to the canvas.
  - a. In the **Add Vertex Pattern** tab, all vertex labels and edge labels of the graph are displayed. Each label is displayed as a card that can be dragged to the canvas. Select a vertex label and drag it to the canvas.

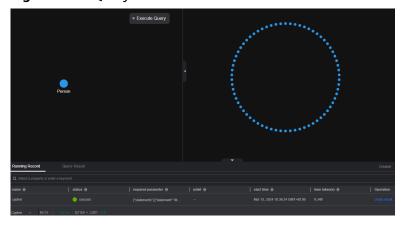
The Cypher query statement below changes with your operations.

result.

These vertex labels and edge labels are the same as those in the metadata list in **Editing Schema**.

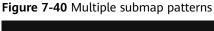
b. Drag the labels you want to use for the query to the canvas and click **Execute Query**. The graph result is displayed on the right of the canvas. You can view the running records of the Cypher query statement in the **Running Record** tab below the canvas. Click **Query Result** to view the

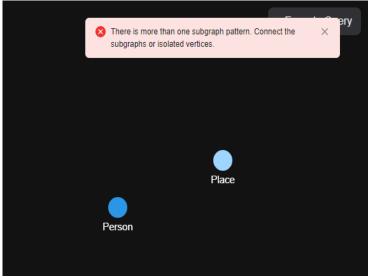
Figure 7-39 Query result



## ■ NOTE

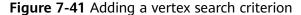
Query results can be displayed only when there is only one submap pattern on the canvas. If there are multiple disconnected submaps or isolated vertices, you must add edges to connect the submaps or isolated vertices. You can also set multiple labels to reconstruct your query mode. Otherwise, when you click **Query**, the system displays a message indicating that there are multiple submap patterns.

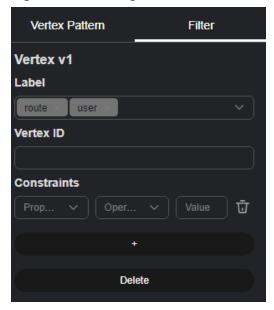




### 3. Add a vertex filter.

Click a vertex in the canvas. The **Filter** tab page is displayed in the left pane. On the **Filter** tab, specify labels, vertex ID, and property search criteria to search for the vertex labels you want to view on the canvas.





- Vertex V1: Cypher variable ID (vertex identifier in the Cypher query statement below the canvas), which is named based on the sequence in which vertices are dragged to the canvas, for example, V1, V2, and more alike.
- Label: Set one or more labels to search for target vertices. The logical operator between each two labels is OR.
- Vertex ID: It is equivalent to a filter criterion. After adding a vertex ID to a vertex label, you can click **Query** to query the vertex labels with the same vertex ID.
- Constraints: Specify a property contained in the vertex label. Currently, a property with multiple values is not supported.
  - Property: Property contained in the label.
  - Operator: Comparison operators (>,>=,<,<=,=,<>), null judgment operators (is null, is not null), and string comparison operators (starts with, ends with, contains) are supported.

#### 

**starts with** searches for a property that starts with a specified string; **ends with** searches for a property that ends with a specified string; contains searches for a property that contains a specified string.

- Value: Property value. The attribute value type must be the same as that in the metadata. If the attribute value is of the character type, you need to use single quotation marks (").
- Ū: Delete the constraint.
- + button: Add a criterion.
- Delete: Delete the added criterion.

Click **Execute Query** in the canvas again. The query result is displayed on the right of the canvas.

4. Add an edge (connect two vertices on the canvas):

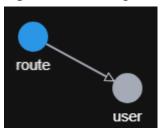
Double-click a vertex. After the border of the vertex turns red (do not move the cursor out of the red border), click and drag a line from the vertex to another vertex.

The Cypher guery statement below changes with your operations.

Figure 7-42 Red border of a vertex



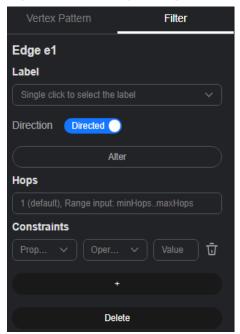
Figure 7-43 Adding an edge



## 5. Add an edge filter.

Click an edge in the canvas. The **Filter** tab page is displayed in the left pane. On the **Filter** tab, specify labels, direction, hops, and property search criteria to search for the edge labels you want to view on the canvas.

Figure 7-44 Adding an edge filter



- **Edge e2**: Cypher variable ID, which is named based on the sequence in which edges are added to the canvas, for example, e1, e2, and more alike.
- **Label**: Set one or more labels to search for target edges. The logical operator between each two labels is OR.
- Direction: Select the direction contained in the edge label.
  - When the slider is toggled on, the edge is a directed one. When the slider is toggled off, the edge is undirected (or called bidirectional).
  - If the edge is directed, the arrow on the canvas indicates the direction of the edge. You can click **Change Direction** to change the direction of the selected edge on the canvas.
- Hops: The default value is 1. The value range is [0, 20). You can specify a number or a range.
  - If you enter an integer, it will be used as the number of hops in the edge pattern.

- If you enter two integers in the format of *minHops..maxHops*, for example, **2..3**, the number of hops in the edge pattern is within the range of [2,3].
- Constraints: Specify a property contained in the edge label. Currently, a property with multiple values is not supported.
  - **Property**: Property contained in the label.
  - Operator: Comparison operators (>,>=,<,<=,=,<>), null judgment operators (is null, is not null), and string comparison operators (starts with, ends with, contains) are supported.

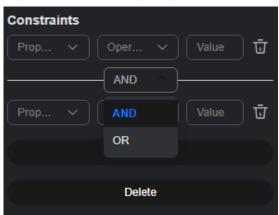
#### 

**starts with** searches for a property that starts with a specified string; **ends with** searches for a property that ends with a specified string; contains searches for a property that contains a specified string.

- Value: Property value. The attribute value type must be the same as that in the metadata. If the attribute value is of the character type, you need to use single quotation marks (").
- Ū: Delete the constraint.
- + button: Add a criterion.

If there is more than one criterion, click next to AND to set the logical operator (AND or OR).

Figure 7-45 Selecting a logical operator



### **NOTE**

The priority of AND is higher than OR. The suggested calculation sequence is as follows:

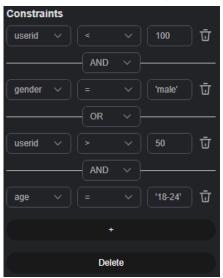
- 1. Arrange all AND operations first.
- 2. Then, perform all OR operations.

In the following example, the edge search criterion is **userid < 100 AND gender** = 'male' OR userid > 50 AND age = '18-24'.

The operation sequence is:

(userid < 100 AND gender = 'male') and (userid > 50 AND age = '18-24') are operated first, and result1 and result2 are recorded respectively.

Then, result1 OR result2 is operated.



Delete: Delete the added criterion.

Click **Execute Query** in the canvas again. The query result is displayed on the right of the canvas.

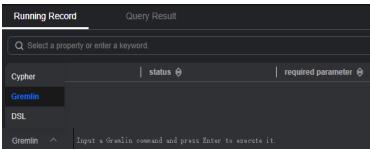
# 7.9 Gremlin Query

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.

The procedure is as follows:

- 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 **Gremlin**. Enter a query statement and press **Enter** to run the statement.

Figure 7-46 Switching to Gremlin query



### **Ⅲ** NOTE

Multi-label graphs do not support Gremlin query.

### **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 gueries 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
  - q.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.

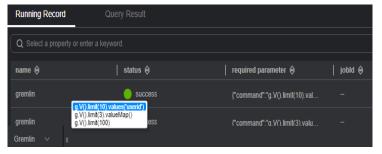
Deleting an edge

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

#### 

- 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 7-47 Historical queries



- 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 7-48 Gremlin keywords



# **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
textClSubString	Substring that ignores cases
textFuzzy	Fuzzy match
textPrefix	Prefix query
textRegex	Regular expression match

## **◯** NOTE

When specifying 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 7-4 shows how Gremlin in GES differs from open source Gremlin.

Table 7-4 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 <b>property()</b> and <b>remove()</b> to modify property values. The value set by <b>property()</b> is determined by the corresponding parameter. <b>remove()</b> converts string properties into empty strings, digital properties into 0, and list properties into empty lists.

Difference	Description
Variables	The GES graph structure does not support the <b>variables</b> 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. For details about the features, visit the **Gremlin official website**.

gremlin> graph.features()
==>FEATURES

#### 

Currently, the following step commands are not supported:

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

# 7.10 Cypher Query

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

The procedure is as follows:

- Access the GES graph editor. For details, see Accessing the GES Graph Editor.
- 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 7-49 Creating an index



### **◯** NOTE

After an index is created, wait for 30 seconds for the index to take effect and then perform Cypher query.

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.

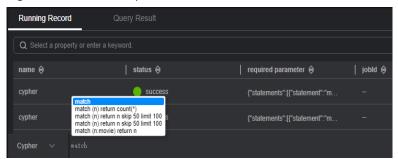
• Querying a schema

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 7-50 Historical queries



- 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 7-51 Cypher keywords



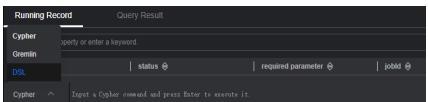
# 7.11 DSL Query

DSL is a graph query language. You can use DSL statements to query and compute graphs, helping you design and run algorithms at low costs. This function applies only to graphs of 2.3.14 or later.

The procedure is as follows:

- 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 **DSL**. Enter a query statement and press **Enter** to run the statement.

Figure 7-52 Switching to DSL query



#### □ NOTE

Multi-label graphs do not support DSL query.

## **Common DSL Query Statements**

The following are typical query statements.

• Querying a vertex

Match<Vertex> v(['Vivian','Eric']);return v: Query vertices whose IDs are Vivian and Eric.

• Querying neighbor vertices in *N* hops

Match<Vertex> v(['Vivian']);v.repeat(bothV()).times(2).emit();return v: Query all neighbor vertices in two hops in both directions of a vertex whose ID is Vivian.

Returning a subgraph

Match<Vertex> v(['Vivian','Eric']); return v.subgraph(): Return vertices Vivian and Eric and the edge set between them.

• Other statements

Match<Vertex> v(); v.pick(1); return v: Randomly match and return one vertex.

Match<Vertex> v(); v.pattern('match (n:user) return n'); return v: // Use Cypher statements to query and return the vertex set.

