Graph Engine Service User Guide

Issue	01
Date	2020-05-09

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.

Contents

Graph Engine Service User Guide

1 Making Preparations

Before using GES, register a HUAWEI CLOUD account.

Registering a HUAWEI CLOUD Account

Skip this step if you already have registered one.

Step 1 Log in to the **HUAWEI CLOUD** official website.

Step 2 Click **Register** in the upper right corner to enter the registration page.

Step 3 Complete the registration as instructed. For detailed operations, see **Account Registration Process**.

----End

2 Permissions Management

2.1 Creating a User and Granting Permissions

This section describes how to use **IAM** to implement fine-grained permissions control for your GES resources. With IAM, you can:

- Create IAM users for employees based on the organizational structure of your enterprise. Each IAM user has their own security credentials, providing access to GES resources.
- Grant only the permissions required for users to perform a specific task.
- Entrust a HUAWEI 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.

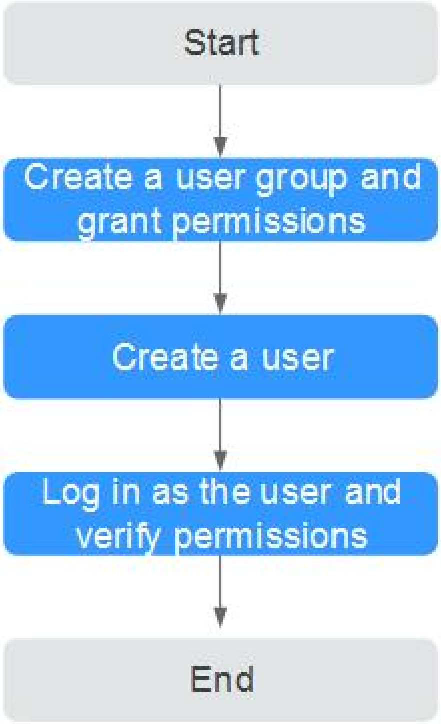
Prerequisites

- **GES ReadOnlyAccess** is a policy.
- Learn about the permissions (see **GES System Permissions**) supported by GES and choose policies or roles according to your requirements.

Process Flow

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

Figure 2-1 Process for granting GES permissions


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

1. **Create a user group and assign permissions** to it.

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

2. **Create an IAM user.**

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

3. **Log in** as the user and verify permissions.

   Log in to the GES console using the newly created user, switch to the authorized region, and verify the user's 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 already taken effect.

2.2 Creating a GES Custom Policy

Custom policies can be created as a supplement to the system policies of GES. For the actions supported for custom policies, see **Permissions Policies and Supported Actions**.

You can create custom policies in either of the following two ways:

- **Visual editor**: Select cloud services, actions, resources, and request conditions without the need to know policy syntax.

- **JSON**: Edit JSON policies from scratch or based on an existing policy.

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

Example Policies

- Example 1: Allowing users to query and operate graphs

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

- Example 2: Denying graph deletion

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

  The following method can be used if you need to assign the **GES FullAccess** policy to a user but also forbid the user from deleting graphs. Create a custom policy for denying graph deletion, and assign both policies to the group the user belongs to. Then the user can perform all operations on GES except deleting graphs. The following is an example deny policy:

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

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

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

- Example 4: Authorizing users to operate some graph resources and view all resources

  The policy consists of the following two parts:

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

  – Part 2: Authorizing users to query the graph, backup, task, and metadata lists, verify metadata files, and view job details

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

2.3 GES System Permissions

GES Permissions

By default, new IAM users do not have permissions assigned. You need to add a user to one or more groups, and attach permissions policies or roles to these groups. Users inherit permissions from the groups to which they are added After authorization, the users can perform specified operations on DLI based on the permissions.

GES is a project-level service deployed and accessed in specific physical regions. To assign DLI permissions to a user group, specify the scope as region-specific projects and select projects for the permissions to take effect. If **All projects** is selected, the permissions will take effect for the user group in all region-specific projects. When accessing GES, the users need to switch to a region where they have been authorized to use GES.

- You can grant users permissions by using roles and policies.
  - Roles are a type of coarse-grained authorization mechanism that defines permissions related to user responsibilities. Only a limited number of service-level roles for authorization are available. When using roles to grant permissions, you need to also assign other roles on which the permissions depend to take effect. However, roles are not the ideal choice for fine-grained authorization and secure access control.
  - Policies are a type of fine-grained authorization mechanism that defines the permissions for performing operations on specific cloud resources under certain conditions. This mechanism allows for flexible policy-based authorization and meets requirements for secure access control. For example, you can grant CS users only the permissions for managing a certain type of cloud servers.
- Dependencies: HUAWEI CLOUD services interact with each other. Therefore, if a GES policy depends on the policies of other services, the permissions of GES take effect only after the dependent policies are granted to users. For details, see **Table 2-1** and **Table 2-2**.

NOTE

Because of the cache, it takes about 13 minutes for an OBS role to take effect after being granted to users and user groups. After a policy is granted, it takes about 5 minutes to take effect.

Table 2-1 GES roles

Role Name	Description
Tenant Guest	Common tenant users ● Permissions: querying GES resources ● Function scope: Project-level service.
GES Administrator	GES administrator ● Permissions: performing any operations on GES resources ● Function scope: Project-level service. **NOTE** If you have the **GES Administrator**, **Tenant Guest**, and **Server Administrator** permissions, you can perform any operations on GES resources. If you do not have the **Tenant Guest** or **Server Administrator** permissions, you cannot use GES properly. ● To bind or unbind an EIP, you must have the **Security Administrator** permissions to create agencies. ● If GES needs to interact with OBS, such as creating and importing data, OBS permissions are required. For details, see **Table 2-5**.

Role Name	Description
GES Manager	GES manager ● Permissions: performing any operations on GES resources other than creating and deleting graphs. ● Function scope: Project-level service. **NOTE** If you have both the **GES Manager** and **Tenant Guest** permissions, you can perform any operations on GES resources except for creating and deleting graphs. If you do not have the **Tenant Guest** permissions, you cannot use GES properly. ● To bind or unbind an EIP, the users must have the **Security Administrator** and **Server Administrator** permissions. ● If GES needs to interact with OBS, such as importing data, OBS permissions are required. For details, see **Table 2-5**.
GES Operator	GES common users ● Permissions: viewing and accessing GES resources ● Function 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. If GES needs to interact with OBS, such as viewing metadata, OBS permissions are required. For details, see **Table 2-5**.

Table 2-2 GES policies

Policy	Description
GES FullAccess	Administrator permissions for GES. Users granted these permissions can perform all operations on GES, including creating, deleting, accessing, and updating graphs. **NOTE** ● To bind or unbind an EIP, you must have the **Security Administrator** permissions to create agencies. ● If GES needs to interact with OBS, such as creating and importing data, OBS permissions are required. For details, see **Table 2-5**.
GES Development	Use permissions for GES. Users granted these permissions can perform any operations on GES except for graph creation and deletion. **NOTE** ● To bind or unbind an EIP, the users must have the **Security Administrator** and **Server Administrator** permissions. ● If GES needs to interact with OBS, such as creating and importing data, OBS permissions are required. For details, see **Table 2-5**.

Policy	Description
GES ReadOnlyAccess	Read-only permissions for ECS. Users granted these permissions can only perform resource querying operations, such as viewing the graph list, metadata, and backups. **NOTE** If GES needs to interact with OBS, such as viewing metadata, OBS permissions are required. For details, see **Table 2-5**.

Table 2-3 Common GES operations supported by each role

Operation	GES Administrator	GES Manager	GES Operator	Tenant Guest
Creating graphs	√	×	×	×
Deleting graphs	√	×	×	×
Querying graphs	√	√	√	√
Accessing graphs	√	√	√	×
Importing data	√	√	×	×
Creating metadata	√	√	×	×
Viewing metadata	√	√	√	√
Copying metadata	√	√	×	×
Editing metadata	√	√	×	×
Deleting metadata	√	√	×	×
Clearing data	√	√	×	×
Backing up graphs	√	√	×	×
Loading backups	√	√	×	×
Deleting backups	√	√	×	×

Operation	GES Administrator	GES Manager	GES Operator	Tenant Guest
Querying backups	√	√	√	√
Starting graphs	√	√	×	×
Stopping graphs	√	√	×	×
Upgrading graphs	√	√	×	×
Exporting graphs	√	√	×	×
Binding an EIP	√	√	×	×
Unbinding an EIP	√	√	×	×
Viewing results in the Task Center	√	√	√	√

Table 2-4 Common GES operations supported by each policy

Operation	GES FullAccess	GES Development	GES ReadOnlyAccess	Resource
Querying the graph list	√	√	√	-
Querying graph details	√	√	√	graphName
Creating graphs	√	x	x	graphName
Accessing graphs	√	x	x	graphName
Stopping graphs	√	√	x	graphName
Starting graphs	√	√	x	graphName
Deleting graphs	√	x	x	graphName
Incrementally importing data to graphs	√	√	x	graphName
Exporting graphs	√	√	x	graphName

Operation	GES FullAccess	GES Development	GES ReadOnlyAccess	Resource
Clearing graphs	√	√	x	graphName
Upgrading graphs	√	√	x	graphName
Binding an EIP	√	√	x	graphName
Unbinding an EIP	√	√	x	graphName
Querying backups of all graphs	√	√	√	-
Querying backups of a graph	√	√	√	-
Adding backups	√	√	x	backupName
Deleting backups	√	√	x	backupName
Querying the metadata list	√	√	√	-
Querying metadata	√	√	√	metadataName
Verifying metadata	√	√	x	-
Adding metadata	√	√	x	metadataName
Deleting metadata	√	√	x	metadataName
Querying the task status	√	√	√	-
Querying the task list	√	√	√	-

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

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

GES Operation	Dependent OBS Permission
Creating a graph (with initial data), and importing or exporting the graph	**OBS Operator** policy or **Tenant Administrator** role

2.4 GES Resources

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

Table 2-6 GES resources and their paths

Specific Resource	Name	Path
graphName	GES graph name	graph.name
backupName	GES backup name	backup.name
metadataName	GES metadata name	metadata.name

2.5 GES 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 filter matching requesters by IP address. The following table shows the condition keys that apply to GES.

Table 2-7 GES request conditions

Condition Key	Operator	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 during 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 Getting Started with GES

3.1 Querying and Analyzing Graphs

If you have never used GES, you can use the sandbox access function to access a demo graph provided by the system to learn and use the functions of the graph editor. You can learn the functions through **Quick Start**. The system runs the PageRank algorithm by default.

This section describes how to use GES to query and analyze the sample graph data. The process is as follows:

Step 1: Creating a Graph

Step 2: Accessing the Graph

Step 3: Querying and Analyzing the Graph

Step 4: Viewing the Analysis Result

Step 1: Creating a Graph

Log in to the GES management console and click **Create Graph**.

Two creation modes are available: **Customize Graph** and **Use Industry-Specific Graph Template**. By default, the system displays the **Customize Graph** tab page.

Method 1: **Customize Graph**

Step 1 On the **Create Graph** page, click the **Customize Graph** tab and set the following parameters:

1. Select **Region**.

   Region: Area where a graph works. You can select the region from the drop-down list in the upper left corner of the page.

2. On the **Configure** tab page, set **Graph Name** and **GES Software Version**.

   – Specify **Graph Name**, for example, **demo**, or use the default name in the system.

     The graph name must comply with the following rules:

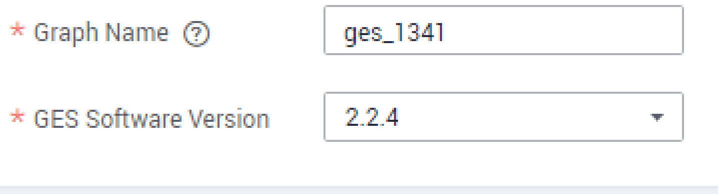
- Contain 4 to 64 characters and start with a letter.
- Letters are case-insensitive.
- Only letters, digits, and underscores (_) are allowed.

– Select the **GES Software Version** as required.

NOTE

Currently, only the default version can be selected.

Figure 3-1 Graph name and software version

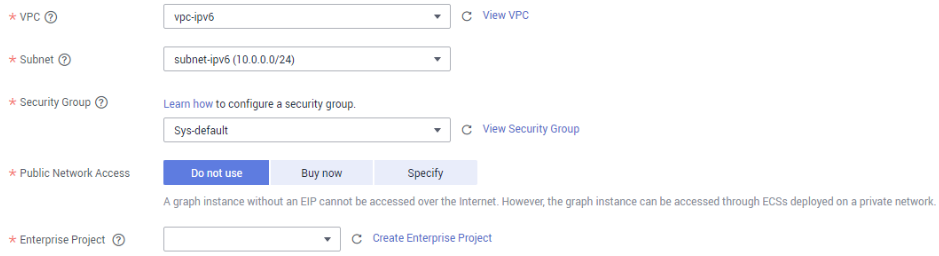


* Graph Name ⓘ: ges_1341

* GES Software Version: 2.2.4

3. Specify network information, including **VPC**, **Subnet**, **Security Group**, and **Public Network Access**.

Figure 3-2 Network information



* VPC ⓘ: vpc-ipv6 View VPC

* Subnet ⓘ: subnet-ipv6 (10.0.0.0/24)

* Security Group ⓘ: Learn how to configure a security group. Sys-default View Security Group

* Public Network Access: ● Do not use ○ Buy now ○ Specify

A graph instance without an EIP cannot be accessed over the Internet. However, the graph instance can be accessed through ECSs deployed on a private network.

* Enterprise Project ⓘ: Create Enterprise Project

a. **VPC**: A Virtual Private Cloud (VPC) is a secure, isolated, and logical network environment.

   Select the VPC for which you want to create the graph and click **View VPC** to view the name and ID of the VPC.

   NOTE

   If your account has a VPC, a VPC will be automatically selected. You can change it as required. If no VPC is available, you need to create a VPC. After the VPC is created, it will be automatically selected.

b. **Subnet**: A subnet provides dedicated network resources that are logically isolated from other networks, improving 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.

c. **Security Group**: A security group is a logical group. It provides access control policies for the ECSs that are mutually trusted and have the same security protection requirements in a VPC.

   - Click **Learn how to configure a security group.** to learn how to configure a security group.

- Click **View Security Group** to learn security group details.

d. Select the **Public Network Access** mode as required. Possible values are **Do not use**, **Buy now**, and **Specify**.

- **Do not use**: A graph instance without an elastic IP (EIP) cannot be accessed over the Internet. However, the graph instance can be accessed through ECSs deployed on a private network.
- **Buy now**: GES automatically allocates an EIP with exclusive bandwidth to the graph instance so that the graph instance can be accessed over the Internet using the EIP. In addition, GES uses the tenant permission to create an agency with the prefix of **ges_agency_default** automatically in the project to support EIP binding.
- **Specify**: An EIP allows the graph instance to be accessed over the Internet.

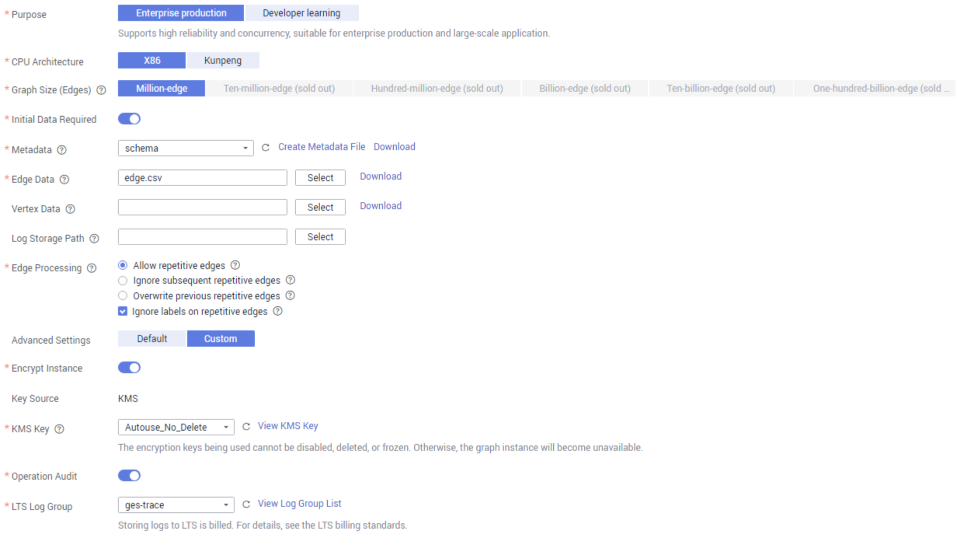
  Click **Create EIP** to access the VPC management console and create an EIP.

e. **Enterprise Project**: An enterprise project facilitates project-level management and grouping of cloud resources and users.

Click **Create Enterprise Project** to go to the **Enterprise Project Management** page.

4. Set graph parameters.

Figure 3-3 Graph parameters



* Purpose: Enterprise production | Developer learning
Supports high reliability and concurrency, suitable for enterprise production and large-scale application.

* CPU Architecture: X86 | Kunpeng

* Graph Size (Edges) ⓘ: Million-edge | Ten-million-edge (sold out) | Hundred-million-edge (sold out) | Billion-edge (sold out) | Ten-billion-edge (sold out) | One-hundred-billion-edge (sold ...

* Initial Data Required: ●

* Metadata ⓘ: schema ▾ ↻ Create Metadata File Download

* Edge Data ⓘ: edge.csv Select Download

Vertex Data ⓘ: Select Download

Log Storage Path ⓘ: Select

* Edge Processing ⓘ:
● Allow repetitive edges ⓘ
○ Ignore subsequent repetitive edges ⓘ
○ Overwrite previous repetitive edges ⓘ
☑ Ignore labels on repetitive edges ⓘ

Advanced Settings: Default | Custom

* Encrypt Instance: ●

Key Source: KMS

* KMS Key ⓘ: Autouse_No_Delete ▾ ↻ View KMS Key
The encryption keys being used cannot be disabled, deleted, or frozen. Otherwise, the graph instance will become unavailable.

* Operation Audit: ●

* LTS Log Group: ges-trace ▾ ↻ View Log Group List
Storing logs to LTS is billed. For details, see the LTS billing standards.

a. **Purpose**: Select **Enterprise production** or **Developer learning**.

- **Enterprise production**: Supports high reliability and concurrency, suitable for enterprise production and large-scale application.
- **Developer learning**: Offers complete function experience, suitable for developer learning.

b. **CPU Architecture**: Currently, GES supports **X86**.

c. **Graph Size (Edges)**: Based on a user's current quota, the system displays the numbers of graphs and edges that can be created. The unit is edge. **Enterprise production** and **Developer learning** have different graph specifications.

   - **Enterprise production**: Currently, **Million-edge**, **Ten-million-edge**, **Hundred-million-edge**, **Billion-edge**, **Billion-edge-pro**, **Ten-billion-edge**, and **One-hundred-billion-edge** are supported.
   - **Developer learning**: Currently, only **Ten-thousand-edge** is supported.

d. **Initial Data Required**: This option is disabled by default. You can create a graph first and then import data. If you enable this option, you need to set the following parameters:

   - **Metadata**: Indicates the graph metadata information.

     If no metadata is available, click **Create Metadata File**. For details about how to create a metadata file, see **Metadata Management**.

     If you already have metadata files, import them to GES.
   - **Edge Data**: Describes edges that form the graph, including information about the edge structures, labels, and properties.
   - **Vertex Data**: Describes vertices that form the graph, including information about all vertex IDs, labels, and properties. If you leave it blank, the vertices in the **Edge Data** set are used as the source of **Vertex Data**.

   NOTE

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

   - **Log Storage Path**: Stores vertex and edge data sets that do not comply with the metadata definition, as well as detailed logs during graph import. Storage on OBS may incur fees, so delete the data in a timely time if you do not need to use it any more.
   - **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.

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 source vertex and target vertex are repetitive edges.

5. **Advanced Settings**: Set this parameter to **Default** or **Custom**.
   - **Default**: Use the default values of the system.
   - **Custom**: Include **Encrypt Instance** and **Operation Audit**.

     Encrypt Instance: Indicate whether to encrypt graph instances. **Key Source** is default to **KMS**. **KMS Key**: Select the corresponding key.

     NOTE

     Disabling or deleting a KMS key affects the instance functions.

     Operation Audit: Indicate whether to enable operation audit. **LTS Log Group**: Select the corresponding log group.

