GaussDB

User Guide

Issue 01

Date 2025-02-17

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


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

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

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: https://www.huaweicloud.com/intl/en-us/

Contents

1 Permissions Management

1.1 Creating a User and Granting Permissions

This section describes how to use **Identity and Access Management (IAM)** for fine-grained permissions management for your GaussDB resources. With IAM, you can:

- Create IAM users for employees based on your enterprise's organizational structure. Each IAM user will have their own security credentials for accessing GaussDB resources.
- Grant only the permissions required for users to perform a specific task.
- Entrust an account of Huawei Cloud or a cloud service to perform professional and efficient O&M on your GaussDB resources.

If your Huawei Cloud account does not require individual IAM users, skip this section.

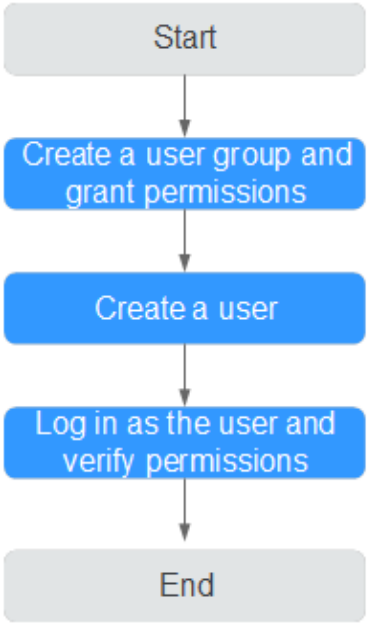
Figure 1-1 describes the process for granting permissions.

Prerequisites

Before assigning permissions to user groups, you should learn about the system-defined permissions of GaussDB listed in **System-defined Permissions**. For the system policies of other services, see **System-defined Permissions**.

Process Flow

Figure 1-1 Process of granting GaussDB permissions



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

   Create a user group on the IAM console, and attach the **GaussDB ReadOnlyAccess** policy to the group.

2. **Create an IAM user and add it to the user group.**

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

3. **Log in** and verify permissions.

   Log in to the console by using the created user, and verify that the user only has read permissions for GaussDB.

   - Under the service list, choose **GaussDB**. In the navigation pane on the left, choose **GaussDB** > **Instances**. Click **Buy DB Instance** in the upper right corner. If a message appears indicating that you have insufficient permissions to perform the operation, the GaussDB ReadOnlyAccess policy has already taken effect.
   - Choose any other service in the service list. If a message appears indicating that you have insufficient permissions to access the service, the **GaussDB ReadOnlyAccess** policy has already taken effect.

1.2 Creating a Custom Policy

Custom policies can be created to supplement the system-defined policies of GaussDB. 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. This does not require knowledge of policy syntax.

- JSON: Create a policy in JSON format or edit the JSON strings of an existing policy.

For details about how to create a custom policy, see the section **Creating a Custom Policy**. The following contains examples of common GaussDB custom policies.

Example Custom Policy

- Example 1: Allowing users to create GaussDB instances

```
{
    "Version": "V2.0-8.103",
    "Statement": [{
  "Effect": "Allow",
  "Action": ["gaussdb:instance:create"]
  }]
}
```

- Example 2: Denying GaussDB instance deletion

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

  The following method can be used if you need to assign permissions of the **GaussDB FullAccess** policy to a user but you want to prevent the user from deleting GaussDB instances. Create a custom policy for denying GaussDB instance deletion, and attach both policies to the group to which the user belongs. Then, the user can perform all operations on GaussDB instances except deleting GaussDB instances. The following is an example of a deny policy:

```
{
 "Version": "V2.0-8.103",
 "Statement": [{
  "Action": ["gaussdb:instance:delete"],
  "Effect": "Deny"
 }]
}
```


2 Buying a GaussDB Instance

Scenarios

You can buy a DB instance on the management console.

GaussDB supports pay-per-use and yearly/monthly billing. GaussDB allows you to tailor your computing resources and storage space to your business needs.

Prerequisites

You have registered a HUAWEI ID and enabled Huawei Cloud services.