#### **Ⅲ** NOTE

- 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.
- 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 7-53 DSL keywords



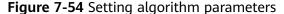
# 7.12 Analyzing Graphs Using Algorithms

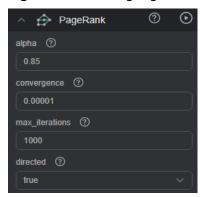
You can analyze graphs using basic graph algorithms, graph analysis algorithms, and graph metric algorithms.

The procedure is as follows:

- Log in to the GES graph editor. For details, see Accessing the GES Graph Editor.
- 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.





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

#### 

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. For details, see **Algorithm APIs**.

Take the sample movie data in the template as an example. The following figure shows the PageRank values.

Figure 7-55 Viewing the analysis result

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

Figure 7-56 Adjusting parameters

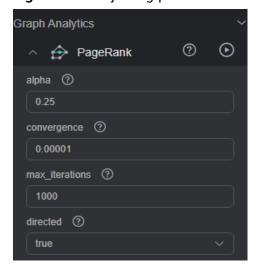


Figure 7-57 Query result after the parameters are adjusted

5. Perform association prediction to obtain the association degree of the two movies. The association degree is 0.029, indicating that only a small group of people have watched both movies.

Figure 7-58 Association analysis

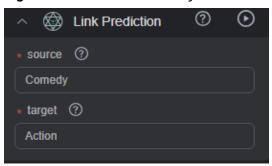


Figure 7-59 Association analysis result

```
Running Record Query Result

1 * {
2          "runtime": 0.000201,
3          "source": "Comedy",
4          "link_prediction": 0.029411764705882353,
5          "target": "Action"
6     }
```

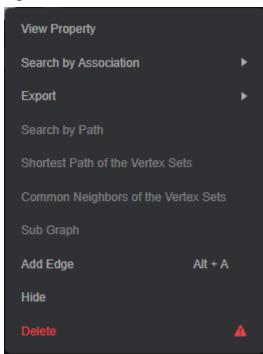
# 7.13 Analyzing Graphs on the Canvas

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 7-3**.

The procedure is as follows:

- **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:

Figure 7-60 Shortcut menu



## • View Property

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

A, and drag the mouse to add an edge. Filtering Property Statistics

Legend FIND\_MGRV ITRUST\_OR... CHANNELV FIN\_PRODV

FIN\_PRODV

FUND\_ES... 20240722 & 
RETURN1... 46.55 & 
RETURN2... 36.33 & 
RETURN2... 36.33 & 
RETURN2... 36.33 & 
RETURN3... & 
RETURN3

Figure 7-61 View Property

## 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 or data displayed on the canvas.

## • Search by Path

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

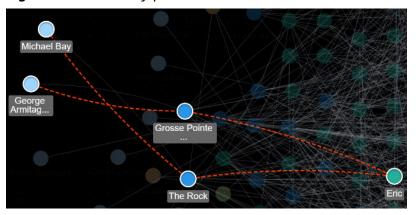
Procedure: Hold down **Ctrl** and click two vertices. The first is the source vertex and the second is the target vertex. Then, Right-click and choose **Search by Path** from the shortcut menu.

#### **Ⅲ** 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.

Figure 7-62 Search by path



#### • 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.

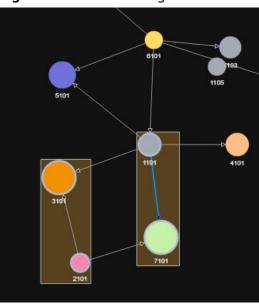


Figure 7-63 Box-selecting vertex sets

ii. Right-click a vertex set and choose **Common Neighbors of Vertex Sets** from the shortcut menu.

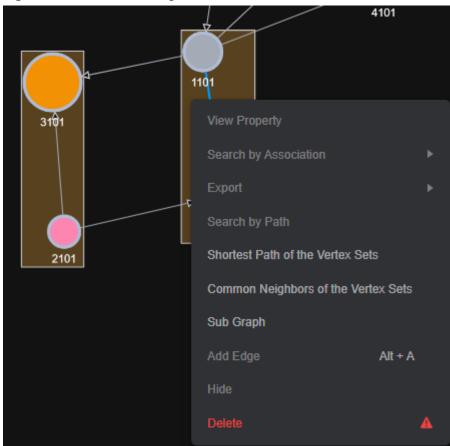


Figure 7-64 Common Neighbors of Vertex Sets

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**.

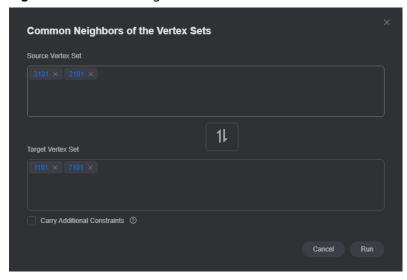


Figure 7-65 Confirming the vertices in the vertex sets

#### 

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.

Figure 7-66 Graph

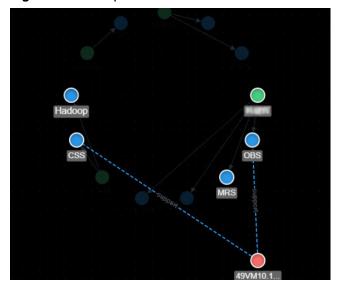
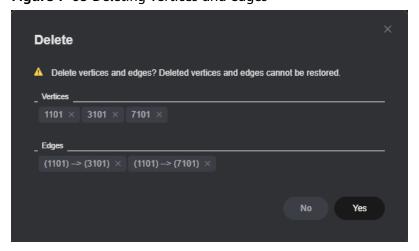


Figure 7-67 Query result

- **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**.

Figure 7-68 Deleting vertices and edges



## 

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.

Figure 7-69 Details



### **◯** 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

# 7.14 Graph Display in 3D View

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

## **Ⅲ** 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.

## Displaying a Graph in 3D View

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

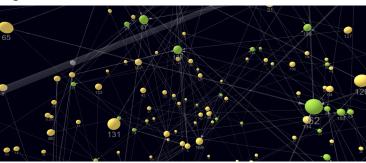


Figure 7-70 3D view of the result

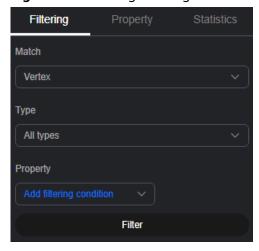
## 7.15 Filter Criteria

You can set filter criteria to filter graph data.

The procedure is as follows:

- Log in to the GES graph editor. For details, see Accessing the GES Graph Editor.
- 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.
- 3. In the **Filtering and Property** 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.

Figure 7-71 Setting filtering conditions



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

# 7.16 Editing Properties

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.

The procedure is as follows:

1. Right-click a vertex/edge on 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.

If the selected vertex has multiple labels, you can click the drop-down box next to the label to view the properties of other labels.

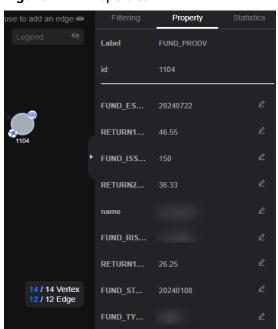
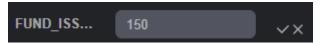


Figure 7-72 Properties

2. Click  $\angle$  next to the property to edit it.



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

3. Click after you finish editing.

## **Ⅲ** 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**.

# 7.17 Statistics Display

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

To display statistics, perform the following steps:

- Log in to the GES graph editor. For details, see Accessing the GES Graph Editor.
- 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 seven tags. There are five vertices tagged with **FUND PRODV** and three vertices tagged with **FIN PRODV**.

In the example graph, the vertex whose ID is 1101 has the largest weight. There are five edges in total. The vertex ranked No. 10 is vertex 1103. There is one edge in total.



Figure 7-73 Tag statistics

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.

# 7.18 View Running Records

The system logs your operations in a table, allowing you to review the execution progress and completion time when analyzing data.

The procedure is as follows:

- Log in to the GES graph editor. For details, see Accessing the GES Graph Editor.
- After executing a Gremlin/Cypher/DSL query or algorithm analysis, the
  operation record name, status, request parameters, job ID, start time, and
  duration will be displayed under the Running Record tab. Clicking the Query
  Result tab will take you to the query results page, where you can view the
  complete results of the operation.

Figure 7-74 Running Record tab



3. You can also export running records in **json**, **csv**, or **excel** format.

# 7.19 Viewing Query Results

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

The procedure is as follows:

- 1. Log in to the GES graph editor. For details, see Accessing the GES Graph Editor.
- 2. Perform a Gremlin/Cypher/DSL query or algorithm analysis and check the query results on the **Query Result** tab page.
  - If the returned results are too large to be fully displayed on the canvas and result area, you can click the export button in the upper right corner to download the analysis results. Currently, three export formats are supported: **json**, **csv**, and **excel**.
  - 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:

Figure 7-75 Gremlin output

 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:

Figure 7-76 Cypher output



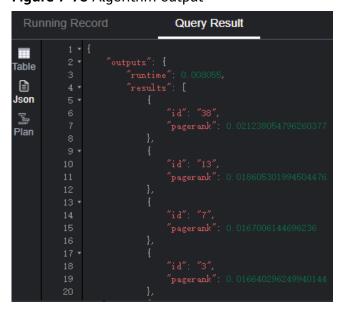
Run a DSL command to display its execution result. For example, if you enter the query command Match<Vertex> v(); v.pick(1); return v, the query result is as follows:

Figure 7-77 DSL output



 Run an algorithm. The running time and result are displayed. For example, if you run PageRank, the result is as follows:

Figure 7-78 Algorithm output

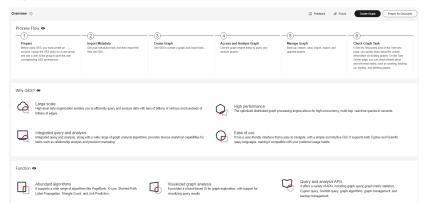


# 8 Viewing Graph Tasks

# 8.1 Graph Overview

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

Figure 8-1 Overview



## **Graph Status**

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

**Table 8-1** 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.

Status	Description
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.
Stopped	Indicates stopped graphs. Graphs in this status cannot be accessed, but can be restarted.
Frozen	Indicates frozen graphs. The user account that created these graphs are frozen.  NOTE  After a user account is frozen, only deletion operations are allowed.
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 following sizes.

## □ NOTE

Only graph names and the number of graphs are displayed.

Table 8-2 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.

Size	Description
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 8-3 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.

## **Payment Details**

This part displays the billing mode, number of instances, and graph expiration time

## 8.2 Task Center

## 8.2.1 Management Plane Task Center

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.

The procedure is as follows:

- 1. In the navigation pane on the left, click **Task Center**.
- 2. On the **Task Center** page, view the task type, task name, graph name, associated graph, start time, end time, status, and running result.