     NOTE

     You will be billed for storing logs to LTS. For details, see the LTS billing standards.

Step 2 Click **Next**. The **Confirm** tab page is displayed.

Step 3 Confirm the information and click **Submit** to create the graph.

Step 4 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.

----End

- Method 2: **Use Industry-Specific Graph Template**

Step 1 On the **Create Graph** page, click the **Use Industry-Specific Graph Template** tab and set the following parameters:

1. Select **Region**.

   Region: Area where a graph works. You can select the region from the drop-down list in the upper left corner of the page.
2. On the **Configure** tab page, set the following parameters:
   - Select the target template, for example, **Asset Management Graph Template**.
   - Set network information.

     Refer to Method 1: **Customize Graph**.

Step 2 Click **Next**. On the **Confirm** tab 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).

Step 3 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.

----End

Step 2: Accessing the Graph

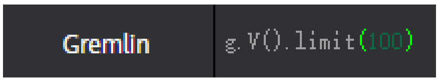
1. On the **Graph Management** page, locate the row containing the graph to be accessed, for example, **demo** created in the previous steps, and click **Access** in the **Operation** column.
2. Enter the displayed graph editor, and query and analyze the current graph. For details, see **Step 3: Querying and Analyzing the Graph**.

Step 3: Querying and Analyzing the Graph

1. You can query and analyze graphs in either of the following ways if a graph is created in **Customize Graph** mode:
   - **Queries using Gremlin commands**
     1. Enter the query command in the Gremlin text box in the lower part of the page, for example, **g.V().limit(100)**.

        > NOTE
        >
        > To prevent the system queries from being time-consuming due to a large amount of returned data, you are advised to add the **limit** parameter and set it to less than **1,000** for a better display effect.

        Figure 3-4 Gremlin query

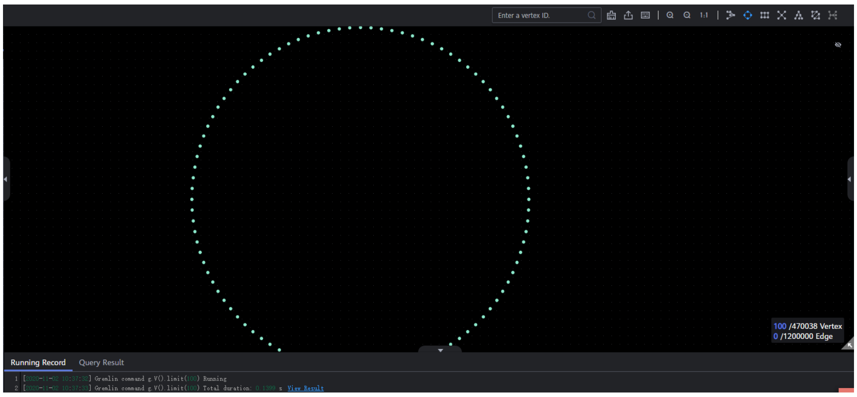
        

        Gremlin g.V().limit(100)

     2. Press **Enter** to run the Gremlin command. The query result is displayed in both the drawing area and result area.

Graph Engine Service User Guide

Figure 3-5 Gremlin query result

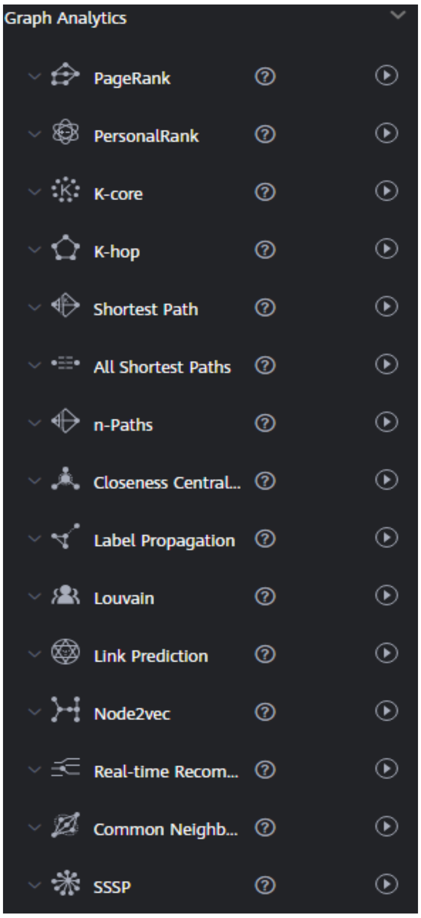


```
1 [2020-11-02 10:37:32] Gremlin command g.V().limit(100) Running
2 [2020-11-02 10:37:33] Gremlin command g.V().limit(100) Total duration: 0.1399 s View Result
```

– **Analysis by selecting algorithms and configuring parameters**

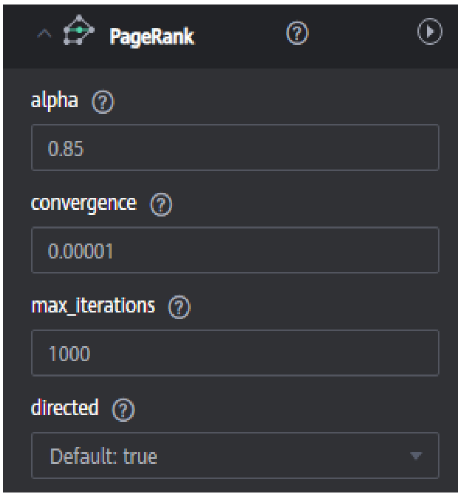
i. The algorithms supported by GES are displayed in the left pane of the graph editor. Select the target analysis algorithm from the list.

Figure 3-6 Algorithm list



ii. Select the algorithm to be used, expand its parameter configuration by clicking ⌄, and input corresponding values based on the required range. Take PageRank as an example. **alpha** indicates the weight coefficient, and its value is **0.85**. **convergence** is the convergence coefficient, and its value is **0.00001**. **max_iterations** indicates the maximum iterations and its value is **1,000**. **directed** indicates whether to consider the edge direction and the default value is **true**.

Figure 3-7 Setting PageRank algorithm parameters



PageRank

alpha: 0.85

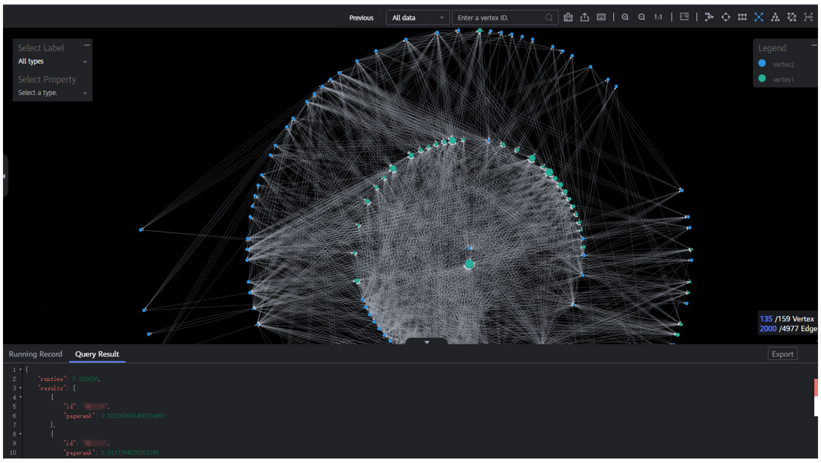
convergence: 0.00001

max_iterations: 1000

directed: Default: true

iii. Execute the algorithm by clicking ⏵. The analysis result is displayed in both the drawing area and result area.

Figure 3-8 Analysis result



```
{
    "runtime": 0.000606,
    "results": [
        {
            "id": [redacted],
            "pagerank": 0.020250601455334853
        },
        {
            "id": [redacted],
            "pagerank": 0.0177394030153798
```

2. For a graph created using an industry-specific template, in addition to using the Gremlin commands and selecting algorithms and configuring parameters, four public operation scenarios of the industry-specific graph template are added.

   – Scenario 1: List administrators and their applications.

     ■ Command: **g.V().hasLabel('admin').outE().otherV().hasLabel('application').path()**

     ■ Procedure: Click **Run**. The graph is displayed on the canvas.

- Scenario 2: Find all paths from administrator *x* to equipment room *x*.
  - Implementation principle: **get${Administrator *x*}** is the source vertex and **${Equipment room *x*}** is the target vertex. Find out all paths using the n-Paths algorithm, where **source** is **Administrator** *x*, **target** is **Equipment room** *x*, **direct** is **false**, **n** is **10**, and **k** is**10**.
  - Procedure: Click **Run**. In the dialog box that is displayed, select administrator *x* and equipment room *x* from the drop-down list box. After the operation is complete, the graph is displayed on the canvas.
- Scenario 3: Find all VMs and physical machines on which application *x* depends.
  - Implementation principle: **get${Application *x*}** is the input to find all points at layer 2. Use the K-Hop algorithm, where **k** is **2**, **source** is *x*, and **mode** is **in**.
  - Procedure: Click **Run**. In the dialog box that is displayed, set the application *x* (select the application from the drop-down list box and drag the vertex of the application entity to the drop-down list box). After the operation is complete, the graph is displayed on the canvas.
- Scenario 4: Find all vertices with a specified label, for example, all administrators, equipment rooms, and physical machines in the graph.
  - Implementation principle: Filter vertices by label.
  - Procedure: Click **Run**. In the dialog box that is displayed, select the **label** from the drop-down list box. After the operation is complete, the graph is displayed on the canvas.

Step 4: Viewing the Analysis Result

You can view the running record and query result in the result area or click the **Export** button on the right to download the analysis result.

4 Introduction to GES Management Console

4.1 Graph Data Formats

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

- GES supports the loading of raw graph data in the standard CSV format. If your raw data is not in the specified format, convert it to the format supported by GES.
- 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

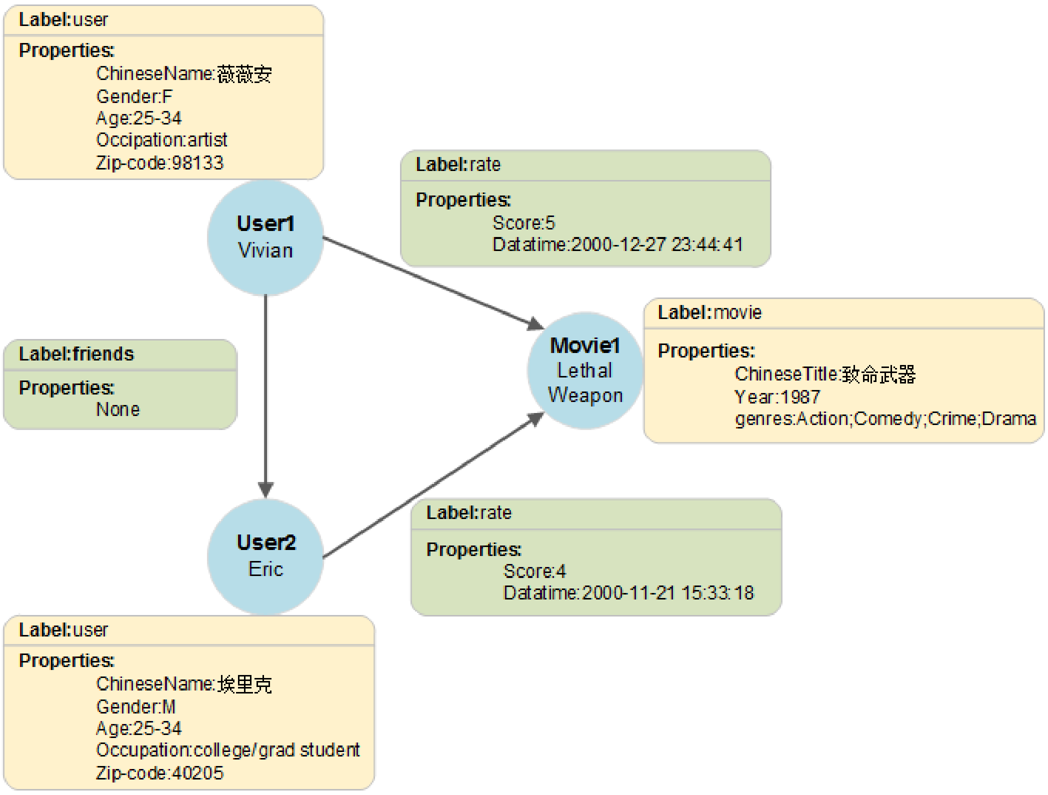
Graph data is imported based on the property graph model in GES, so you must learn the concept of the property graph.

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.
- The metadata is used to describe vertex and edge properties. The metadata consists of 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.

In the following example, the graph data consists of three vertices and three edges. Vivian, Eric, and Lethal Weapon indicate vertices. (Vivian, Eric), (Vivian, Lethal Weapon), and (Eric, Lethal Weapon) indicate edges. **user** and **movie** indicate the vertex types (labels), and **rate** and **friends** indicate the relationship types (labels).

Figure 4-1 Graph data format example



Metadata

The GES metadata is a file in XML format and is used to define vertex and edge properties.

It contains labels and properties.

- **Label**

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

  NOTE

  If the same **Property Name** is defined in different labels, the **Cardinality** and **Data Type** in different labels must be the same.

- **Property**

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

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

    NOTE

    A label cannot contain two properties with the same name.

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

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

      NOTE

      If the value of a **single** property in a data file is **value1;value2**, **value1;value2** is regarded as a single value.

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

      NOTE

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

- **Data Type**: Indicates the data type. The following table lists the data types supported by GES.

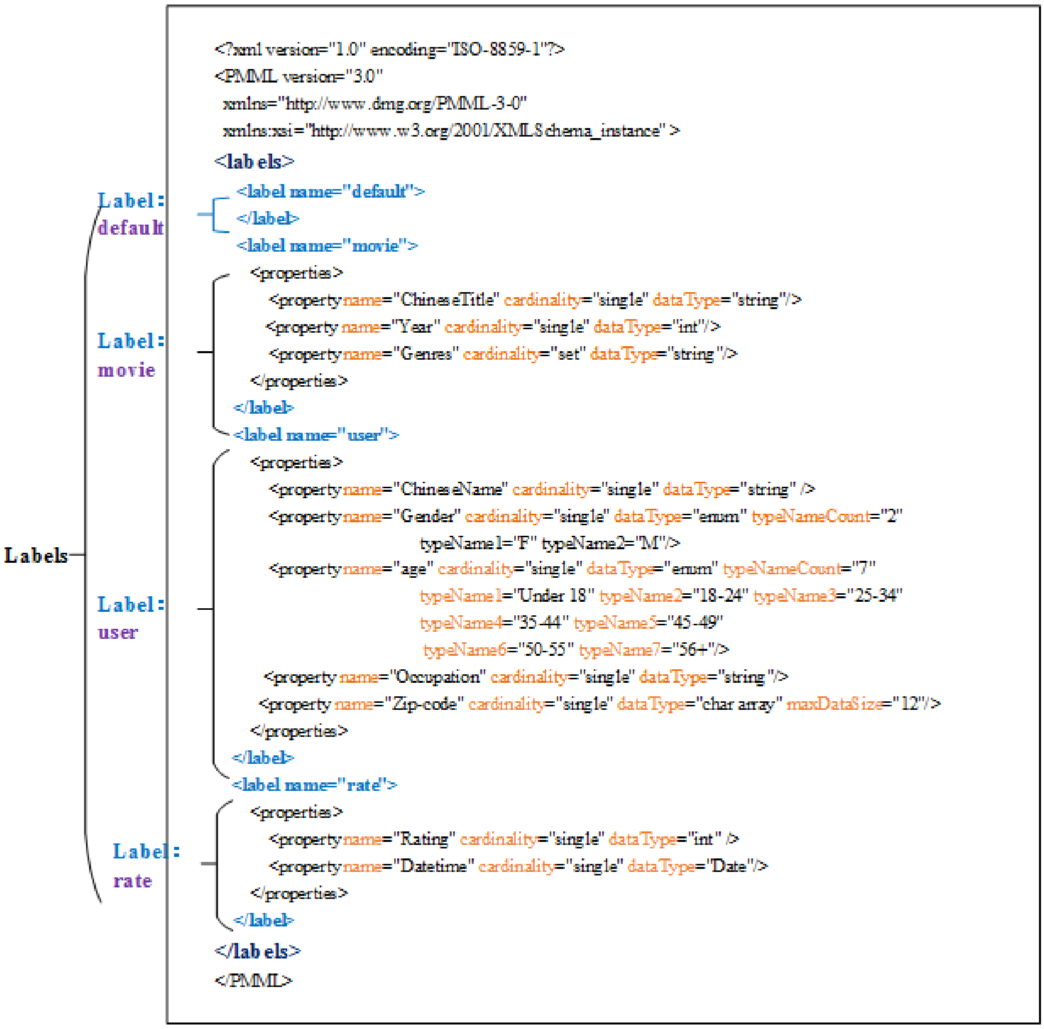
Table 4-1 Supported data types

Type	Description
char	Character
char array	Fixed-length character string (The maximum length must be specified.) **NOTE** ● Only **single** supports the data type. ● If the data is a character string, you are advised to set this parameter to **char array**. If it is set to **string**, the import is slower.
float	Float type (32-bit float)
double	Double float type (64-bit float)
bool	bool type. Possible values are **(0/1)** and **(true/false)**.
long	Long integer (value range: **-2^63** to **2^63-1**)
int	Integer (value range: **-2^31** to **2^31-1**)
date	Date. Currently, the following formats are supported: ● YYYY-MM-DD HH:MM:SS ● YYYY-MM-DD **NOTE** The value of MM or DD must consist of two digits. If the day or month number contains only one digit, add 0 before it, for example, 05/01.

Type	Description
enum	Enumeration (you need to specify the number of enumeration types and each enumeration value). For details, see **Figure 4-2**.
string	Variable character string **NOTE** The string length is not fixed, which affects data import efficiency. You are advised to use char array instead. You can define the length of char array based on service requirements. It is recommended that the length be less than or equal to 32 characters.

The following figure shows the metadata example:

Figure 4-2 Metadata example



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

Vertex Files

A vertex file contains the data of each vertex. A vertex of data is generated for each behavior. The following shows the format. **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 vertex ID type. Its default value is **string**.