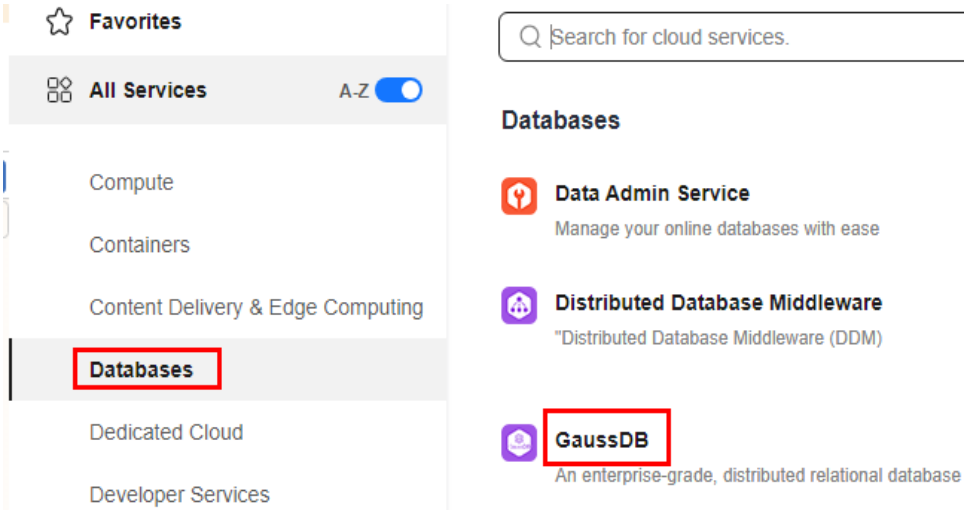
Procedure

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Figure 2-1 GaussDB



Step 4 On the **Instances** page, click **Buy DB Instance**.

Step 5 On the displayed page, select a billing mode, configure parameters about the instance, and click **Next**.