Figure 8-2 Task center



 In the Running Result column, click View Details to view the detailed information. You can also click Cause of Failure or Job ID.

Figure 8-3 Viewing details



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.

Figure 8-4 Partially successful task



4. On the **Task Center** page, search for a task in any of the following ways:

Figure 8-5 Searching for a task



- a. Selecting the task type
- b. Selecting the task name
- c. Entering an associated graph

- d. Entering a task status
- e. Entering a task ID
- f. Setting the time

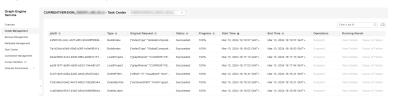
## 8.2.2 Service Plane Task Center

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

The procedure is as follows:

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

Figure 8-6 Query task center



## **◯** NOTE

- The query task center is available for graphs of version 2.2.23 and later.
- You can access the query task center of graphs that 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
  - **Graph Name**: name of graphs of the database edition
  - 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.

# 8.3 Viewing Monitoring Metrics

Cloud Eye monitors the running status of your graphs. You can view the monitoring metrics of GES on the Cloud Eye management console.

It takes a period of time for transmitting and displaying monitoring 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 monitoring metrics.
- You can view monitoring data of graphs in the running, importing, exporting, and clearing states. The monitoring metrics can be viewed after the real-time service starts or recovers.

## **Viewing Metrics**

- 1. Log in to the GES management console.
- In the navigation pane, choose Graph Management. In the Operation column, choose More > View Metrics. The Cloud Eye management console is displayed.
- 3. On the monitoring page for GES, you can view the figures of all monitoring metrics.

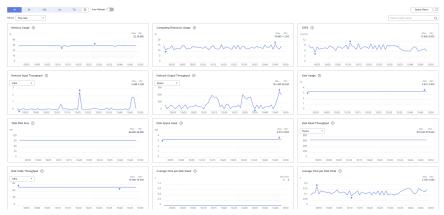


Figure 8-7 Viewing monitoring metrics

4. To view the monitored data of a longer time range, click  $\ ^{\searrow}$  .

Memory Usage ②

Max Min
32 29.660

40
30
20
10
0
09:55 10:00 10:05 10:10 10:15 10:20 10:25 10:30 10:35 10:40 10:45 10:50

Figure 8-8 Enabling the full image mode

Figure 8-9 Viewing monitoring data in a bigger view



- 5. The system allows you to select a fixed time range or use automatic refresh.
  - a. Fixed time ranges include 1h, 3h, 12h.
  - b. The automatic refresh interval is 60s, which is the user monitoring period.

## **Metrics**

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

## ■ NOTE

The namespace of the metrics reported by GES to Cloud Eye is SYS.GES.

Table 8-4 GES metrics

Metric ID	Metric	Description	Value Range	Monitore d Object	Monitor ing Period (Origin al Metric)
ges001_vert ex_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_edg e_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_ave rage_impor t_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
ges004_req uest_count	Request Quantity	Number of requests received by a graph instance Unit: count	≥ 0 Value type: Int	GES instance	1 minute
ges005_ave rage_respon se_time	Average Response Time	Average response time of requests received by a graph instance Unit: ms	≥ 0 Value type: Int	GES instance	1 minute

Metric ID	Metric	Description	Value Range	Monitore d Object	Monitor ing Period (Origin al Metric)
ges006_min _response_t ime	Minimum Response Time	Minimum response time of requests received by a graph instance Unit: ms	≥ 0 Value type: Int	GES instance	1 minute
ges007_ma x_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_rea d_task_pen ding_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
ges009_rea d_task_pen ding_max_ti me	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_pen ding_max_ti me_ read_task_t ype	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 8-6 to find the corresponding task name.	≥ 1 Value type: Int	GES instance	1 minute

Metric ID	Metric	Description	Value Range	Monitore d Object	Monitor ing Period (Origin al Metric)
ges011_rea d_task_runn ing_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_rea d_task_runn ing_max_ti me	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
ges013_run ning_max_ti me_ read_task_t ype	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_writ e_task_pen ding_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

Metric ID	Metric	Description	Value Range	Monitore d Object	Monitor ing Period (Origin al Metric)
ges015_writ e_task_pen ding_max_ti me	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_pen ding_max_ti me_ write_task_t ype	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 8-6 to find the corresponding task name.	≥ 1 Value type: Int	GES instance	1 minute
ges017_writ e_task_runn ing_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_writ e_task_runn ing_max_ti me	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

Metric ID	Metric	Description	Value Range	Monitore d Object	Monitor ing Period (Origin al Metric)
ges019 _running_m ax_time_ write_task_t ype	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_co mputer_res ource_usag e	Computin g Resource Usage	Computing resource usage of each graph instance Unit: %	0-100 Value type: Float	GES instance	1 minute
ges021_me mory_usage	Memory Usage	Memory usage of each graph instance Unit: %	0-100 Value type: Float	GES instance	1 minute
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_byt es_in	Network Input Throughp ut	Data input to each graph instance per second over the network Unit: byte/s	≥ 0 Value type: Float	GES instance	1 minute
ges024_byt es_out	Network Output Throughp ut	Data sent to the network per second from each graph instance Unit: byte/s	≥ 0 Value type: Float	GES instance	1 minute

Metric ID	Metric	Description	Value Range	Monitore d Object	Monitor ing Period (Origin al Metric)
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_throu ghput	Disk Read Throughp ut	Data volume read from the disk in a graph instance per second Unit: byte/s	≥ 0 Value type: Float	GES instance	1 minute
ges029_disk _write_thro ughput	Disk Write Throughp ut	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_p er_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_p er_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

Metric ID	Metric	Description	Value Range	Monitore d Object	Monitor ing Period (Origin al Metric)
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 8-5 Dimensions

Кеу	Value
instance_id	GES instance

# **Mapping Between Task Types and Names**

**Table 8-6** Mapping table

Туре	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

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

# **8.4 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 8-10 Connection management page



## **Downloading SDK and Driver**

**Figure 8-11** SDK and driver



- 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 8-12 Instance information



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.

# **9** Configuring Permissions

# 9.1 Configuring Granular Permissions

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 memory edition graphs of version 2.2.21 or later and database edition graphs of version 2.4.0 or later. You can upgrade a graph of an earlier version to 2.2.21 or a later version and then set granular permissions.
- Configuring fine-grained permissions for the graph requires IAM user viewing permissions and **GES Manager** or higher permissions. If there is no IAM user viewing permission, refer to **User Details** to import IAM users.

#### **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.

## 

- 1. Only graphs in the **Running** status are displayed on this page.
- 2. 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.

Figure 9-1 Setting permissions



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.

Figure 9-2 Creating permission



- 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.

#### ∩ NOTE

To use the Cypher query function, you need to configure the metadata permission and select the read and write permissions for all labels (including the default label \_\_DEFAULT\_\_) when configuring the graph permission.

Figure 9-3 Configuring permissions



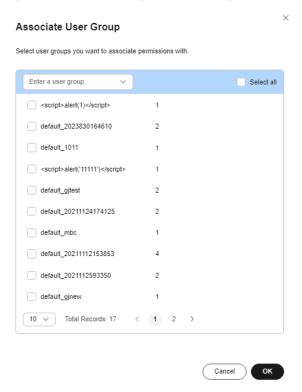
7. Click **Save**. The **Set Permission** page is displayed. You can view the created permission policy in the **Granular Permission Policy** pane.

Figure 9-4 Created policies



8. Click **Set** in the **Operation** column to associate the created granular permission with a user group.

Figure 9-5 Associating with a user group



9. Click **OK**. On the **Granular Permission Policy** pane, you can view the number of users who have been granted the permission.

**Figure 9-6** Users granted the permission



# 9.2 User Groups

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

The procedure is as follows:

- Before creating a user group, you need to understand the concept of the User Group.
- 2. On the **User Groups** page, click **Create User Group** in the upper right corner. The **Create User Group** page is displayed.

Figure 9-7 User groups

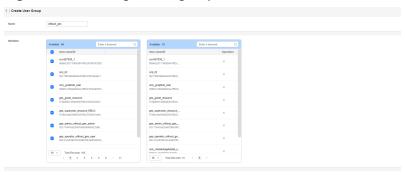


- 3. Set the user group name and add group members.
  - Name: Set a name for the user group or use the default name.
  - Members: All IAM users created under your account are displayed in this area. Select members you want to add to the user group. The selected members are displayed on the right.
    - Click next to User name/ID to select or deselect all the members on the current page.

#### 

If the IAM user is not found due to insufficient permissions, manually import the IAM user by referring to **User Details**.

Figure 9-8 Creating a user group



4. 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.

Figure 9-9 Available operations



#### 

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

# 9.3 User Details

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

The procedure is as follows:

1. On the **User Permissions** page, click vertex next to the target username to view its fine-grained permissions.

Figure 9-10 Viewing granular permissions



2. Click the permission name to view the details.

Figure 9-11 Permission details



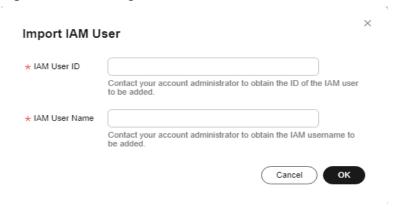
3. If you do not have such permission, you can click Import IAM User in the upper right corner to manually import IAM users.

Figure 9-12 Importing IAM users



In the **Import IAM User** dialog box, enter the ID and username of the IAM user to be added and click **OK**. The system will add the IAM user to GES so that the IAM user can be selected in the user group.

Figure 9-13 Entering IAM user information



# 10 O&M Monitoring and Alarm Reporting

# **10.1 Monitoring Metrics**

By using the O&M monitoring function of the graph instance, you can check the instance status, available resources, and real-time resource consumption.

Table 10-1 lists the monitoring metrics for GES.