Example:

```
Lethal Weapon, movie, Lethal Weapon, 1987, Action; Comedy; Crime; Drama
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. Graph specifications in GES are defined based on the edge quantity, for example, one million edges. The following shows the format. **id 1** and **id 2** are the IDs of the two endpoints of an edge.

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

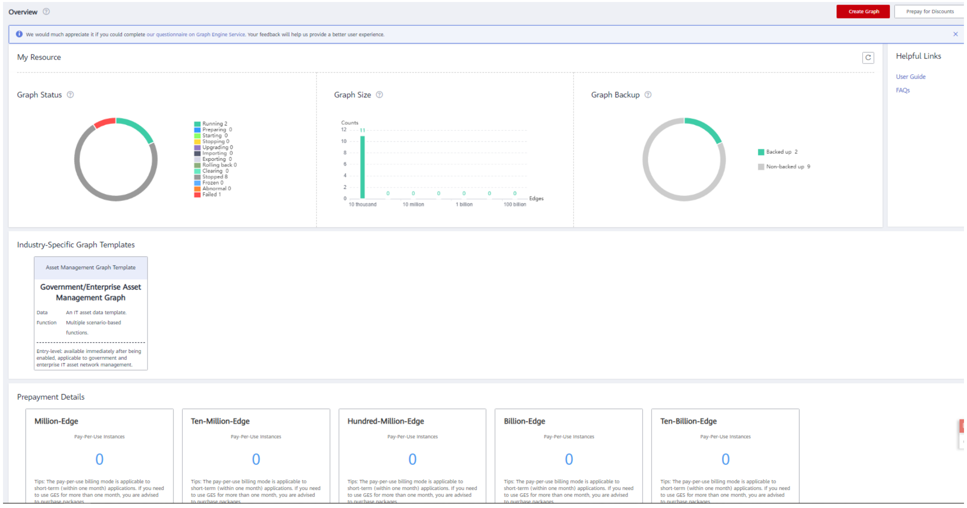
Example:

```
Vivian,Lethal Weapon,rate,5,2000-12-27 23:44:41
Eric,Lethal Weapon,rate,4,2000-11-21 15:33:18
Vivian,Eric,friends
```

4.2 Overview

The **Overview** page displays the **My Resource** information, including **Graph Status**, **Graph Size**, **Prepayment Details**, and **Graph Backup**, enabling you to quickly learn the information about existing graphs and charging details.

Figure 4-3 Overview



We would much appreciate it if you could complete our questionnaire on Graph Engine Service. Your feedback will help us provide a better user experience.

Government/Enterprise Asset Management Graph

Data An IT asset data template.

Function Multiple scenario-based functions.

Entry-level: available immediately after being enabled, applicable to government and enterprise IT asset network management.

Million-Edge	Ten-Million-Edge	Hundred-Million-Edge	Billion-Edge	Ten-Billion-Edge
Pay-Per-Use Instances	Pay-Per-Use Instances	Pay-Per-Use Instances	Pay-Per-Use Instances	Pay-Per-Use Instances
0	0	0	0	0
Tips: The pay-per-use billing mode is applicable to short-term (within one month) applications. If you need to use GES for more than one month, you are advised to purchase packages.	Tips: The pay-per-use billing mode is applicable to short-term (within one month) applications. If you need to use GES for more than one month, you are advised to purchase packages.	Tips: The pay-per-use billing mode is applicable to short-term (within one month) applications. If you need to use GES for more than one month, you are advised to purchase packages.	Tips: The pay-per-use billing mode is applicable to short-term (within one month) applications. If you need to use GES for more than one month, you are advised to purchase packages.	Tips: The pay-per-use billing mode is applicable to short-term (within one month) applications. If you need to use GES for more than one month, you are advised to purchase packages.

Graph Status

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

Table 4-2 Status description

Status	Description
Running	Indicates running graphs. Graphs in this status can be accessed.
Preparing	Indicates graphs whose ECSs are being created or started.
Starting	Indicates graphs being started.
Stopping	Indicates graphs being stopped.
Upgrading	Indicates graphs being upgraded.
Importing	Indicates graphs being imported.
Exporting	Indicates graphs being exported.
Rolling back	Indicates graphs being rolled back.
Clearing	Indicates graphs being cleared.
Stopped	Indicates stopped graphs. Graphs in this status cannot be accessed, but can be restarted.
Frozen	Indicates that the user's account and resources 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

Graph Size displays the number of graphs in different sizes. Currently, the system supports seven sizes.

Table 4-3 Specifications

Specifications	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 one million.

Specifications	Description
10 million	Indicates that the number of edges of a graph cannot exceed 10 million.
100 million	Indicates that the number of edges of a graph cannot exceed 100 million.
1 billion	Indicates that the number of edges of a graph cannot exceed one 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

The graph data can be backed up to prevent data loss. **Graph Backup** displays the numbers of graphs that are backed up and are not backed up.

Table 4-4 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.

Industry-Specific Graph Templates

The industry-specific graph templates are product packages provided for common scenarios of governments and enterprises. Each product package has built-in graph models and sample data, which can be used out of the box. Only two steps are required from order placement to graph visualization. Those templates enable you to quickly experience the powerful analysis and scenario response capabilities of graphs.

Payment Details.

This part displays the purchase methods, number of instances, and expiration time of different graph sizes.

4.3 Graph Management

4.3.1 Graph Management Overview

On the **Graph Management** page, you can view the running status, internal access address, external access address, and creation time of a graph, and perform the following operations:

- **Creating and Accessing a Graph**
- **Backing Up a Graph on Graph Management**
- **Importing Data**
- **Starting a Graph**
- **Stopping a Graph**
- **Deleting a Graph**
- **Upgrading a Graph**
- **Exporting a Graph**
- **Binding an EIP**
- **Unbinding an EIP**
- **Clearing Data**
- **Viewing Monitoring Metrics**
- **Querying Schema**

Click ⌄ next to a graph name, you can view details about the graph, including the graph ID, VPC, subnet, security group, graph size (edges), vertex data set, edge data set, metadata, graph version, cross-AZ HA status, creator, enterprise project, encryption status, and CPU architecture.

4.3.2 Creating and Accessing a Graph

Scenario

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

Procedure

For details about how to create and access a graph, see **Querying and Analyzing Graphs**.

4.3.3 Backing Up a Graph on Graph Management

Scenario

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

Procedure

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

- For details about operations on the **Backup Management** page, see **Backing Up a Graph**.
- Operations on the **Graph Management** are as follows:

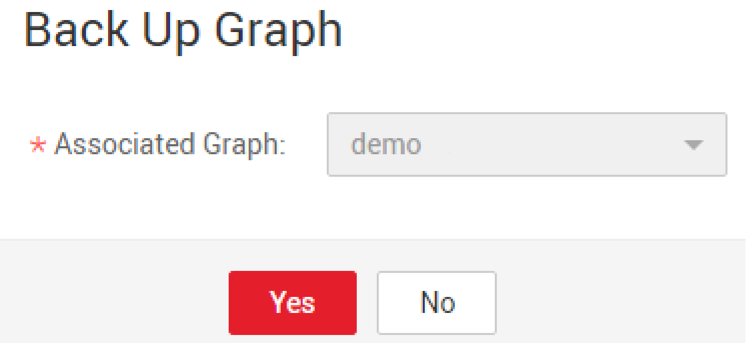
  a. Log in to the GES management console.

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

  c. Locate the target graph in the graph list and select **Back Up** in the **Operation** column.

  d. In the displayed dialog box, click **Yes**.

Figure 4-4 Backup on the Graph Management page



Back Up Graph

* Associated Graph: demo

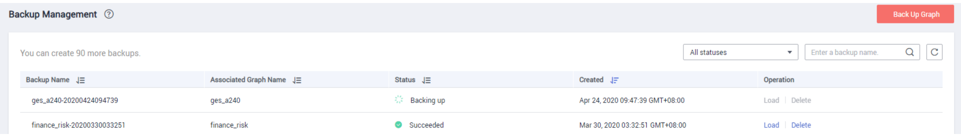
NOTE

On the **Graph Management** page, the backup operation can be performed only on the selected graph. The associated graph cannot be changed.

  e. In the navigation tree on the left, select **Backup Management**. You can view the data being backed up and already backed up in the backup list.

  If **Status** changes from **Backing up** to **Succeeded**, the backup is successful.

Figure 4-5 Backup management list



Backup Management

You can create 90 more backups.

Backup Name	Associated Graph Name	Status	Created	Operation
ges_a240-20200424094739	ges_a240	Backing up	Apr 24, 2020 09:47:39 GMT+08:00	Load | Delete
finance_risk-20200330033251	finance_risk	Succeeded	Mar 30, 2020 03:32:51 GMT+08:00	Load | Delete

4.3.4 Importing Data

Scenario

GES allows you to select initial data during graph creation or incrementally import data after the graph is created.

- Select initial data during cluster creation.

  If you select the initial data when creating a graph, the data is imported to the graph by default. You do not need to import it again.
- Incrementally import data after the graph is created.

If you need to add data after a graph is created, click **Import** in the graph list to import the data incrementally.

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.

The following provides the procedure for incrementally importing data:

Procedure

Step 1 Log in to the GES management console.

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

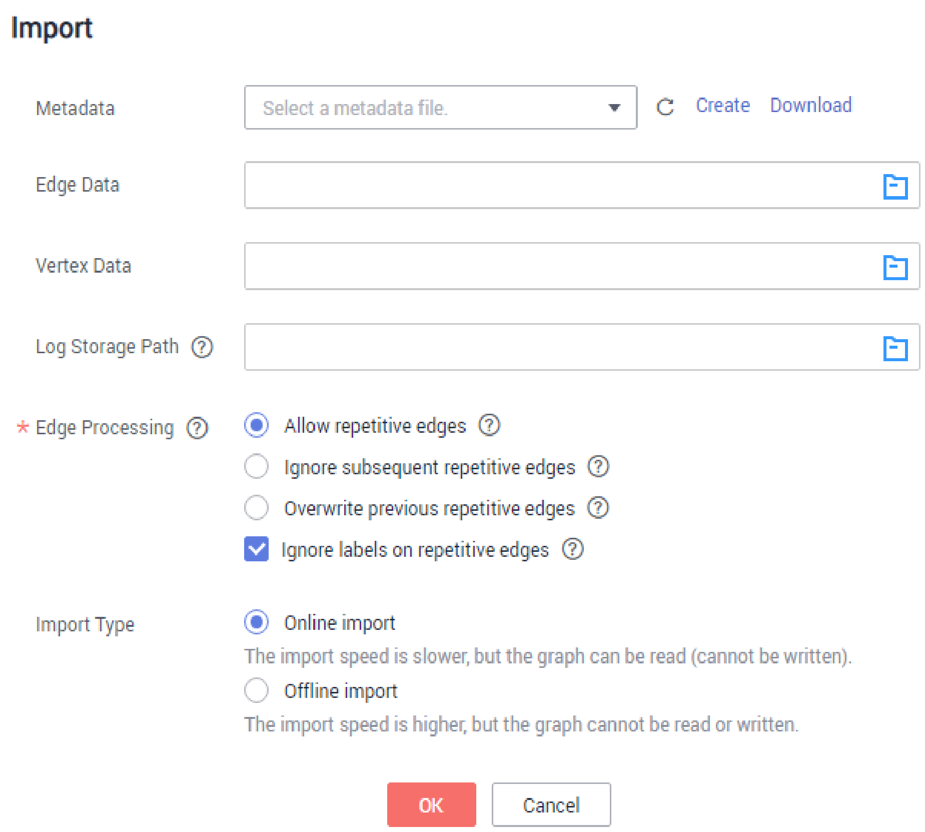
Step 3 Locate the target graph in the graph list and select **Import** in the **Operation** column.

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

- **Metadata**: Select an existing metadata file or create one. For details, see **Metadata Management**.
- **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**. For details, see **Edge Processing**.
- **Import Type**: The value can be **Online import** or **Offline import**.

Graph Engine Service User Guide

Figure 4-6 Importing data



Import

Metadata: Select a metadata file. ▾ ⟳ Create Download

Edge Data:

Vertex Data:

Log Storage Path ?:

* Edge Processing ?:

● Allow repetitive edges ?

○ Ignore subsequent repetitive edges ?

○ Overwrite previous repetitive edges ?

☑ Ignore labels on repetitive edges ?

Import Type:

● Online import

The import speed is slower, but the graph can be read (cannot be written).

○ Offline import

The import speed is higher, but the graph cannot be read or written.

Step 5 Click **OK**.

----End

4.3.5 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

4.3.6 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.

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

----End

4.3.7 Deleting a Graph

Scenario

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

NOTE

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

Procedure

Step 1 Log in to the GES management console.

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

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

Step 4 In the dialog box that is displayed, select or deselect **Release the EIP bound with the graph instance**. If you do not select this option, the EIP will continue to be charged.

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

----End

4.3.8 Upgrading a Graph

Scenario

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.

Procedure

Step 1 Log in to the GES management console.

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

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

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

NOTE

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

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

NOTE

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

----End

4.3.9 Exporting a Graph

Scenario

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

NOTE

Currently, only graphs of version 1.0.3 and later support this function.

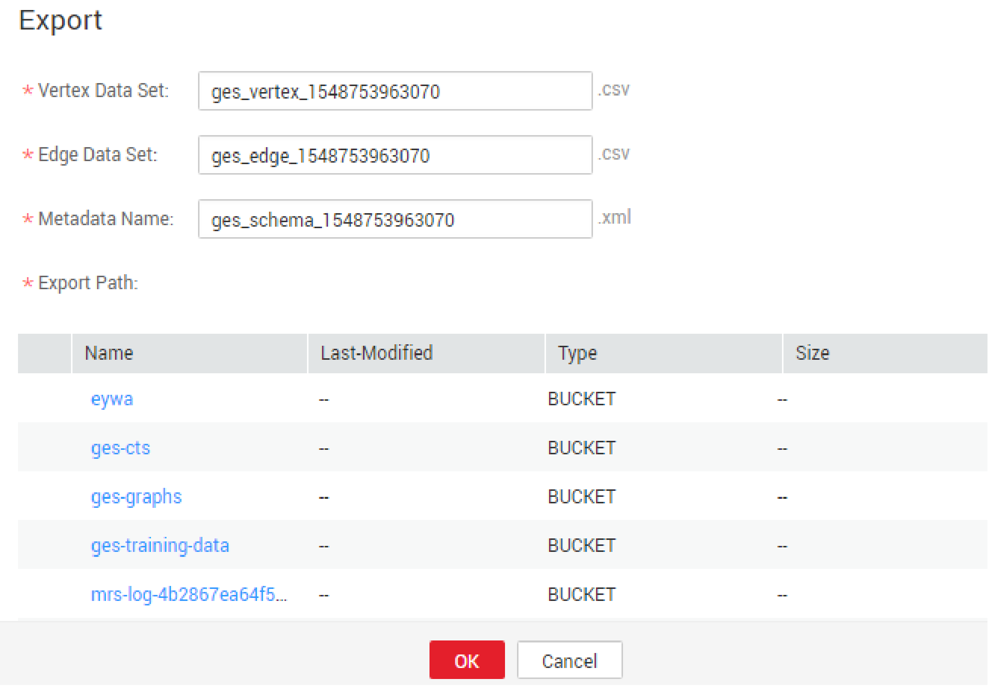
Procedure

Step 1 Log in to the GES management console.

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

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

Figure 4-7 Exporting a graph



Export

* Vertex Data Set: ges_vertex_1548753963070 .csv

* Edge Data Set: ges_edge_1548753963070 .csv

* Metadata Name: ges_schema_1548753963070 .xml

* Export Path:

	Name	Last-Modified	Type	Size
	eywa	--	BUCKET	--
	ges-cts	--	BUCKET	--
	ges-graphs	--	BUCKET	--
	ges-training-data	--	BUCKET	--
	mrs-log-4b2867ea64f5...	--	BUCKET	--

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

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

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

----End

4.3.10 Binding an EIP

Scenario

To access GES over the public network, you can bind an Elastic IP Address (EIP).

Procedure

Step 1 Log in to the GES management console.

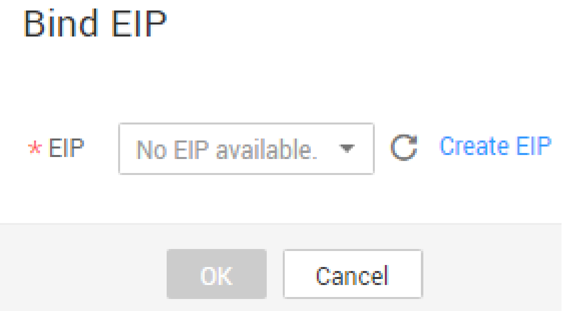
Step 2 In the navigation tree on the left, select **Graph Management**.

Step 3 Locate the target graph in the graph list and choose **More** > **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 4-8 Binding an EIP



Step 5 Click **OK**.

----End

4.3.11 Unbinding an EIP

Scenario

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

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** > **Unbind EIP** in the **Operation** column.

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

----End

4.3.12 Clearing Data

Scenario

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

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

NOTE

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

Procedure

Step 1 Log in to the GES management console.

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

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

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

NOTE

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

Step 5 Click **Yes**.

----End

4.3.13 Viewing Monitoring Metrics

Scenario

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

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

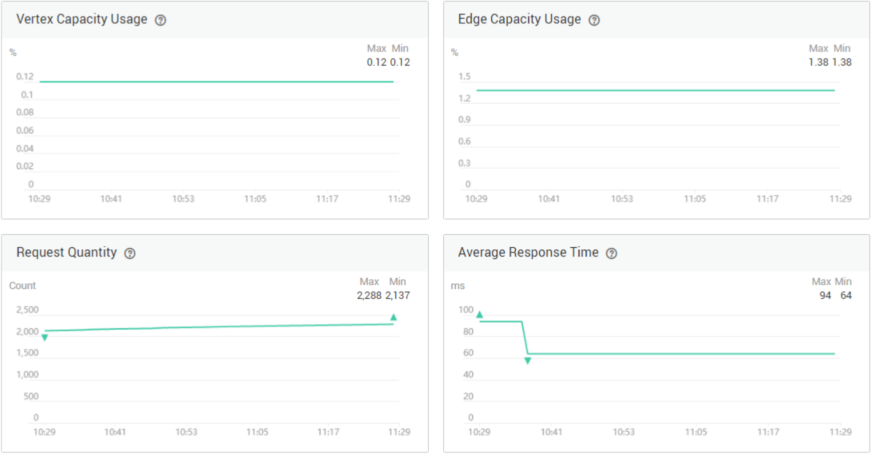
Prerequisites

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

Procedure

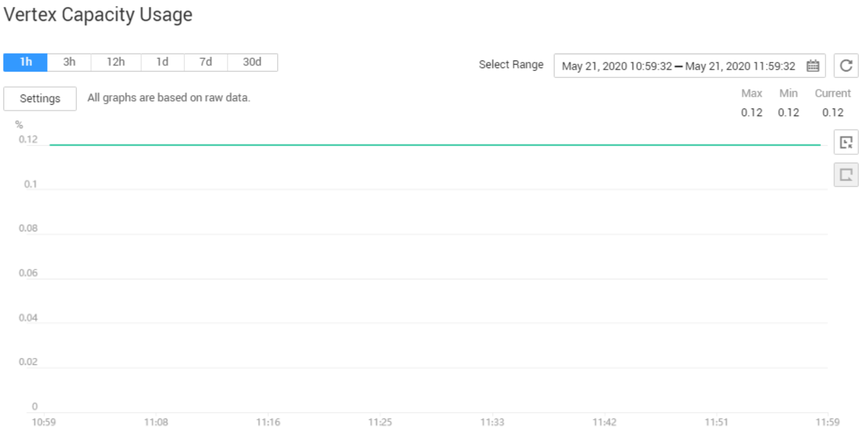
1. Log in to the management console.
2. In the navigation pane, choose **Graph Management**. In the **Operation** column, choose **More** > **View Metric**. The Cloud Eye management console is displayed.
3. On the monitoring page for GES, you can view the figures of all monitoring metrics.

Figure 4-9 Viewing GES monitoring metrics



4. To view the monitoring curve in a longer time range, click ⤢ to enlarge each figure.

Figure 4-10 Zoomed in monitoring graph



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

4.3.14 Querying Schema

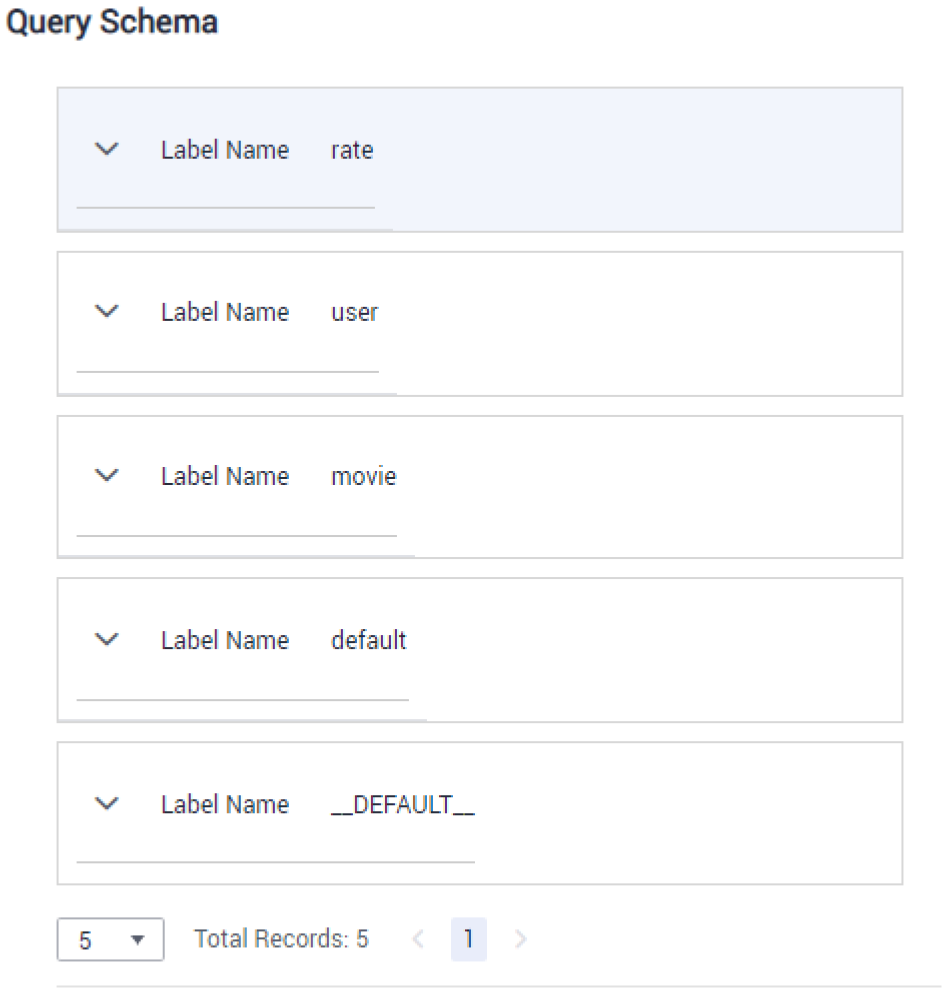
Scenario

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

Procedure

1. Log in to the management console.
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 4-11 Querying schema



Query Schema

Label Name rate

Label Name user

Label Name movie

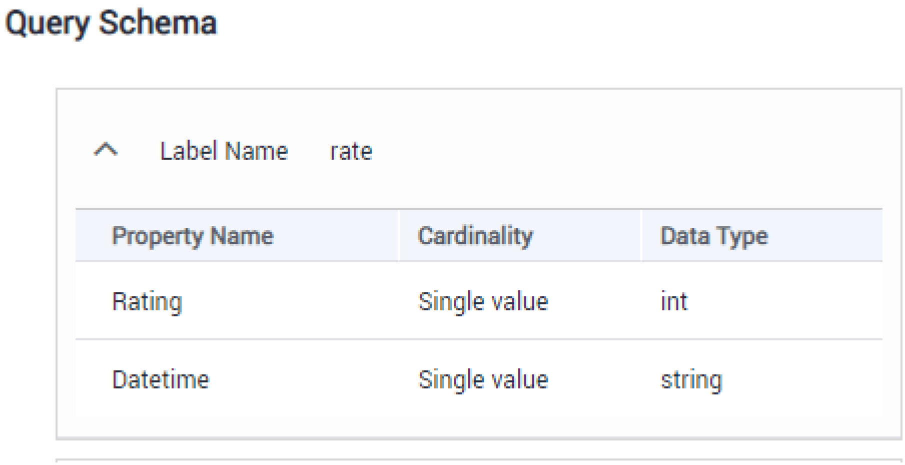
Label Name default

Label Name __DEFAULT__

5 ▾ Total Records: 5 ‹ 1 ›

3. To view the properties contained in labels, click ⌄ of each label.

Figure 4-12 Viewing properties in labels



Query Schema

Label Name rate

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

4.4 Backup Management

4.4.1 Overview of Backup Management

On the **Backup Management** page, you can view the associated graph name, status, and creation time of a graph backup, and perform the following operations:

- Backing Up a Graph
- Loading a Graph
- Deleting a Backup

4.4.2 Backing Up a Graph

Scenario

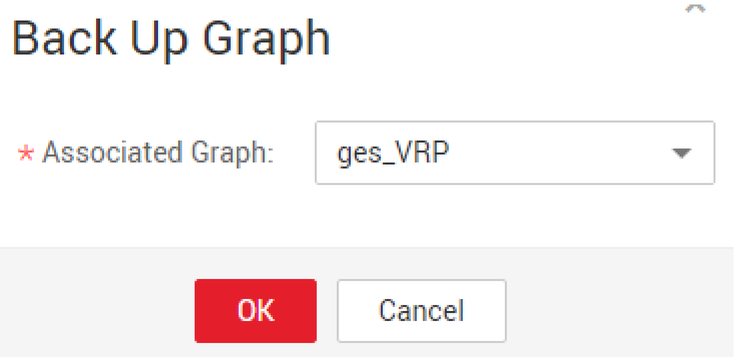
To ensure data security, back up the graph data so that you can restore it when faults occur.

Procedure

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

- For details about operations on the **Graph Management** page, see Backing Up a Graph on Graph Management.
- Operations on the **Backup Management** are as follows:
  a. Log in to the GES management console.
  b. In the navigation tree on the left, select **Backup Management**.
  c. In the upper right corner of the **Backup Management** page, click **Back Up Graph**.
  d. On the **Back Up Graph** page, select an **Associated Graph** (graph created by the current user) and click **OK** to start backup.

Figure 4-13 Backup on the Backup Management page



Back Up Graph

* Associated Graph: ges_VRP

NOTE

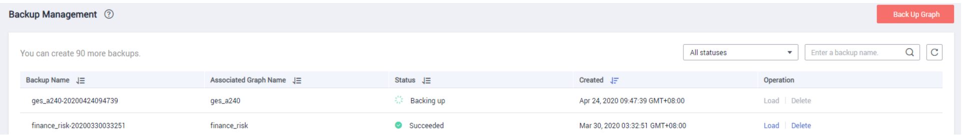
You can select an **Associated Graph** if you perform the backup operation on the **Backup Management** page. However, if the system has only one graph, you cannot change the value of **Associated Graph**.

Graph Engine Service User Guide

e. 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 4-14 Backup management



You can create 90 more backups.

Backup Name	Associated Graph Name	Status	Created	Operation
ges_a240-20200424094739	ges_a240	Backing up	Apr 24, 2020 09:47:39 GMT+08:00	Load | Delete
finance_risk-20200330033251	finance_risk	● Succeeded	Mar 30, 2020 03:32:51 GMT+08:00	Load | Delete

4.4.3 Loading a Graph

Scenario

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

Procedure

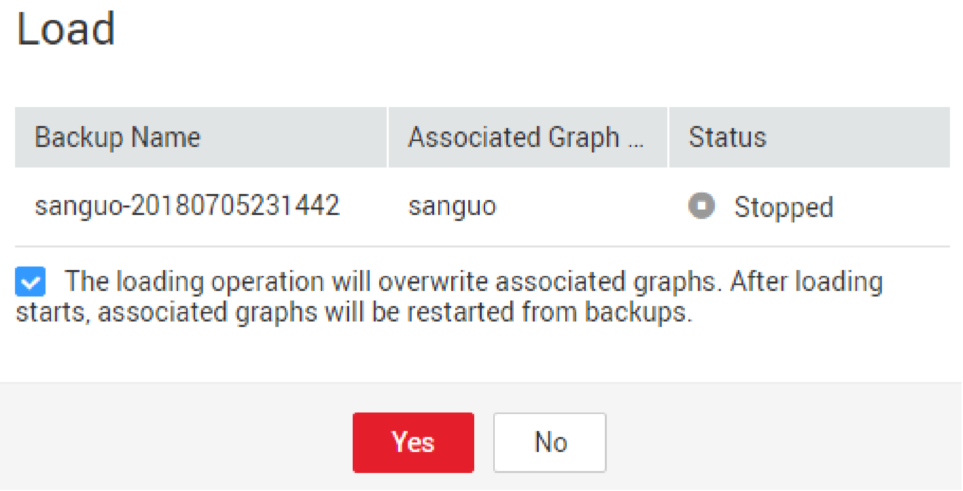
Step 1 Log in to the GES management console.

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

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

Step 4 On the **Load** dialog box, confirm the backup data, select **The loading operation will overwrite associated graphs. After loading starts, associated graphs will be restarted from backups**, and click **Yes**. After loading starts, the associated graph will be restarted from its backup. Click **Yes**.

Figure 4-15 Data loading



Backup Name	Associated Graph ...	Status
sanguo-20180705231442	sanguo	● Stopped

☑ The loading operation will overwrite associated graphs. After loading starts, associated graphs will be restarted from backups.

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

----End

4.4.4 Deleting a Backup

Scenario

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

Procedure

Step 1 Log in to the GES management console.

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

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

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

NOTE

The deleted data cannot be recovered. Exercise caution when performing this operation.

----End

4.5 Metadata Management

Metadata refers to the metadata in the graph data, specifying the graph data type.

On the **Metadata Management** page, you can view the storage path, status, encryption status, and creation time of a metadata file, and perform the following operations:

- **Importing a Metadata File**
- **Creating a Metadata File**
- **Searching for a Metadata File**
- **Editing a Metadata File**
- **Copying a Metadata File**
- **Deleting a Metadata File**

NOTE

The maximum number of metadata files that can be imported or created is 50.

Importing a Metadata File

If you already have metadata files, import them to GES for subsequent graph creation.

The procedure is as follows:

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. On the **Import** page, select **Local** or **OBS** in **Type** to import a metadata file form a local path or OBS.
   - Importing a metadata file from a local path

     Select Local File: Click **Upload** to select a local file.

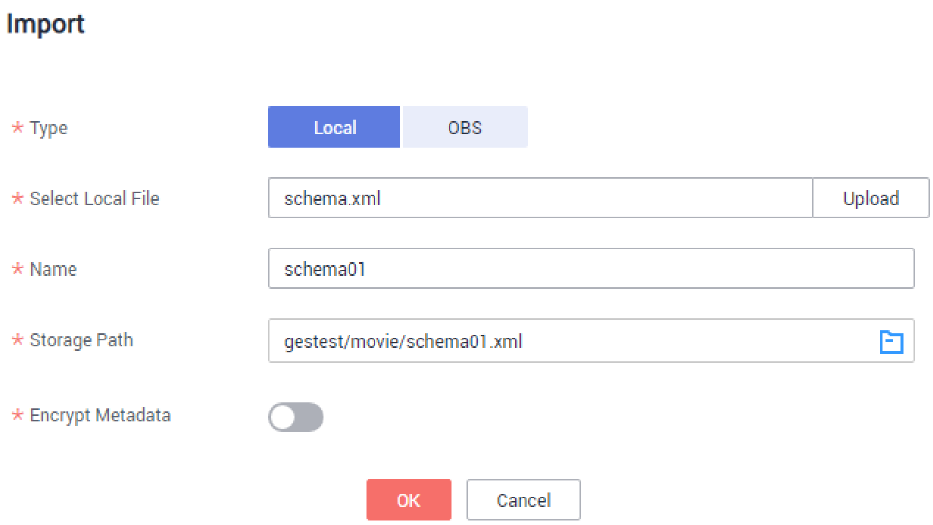
     NOTE

     The file must be in the XML format.

     Name: Enter the metadata file name.

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

     Figure 4-16 Importing metadata from a local path



Import

* Type: Local | OBS

* Select Local File: schema.xml Upload

* Name: schema01

* Storage Path: gestest/movie/schema01.xml

* Encrypt Metadata: ○

   - Importing a metadata file from OBS

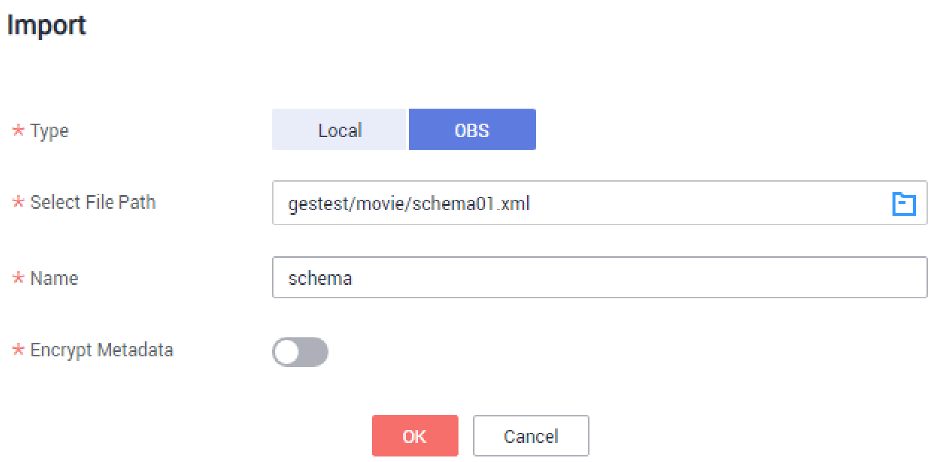
     Select File Path: Select a metadata file from OBS by clicking 🗀 .

     NOTE

     - The file must be in the XML format.
     - You need to upload the metadata file to your OBS bucket in advance.

     Name: Enter the metadata file name.

Figure 4-17 Importing data from OBS



Import

* Type: Local | OBS

* Select File Path: gestest/movie/schema01.xml

* Name: schema

* Encrypt Metadata

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

   After the import is complete, the metadata file is displayed on the **Metadata Management** page.

Creating a Metadata File

If no metadata file is available in the local path or on OBS, you can manually create metadata files.

The procedure is as follows:

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 to automatically create OBS buckets for storing the metadata.

   Encrypt Metadata: This parameter determines whether to encrypt the metadata. It is disabled by default. **Key Source** is default to **KMS**. **KMS Key**: Select the corresponding key.

   NOTE

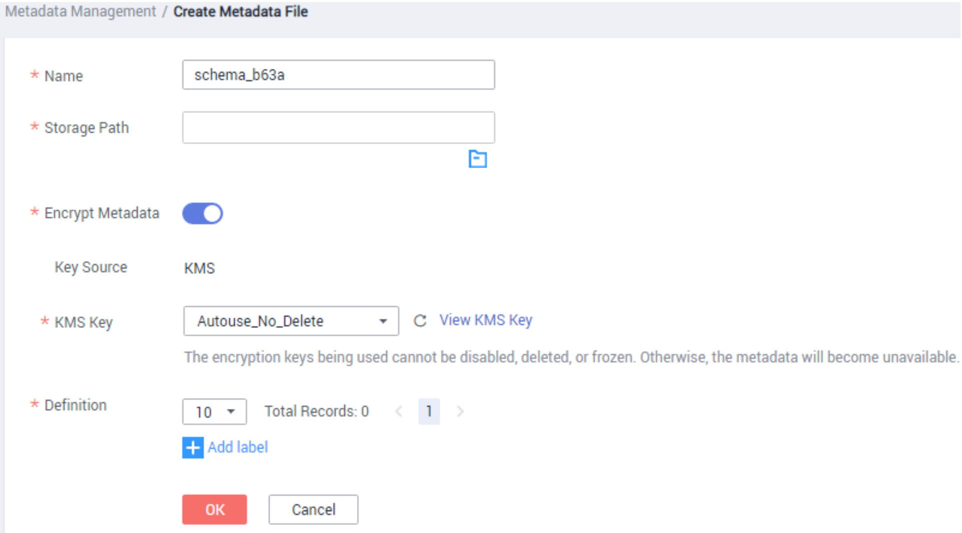
   Disabling or deleting a KMS key affects the instance functions.

   Definition: Detailed label definitions in the metadata file. Multiple labels can be defined in a metadata file. Click **Add label** to add labels as required.

   In **Definition**, specify the **Label Name** and add properties under a label. You can click **Add** to add properties and click **Up** and **Down** to sort the properties. **Table 4-5** lists the property parameters. For details about other metadata information, see section **Graph Data Formats**.

Graph Engine Service User Guide

Figure 4-18 Creating a metadata file



Metadata Management / **Create Metadata File**

* Name: schema_b63a

* Storage Path:

* Encrypt Metadata: ●

Key Source: KMS

* KMS Key: Autouse_No_Delete

The encryption keys being used cannot be disabled, deleted, or frozen. Otherwise, the metadata will become unavailable.

* Definition: 10 Total Records: 0 1

Table 4-5 Property parameters

Name	Description
Property Name	Indicates the name of a property. It contains 1 to 256 characters and cannot contain special characters such as angle brackets (<>) and ampersands (&).
Cardinality	Indicates the composite type of data. ● **Single value**: indicates that the data of this property has a single value, such as a digit or a character string. ● **Multiple values**: indicates that data of this property consists of multiple values separated by semicolons (;). You can determine whether to allow repetitive values.
Data Type	Indicates the data type of a property. Possible values are **char**, **float**, **double**, **bool**, **long**, **int**, **date**, **enum**, **string**, and **char array**. For details, see **Table 4-1**. **NOTE** Only the single-value type supports the **char array** data type.
Operation	Click **Remove** to delete unnecessary properties.

3. Click **OK**. After the metadata file is created, it will be displayed on the **Metadata Management** page.

Searching for a Metadata File

On the **Metadata Management** page, enter the name of a metadata file in the search box to search for it.

Editing a Metadata File

If the metadata file you imported or created does not meet service requirements and needs to be modified, you can modify its labels and properties online.

NOTE

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

The procedure is as follows:

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** at the bottom of the page.
   - Click **Edit** in the **Operation** column of the target metadata file.
2. On the **Edit** page, you can add labels or properties, change the label names, and sort properties by clicking **Up** and **Down**.
3. After the modification is complete, click **OK**.

Copying a Metadata File

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

The procedure is as follows:

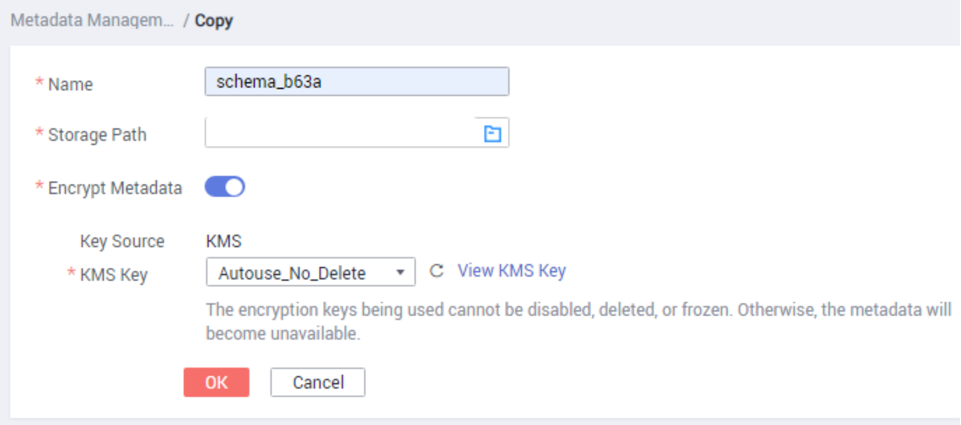
1. GES provides two methods for you to copy a metadata file on the **Data Management** page.
   - Click the metadata file name. On the metadata details page, click **Copy** at the bottom of the page.
   - Click **Copy** in the **Operation** column of the target metadata file.
2. Define the metadata file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

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

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

   Encrypt Metadata: This parameter determines whether to encrypt the copied metadata. It is disabled by default. **Key Source** is default to **KMS**. **KMS Key**: Select the corresponding key.

Figure 4-19 Copying a metadata file



Metadata Managem... / Copy

* Name: schema_b63a

* Storage Path:

* Encrypt Metadata: ●

Key Source: KMS

* KMS Key: Autouse_No_Delete

The encryption keys being used cannot be disabled, deleted, or frozen. Otherwise, the metadata will become unavailable.

3. Click **OK**.

   After the file is copied, the new metadata file will be displayed on the **Metadata Management** page.

Deleting a Metadata File

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

NOTE

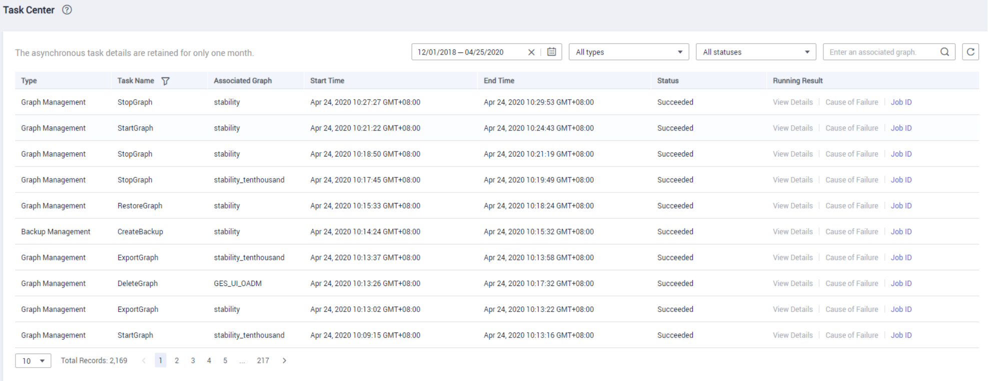
The deleted data cannot be recovered. Exercise caution when performing this operation.

4.6 Task Center

The task center displays details about asynchronous tasks, such as creating, backing up, starting, and deleting graphs.

You can view the information about a graph's tasks, including the task types, names, associated graphs, start time, end time, task statuses, and running results.

Figure 4-20 Task center



Task Center

The asynchronous task details are retained for only one month.

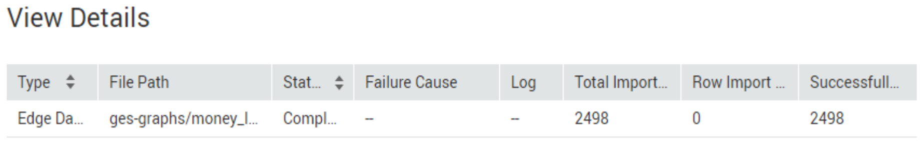
12/01/2018 — 04/25/2020 | All types | All statuses | Enter an associated graph.

Type	Task Name	Associated Graph	Start Time	End Time	Status	Running Result
Graph Management	StopGraph	stability	Apr 24, 2020 10:27:27 GMT+08:00	Apr 24, 2020 10:29:53 GMT+08:00	Succeeded	View Details | Cause of Failure | Job ID
Graph Management	StartGraph	stability	Apr 24, 2020 10:21:22 GMT+08:00	Apr 24, 2020 10:24:43 GMT+08:00	Succeeded	View Details | Cause of Failure | Job ID
Graph Management	StopGraph	stability	Apr 24, 2020 10:18:50 GMT+08:00	Apr 24, 2020 10:21:19 GMT+08:00	Succeeded	View Details | Cause of Failure | Job ID
Graph Management	StopGraph	stability_tenthousand	Apr 24, 2020 10:17:45 GMT+08:00	Apr 24, 2020 10:19:49 GMT+08:00	Succeeded	View Details | Cause of Failure | Job ID
Graph Management	RestoreGraph	stability	Apr 24, 2020 10:15:33 GMT+08:00	Apr 24, 2020 10:18:24 GMT+08:00	Succeeded	View Details | Cause of Failure | Job ID
Backup Management	CreateBackup	stability	Apr 24, 2020 10:14:24 GMT+08:00	Apr 24, 2020 10:15:32 GMT+08:00	Succeeded	View Details | Cause of Failure | Job ID
Graph Management	ExportGraph	stability_tenthousand	Apr 24, 2020 10:13:37 GMT+08:00	Apr 24, 2020 10:13:58 GMT+08:00	Succeeded	View Details | Cause of Failure | Job ID
Graph Management	DeleteGraph	GES_UI_OADM	Apr 24, 2020 10:13:26 GMT+08:00	Apr 24, 2020 10:17:32 GMT+08:00	Succeeded	View Details | Cause of Failure | Job ID
Graph Management	ExportGraph	stability	Apr 24, 2020 10:13:02 GMT+08:00	Apr 24, 2020 10:13:22 GMT+08:00	Succeeded	View Details | Cause of Failure | Job ID
Graph Management	StartGraph	stability_tenthousand	Apr 24, 2020 10:09:15 GMT+08:00	Apr 24, 2020 10:13:16 GMT+08:00	Succeeded	View Details | Cause of Failure | Job ID

10 | Total Records: 2,169 | < 1 2 3 4 5 ... 217 >

- You can click **View Details**, **Cause of Failure**, and **Job ID** in the **Running Result** column to view corresponding information.

Figure 4-21 Viewing details

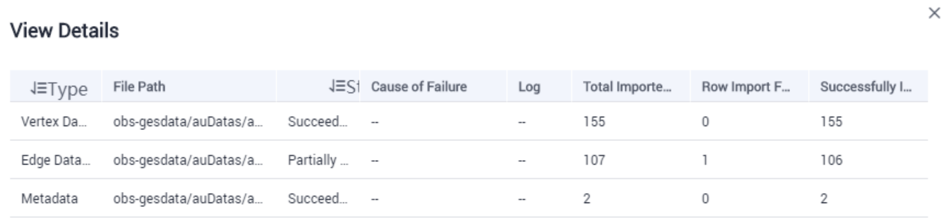


View Details

Type	File Path	Stat...	Failure Cause	Log	Total Import...	Row Import ...	Successfull...
Edge Da...	ges-graphs/money_l...	Compl...	--	--	2498	0	2498

If the task status is **Partially successful** when you import a graph, 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 4-22 Partially successful



View Details

Type	File Path	S	Cause of Failure	Log	Total Importe...	Row Import F...	Successfully I...
Vertex Da...	obs-gesdata/auDatas/a...	Succeed...	--	--	155	0	155
Edge Data...	obs-gesdata/auDatas/a...	Partially ...	--	--	107	1	106
Metadata	obs-gesdata/auDatas/a...	Succeed...	--	--	2	0	2

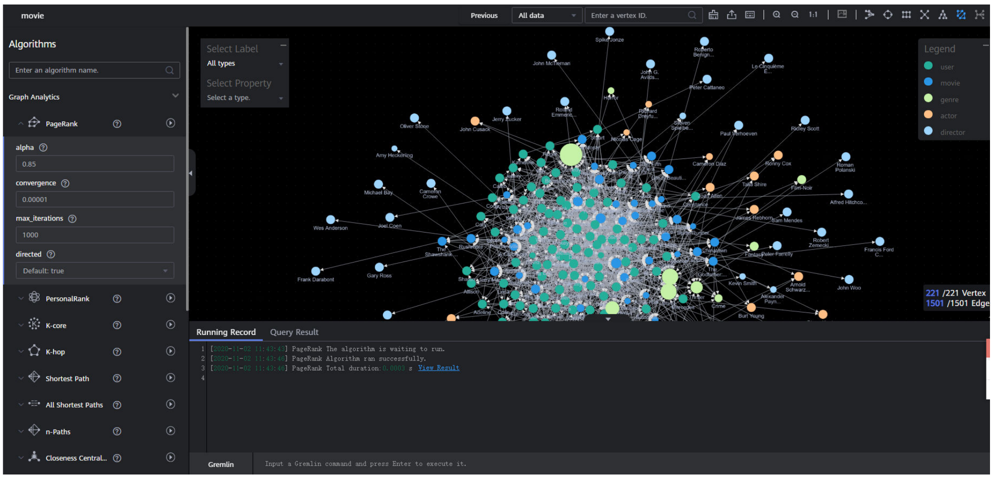
- Use any of the following methods to search tasks:
  - Setting the time
  - Selecting the task type
  - Selecting the task status
  - Entering an associated graph
  - Filtering the task name in the task list

4.7 Sandbox Access

You can use the sandbox access function to access a demo graph provided by the system to learn and use the functions of the graph editor.

In the navigation pane of the GES management console, click **Access Sandbox**. You can learn the page functions through **Quick Start**. The system runs the PageRank algorithm by default.

Figure 4-23 Sandbox access



alpha: 0.85

convergence: 0.00001

max_iterations: 1000

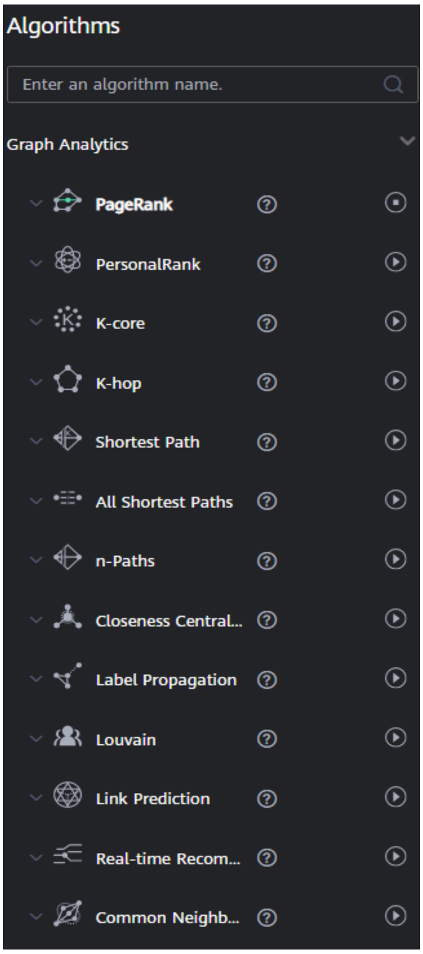
directed: Default: true

```
[2020-11-02 11:43:43] PageRank The algorithm is waiting to run.
[2020-11-02 11:43:46] PageRank Algorithm ran successfully.
[2020-11-02 11:43:46] PageRank Total duration:0.0003 s View Result
```

Algorithm Library

The algorithm library lists all algorithms supported by GES. You can set the properties of each algorithm in this area. **Table 5-2** describes the functions of the algorithm area.

Figure 4-24 Algorithm library



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. For details, see **Gremlin Query**.

Figure 4-25 Gremlin query

Gremlin

Input a Gremlin command and press Enter to execute it.

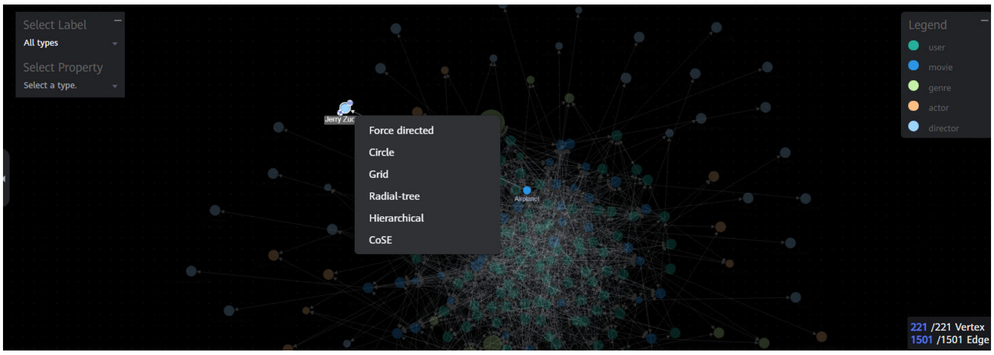
NOTE

Gremlin in the sandbox experience environment does not support write and cyclic operations, such as **repeat()**, **times()**, **until()**, **emit()**, and **loops()**.

Viewing Vertex Properties

On the canvas, select a vertex or an edge, right-click, and select **View Property** to view the property information about the selected vertex or edge on the **Property** tab page.

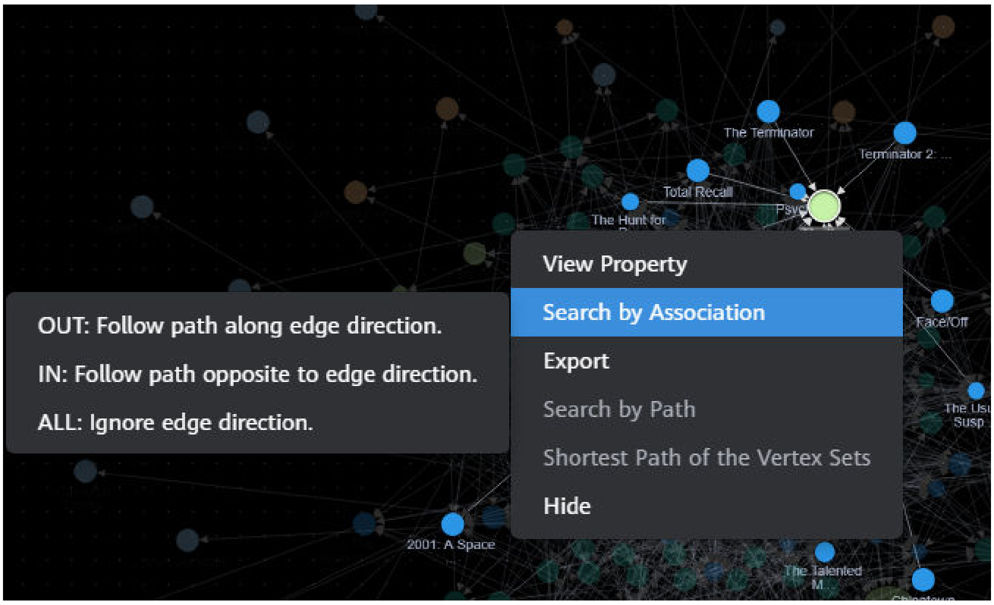
Figure 4-26 Viewing vertex properties



Search by Association

On the canvas, select a vertex, right-click, and select **Search by Association** to find vertices associated with the selected vertex. You can select **OUT**, **IN**, and **ALL**.

Figure 4-27 Search by Association



OUT: Follow path along edge direction.

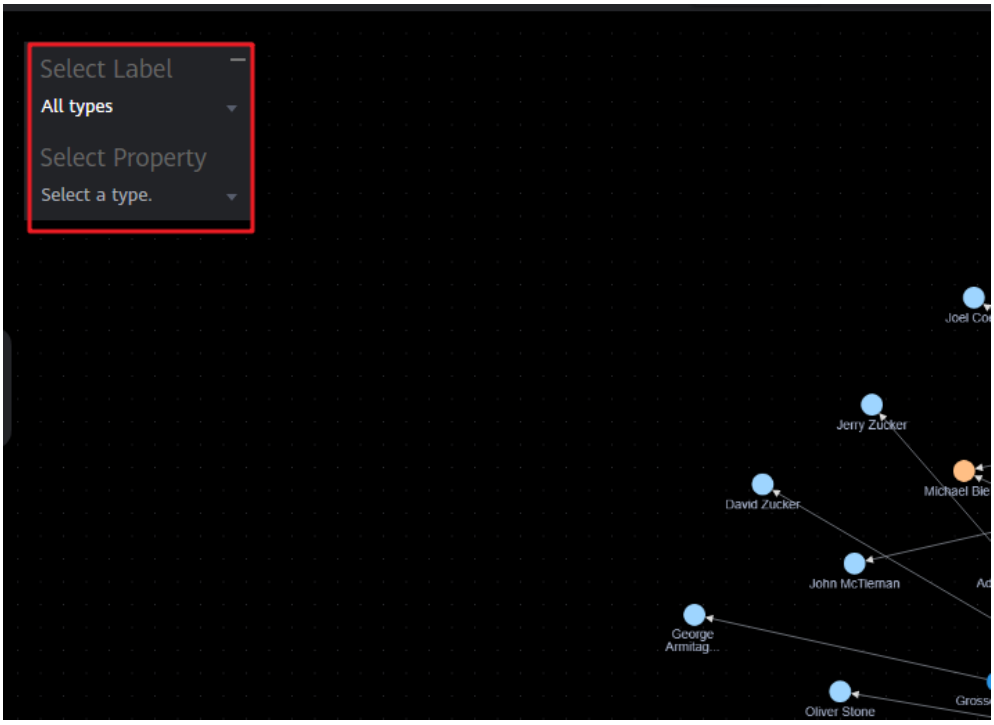
IN: Follow path opposite to edge direction.

ALL: Ignore edge direction.

Filtering Labels and Properties

You can select a label and its properties to filter and display the required graph information.

Figure 4-28 Filtering labels and properties



Select Label
All types

Select Property
Select a type.

5 Introduction to the Graph Editor

5.1 Accessing the GES Graph Editor

Scenario

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

Procedure

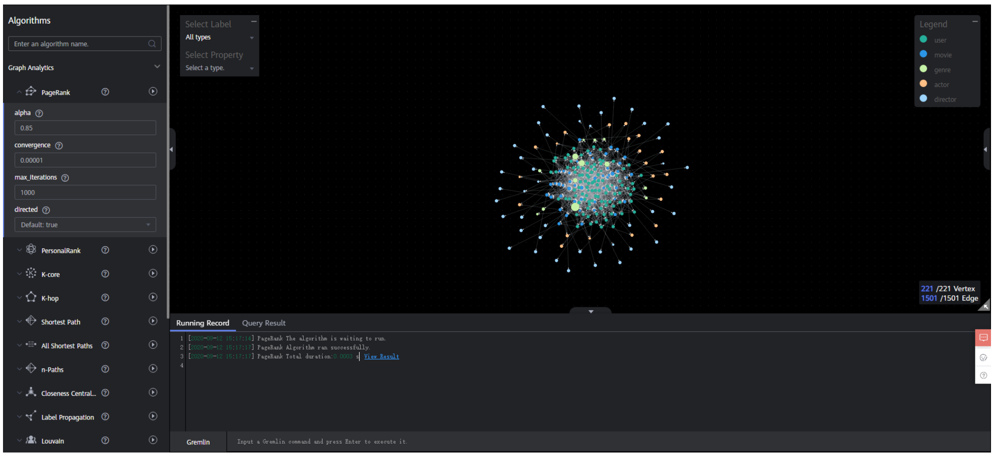
Step 1 Log in to the GES management console and select **Graph Management**.

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

#ges_01_0022/fig1567410497529 shows the graph editor page. You can analyze the graph data on the graph editor. For details, see **Querying and Analyzing Graphs**.

Graph Engine Service User Guide

Figure 5-1 Graph editor



alpha: 0.85

convergence: 0.00001

max_iterations: 1000

directed: Default: true

221 /221 Vertex

1501 /1501 Edge