Table 10-1 GES monitoring metrics

Monitor ed Object	Metric	Description	Value Range	Monitor ing Period (Origin al Metric)
Instance overview	Cluster Information	Size and CPU architecture	String	-
metrics	Cluster Capacity	Total and used vertices and edges, and usage	≥ 0	Real- time
	Cluster Node	Node type, available quantity, and total quantity	≥ 0	Real- time
	Cluster Request Statistics	Number of waiting and running read and write requests on an instance	≥ 0	Real- time
Instance alarm metrics	Alarm Statistics	Number of critical, major, minor, and info alarms on an instance	≥ 0	5 min

Monitor ed Object	Metric	Description	Value Range	Monitor ing Period (Origin al Metric)
Instance workloa d metrics	QPS	Number of requests processed by an instance per second	≥ 0	5 min
Resourc e	Average CPU Usage	Average CPU usage of the active node	0%-100%	5 min
ption metrics	Memory Usage	Average memory usage of the active node	0%-100%	5 min
of graph instance s	Disk Usage	Average disk usage of the active node	0%-100%	5 min
	Disk I/O Usage	Average disk I/O usage of the active node	0%-100%	5 min
	Network I/O Usage	Average network I/O usage of the active node	0%-100%	5 min
Overvie	Node Name	Name of a node	String	-
W	CPU Usage (%)	CPU usage of a node	0%–100%	5 min
	Memory Usage (%)	Memory usage of a node	0%–100%	5 min
	Average Disk Usage (%)	Disk usage of a node	0%-100%	5 min
	IP Address	Service IP address of a node	String	5 min
	Disk I/O (KB/S)	Disk I/O of a node, in KB/s	≥ 0KB/s	5 min
	TCP Protocol Stack Retransmissio n Rate (%)	Retransmission rate of TCP packets per unit time	0%-100%	5 min
	Status	Status of a node	Running/ Faulty	5 min
Disks	Node Name	Name of a node	String	5 min
	Disk Name	Name of a disk on a node	String	5 min

Monitor ed Object	Metric	Description	Value Range	Monitor ing Period (Origin al Metric)
	Disk Capacity (GB)	Capacity of a disk on a node, in GB	≥ 0 GB	5 min
	Disk Usage (%)	Disk usage of a node	0%-100%	5 min
	Disk Read Rate (KB/S)	Disk read rate of a node, in KB/s	≥ 0KB/S	5 min
	Disk Write Rate (KB/S)	Disk write rate of a node, in KB/s	≥ 0KB/S	5 min
	I/O Wait Time (ms)	Average waiting time for each I/O request, in ms	≥ 0 ms	5 min
	I/O Service Time (ms)	Average processing time for each I/O request, in ms	≥ 0 ms	5 min
	I/O Usage (%)	Disk I/O usage of a host	0%-100%	5 min
Network	Node Name	Name of a node	String	5 min
S	NIC Name	Name of the NIC on a node	String	5 min
	NIC Status	NIC status	up/down	5 min
	NIC Speed	Working rate of a NIC, in Mbit/s	≥ 0	5 min
	Received Packets	Number of packets received by a NIC	≥ 0	5 min
	Transmitted Packets	Number of packets transmitted by a NIC	≥ 0	5 min
	Lost Received Packets	Number of lost packets received by a NIC	≥ 0	5 min
	Receive Rate (KB/S)	Number of bytes received by a NIC per unit time, in KB/s	≥ 0KB/s	5 min
	Transmit Rate (KB/S)	Number of bytes transmitted by a NIC per unit time, in KB/s	≥ 0KB/s	5 min
Perform ance	Cluster CPU Usage	Average CPU usage of the active node	0%-100%	5 min

Monitor ed Object	Metric	Description	Value Range	Monitor ing Period (Origin al Metric)
	Cluster Memory Usage	Average memory usage of the active node	0%-100%	5 min
	Cluster Disk Usage	Average disk usage of the active node	0%-100%	5 min
	Cluster Disk I/O	Average disk I/O of the active node	0%-100%	5 min
	Cluster Network I/O	Average network I/O of the NIC of the active node	0%-100%	5 min
	Tomcat Connection Usage	HTTP connection usage of the active node	0%-100%	5 min
	Cluster Swap Disk Usage	Swap partition disk usage of the active node	0%-100%	5 min
	JVM Heap Memory Usage	JVM heap memory usage of the active node	0%-100%	5 min
	Read Requests in Running Queue	Number of running read requests on the current instance	≥ 0	5 min
	Read Requests in Blocked Queue	Number of blocked read requests on the current instance	≥ 0	5 min
Real- Time	Request ID	ID of the current query request	String	Real- time
Queries	Job Name	Name of the current query job	String	Real- time
	Request Parameters	Request parameters for the current query	String	Real- time
	Progress	Progress of the current query	0%-100%	Real- time
	Blocking Duration (S)	Blocking duration of the current query, in seconds	≥ 0	Real- time

Monitor ed Object	Metric	Description	Value Range	Monitor ing Period (Origin al Metric)
	Started	Start time of the current query	String	Real- time
	Ended	End time of the current query	String	Real- time
	Running Duration	Running duration of the current query, in seconds	≥ 0	Real- time
Historica	Job ID	ID of a historical query job	String	Real- time
Queries	Туре	Type of a historical query job	String	Real- time
	Original Request	Original request for a historical query	String	Real- time
	Status	Status of a historical query job	String	Real- time
	Progress	Execution progress of a historical query job	0%-100%	Real- time
	Start Time	Start time of a historical query job	String	Real- time
	End Time	End time of a historical query job	String	Real- time
	Running Result	Execution results of a historical query job	String	Real- time

# 10.2 Graph Instance O&M Monitoring

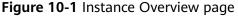
GES offers a multi-dimensional O&M monitoring interface that guarantees the smooth operations of graph instances. This feature gathers, monitors, and analyzes disk, network, and OS metrics utilized by graph instances, along with key cluster performance metrics. It promptly identifies significant database faults and performance issues and provides recommendations to optimize and resolve them.

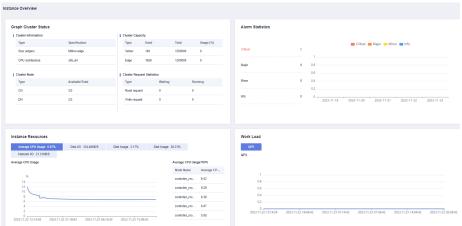
#### 

- The graph instance O&M monitoring dashboard supports only graphs of version 2.3.17 or later.
- The ten-thousand-edge size is for development learning and does not support the O&M monitoring dashboard.
- Database edition graph databases do not support graph instance monitoring.

#### **O&M Monitoring Page**

- 1. Log in to the GES management console. In the navigation pane on the left, choose **Graph Management**.
- In the graph list, locate the target graph instance, click More in the
   Operation column, and select View Metric to access the Instance Overview page. For details about monitoring metrics, see Monitoring Metrics.





#### **Instance Overview**

On the **Instance Overview** page of a graph instance, you can check the graph instance status, real-time resource consumption, alarm statistics, and service workload. The functions of these areas are as follows:

- Graph Cluster Status
  - In this area, you can check the basic information, cluster capacity, and number of requests of the current graph instance.
  - a. **Cluster Information**: includes graph size and CPU architecture.
  - b. **Cluster Capacity**: includes the number of used and total vertices and edges, as well as the usage.
  - c. **Cluster Node**: includes the number of available/total CNs/DNs.
  - d. **Cluster Request Statistics**: includes the number of waiting read requests, running read requests, waiting write requests, and running write requests.

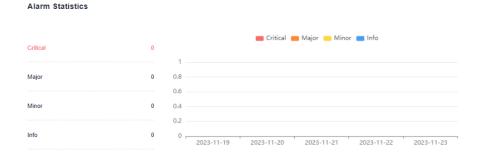
**Graph Cluster Status** Cluster Information Cluster Capacity Usage (%) Type Specification Type Used Total Size (edges) Million-edge Vertex 1200000 1200000 CPU architecture x86\_64 1659 Edge Cluster Node Cluster Request Statistics Available/Total Waiting Running Type Type CN 3/3 Read request 0 0 DN 2/2 Write request

Figure 10-2 Graph Cluster Status

#### Alarm Statistics

In this area, you can check all alarms that are not cleared for the current instance and all alarms generated for the instance in the last seven days.

Figure 10-3 Alarm Statistics



#### • Instance Resources

In this area, you can check the resource usage of the current instance, including the CPU usage, disk I/O, disk usage, memory usage, and network I/O. You can click a resource metric to view its change trend in the last 72 hours and the top 5 nodes with the highest usage of the resource at the current time.



Figure 10-4 Instance Resources

#### Workload

In this area, you can check the change trend of the database service load metric QPS in the last 72 hours.



Figure 10-5 Workload

# 10.3 Monitoring

#### 10.3.1 Nodes

In the navigation pane on the left of the O&M monitoring page, choose **Monitoring** > **Nodes**. The node monitoring page is displayed, showing the real-time consumption of nodes, memory, disks, disk I/O, and network I/O.

#### Overview

On the **Overview** page, you can check the key resources of a specified node based on the node name, including the node name, CPU usage (%), memory usage (%), average disk usage (%), IP address, disk I/O (KB/s), TCP protocol stack retransmission rate (%), network I/O (KB/s), node status, and node monitoring status.

| Part |

Figure 10-6 Overview page

You can click **Monitor** on the right of the row where a specified node is located to access the monitoring overview page and check the performance metric topology of the node in a specified period.

The period options are Last 1 hour, Last 3 hour, Last 12 hour, Last 24 hour, and Last 3 days. If you stay on the page for a long time, you can click **Refresh** in the upper right corner to update the monitoring data.

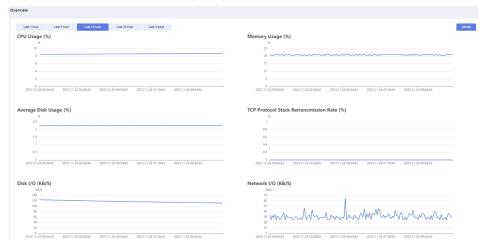


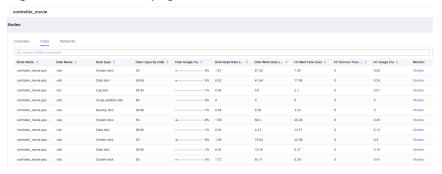
Figure 10-7 Node monitoring page

#### Disks

On the **Disks** tab page, you can check the real-time disk usage of a node based on the node name and disk name. The metrics include **Node Name**, **Disk Name**, **Disk Type**, **Disk Capacity (GB)**, **Disk Usage (%)**, **Disk Read Rate (KB/s)**, **Disk Write Rate (KB/s)**, **I/O Wait Time (ms)**, **I/O Service Time (ms)**, **I/O Usage (%)**, and **Monitor**.

The disk types include system disk, data disk, log disk, swap partition disk, and backup disk.

Figure 10-8 Disks tab page



You can click **Monitor** on the right of the row where a specified node is located to access the monitoring overview page and check the performance metric topology of the disk in a specified period.

The options are Last 1 hour, Last 3 hour, Last 12 hour, Last 24 hour, and Last 3 days. If you stay on the page for a long time, you can click Refresh in the upper right corner to update the monitoring data.