```
[2020-09-12 15:17:14] PageRank The algorithm is waiting to run.
[2020-09-12 15:17:17] PageRank Algorithm ran successfully.
[2020-09-12 15:17:17] PageRank Total duration:0.0003 s View Result
```

Input a Gremlin command and press Enter to execute it.

----End

5.2 Graph Editor Overview

The graph editor consists of the algorithm library, Gremlin, canvas, result display, and filtering and property areas.

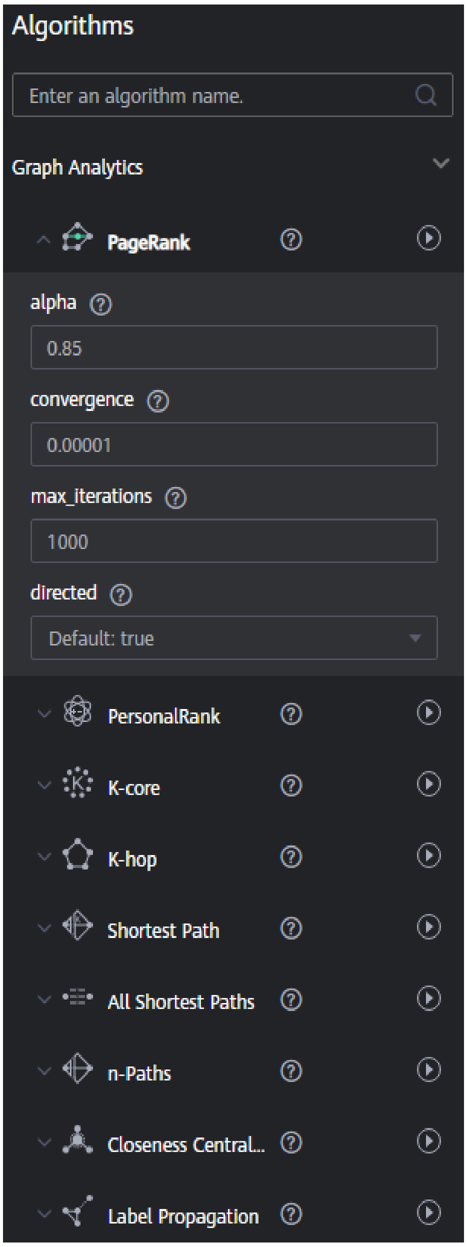
Table 5-1 Area description

Area	Description
Algorithm Library	Lists all algorithms supported by GES. You can set the properties of each algorithm in this area. **Table 5-2** 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.
Gremlin	Allows you to enter Gremlin query statements to query a graph.
Canvas	Displays the graph data in a visualized manner. Shortcut operations are preset in the drawing area for you to easily analyze the graph data. **Table 5-3** describes the functions of the drawing area.
Result Display	Contains the following two tab pages: ● **Running Record**: For details, see **View Running Records**. ● **Query Result**: For details, see **Viewing Query Results**.

Graph Engine Service User Guide

Area	Description
Filtering and Property	On the canvas, select a vertex and right-click it. Then, choose **View Property** from the shortcut menu to display the area. It contains the following two tab pages: ● The **Filtering** tab page allows you to set properties and conditions to filter the data for analysis. For details, see **Filtering Conditions**. ● The **Property** tab page displays the property information about a vertex or an edge.

Figure 5-2 Algorithm Library



Algorithms

Enter an algorithm name.

Graph Analytics

PageRank

alpha: 0.85

convergence: 0.00001

max_iterations: 1000

directed: Default: true

Table 5-2 Algorithm library description

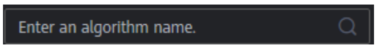
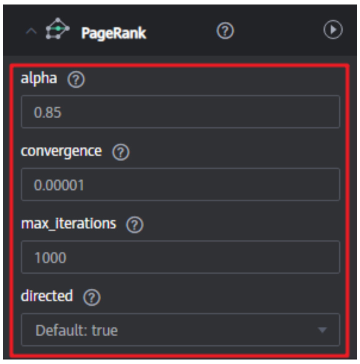
Interface Element	Description
 Enter an algorithm name.	Enter the algorithm name to quickly find it.
⌄	Expand the algorithm parameter configuration area.
⏵	Run the algorithm.
 PageRank alpha 0.85 convergence 0.00001 max_iterations 1000 directed Default: true	Set the properties of an algorithm. Different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

Figure 5-3 Canvas

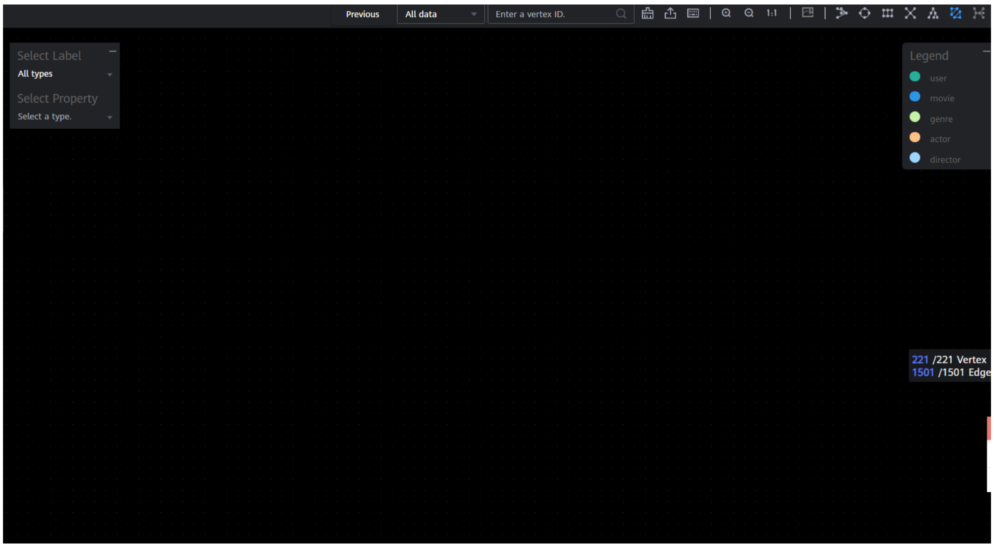
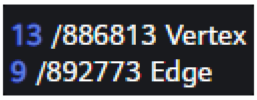
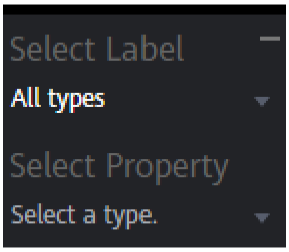
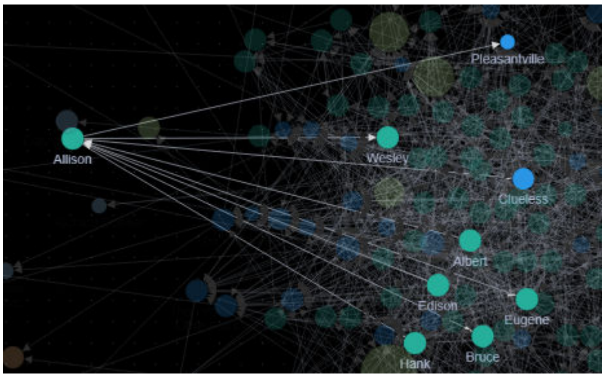


Table 5-3 Canvas description

Interface Element	Description
 13 /886813 Vertex 9 /892773 Edge	Row 1: **13** indicates the number of vertices displayed on the current canvas and **886813** indicates the total number of vertices in the entire graph. Row 2: **9** indicates the number of edges displayed on the current canvas and **892773** indicates the total number of edges in the entire graph.
 Select Label All types Select Property Select a type.	● Select a label as required. ● Select the label's property to be displayed.
 Previous	Cancel the previous operation.
 All data	Select **All data** or **Current data**. ● **All data** indicates all data of a graph. ● **Current data** indicates the data rendered on the canvas.
 Enter a vertex ID.	After you select **All data** or **Current data**, enter the node ID in the search box, for example, **2**. Press **Enter** or click the query icon to search for the corresponding vertex and render it to the canvas. **NOTE** Currently, only a single vertex ID can be entered.
🧹	Click **Clear** to clear all content on the canvas.
⇪	Export the canvas content as a PNG or CSV file (snapshot or vertex and edge file of the current canvas).

Interface Element	Description
⌨	Keyboard shortcuts - Ctrl+S: Save the canvas. - Ctrl+E: Clear all content on the canvas. - Ctrl++: Zoom in. - Ctrl+-: Zoom out. - Ctrl+Z: Cancel the undo operation. - Ctrl+A: Select all content on the canvas. - del: Delete the selected vertex from the canvas. - Ctrl+Click: Select multiple vertices and edges.
⊕	Zoom in the graph. You can zoom in a graph to at most 600%.
⊖	Zoom out the graph. You can zoom out a graph to 5%.
1:1	Automatic screen adaptation When the displayed graph data is too large (cannot be completely displayed) or too small, you can click this button to quickly adjust it based on the screen size.
🖼	Whether to display legends
	Quick layout switchover. From left to right: **Force directed**, **Circle**, **Grid**, **Radial-tree**, **Hierarchical**, **CoSE**, and **Double-core**. The layouts are shown in #ges_01_0023/fig651684751424, #ges_01_0023/fig0780817165314, #ges_01_0023/fig87011829185314, #ges_01_0023/fig209767427538, #ges_01_0023/fig1659214578534, #ges_01_0023/fig66501517135419, and #ges_01_0023/fig1192873075410. **NOTE** The **Double-core** takes effect only when two nodes are selected.

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

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

Figure 5-4 Force directed

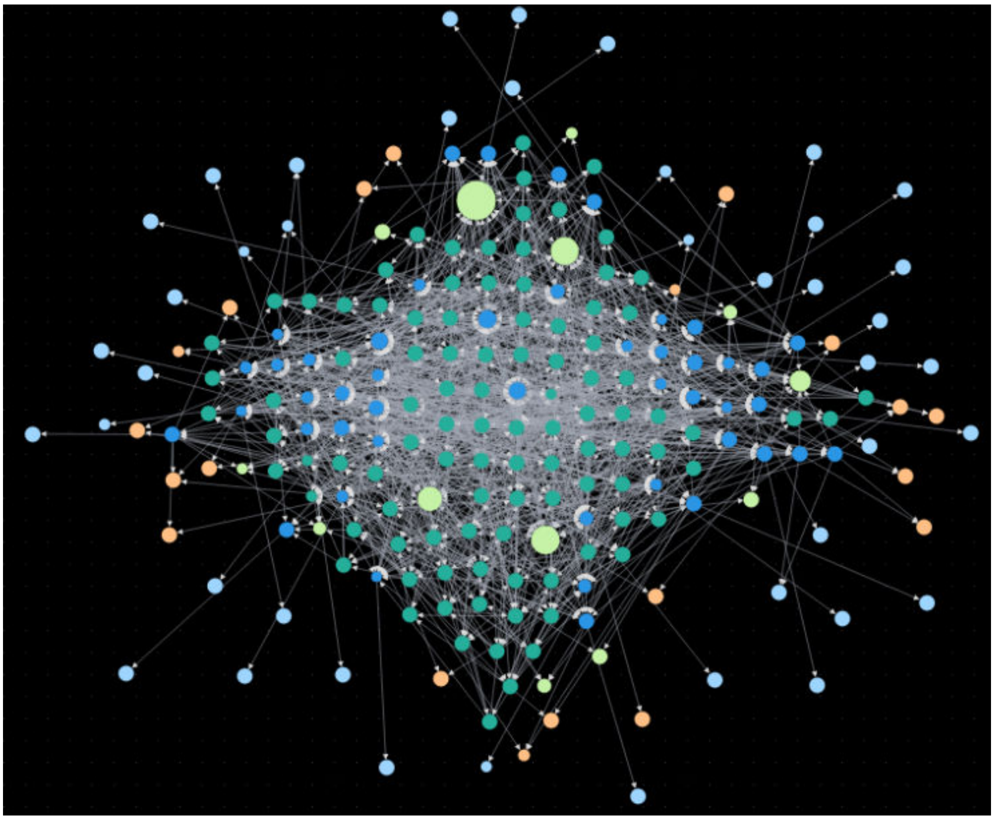


Figure 5-5 Circle

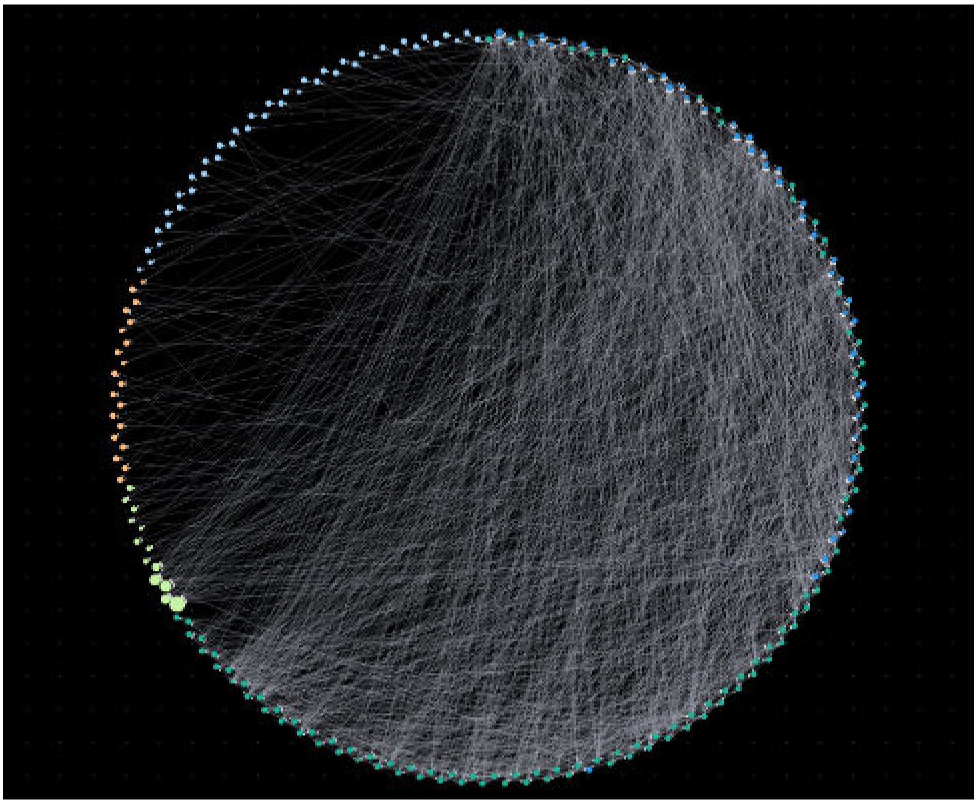
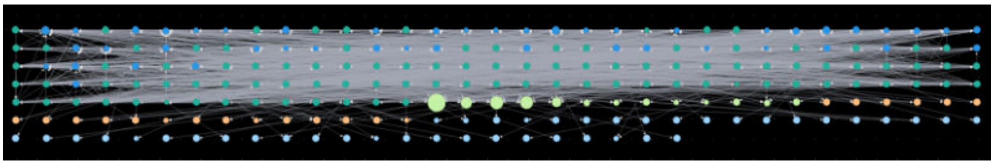


Figure 5-6 Grid



Graph Engine Service User Guide

Figure 5-7 Radial-tree

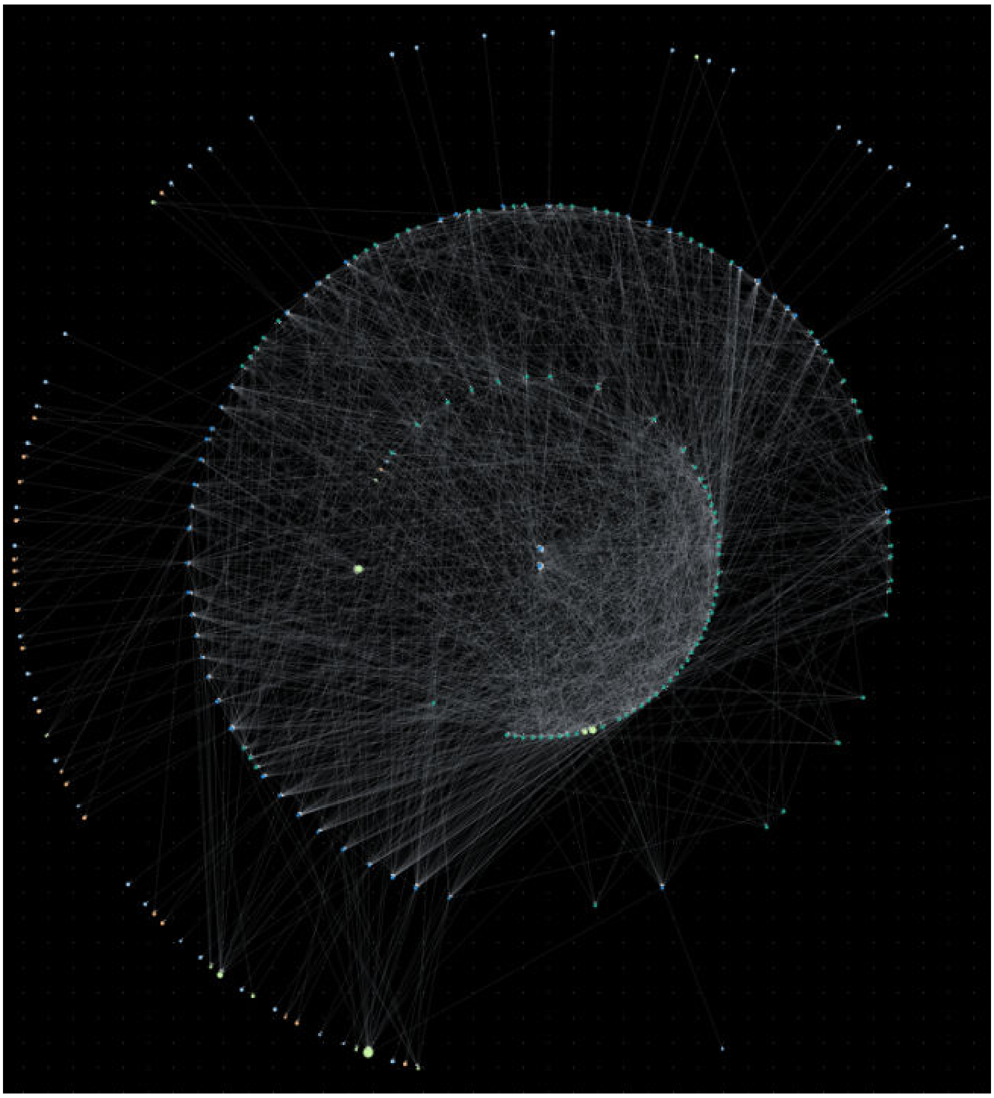


Figure 5-8 Hierarchical

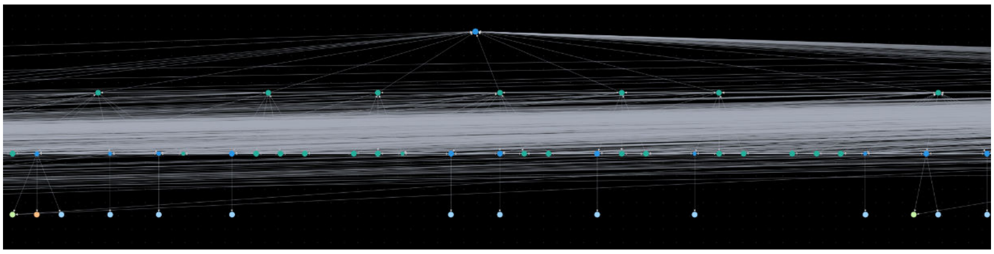


Figure 5-9 CoSE

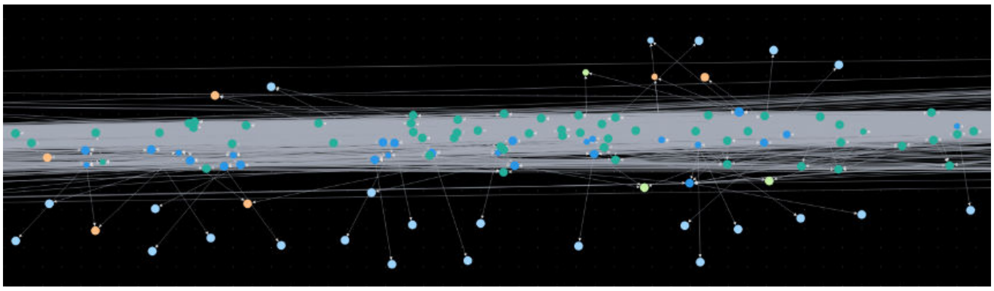
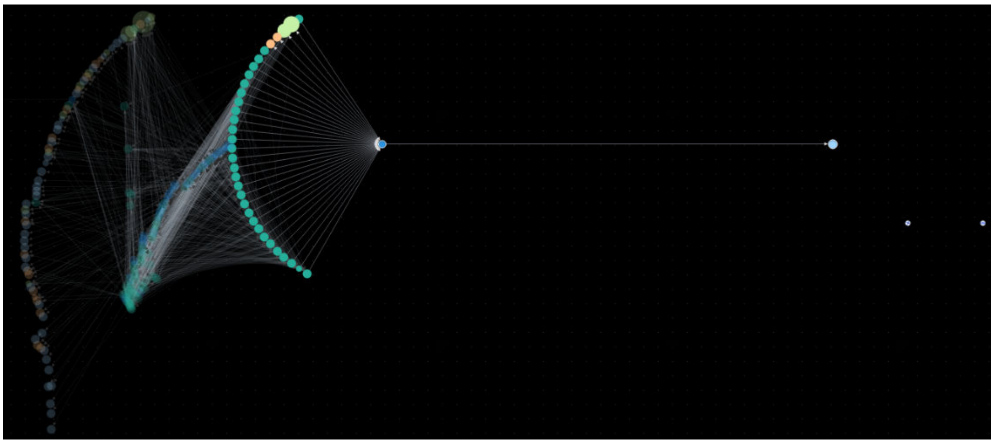


Figure 5-10 Double-core



5.3 Gremlin Query

Scenario

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

Procedure

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

Step 2 In the Gremlin text box, enter the query commands and press **Enter** to run the commands.

Common query commands are as follows:

- Vertex query

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

  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

The **g.V ()** syntax is not recommended because it affects the display effect if the vertex scale is large.

- Edge query

  g.E(): This command is used to query all edges. You are not advised to use this command because you need to add filter criteria or limit the returned results when the edge volume is too large.

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

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

  g.V('46').outE('rate'): This command is used to query the edge whose ID is **48** and all labels are **rate**.

- Property query

  g.V().limit(3).valueMap(): This command is used to query all properties of a vertex. (This is an optional parameter and only one vertex is queried. All properties of the vertex are 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. (This parameter can be left blank and all properties are queried. Each property is displayed in one row, containing only the value).

- Adding a vertex
  - Method 1:

    a = graph.addVertex(label,'user',id,'500','age','18-24'): This command is used to add a vertex whose label is **user**, ID is **500**, and age is **18** to **24**.
  - Method 2:

    g.addV('user').property(id,'600').property('age','18-24'): This command is used to add a vertex whose label is **user**, ID is **500**, and age is **18** to **24**.

- Deleting a vertex

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

- Adding an edge
  - Method 1:

    a = graph.addVertex(label,'user',id,'501','age','18-24');

    b = graph.addVertex(label,'movie',id,'502','title','love');

    a.addEdge('rate',b,'Rating','4'): This command is used to add an edge. IDs of the edge's two endpoints are **501** and **502**.
  - Method 2:

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

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

    g.addE('rate').property('Rating', '4').from(a).to(b): This command is used to add an edge. IDs of the edge's two endpoints are **501** and **502**.

- Deleting an edge

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

----End

Related Information

Table 5-4 shows the differences between the Gremlin in GES and that in the open source community.

Table 5-4 Differences

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

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

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

NOTE

Currently, the following step commands are not supported:

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

5.4 Analyzing Graphs Using Algorithms

Scenario

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

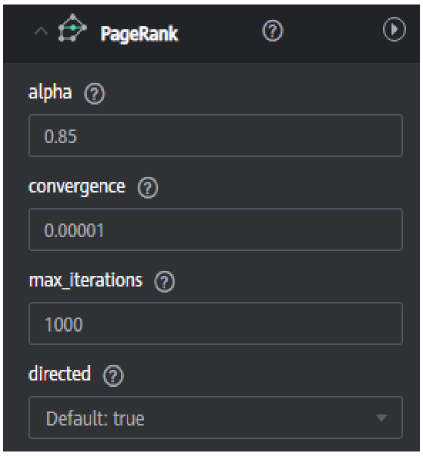
Procedure

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

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

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

Figure 5-11 Setting algorithm parameters



PageRank

alpha: 0.85

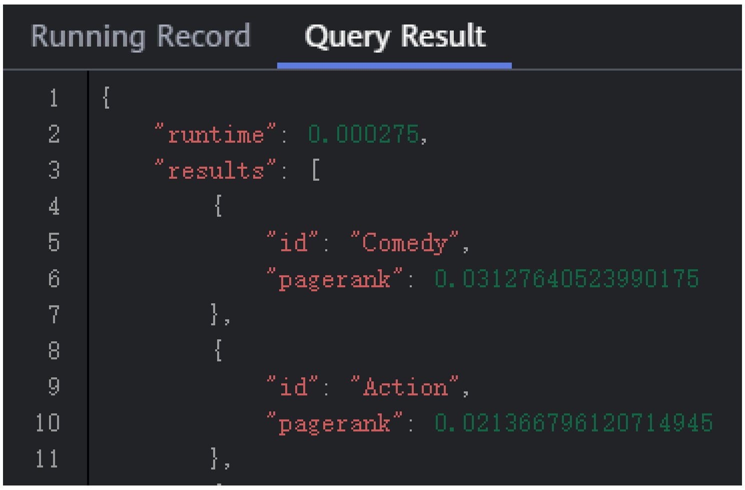
convergence: 0.00001

max_iterations: 1000

directed: Default: true

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

Figure 5-12 Viewing the analysis result

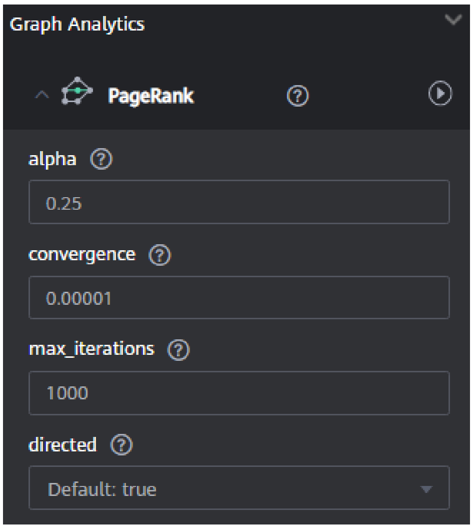