Figure 10-9 Disks page



#### **◯** NOTE

According to the disk usage displayed on the page, the sum of the used disk space and available disk space is not equal to the total disk space. This is because a small amount of space is reserved in each default partition for system administrators to use. Even if common users have run out of space, system administrators can log in to the system and use their space required for solving problems.

The disk capacity is collected by running the **df** command on Linux. The following is an example:

[Ruby@host-	10-0-16-43 8_1	0]# df ·	-x tmpfs -x	devt	npfs
Filesystem	1K-blocks	Used	Available	Use%	Mounted on
/dev/sda4	569616888	5757444	540228616	2%	/
/dev/sda2	999320	107584	822924	12%	/boot
/dev/sdal	204580	8368	196212	5%	/boot/efi
/dev/sdd	3513495364	390076	3513105288	1%	/var/chroot/DWS/datal
/dev/sde	3513495364	274192	3513221172	1%	/var/chroot/DWS/data2
/dev/sdb	3513495364	34224	3513461140	1%	/var/chroot/DWS/data3
/dev/sdc	3513495364	34224	3513461140	1%	/var/chroot/DWS/data4
[Rubvahost-	10-0-16-43 8 1	01#			

/dev/sda4: Used(5757444) + Available(540228616) != Total(569616888)

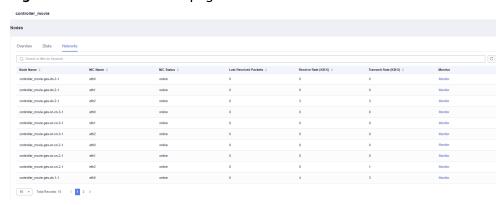
The parameters are as follows:

- **Filesystem**: path name of the device file corresponding to the file system. Generally, it is a hard disk partition.
- IK-blocks: number of data blocks (1,024 bytes) in a partition.
- Used: number of data blocks used by the disk.
- Available: number of available data blocks on the disk.
- **Use%**: percentage of the space used by common users. Even if the space is used up, the partition still reserves the space for system administrators.
- Mounted on: mount point of the file system.

#### Networks

On the **Networks** tab page, you can check the real-time network resource consumption of a node based on the node and NIC name. The metrics include **Node Name**, **NIC Name**, **NIC Status**, **Lost Received Packets**, **Receive Rate (KB/S)**, **Transmit Rate (KB/S)**, and **Monitor**.

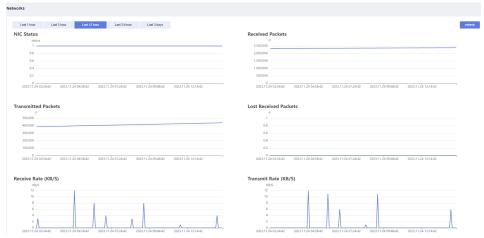
Figure 10-10 Networks tab page



You can click **Monitor** on the right of the row where a specified node is located to access the monitoring overview page and check the performance metric topology of the network in a specified period.

The options are Last 1 hour, Last 3 hour, Last 12 hour, Last 24 hour, and Last 3 days. If you stay on the page for a long time, you can click **Refresh** in the upper right corner to update the monitoring data.

Figure 10-11 Networks page



# 10.3.2 Performance

In the navigation pane on the left of the O&M monitoring page, choose **Monitoring** > **Performance**. The performance monitoring page displays the trends of the following performance metrics:

- CPU Usage (%)
- Memory Usage (%)
- Disk Usage (%)
- Disk I/O (KB/s)
- Network I/O (KB/s)
- Tomcat Connection Usage (%)
- Swap Disk Usage
- JVM Heap Memory Usage
- Read Requests in Running Queue
- Read Requests in Blocked Queue

You can select a time range to check the performance trends within this range.

The options are Last 1 hour, Last 3 hour, Last 12 hour, Last 24 hour, and Last 3 days. If you stay on the page for a long time, you can click **Refresh** in the upper right corner to update the monitoring data.

Figure 10-12 Performance page

# 10.3.3 Real-Time Queries

In the navigation pane on the left of the O&M monitoring page, choose **Monitoring** > **Real-Time Queries**. The **Real-Time Queries** page is displayed, showing the real-time information about all queries running on the instance. The information includes **Request ID**, **Job Name**, **Request Parameters**, **Progress**, **Blocking Duration (S)**, **Started**, **Ended**, and **Running Duration**.

Figure 10-13 Real-Time Queries page



# 10.3.4 Historical Queries

In the navigation tree on the left of the O&M monitoring page, choose **Monitoring** > **History Queries**. The **History Queries** page is displayed, showing details about historical asynchronous tasks running on the graph instance (the same as those displayed in the task center on the service plane).

Figure 10-14 Historical Queries page



# 10.4 Monitoring Clusters Using Cloud Eye

This section describes metrics reported by GES to Cloud Eye as well as their namespaces, lists, and dimensions. You can use APIs provided by Cloud Eye to query the metric information generated for GES.

# Namespace

SYS.GES

# **Monitoring Metrics**

Table 10-2 GES metrics

Metric ID	Metric	Description	Value Range	Monitored Object
ges001_vertex_ util	Vertex Capacity Usage	Vertex usage in a graph instance. The value is the ratio of used vertices to the total vertices. Unit: %	0–100 Type: float	GES instance
ges002_edge_ut il	Edge Capacity Usage	Edge usage of a graph instance. The value is the ratio of the used edges to the total edges. Unit: %	0–100 Type: float	GES instance
ges003_average _import_rate	Average Import Rate	Average rate of importing vertices or edges to a graph instance Unit: count/s	0- 400000 Type: float	GES instance
ges004_request _count	Request Quantity	Number of requests received by a graph instance Unit: count	≥ 0 Type: integer	GES instance
ges005_average _response_time	Average Response Time	Average response time of requests received by a graph instance Unit: ms	≥ 0 Type: integer	GES instance

Metric ID	Metric	Description	Value Range	Monitored Object
ges006_min_res ponse_time	Minimum Response Time	Minimum response time of requests received by a graph instance Unit: ms	≥ 0 Type: integer	GES instance
ges007_max_res ponse_time	Maximum Response Time	Maximum response time of requests received by a graph instance Unit: ms	≥ 0 Type: integer	GES instance
ges008_read_ta sk_pending_que ue_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 Type: integer	GES instance
ges009_read_ta sk_pending_ma x_time	Maximum Waiting Duration of Read Tasks	Maximum waiting duration of read requests received by a graph instance Unit: ms	≥ 0 Type: integer	GES instance
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. You can find the corresponding task name in GES documents.	≥ 1 Type: integer	GES instance
ges011_read_ta sk_running_que ue_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 Type: integer	GES instance
ges012_read_ta sk_running_max _time	Maximum Running Duration of Read Tasks	Maximum running duration of read requests received by a graph instance Unit: ms	≥ 0 Type: integer	GES instance

Metric ID	Metric	Description	Value Range	Monitored Object
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 Type: integer	GES instance
ges014_write_ta sk_pending_que ue_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 Type: integer	GES instance
ges015_write_ta sk_pending_ma x_time	Maximum Waiting Duration of Write Tasks	Maximum waiting duration of write requests received by a graph instance Unit: ms	≥ 0 Type: integer	GES instance
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. You can find the corresponding task name in GES documents.	≥ 1 Type: integer	GES instance
ges017_write_ta sk_running_que ue_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 Type: integer	GES instance
ges018_write_ta sk_running_max _time	Maximum Running Duration of Write Tasks	Maximum running duration of write requests received by a graph instance Unit: ms	≥ 0 Type: integer	GES instance

Metric ID	Metric	Description	Value Range	Monitored Object
ges019 _running_max_t ime_ 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 Type: integer	GES instance
ges020_comput er_resource_usa ge	Computin g Resource Usage	Compute resource usage of each graph instance Unit: %	0–100 Type: float	GES instance
ges021_memor y_usage	Memory Usage	Memory usage of each graph instance Unit: %	0–100 Type: float	GES instance
ges022_iops	IOPS	Number of I/O requests processed by each graph instance per second Unit: count/s	≥ 0 Type: integer	GES instance
ges023_bytes_in	Network Input Throughp ut	Data input to each graph instance per second over the network Unit: byte/s	≥ 0 Type: float	GES instance
ges024_bytes_o ut	Network Output Throughp ut	Data sent to the network per second from each graph instance Unit: byte/s	≥ 0 Type: float	GES instance
ges025_disk_us age	Disk Usage	Disk usage of each graph instance Unit: %	0–100 Type: float	GES instance
ges026_disk_tot al_size	Total Disk Size	Total data disk space of each graph instance Unit: GB	≥ 0 Type: float	GES instance
ges027_disk_us ed_size	Disk Space Used	Used data disk space of each graph instance Unit: GB	≥ 0 Type: float	GES instance
ges028_disk_rea d_throughput	Disk Read Throughp ut	Data volume read from the disk in a graph instance per second Unit: byte/s	≥ 0 Type: float	GES instance

Metric ID	Metric	Description	Value Range	Monitored Object
ges029_disk_wri te_throughput	Disk Write Throughp ut	Data volume written to the disk in a graph instance per second Unit: byte/s	≥ 0 Type: float	GES instance
ges030_avg_dis k_sec_per_read	Average Time per Disk Read	Average time per disk read for a graph instance Unit: second	≥ 0 Type: float	GES instance
ges031_avg_dis k_sec_per_write	Average Time per Disk Write	Average time per disk write for a graph instance Unit: second	≥ 0 Type: float	GES instance
ges032_avg_dis k_queue_length	Average Disk Queue Length	Average I/O queue length of the disk in a graph instance Unit: count	≥ 0 Type: integer	GES instance

# **Dimensions**

Кеу	Value
instance_id	GES instance

# **Mapping Between Task Types and Names**

**Table 10-3** Mapping between task types and names

Туре	Name		
100	Querying vertices		
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 edges		
201	Creating an edge		

Туре	Name		
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 a graph		
405	Deleting a graph		
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		
602	Querying indexes		
603	Updating an index		
604	Deleting an index		
700	Running an algorithm		

# **Viewing Instance Monitoring Information**

- 1. Log in to the GES management console and choose **Graph Management**.
- In the graph list, locate the row that contains the target graph, choose More, and select View Metric to access the Cloud Eye management console. By default, the graph instance monitoring information is displayed.

You can select a monitoring metric name and time range to check the performance curve.

#### Creating an Alarm Rule

By setting alarm rules for GES, you can customize monitoring objects and notification policies to promptly understand the operational status of GES and serve as an early warning.

Alarm rule settings for GES include parameters such as alarm rule name, monitoring object, monitoring metrics, alarm threshold, monitoring cycle, and notification sending.

This part describes how to set an alarm rule for GES.