```
{
    "runtime": 0.000275,
    "results": [
        {
            "id": "Comedy",
            "pagerank": 0.03127640523990175
        },
        {
            "id": "Action",
            "pagerank": 0.021366796120714945
        },
```

Step 4 Take the movie data in the template as an example. After the parameters are adjusted, the PageRank value changes, but the top rank is nearly the same.

The two most influential movies, that is, the ID Lethal Weapon and ID Jaws, are selected.

Figure 5-13 Adjusting parameters



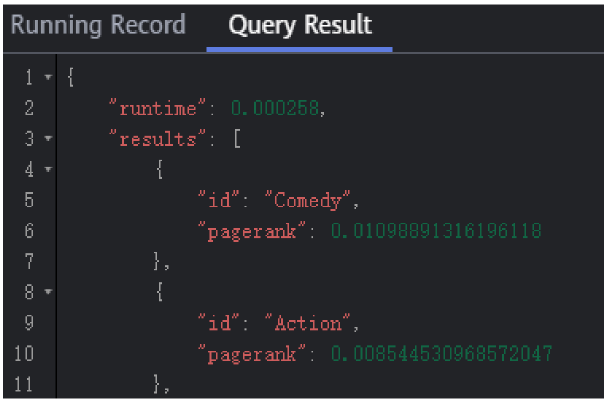
alpha: 0.25

convergence: 0.00001

max_iterations: 1000

directed: Default: true

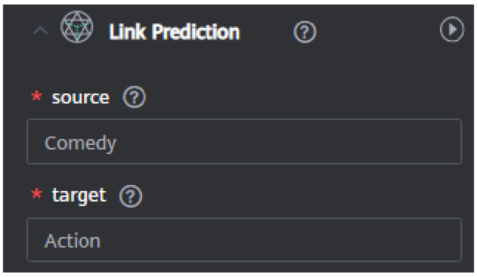
Figure 5-14 Query result after the parameters are adjusted



```
{
    "runtime": 0.000258,
    "results": [
        {
            "id": "Comedy",
            "pagerank": 0.010988913161961 18
        },
        {
            "id": "Action",
            "pagerank": 0.008544530968572047
        },
```

Step 5 Run the Link Prediction algorithm to analyze the association degree of the two movies. The degree is 0.35, indicating that many people have watched both the two movies.

Figure 5-15 Association analysis



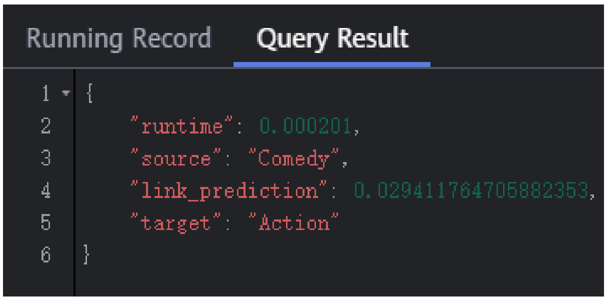
* source

Comedy

* target

Action

Figure 5-16 Association analysis result



```
{
    "runtime": 0.000201,
    "source": "Comedy",
    "link_prediction": 0.029411764705882353,
    "target": "Action"
}
```

----End

5.5 Analyzing Graphs on the Canvas

Scenario

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

For details about the shortcut keys and interface elements on the canvas, see **Table 5-3**.

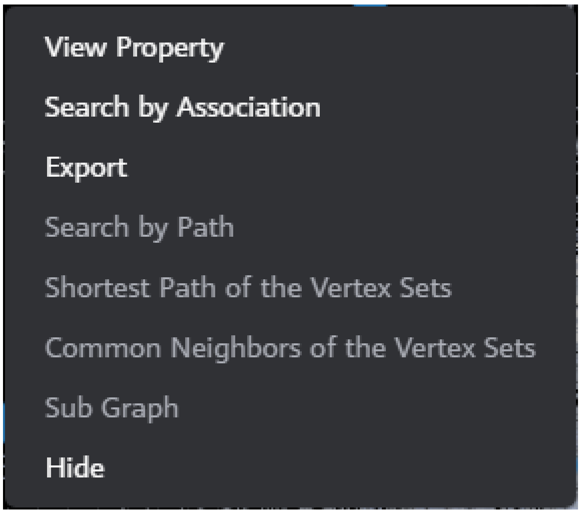
Graph Engine Service User Guide

Procedure

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

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

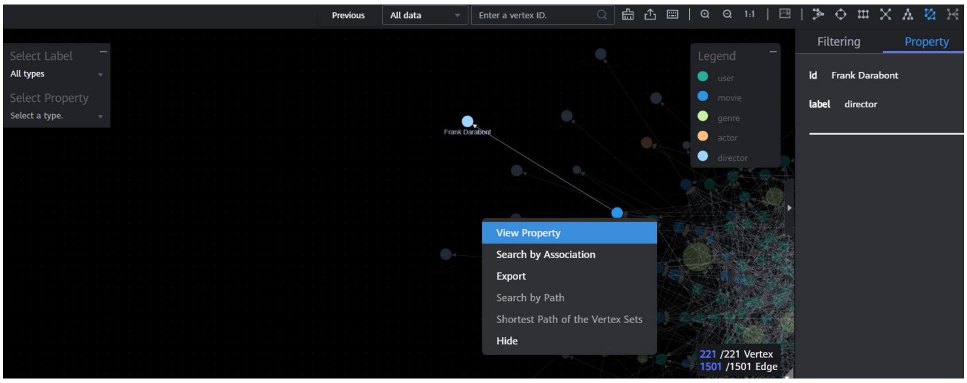
Figure 5-17 Shortcut menu


View Property
Search by Association
Export
Search by Path
Shortest Path of the Vertex Sets
Common Neighbors of the Vertex Sets
Sub Graph
Hide

- **View Property**

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

  Figure 5-18 View Property

  Id Frank Darabont

  label director

- **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.

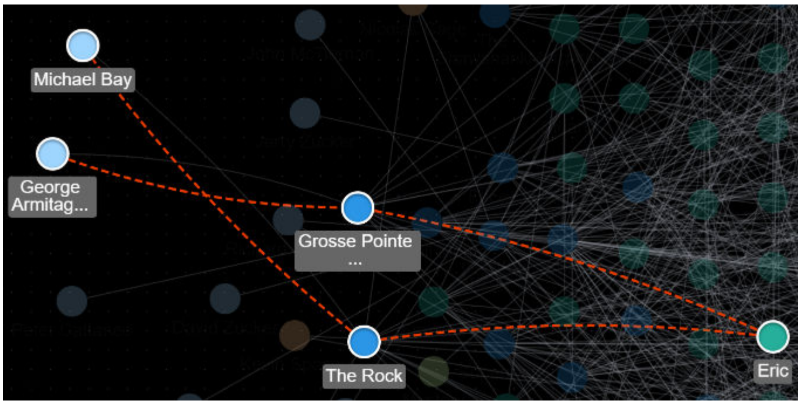
  Procedure: Hold down **Ctrl** and click two vertices. The first is the target vertex and the second is the source vertex. Then, Right-click and choose **Search by Path** from the shortcut menu.

  NOTE

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

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

  Figure 5-19 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

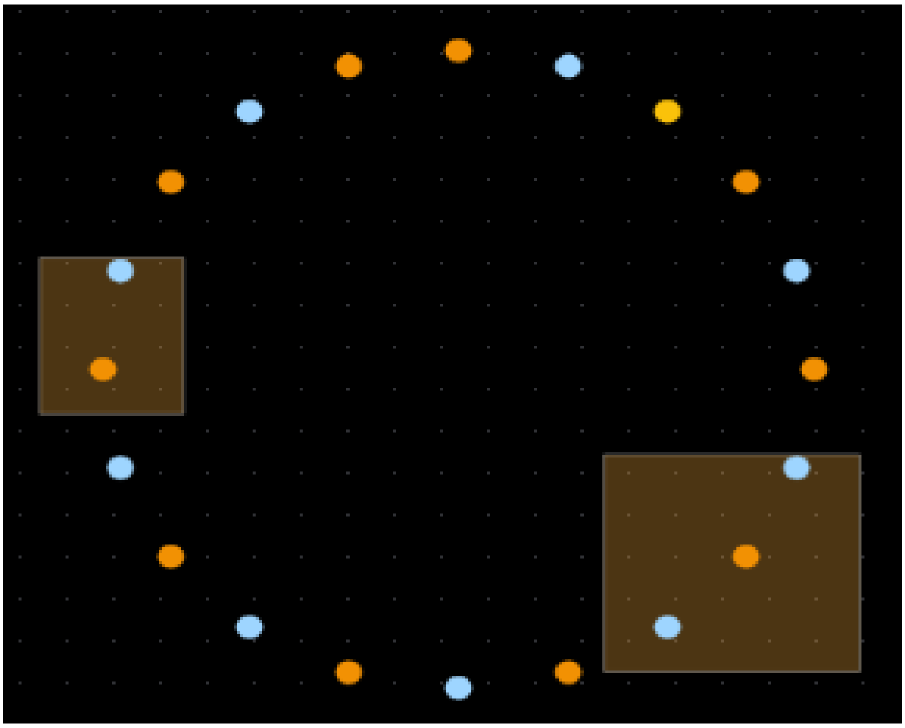
    By box-selecting the common neighbors of two vertex sets, you can intuitively discover the objects associated with the two sets.

  – Procedure

    i. Hold down **Shift** and box-select two vertex sets.

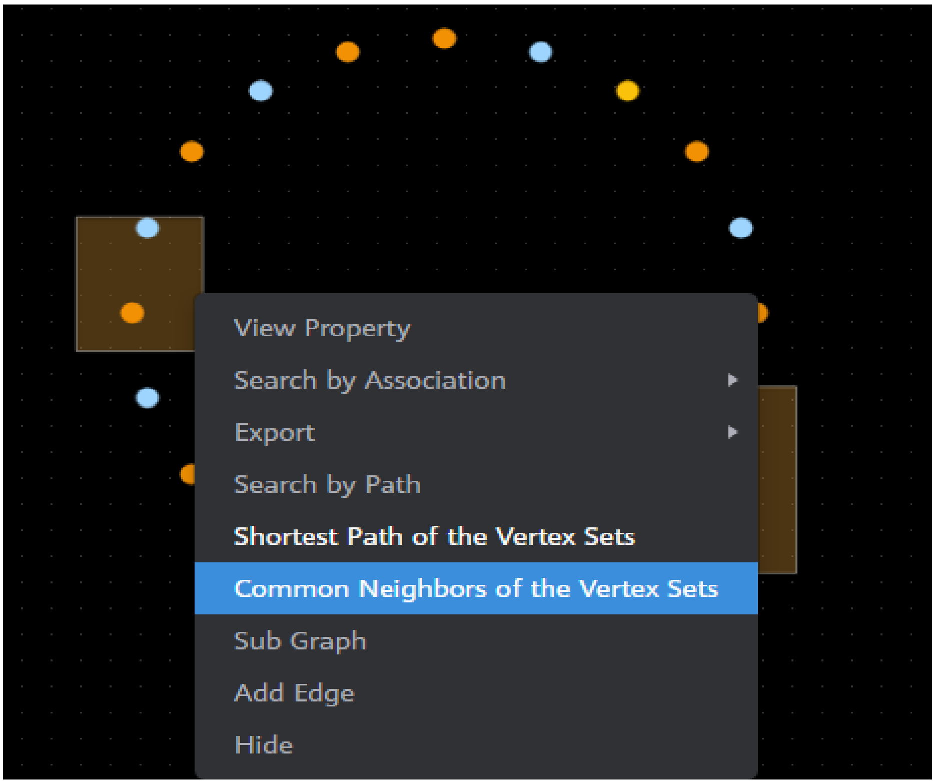
Graph Engine Service User Guide

Figure 5-20 Box-selecting vertex sets



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

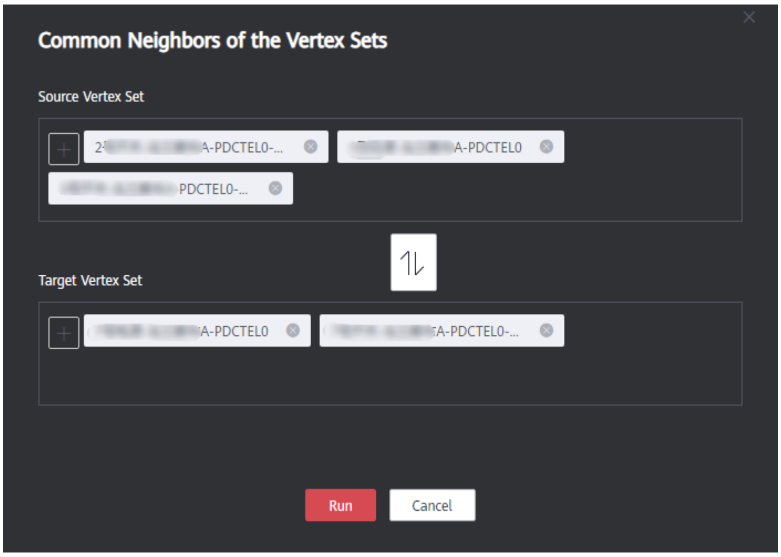
Figure 5-21 Common Neighbors of Vertex Sets



iii. In the dialog box that is displayed, confirm the vertices in the vertex sets. You can add or delete vertices as required, and then click **Run**.

Graph Engine Service User Guide

Figure 5-22 Confirming the vertices in the vertex sets



Common Neighbors of the Vertex Sets

Source Vertex Set

2[redacted]A-PDCTEL0-...
[redacted]A-PDCTEL0
[redacted]PDCTEL0-...

Target Vertex Set

[redacted]A-PDCTEL0
[redacted]A-PDCTEL0-...

iv. Display the result.

Figure 5-23 Graph Display

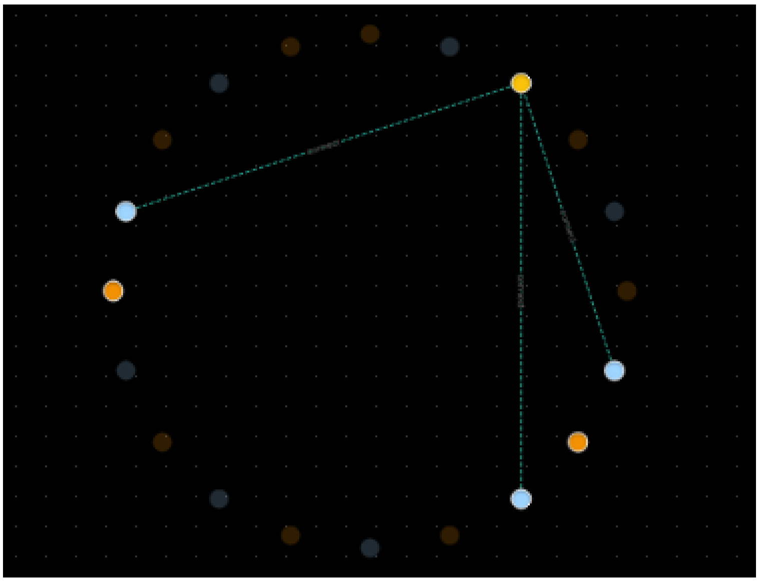
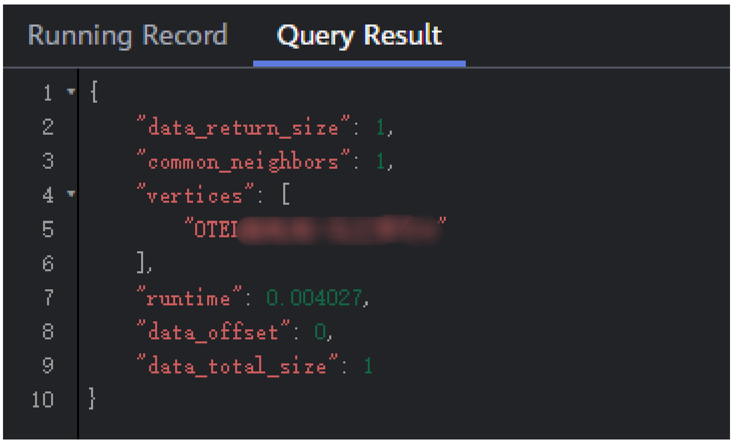


Figure 5-24 Query Result