- 1. Log in to the GES management console and choose **Graph Management** from the navigation pane on the left.
- Locate the row containing the target instance, choose More in the Operation column, and select View Metric to access the Cloud Eye management console and check the GES monitoring information.

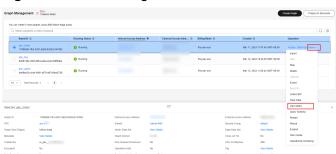


Figure 10-15 Selecting View Metrics

#### 

Ensure that the status of the instance whose monitoring information you want to view is **Running**. Otherwise, you cannot create an alarm.

3. In the navigation pane on the left of the Cloud Eye management console, choose **Alarm Management** > **Alarm Rules**. On the page displayed, click **Create Alarm Rule** in the upper right corner or in the middle.



- 4. On the **Create Alarm Rule** page, set parameters as prompted.
  - a. Setting alarm parameters

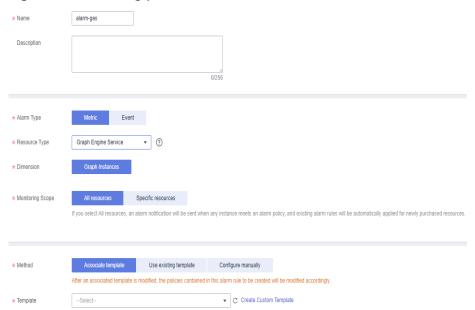


Figure 10-16 Setting parameters

Table 10-4 Alarm parameters

Paramet er	Description	Example Value	
Alarm Type	Alarm type the alarm rule applies to. The value can be <b>Metric</b> or <b>Event</b> .	Metric	
Resource Type	Name of the cloud service the alarm rule is created for	Graph Engine Service	
Dimensio n	Metric dimension of the selected resource type. Select <b>Graph Instance</b> .	Graph Instance	
Monitori ng Scope	Resource scope the alarm rule applies to. Select <b>Specified resources</b> and select one or more monitored objects. Click <b>Select Specific Resources</b> and select the cluster instance you have created.	Specific resources	

Paramet er	Description	Example Value	
Method	You can create an alarm rule by using the template or manually creating it.  If no alarm template is available, set Method to Create manually and configure related parameters to create an alarm rule.  If you have available alarm rule templates, set Method to Use template, so that you can use a template to quickly create alarm rules.	Create manually	
Method	There are three options: Associate template, Use existing template, and Configure manually.	Associate template	
Template	This parameter is available only when <b>Use template</b> is selected. Select the template to be used. If no alarm template is available, click <b>Create Custom Template</b> to create one that meets your requirements.	-	
Alarm Policy	This parameter is available only when <b>Configure manually</b> is selected for <b>Method</b> .  Set the policy that triggers an alarm. For example, trigger an alarm if the CPU usage equals to or is greater than 80% for 3 consecutive periods.  For details about GES monitoring metrics, see <b>Monitoring Metrics</b> .	-	
Alarm Severity	Alarm severity, which can be Critical, Major, Minor, or Informational.	Major	

b. Configure the alarm notification parameters as prompted.

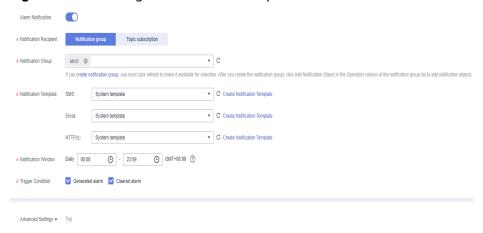


Figure 10-17 Setting alarm notification parameters

**Table 10-5** Alarm notification parameters

Parame ter	Description	Example Value
Alarm Notifica tion	Whether to send email, SMS, HTTP, or HTTPS notifications to users when an alarm is triggered	Enable this function
	You can enable (recommended) or disable this function.	
Notifica tion Recipie nt	You can select <b>Notification group</b> or <b>Topic subscription</b> .	Topic subscription
Notifica tion Object	This parameter is mandatory when <b>Notification Recipient</b> is set to <b>Topic subscription</b> .	SMN topic
	Name of the topic the alarm notification is to be sent to. If you have enabled <b>Alarm Notification</b> , select a topic. If no desirable topics are available, create one first, whereupon the SMN service is invoked.	
	For details about how to create a topic, see <b>Creating a Topic</b> .	
Notifica tion Group	This parameter is mandatory when Notification Recipient is set to Notification group. You can select or create a notification group. After creating a notification group, you need to click Add Notification Recipient in the Operation column of the notification group list to add group members and notification methods.	Notification group name

Parame ter	Description	Example Value
Notifica tion Templat e	You can select a system template or create a custom notification template.	System template
Notifica tion Window	Notifications are sent only within the notification window specified in the alarm rule.	-
	For example, if <b>Notification Window</b> is set to <b>00:00–08:00</b> , Cloud Eye sends notifications only within this period.	
Trigger Conditi on	Condition for triggering the alarm notification. You can select <b>Generated alarm</b> (when an alarm is generated), <b>Cleared alarm</b> (when an alarm is cleared), or both.	-

5. Click **Create**. After the alarm rule is created, if the metric data reaches the specified threshold, Cloud Eye will immediately inform you that an exception has occurred.

# **Transferring Data to OBS**

On Cloud Eye, raw metric data is only stored for two days. However, if you subscribe to OBS, you can synchronize the raw data and extend the storage period.

# 11 Algorithms

# 11.1 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 11-1 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.
Filtered Shortest Path	This algorithm searches for the shortest path that meets the filter criteria between vertices. If there are multiple shortest paths, any one of them is returned.
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 <i>n</i> 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 <b>Closeness Centrality</b> 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.

Algorithm	Description		
Link Prediction	This algorithm is used to calculate the similarity between two nodes and predict their relationship based on the Jaccard measurement method.		
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 <b>P</b> and forward parameter <b>Q</b> . 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 Recommenda tion	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 purchasing or 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 products 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.		

Algorithm	Description	
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.	
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.	
Filtered Circle Detection	This algorithm searches for all circles that meet a specified filter criteria in the graph. It is applicable to scenarios such as detecting round-trip transfers and anti-money laundering for financial risk control, locating abnormal links in network routes, and risk identification in enterprise finance guarantee.	
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.	

Algorithm	Description
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.
Filtered n- Paths	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.
Temporal Paths	Different from path analysis on static graphs, the Temporal Paths algorithm combines the order of information transmission on dynamic graphs. The passing time of an edge on a path must be later than or the same as that of the previous edge, showing the increment (or non-decrement) of time.

# 11.2 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 11-2** PageRank algorithm parameters

Parameter	Mandat ory	Description	Туре	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

Parameter	Mandat ory	Description	Туре	Value Range	Default Value
convergen ce	No	Convergence	Double	A real number between 0 and 1 (excluding 0 and 1)	0.00001
max_iterati ons	No	Maximum iterations	Int	1–2000	1000
directed	No	Whether an edge is directed	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

Select the algorithm in the algorithm area of the graph engine editor. For details, see **Analyzing Graphs Using Algorithms**.

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.

# 11.3 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 product, friend, and web page recommendations.

#### **Parameter Description**

**Table 11-3** PersonalRank algorithm parameters

Paramet er	Mandato ry	Descriptio n	Туре	Value Range	Default Value
source	Yes	Node ID	String	-	-
alpha	No	Weight coefficient	Doubl e	A real number between 0 and 1 (excluding 0 and 1)	0.85
converge nce	No	Convergen ce	Doubl e	A real number between 0 and 1 (excluding 0 and 1)	0.00001
max_iter ations	No	Maximum iterations	Int	1–2000	1000
directed	No	Whether an edge is directed	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**

Select the algorithm in the algorithm area of the graph engine editor. For details, see **Analyzing Graphs Using Algorithms**.

Set **source** to **Lee**, **alpha** to **0.85**, **convergence** to **0.00001**, **max\_iterations** to **1000**, 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.

# 11.4 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 11-4** K-core algorithm parameters

Parame	Mandat	Description	Typ	Value	Default
ter	ory		e	Range	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  $\mathbf{k}$  to  $\mathbf{10}$ . 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.

# 11.5 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 11-5** K-hop algorithm parameters

Parame ter	Mandat ory	Description	Туре	Value Range	Default Value
k	Yes	Number of hops	Integer	1-100	-
source	Yes	Node ID	String	-	-
mode	No	<ul> <li>Direction:</li> <li>OUT: Hop from the outgoing edges.</li> <li>IN: Hop from the incoming edges.</li> <li>All: Hop from edges in both directions.</li> </ul>	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

Select the algorithm in the algorithm area of the graph engine editor. For details, see **Analyzing Graphs Using Algorithms**.

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.

# 11.6 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 11-6** Shortest Paths algorithm parameters

Paramet er	Mandat ory	Description	Туре	Value Range	Defau lt Value
source	Yes	Enter the source ID of a path.	String	-	-
target	Yes	Enter the target ID of a path.	String	-	-
directed	No	Whether an edge is directed	Bool	true or false	false
weight	No	Weight of an edge	String	<ul> <li>Empty or null character string</li> <li>Empty: The default weight and distance are 1.</li> <li>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.</li> <li>NOTE         <ul> <li>The weight of an edge must be greater than 0.</li> </ul> </li> </ul>	-

Paramet er	Mandat ory	Description	Туре	Value Range	Defau lt Value
timeWin dow	No	Time window used for time filtering	Json	For details, see Table 11-7.  NOTE timeWindow does not support the shortest path with weight. That is, parameters timeWindow and weight cannot be both specified.	-

**Table 11-7** timeWindow parameters

Parame ter	Man dator y	Description	Typ e	Value Range	Def ault Valu e
filterNa me	Yes	Name of the time attribute used for time filtering	Stri ng	Character string: The attribute on the corresponding vertex/ edge is used as the time.	-
filterTy pe	No	Filtering by vertex or edge	Stri ng	V: Filtering by vertex E: Filtering by edge BOTH: Filtering by vertex and edge	BOT H
startTi me	No	Start time	Stri ng	Date character string or timestamp	-
endTim e	No	End time	Stri ng	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.

## 11.7 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 11-8** All Shortest Paths algorithm parameters

Paramet er	Mandato ry	Description	Туре	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 an edge is directed	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.

# 11.8 Filtered Shortest Path

#### Overview

The Filtered Shortest Path algorithm is used to search for the shortest path that meets the filtering criteria between two vertices. If there are multiple shortest paths, any one of them is returned.

## **Application Scenarios**

This algorithm applies to path design and network planning. It generates the shortest path based on vertex and edge filtering criteria.

## **Parameter Description**

**Table 11-9** Filtered Shortest Path algorithm parameters

Paramet er	Mandat ory	Туре	Description
source	Yes	String	Enter the source vertex ID of a path.
target	Yes	String	Enter the target vertex ID of a path.
directed	No	Boole an	Whether to consider the edge direction The default value is <b>false</b> .

#### **Precautions**

This algorithm only returns one shortest path.

## 11.9 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.

**Table 11-10** SSSP algorithm parameters

Paramet er	Mandatory	Description	Туре	Value Range	Default Value
source	Yes	Node ID	Strin g	-	-
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**.

# 11.10 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.

**Table 11-11** Shortest Path of Vertex Sets algorithm parameters

Parame ter	Mandato ry	Descripti on	Туре	Value Range	Defa ult Value
sources	Yes	Source vertex ID set	String	The value is in the standard CSV format. IDs are separated by commas (,), for example, <b>Alice, Nana</b> . The maximum ID number is 100,000.	1
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.	-
directed	No	Whether an edge is directed	Bool	true or false	false
timeWin dow	No	Time window used for time filtering	Json	For details, see <b>Table 11-12</b> .	-

**Table 11-12** timeWindow parameters

Parame ter	Man dator y	Description	Typ e	Value Range	Def ault Valu e
filterNa me	No	Name of the time attribute used for time filtering	Stri ng	Character string: The attribute on the corresponding vertex/ edge is used as the time.	-
filterTy pe	No	Filtering by vertex or edge	Stri ng	V: Filtering by vertex E: Filtering by edge BOTH: Filtering by vertex and edge	BOT H
startTi me	No	Start time	Stri ng	Date character string or timestamp	-
endTim e	No	End time	Stri ng	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.

# 11.11 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 11-13** n-Paths algorithm parameters

Paramet er	Mandator y	Description	Туре	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 an edge is directed	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.

# 11.12 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.

**Table 11-14** Closeness Centrality algorithm parameters

Paramet er	Mandato ry	Description	Туре	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.

# 11.13 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.

**Table 11-15** Label Propagation algorithm parameters

Paramete r	Mandato ry	Descripti on	Туре	Value Range	Default Value
convergen ce	No	Converge nce	Double	A real number between 0 and 1 (excluding 0 and 1)	0.00001
max_itera tions	No	Maximum iterations	Int	1-2,000	1,000

Paramete r	Mandato ry	Descripti on	Туре	Value Range	Default Value
initial	No	Name of the property used as the initializati on label on a vertex	String	<ul> <li>Null: Each vertex is allocated with a unique initialization label. This method is applicable to scenarios where no vertex label information exists.</li> <li>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.</li> </ul>	

Paramete r	Mandato ry	Descripti on	Туре	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.

# 11.14 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.

**Table 11-16** Louvain algorithm parameters

Parameter	Mandat ory	Description	Туре	Value Range	Default Value
convergen ce	No	Convergence	Doubl e	A real number between 0 and 1 (excluding 0 and 1)	0.00001

Parameter	Mandat ory	Description	Туре	Value Range	Default Value
max_iterat ions	No	Maximum iterations	Int	1-2,000	100
weight	No	Weight of an edge	String	<ul> <li>Empty or null character string</li> <li>Empty: The default weight and distance are</li> <li>1.</li> <li>Character string: The attribute of the correspondin g edge is the weight. When the edge does not have correspondin g attribute, the weight is</li> <li>1 by default.</li> <li>NOTE         <ul> <li>The weight of an edge must be greater than 0.</li> </ul> </li> </ul>	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.

# 11.15 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 11-17** Link Prediction algorithm parameters

Paramet er	Mandator y	Description	Туре	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.

## 11.16 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 11-18** Node2vec algorithm parameters

Parame ter	Mandato ry	Description	Туре	Value Range	Defa ult Valu e
Р	No	Rollback parameter	Doubl e	-	1
Q	No	Forward parameter	Doubl e	-	1
dim	No	Mapping dimension	Int	1 to 200, including 1 and 200	50
walkLen gth	No	Random walk length	Int	1 to 100, including 1 and 100	40
walkNu mber	No	Number of random walk steps of each node.	Int	1 to 100, including 1 and 100	10
iteration s	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.

# 11.17 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 purchasing or browsing data or recommend potential friends with similar preferences.

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

**Table 11-19** Real-time Recommendation algorithm parameters

Parame ter	Mandat ory	Description	Туре	Value Range	Defa ult Value
sources	Yes	Node ID. Multiple node IDs separated by commas (,) are supported (standard CSV input format).	Strin g	The number of source nodes cannot exceed 30.	-
alpha	No	Weight coefficient. A larger value indicates a longer step.	Dou ble	A real number between 0 and 1 (excluding 0 and 1)	0.85
N	No	Total number of walk steps	Int	1–200,000	10,00 0
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	1–2000	1000

Parame ter	Mandat ory	Description	Туре	Value Range	Defa ult 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.	Strin g	Node label	1
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.

# 11.18 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 products 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 11-20** Common Neighbors algorithm parameters

Parame ter	Mandat ory	Description	Туре	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.

# 11.19 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.

# 11.20 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.

# 11.21 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**

Paramet er	Manda tory	Description	Туре	Value Range
statistics	No	Whether to export only the total statistical result.	Boolea n	<b>true</b> or <b>false</b> . The default value is
		• <b>true</b> : Export only the statistical result.		true.
		• false: Export the number of triangles corresponding to each vertex.		

#### **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.

# 11.22 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.

# 11.23 Betweenness Centrality

#### Overview

Betweenness centrality is a measure of centrality in a graph based on shortest paths. This algorithm calculates shortest paths that pass through a vertex.

## **Application Scenarios**

The Betweenness Centrality algorithm can be used for tracing man-in-the-middle in social networks and risk control networks and identifying key vertices in transportation networks. This algorithm is widely used for social networking, financial risk control, transportation networking, and city planning.

**Table 11-21** Algorithm parameters

Parame ter	Manda tory	Descriptio n	Туре	Value Range	Default Value
directed	No	Whether an edge is directed	Boolean	The value can be <b>true</b> or <b>false</b> .	true
weight	No	Weight of an edge	String	The value can be an empty string. If this parameter is left blank, the weight and distance of this edge are 1 by default. You can set this parameter to a property of the edge, and the property value will be the weight. If the edge does not have the specified property, the weight is 1 by default.  NOTE  The weight of an edge must be greater than 0.	-
seeds	No	Vertex ID	String	If the graph is large, betweenness calculation can be slow. You can set <b>seeds</b> to the sampling nodes for approximate calculation. The more <b>seeds</b> nodes, the closer results to the accurate calculation. The number of vertices cannot be greater than 100,000.	
k	No	Number of samples	Integer	If the graph is large, betweenness calculation can be slow. You can set <b>k</b> to randomly select k sampling vertices from the graph. The larger value, the closer results to the accurate calculation. The value cannot be greater than 100,000.	-

#### □ NOTE

When you perform approximate betweenness calculation, either **seeds** or  $\mathbf{k}$  must be specified. If both are specified, **seeds** vertices will be sampled by default and  $\mathbf{k}$  will be ignored.

#### **Precautions**

None

#### Example

Set weight="length", directed=true, seeds ="Lee,Alice" and view the result.

# 11.24 Edge Betweenness Centrality

#### Overview

The Edge Betweenness Centrality algorithm calculates shortest paths that pass through an edge.

### **Application Scenarios**

The Edge Betweenness Centrality algorithm can be used for key relationship mining. It is applicable to social networking, financial risk control, transportation networking, and city planning.

**Table 11-22** Algorithm parameters

Parame ter	Manda tory	Descriptio n	Туре	Value Range	Default Value
directed	No	Whether an edge is directed	Boolean	The value can be <b>true</b> or <b>false</b> .	true

Parame ter	Manda tory	Descriptio n	Туре	Value Range	Default Value
weight	No	Weight of an edge	String	The value can be an empty string. If this parameter is left blank, the weight and distance of this edge are 1 by default. You can set this parameter to a property of the edge, and the property value will be the weight. If the edge does not have the specified property, the weight is 1 by default.  NOTE  The weight of an edge must be greater than 0.	
seeds	No	Vertex ID	String	If the graph is large, betweenness calculation can be slow. You can set <b>seeds</b> to the sampling nodes for approximate calculation. The more <b>seeds</b> nodes, the closer results to the accurate calculation. The number of vertices cannot be greater than 100,000.	1
k	No	Number of samples	Integer	If the graph is large, betweenness calculation can be slow. You can set <b>k</b> to randomly select k sampling vertices from the graph. The larger value, the closer results to the accurate calculation. The value cannot be greater than 100,000.	-

## □ NOTE

When you perform approximate edge-betweenness calculation, either **seeds** or  $\mathbf{k}$  must be specified. If both are specified, **seeds** vertices will be sampled by default and  $\mathbf{k}$  will be ignored.

#### **Precautions**

None

### **Example**

Set weight="length", directed=true, seeds ="Lee,Alice" and view the result.

# 11.25 Origin-Destination Betweenness Centrality

#### Overview

The Origin-Destination Betweenness Centrality algorithm calculates shortest paths that pass through a vertex/edge, with the origin and destination (OD) specified.

### **Application Scenarios**

OD Betweenness Centrality can be used for tracing man-in-the-middle in social networks and risk control networks and identifying key vertices in transportation networks. It is suitable for simulating busy transportation sections during peak hours. It is also widely used for social networking, financial risk control, transportation networking, and city planning.

Table 11-23 Algorithm parameters

Parame ter	Manda tory	Descriptio n	Туре	Value Range	Default Value
directed	No	Whether an edge is directed	Boole an	The value can be <b>true</b> or <b>false</b> .	true
weight	No	Weight of an edge	String	The value can be an empty string. If this parameter is left blank, the weight and distance of this edge are 1 by default. You can set this parameter to a property of the edge, and the property value will be the weight. If the edge does not have the specified property, the weight is 1 by default.  NOTE  The weight of an edge must be greater than 0.	

Parame ter	Manda tory	Descriptio n	Туре	Value Range	Default Value
OD_pai rs	No	Pairs of OD vertices	String	The value must be in the standard CSV format. The start vertex (origin) and end vertex (destination) are separated by commas (,), and the start and end vertex pairs are separated by newline characters (\n), for example, Alice,Nana\nLily,Amy.	-
seeds	No	ID of the hot spot vertex	String	Data that will be imported when the data of OD vertex pairs is unknown. The value is in the standard CSV format. IDs are separated by commas (,), for example, Alice, Nana. A maximum of 30 IDs are allowed.	
modes	No	Hot spot vertex type	String	<ul> <li>IN: The hot spot vertex ID is used as the start vertex ID.</li> <li>OUT: The hot spot vertex ID is used as the end vertex ID.</li> </ul>	-
attende es	No	Number of participant s at each hot spot in seeds	String	The value is in the standard CSV format. Numbers are separated by commas (,), for example, 10,20. The value ranges from 1 to 1,000,000.	-

#### □ NOTE

When you perform approximate OD-Betweenness calculation, either **OD\_pairs** or **seeds** must be specified. If both are specified, the **OD\_pairs** vertices will be used for calculation by default and **seeds** will be ignored.