```
{
    "data_return_size": 1,
    "common_neighbors": 1,
    "vertices": [
        "OTEI[redacted]"
    ],
    "runtime": 0.004027,
    "data_offset": 0,
    "data_total_size": 1
}
```

- **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.

----End

5.6 Filtering Conditions

Scenario

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

Procedure

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

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

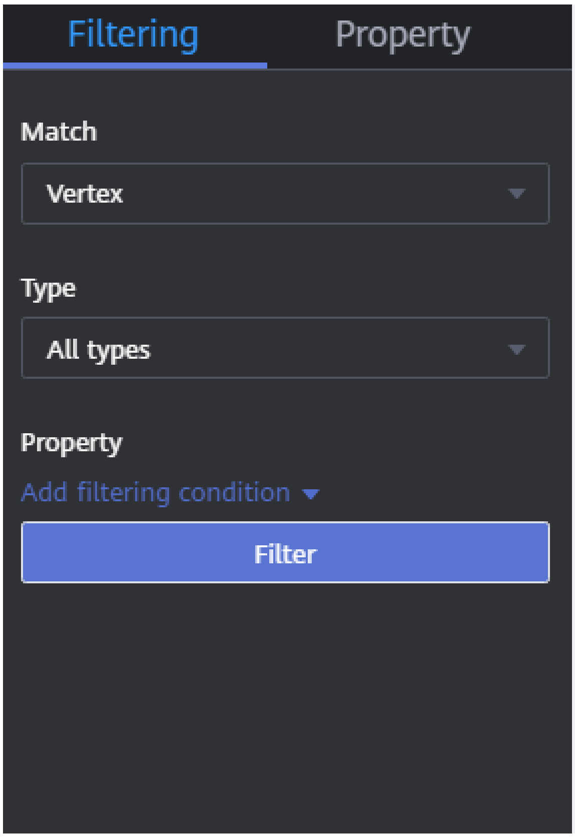
Step 3 In the **Filtering and Property** area, set the filtering conditions and click **Filter**.

- **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**

Graph Engine Service User Guide

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 5-25 Setting filtering conditions



Match

Vertex

Type

All types

Property

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

----End

5.7 View Running Records

Scenario

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

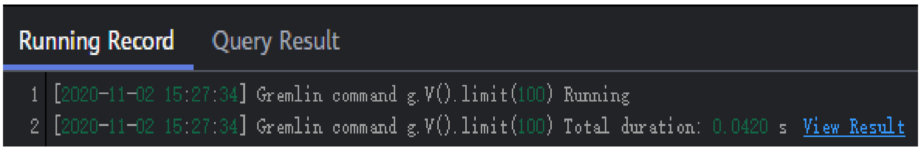
Procedure

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

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

- After a Gremlin command is executed, the local time, and the command name and content are displayed in the **Running Record** tab page.

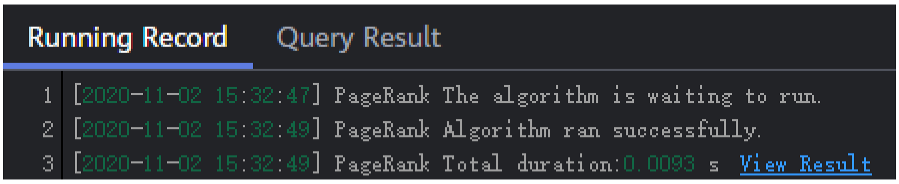
Figure 5-26 Gremlin running record



```
[2020-11-02 15:27:34] Gremlin command g.V().limit(100) Running
[2020-11-02 15:27:34] Gremlin command g.V().limit(100) Total duration: 0.0420 s View Result
```

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

Figure 5-27 Algorithm running record



```
[2020-11-02 15:32:47] PageRank The algorithm is waiting to run.
[2020-11-02 15:32:49] PageRank Algorithm ran successfully.
[2020-11-02 15:32:49] PageRank Total duration:0.0093 s View Result
```

----End

5.8 Viewing Query Results

Scenario

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

Procedure

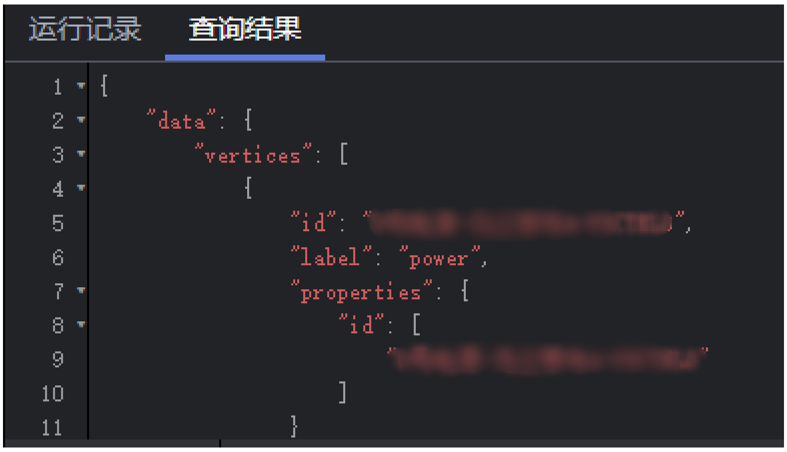
Step 1 Log in to the GES graph editor. For details, see **Accessing the GES Graph Editor**.

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

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

- Run a Gremlin command. The command output is quickly displayed. For example, if you run the **g.V().limit(100)** command, the result is as follows:

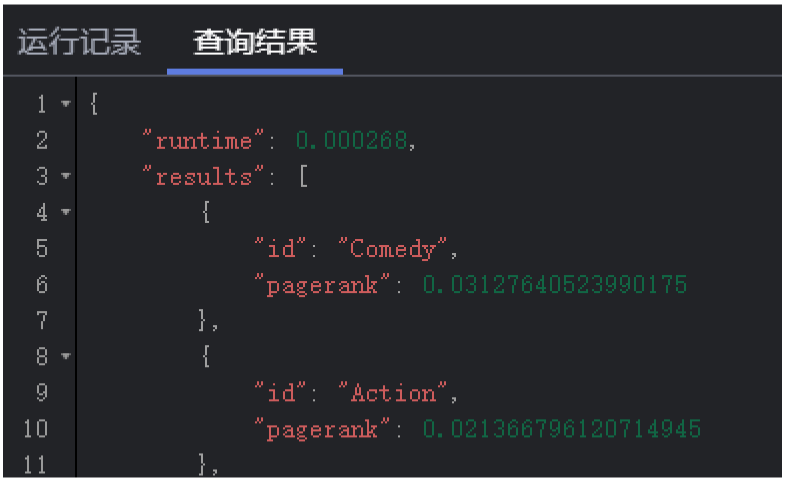
Figure 5-28 Gremlin output



```
{
    "data": {
        "vertices": [
            {
                "id": "[redacted]",
                "label": "power",
                "properties": {
                    "id": [
                        "[redacted]"
                    ]
                }
```

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

Figure 5-29 Algorithm output



```
{
    "runtime": 0.000268,
    "results": [
        {
            "id": "Comedy",
            "pagerank": 0.03127640523990175
        },
        {
            "id": "Action",
            "pagerank": 0.021366796120714945
        },
```

----End

6 Algorithms

6.1 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 6-1 Algorithm List

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

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

Algorithm	Description
Node2vec	By invoking the Word2vec algorithm, the Node2vec algorithm maps nodes in the network to the Euclidean space, and uses vectors to represent the node characteristics. The Node2vec algorithm generates random steps from each node using the rollback parameter **P** and forward parameter **Q**. It combines BFS and DFS. The rollback probability is proportional to 1/P, and the forward probability is proportional to 1/Q. Multiple random steps are generated to reflect the network structures.
Real-time Recommendation	The Real-time Recommendation algorithm is based on the random walk model and is used to recommend nodes that are similar (have similar relationships or preferences) to the input node. This algorithm can be used to recommend similar products based on historical 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 commodities that interest both nodes in the consumption field.
Connected Component	A connected component stands for a sub-graph, in which all nodes are connected with each other. Path directions are involved in the strongly connected components and are not considered in the weakly connected components. **NOTE** This algorithm generates weakly connected components.
Degree Correlation	The Degree Correlation algorithm calculates the Pearson correlation coefficient between the source vertex degree and the target vertex degree of each edge. It is used to indicate whether the high-degree nodes are connected to other high-degree nodes in a graph.
Triangle Count	The Triangle Count algorithm counts the number of triangles in a graph without considering the edge directions. More triangles mean higher node association degrees and closer organization relationships.
Cluster Coefficient	The cluster 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.
点集共同邻居（Common Neighbors of Vertex Sets）	可以得到两个点集合（群体集合）所共有的邻居（即两个群体临域的交集），直观的发现与两个群体共同联系的对象，如发现社交场合中的共同好友、消费领域共同感兴趣的商品、社区群体共同接触过的人，进一步推测两点集合之间的潜在关系和联系程度。

Algorithm	Description
点集全最短路（All Shortest Paths of Vertex Sets）	点集最短路算法用于发现两个点集之间的所有最短路径，可应用于互联网社交、金融风控、路网交通、物流配送等场景下的区块之间关系的分析。

6.2 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 6-2 PageRank algorithm parameters

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

NOTE

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

Precautions

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

Example

Set parameters **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.

6.3 PersonalRank

Overview

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

Application Scenarios

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

Parameter Description

Table 6-3 PersonalRank algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
source	Yes	Node ID	String	-	-
alpha	No	Weight coefficient	Double	A real number between 0 and 1 (excluding 0 and 1)	0.85

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

NOTE

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

Precautions

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

Example

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

6.4 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 6-4 K-core algorithm parameters

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

Precautions

None

Example

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

6.5 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 6-5 K-hop algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
k	Yes	Number of hops	Integer	1-100	-
source	Yes	Node ID	String	-	-

Parameter	Mandatory	Description	Type	Value Range	Default Value
mode	No	Direction: ● OUT: Hop from the outgoing edges. ● IN: Hop from the incoming edges. ● All: Hop from edges in both directions.	String	OUT, IN, ALL	OUT

Precautions

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

Example

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

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

6.6 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 6-6 Shortest Paths algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
source	Yes	Enter the source ID of a path.	String	-	-
target	Yes	Enter the target ID of a path.	String	-	-
directed	No	Whether to consider the edge direction	Bool	**true** or **false**	false
weight	No	Weight of an edge	String	Empty or null character string ● Empty: The default weight and distance are **1**. ● Character string: The attribute of the corresponding edge is the weight. When the edge does not have corresponding attribute, the weight is **1** by default. **NOTE** The weight of an edge must be greater than **0**.	-
timeWindow	No	Time window used for time filtering	JSON	For details, see **Table 6-7**. **NOTE** **timeWindow** does not support the shortest path with weight. That is, parameters **timeWindow** and **weight** cannot be both specified.	-

Table 6-7 timeWindow parameters

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

Precautions

This algorithm only returns one shortest path.

Example

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

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

6.7 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 6-8 All Shortest Paths algorithm parameters

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

Precautions

None

Example

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

6.8 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 6-9 Filtered Shortest Path algorithm parameters

Parameter	Mandatory	Type	Description
source	Yes	String	Enter the source vertex ID of a path.

Parameter	Mandatory	Type	Description
target	Yes	String	Enter the target vertex ID of a path.
directed	No	Boolean	Whether to consider the edge direction The default value is **false**.

Precautions

This algorithm only returns one shortest path.

6.9 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.

Parameter Description

Table 6-10 SSSP algorithm parameters

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

6.10 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.

Parameter Description

Table 6-11 Shortest Path of Vertex Sets algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
sources	Yes	Source vertex ID set	String	The value is in the standard CSV format. IDs are separated by commas (,), for example, **Alice, Nana**. The maximum ID number is 100,000.	-
targets	Yes	Target vertex ID set	String	The value is in the standard CSV format. IDs are separated by commas (,), for example, **Alice, Nana**. The maximum ID number is 100,000.	-
directed	No	Whether to consider the edge direction	Bool	**true** or **false**	false
timeWindow	No	Time window used for time filtering	JSON	For details, see **Table 6-12**.	-

Table 6-12 timeWindow parameters

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

NOTE

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

Example

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

6.11 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 6-13 n-Paths algorithm parameters

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

Example

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

6.12 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.

Parameter Description

Table 6-14 Closeness Centrality algorithm parameters

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

Example

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

6.13 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.

Parameter Description

Table 6-15 Label Propagation algorithm parameters

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

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

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

Precautions

Label Propagation uses IDs as labels by default.

Example

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

6.14 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.

Parameter Description

Table 6-16 Louvain algorithm parameters

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

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

Precautions

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

Example

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

6.15 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 6-17 Link Prediction algorithm parameters

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

Example

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

6.16 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 6-18 Node2vec algorithm parameters

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

Precautions

None

Example

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

6.17 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.

Parameter Description

Table 6-19 Real-time Recommendation algorithm parameters

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

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

NOTE

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

Precautions

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

Example

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

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

6.18 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 commodities that interest both nodes in the consumption field.

Application Scenarios

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

Parameter Description

Table 6-20 Common Neighbors algorithm parameters

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

Precautions

None

Example

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

6.19 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.

6.20 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.

6.21 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

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

Instructions

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

Example

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

6.22 Cluster Coefficient

Overview

The cluster 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 cluster coefficient of a graph. The JSON result is displayed in the query result area.

6.23 Common Neighbors of Vertex Sets

Overview

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

Application Scenarios

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

Parameter Description

Table 6-21 Common Neighbors of Vertex Sets algorithm parameters

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

Precautions

None

Example

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

6.24 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 6-22 All Shortest Paths of Vertex Sets algorithm parameters

Parameter	Mandatory	Description	Type	Value Range	Default Value
sources	Yes	Source vertex ID set	String	The value is in the standard CSV format. IDs are separated by commas (,), for example, **Alice, Nana**. The maximum ID number is 100,000.	-
targets	Yes	Target vertex ID set	String	The value is in the standard CSV format. IDs are separated by commas (,), for example, **Alice, Nana**. The maximum ID number is 100,000.	-
directed	No	Whether to consider the edge direction	Boolean	**true** or **false**. 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.

7 GES Metrics

Function

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.

Namespace

SYS.GES

Metrics

Table 7-1 GES metrics

Metric ID	Metric	Description	Value Range	Monitored Object and Dimension	Monitoring Period (Original Metric)
ges001_vertex_util	Vertex Capacity Usage	Capacity usage of vertices in a graph instance. The value is the ratio of the number of used vertices to the total vertex capacity. Unit: %	0-100 Value type: Float	Monitored object: GES instance Dimension: instance_id	1 minute

Metric ID	Metric	Description	Value Range	Monitored Object and Dimension	Monitoring Period (Original Metric)
ges002_edge_util	Edge Capacity Usage	Capacity usage of edges in a graph instance. The value is the ratio of the number of used edges to the total edge capacity. Unit: %	0-100 Value type: Float	Monitored object: GES instance Dimension: instance_id	1 minute
ges003_average_import_rate	Average Import Rate	Average rate of importing vertices or edges to a graph instance Unit: count/s	0-400000 Value type: Float	Monitored object: GES instance Dimension: instance_id	1 minute
ges004_request_count	Request Quantity	Number of requests received by a graph instance Unit: count	≥ 0 Value type: Int	Monitored object: GES instance Dimension: instance_id	1 minute
ges005_average_response_time	Average Response Time	Average response time of requests received by a graph instance Unit: ms	≥ 0 Value type: Int	Monitored object: GES instance Dimension: instance_id	1 minute
ges006_min_response_time	Minimum Response Time	Minimum response time of requests received by a graph instance Unit: ms	≥ 0 Value type: Int	Monitored object: GES instance Dimension: instance_id	1 minute
ges007_max_response_time	Maximum Response Time	Maximum response time of requests received by a graph instance Unit: ms	≥ 0 Value type: Int	Monitored object: GES instance Dimension: instance_id	1 minute

Metric ID	Metric	Description	Value Range	Monitored Object and Dimension	Monitoring Period (Original Metric)
ges008_read_task_pending_queue_size	Length of the Waiting Queue for Read Tasks	Length of the waiting queue for read requests received by a graph instance. This metric is used to view the number of read requests waiting in the queue. Unit: count	≥ 0 Value type: Int	Monitored object: GES instance Dimension: instance_id	1 minute
ges009_read_task_pending_max_time	Maximum Waiting Duration of Read Tasks	Maximum waiting duration of read requests received by a graph instance Unit: ms	≥ 0 Value type: Int	Monitored object: GES instance Dimension: instance_id	1 minute
ges010_pending_max_time_ read_task_type	Type of the Read Task That Waits the Longest	Type of the read request that waits the longest in a graph instance. You can find the corresponding task name in documents at the official website. Unit: count	≥ 1 Value type: Int	Monitored object: GES instance Dimension: instance_id	1 minute
ges011_read_task_running_queue_size	Length of the Running Queue for Read Tasks	Length of the running queue for read requests received by a graph instance. This metric is used to view the number of running read requests. Unit: count	≥ 0 Value type: Int	Monitored object: GES instance Dimension: instance_id	1 minute

Metric ID	Metric	Description	Value Range	Monitored Object and Dimension	Monitoring Period (Original Metric)
ges012_read_task_running_max_time	Maximum Running Duration of Read Tasks	Maximum running duration of read requests received by a graph instance Unit: ms	≥ 0 Value type: Int	Monitored object: GES instance Dimension: instance_id	1 minute
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 documents at the official website. Unit: count	≥ 1 Value type: Int	Monitored object: GES instance Dimension: instance_id	1 minute
ges014_ write_task_pending_queue_size	Length of the Waiting Queue for Write Tasks	Length of the waiting queue for write requests received by a graph instance. This metric is used to view the number of write requests waiting in the queue. Unit: count	≥ 0 Value type: Int	Monitored object: GES instance Dimension: instance_id	1 minute
ges015_ write_task_pending_max_time	Maximum Waiting Duration of Write Tasks	Maximum waiting duration of write requests received by a graph instance Unit: ms	≥ 0 Value type: Int	Monitored object: GES instance Dimension: instance_id	1 minute

Metric ID	Metric	Description	Value Range	Monitored Object and Dimension	Monitoring Period (Original Metric)
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 documents at the official website. Unit: count	≥ 1 Value type: Int	Monitored object: GES instance Dimension: instance_id	1 minute
ges017_ write_task_running_queue_size	Length of the Running Queue for Write Tasks	Length of the running queue for write requests received by a graph instance. This metric is used to view the number of running write requests. Unit: count	≥ 0 Value type: Int	Monitored object: GES instance Dimension: instance_id	1 minute
ges018_ write_task_running_max_time	Maximum Running Duration of Write Tasks	Maximum running duration of write requests received by a graph instance Unit: ms	≥ 0 Value type: Int	Monitored object: GES instance Dimension: instance_id	1 minute
ges019_running_max_time_ write_task_type	Type of the Write Task That Runs the Longest	Type of the write request that runs the longest in a graph instance. You can find the corresponding task name in documents at the official website. Unit: count	≥ 1 Value type: Int	Monitored object: GES instance Dimension: instance_id	1 minute

Metric ID	Metric	Description	Value Range	Monitored Object and Dimension	Monitoring Period (Original Metric)
ges020_computer_resource_usage	Computing Resource Usage	Computing resource usage of each graph instance Unit: %	0-100 Value type: Float	Monitored object: GES instance Dimension: instance_id	1 minute
ges021_memory_usage	Memory Usage	Memory usage of each graph instance Unit: %	0-100 Value type: Float	Monitored object: GES instance Dimension: instance_id	1 minute
ges022_iops	IOPS	Number of I/O requests processed by each graph instance per second Unit: count/s	≥ 0 Value type: Int	Monitored object: GES instance Dimension: instance_id	1 minute
ges023_bytes_in	Network Input Throughput	Data input to each graph instance per second over the network Unit: byte/s	≥ 0 Value type: Float	Monitored object: GES instance Dimension: instance_id	1 minute
ges024_bytes_out	Network Output Throughput	Data sent to the network per second from each graph instance Unit: byte/s	≥ 0 Value type: Float	Monitored object: GES instance Dimension: instance_id	1 minute
ges025_disk_usage	Disk Usage	Disk usage of each graph instance Unit: %	0-100 Value type: Float	Monitored object: GES instance Dimension: instance_id	1 minute
ges026_disk_total_size	Total Disk Size	Total data disk space of each graph instance Unit: GB	≥ 0 Value type: Float	Monitored object: GES instance Dimension: instance_id	1 minute

Metric ID	Metric	Description	Value Range	Monitored Object and Dimension	Monitoring Period (Original Metric)
ges027_disk_used_size	Disk Space Used	Used data disk space of each graph instance Unit: GB	≥ 0 Value type: Float	Monitored object: GES instance Dimension: instance_id	1 minute
ges028_disk_read_throughput	Disk Read Throughput	Data volume read from the disk in a graph instance per second Unit: byte/s	≥ 0 Value type: Float	Monitored object: GES instance Dimension: instance_id	1 minute
ges029_disk_write_throughput	Disk Write Throughput	Data volume written to the disk in a graph instance per second Unit: byte/s	≥ 0 Value type: Float	Monitored object: GES instance Dimension: instance_id	1 minute
ges030_avg_disk_sec_per_read	Average Time per Disk Read	Average time used each time when the disk of a graph instance reads data Unit: second	≥ 0 Value type: Float	Monitored object: GES instance Dimension: instance_id	1 minute
ges031_avg_disk_sec_per_write	Average Time per Disk Write	Average time used each time when data is written to the disk of a graph instance Unit: second	≥ 0 Value type: Float	Monitored object: GES instance Dimension: instance_id	1 minute
ges032_avg_disk_queue_length	Average Disk Queue Length	Average I/O queue length of the disk in a graph instance Unit: count	≥ 0 Value type: Int	Monitored object: GES instance Dimension: instance_id	1 minute

Dimensions

Table 7-2 Dimensions

Key	Value
instance_id	GES instance