#### **Precautions**

None

## **Example**

Ser weight=length, directed=true, OD = Alice, Nana\nLily, Amy and view the result.

# 11.26 Circle Detection with a Single Vertex

#### Overview

This algorithm solves a classic graph problem: detecting loops in a graph. The vertices on a loop (circle) are import.

### **Application Scenarios**

This algorithm is widely used for transportation networking and financial risk control.

### **Parameter Description**

Table 11-24 Algorithm parameters

Parameter	Man dato ry	Description	Туре	Value Range	Default Value
source	Yes	ID of the given vertex	String	-	-
min_circle_le ngth	No	Minimum circle length	Int	[3,15]	3
max_circle_le ngth	No	Maximum circle length. The value must be bigger than min_circle_len gth.	Int	[3,15]	10
limit_circle_n umber	No	Maximum number of circles you want to search for	Int	[1,100000]	100

# 11.27 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 product 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 11-25** Common Neighbors of Vertex Sets algorithm parameters

Parameter	Mand atory	Descripti on	Туре	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, Mike,Amy.	-
				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.

# 11.28 All Shortest Paths of Vertex Sets

#### Overview

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

## **Application Scenarios**

This algorithm can be used to analyze relationships between blocks in scenarios such as Internet social networking, financial risk control, road network traffic, and logistics delivery.

## **Parameter Description**

**Table 11-26** All Shortest Paths of Vertex Sets algorithm parameters

Param eter	Man dato ry	Descripti on	Туре	Value Range	Default Value
sources	Yes	Source vertex ID set	Strin g	The value is in the standard CSV format. IDs are separated by commas (,), for example, <b>Alice, Nana</b> . The maximum ID number is 100,000.	-
targets	Yes	Target vertex ID set	Strin g	The value is in the standard CSV format. IDs are separated by commas (,), for example, Alice, Nana. The maximum ID number is 100,000.	-
directe d	No	Whether to consider the edge direction	Boole an	<b>true</b> or <b>false</b> . It is a Boolean value.	false

#### **Precautions**

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.

## 11.29 Filtered Circle Detection

#### Overview

The Filtered Circle Detection algorithm finds all circles that meet the filter criteria.

## **Application Scenarios**

The Filtered Circle Detection algorithm is applicable to scenarios such as cyclic transfer detection and anti-money laundering in financial risk control, abnormal connection detection in network routing, and loan risk identification in enterprise guarantee circles.

Table 11-27 Parameter description

Paramet er	Ma nda tor y	Description	Туре	Value Range	Default Value
sources	No	Set of source vertex IDs to be queried	Strin g	-	The value is in the standard CSV format. IDs are separated by commas (,), for example, Alice, Nana.
n	No	Upper limit of the number of enumerated circles that meet the filter criteria	Int	[1,100000]	100
statistics	No	Whether to export the number of circles that meet the filter criteria	Bool ean	true or false	false
batch_nu mber	No	Number of source vertices for batch processing	Int	[1,1000]	10

Paramet er	Ma nda tor y	Description	Туре	Value Range	Default Value
output_f ormat	No	Output format	Strin g	vertexId, edgeId, or edgeObject	edgeObject
filters	Yes	Filter criteria. Each element in the array corresponds to the filter criteria of each layer.	Json	-	-

# 11.30 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.

**Table 11-28** Subgraph matching parameters

Name	Manda tory	Description	Туре	Value Range
edges	Yes	Edge set of the subgraph to be matched. The vertex ID must be a nonnegative 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.

Name	Manda tory	Description	Туре	Value Range
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 <b>true</b> or <b>false</b> . The default value is <b>true</b> .
n	No	Maximum number of subgraphs to be searched for	Int	The value range is [1,100000]. The default value is <b>100</b> .
batch_num ber	No	Number of queries processed in batches each time	Int	The value range is [1,1000000]. The default value is <b>10000</b> .
statistics	No	Whether to display the number of all subgraphs that meet the conditions	Bool	The value can be <b>true</b> or <b>false</b> . The default value is <b>false</b> .

# 11.31 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.

**Table 11-29** Parameters

Name	Mand atory	Description	Туре	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 (,).	Strin g	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 (,).	Strin g	The number of target vertices cannot exceed 10,000.	-
directed	No	Whether the edges are directed	Bool	The value can be <b>true</b> or <b>false</b> .	-
cutoff	No	Maximum length	Int	1-100	6
path_lim it	No	Maximum number of paths	Int	<ul> <li>For synchronous tasks:</li> <li>The value ranges from 1 to 100000. The default value is 100000.</li> <li>For asynchronous tasks:</li> <li>The value ranges from 1 to 1000000. The default value is 1000000.</li> <li>10000000.</li> </ul>	100000/10

## 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.

## 11.32 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 11-30** Filtered All Shortest Paths algorithm parameters

Paramete r	Mand atory	Descrip tion	Туре	Value Range	Default Value
source	Yes	Source vertex ID	String	-	-
target	Yes	Target vertex ID	String	-	-
directed	No	Whethe r an edge is directed	Bool	The value can be <b>true</b> or <b>false</b> .	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.

# 11.33 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.

**Table 11-31** TopicRank parameters

Name	Ma nda tor y	Description	Туре	Value Range	Default
sources	Yes	Vertex ID. You can specify multiple IDs in CSV format and separate them with commas (,).	Strin g	Currently, a maximum of 100000 IDs are allowed.	-
actived_p	No	Initial weight of the source vertices	Dou ble	The value ranges from 0 to 100000.	1
default_p	No	Initial weight of a non-source vertices	Dou ble	The value ranges from 0 to 100000.	1
filtered	No	Whether to filter results	Bool ean	The value can be true or false.	false
only_neig hbors	No	Whether to display only the neighboring vertices of the sources	Bool ean	The value can be true or false.	false
alpha	No	Weight coefficient	Real num ber	A real number between 0 and 1	0.85
converge nce	No	Convergence	Real num ber	A real number between 0 and 1	0.00001

Name	Ma nda tor y	Description	Туре	Value Range	Default
max_iter ations	No	Maximum iterations	Posit ive integ er	The value ranges from 1 to 2000.	1000
directed	No	Whether the edges are directed	Bool ean	The value can be true or false.	true
num_thr ead	No	Number of threads	Posit ive integ er	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.

# 11.34 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

**Table 11-32** filtered\_n\_paths parameters

Parameter	Mandato ry	Descripti on	Туре	Value Range	Default Value
source	Yes	Source vertex	String	Internal vertices	None

Parameter	Mandato ry	Descripti on	Туре	Value Range	Default Value
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

# 11.35 Temporal Paths

#### Overview

Different from path analysis on static graphs, the Temporal Paths algorithm combines the order of information transmission on dynamic graphs. The passing time of an edge on a path must be later than or the same as that of the previous edge, showing the increment (or non-decrement) of time.

- Temporal paths do not meet transitivity: If there is one temporal path from the vertex i to the vertex j, and there is one temporal path from the vertex j to the vertex k, it does not indicate that there is one temporal path from the vertex i to the vertex k. So, in terms of solving a problem, solving a path on a dynamic graph is more complex than on a static graph, and the calculation is much more difficult. However, temporal path analysis is widely used in actual life, for example, calculating a travel route and simulating/searching for an information propagation path.
- Temporal Paths can be classified into Shortest, Foremost, and Fastest Temporal Paths based on the problem-solving objective.
  - Shortest Temporal Paths: indicates the temporal path with the shortest distance.
  - Foremost Temporal Paths: indicates the temporal path that reaches the target node as early as possible.
  - Fastest Temporal Paths: indicates the temporal path that takes the shortest time.

# **Application Scenarios**

It is applicable to scenarios such as epidemic or disease transmission source tracing, information transmission and public opinion analysis, time sequence-based path planning, and fund circulation path.

**Table 11-33** Temporal Paths parameters

Parameter	Mand atory	Descriptio n	Туре	Value Range	Default Value
source	Yes	Source vertex ID	String	-	-
targets	Yes	Target vertex ID set	String	The value is in CSV format. IDs are separated by commas (,), for example, Alice,Nana. The number of IDs cannot exceed 100,000.	1000
directed	No	Whether an edge is directed	Boolea n	The value can be <b>true</b> or <b>false</b> .	false
k	No	Maximum depth	Integer	1 to 100, including 1 and 100	3
strategy	No	Algorithm policy	String	The value can be shortest, foremost, or fastest.  (Note: fastest is not supported currently.)  • shortest: Runs the shortest temporal paths algorithm to return the temporal path with the shortest distance.  • foremost: Runs the foremost temporal paths algorithm to return the temporal path that reaches the target node as early as possible.  • fastest: Runs the fastest temporal paths algorithm to return the temporal paths algorithm to return the temporal path that takes the shortest time.	shortes

Table 11-34 dynamicRange description

Paramete r	Mandat ory	Descriptio n	Туре	Value Range	Defau lt Value
start	Yes	Start time for dynamic analysis	Date/Integer	-	-
end	Yes	End time for dynamic analysis	Date/Integer	-	-
time_prop s	Yes	Time properties for dynamic analysis	Object	-	-

Table 11-35 time\_props description

Paramete r	Mand atory	Description	Туре	Value Range	Defau lt Value
stime	Yes	Name of the start time property	String	-	-
etime	Yes	Name of the end time property	String	-	-

#### **Precautions**

Temporal path analysis needs to be performed on dynamic graphs.

## **Example**

Select the algorithm in the algorithm area of the graph engine editor. For details, see **Analyzing Graphs Using Algorithms**.

- 1. To set the dynamic time range parameters, run the following command: start=1646092800, end =1646170716, stime="startTime", etime="endTime"
- 2. Set the parameters of the temporal paths algorithm. source="Person00014"

targets="Person00055,Person00058,Person00052,Person00061,Person00060,Pl ace00032,Place00016,Place00026,Place00015,Place00043" directed="false"

k="5"

3. Select the algorithm search policy **shortest** or **foremost**. Click **Run** to run the temporal paths algorithm. The graph engine calculates and returns the temporal analysis path based on the selected algorithm search policy. The path dynamically extends with the time axis until it reaches the target node. The JSON results are displayed in the query result area.