Figure 2-2 Billing mode and basic information

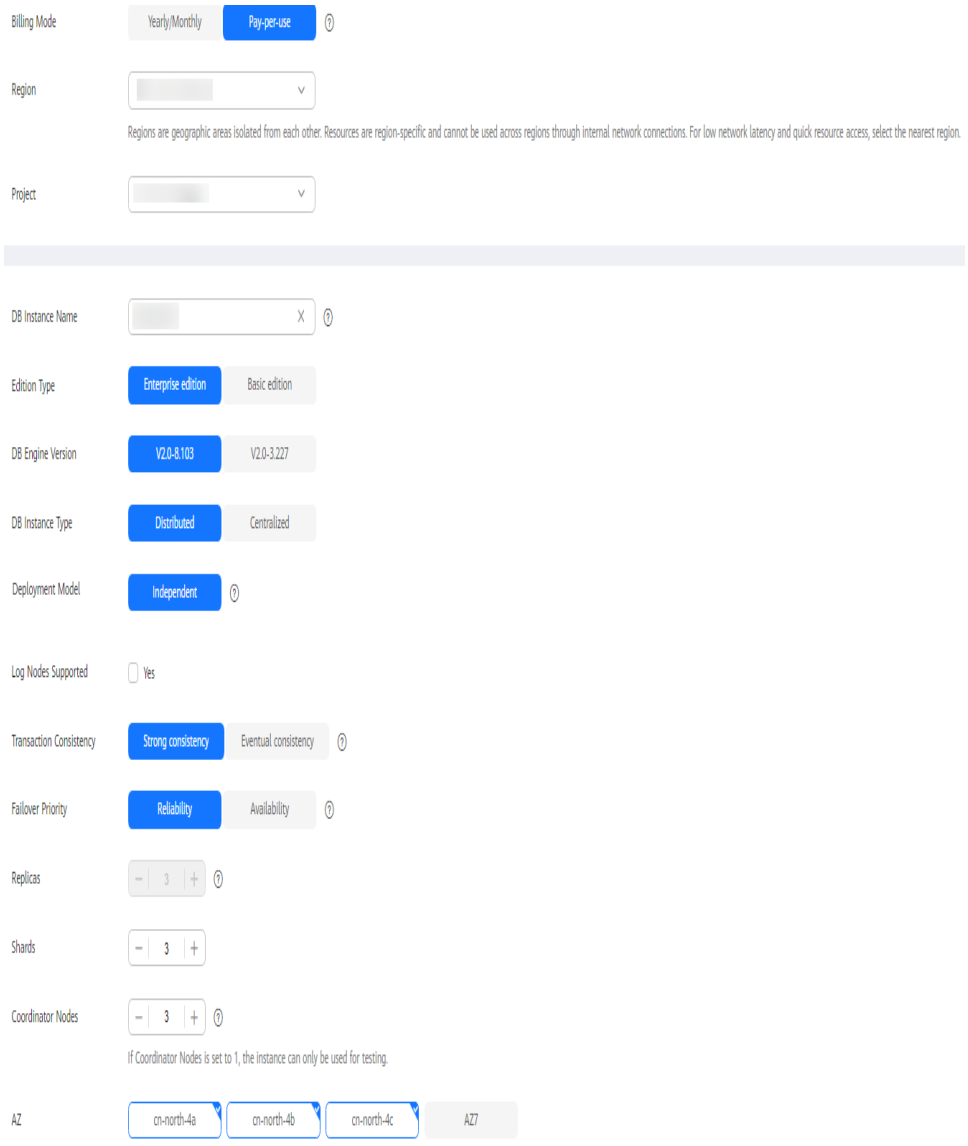


Table 2-1 Basic information

Parameter	Description
Billing Mode	GaussDB provides yearly/monthly billing and pay-per-use billing. • **Yearly/Monthly**: You pay upfront for the amount of time you expect to use the DB instance for. You will need to make sure you have a top-up account with a sufficient balance or have a valid payment method configured first. • **Pay-per-use**: You can start using the DB instance first and then pay as you go. Pricing is listed on a per-hour basis, but bills are calculated based on the actual usage duration.
Region	A region where the tenant is located. You can change the region on the instance creation page, or go back to the **Instances** page and change it in the upper left corner. **NOTE** Products in different regions cannot communicate with each other over a private network. After the DB instance is created, you cannot change its region.
DB Instance Name	The instance name must start with a letter and can contain 4 to 64 characters. Only letters (case-sensitive), digits, hyphens (-), and underscores (_) are allowed.
Edition Type	GaussDB provides **Basic edition** and **Enterprise edition**. The basic edition lacks certain advanced features that are available in the enterprise edition. The basic edition delivers the same level of performance as the enterprise edition at a more affordable price. This edition is ideal for users who prioritize cost and do not need advanced features.
DB Engine Version	Select the GaussDB database version of the new instance.
DB Instance Type	• **Distributed**: You can add nodes for distributed instances as needed to handle large volumes of concurrent requests. • **Centralized**: Centralized instances are suitable for scenarios with small and stable volumes of data, where data reliability and service availability are extremely important.

Parameter	Description
Deployment Model	• Distributed instances – **Independent**: Database components are deployed on different nodes. This model is suitable for where high availability and stability are required and the instance scale is large. – **Combined**: Different database components are deployed on the same node. This option is available only when **Edition Type** is **Basic edition**. • centralized – **HA (1 primary + 2 standby)**: 3-node deployment where there is a shard. The shard contains one primary DN and two standby DNs. – **Single**: single-node deployment where there is only one CMS component and one DN. To create a single-replica instance, ensure that the instance version is V2.0-2.2 or later. – **1 primary + 1 standby + 1 log**: 3-node deployment where there is one shard with three replicas. The shard contains one primary DN, one standby DN, and one log-dedicated DN. This model is available only for instances of version V2.0-3.200 or later. **CAUTION** **Single**: The single-replica deployment model is only available to whitelisted users. You can submit a service ticket to request it at **Service Tickets > Create Service Ticket** in the upper right corner of the management console. The availability (SLA) cannot be guaranteed because the instance is deployed on a single server. **NOTE** • The combined deployment model has the following restrictions: – This model is available only for instances of version V2.0-3.223 or later. – Instance specifications cannot be changed. – Storage autoscaling is not supported. – Yearly/Monthly billing is not supported.
Log Nodes Supported	This parameter is available only for distributed instances. If this option is selected, the distributed instance will be created using the 1 primary + 1 standby + 1 log deployment model. By default, centralized instances support the 1 primary + 1 standby + 1 log deployment model. You can simply set **Deployment Model** to **1 primary + 1 standby + 1 log** as needed when creating a centralized instance.

Parameter	Description
Transaction Consistency	This parameter is available only to distributed instances. • **Strong consistency**: When an application updates data, every user can query all data that has been successfully committed, but performance is affected. • **Eventual consistency**: When an application updates data, the data users queried may be different, and some users may not obtain the most current value. The most current data may take a bit of time to become available for query by all users. However, DB instances with eventual consistency generally have higher performance. Eventual consistency cannot ensure strong read consistency of distributed transactions and consistency of transactions that depend on query results, such as INSERT INTO SELECT * FROM. Write operations that are split into multiple statements or involve in multiple nodes are not supported.
Failover Priority	This function is available only to distributed instances. To use this parameter, contact customer service to apply for the required permissions. The default value is **Reliability**. For details about how to change the failover priority for an existing instance, see **Changing Failover Priority**. • **Reliability**: Data consistency is given priority during a failover. This is recommended for applications with highest priority for data consistency. • **Availability**: Database availability is given priority during a failover. This is recommended for applications that require their databases to provide uninterrupted online services. **NOTE** If **Availability** is selected, exercise caution when modifying the following database parameters. For details about how to modify parameters, see **Modifying Instance Parameters**. – **recovery_time_target**: If this parameter is changed, the DB instance will undergo frequent forced failovers. To change this parameter, contact technical support first. – **audit_system_object**: If this parameter is changed, DDL audit logs will be lost. To change this parameter, contact technical support first.
Replicas	This parameter is available only for distributed instances. Total number of DNs each shard, primary and standby DNs combined. There are three replicas in a shard, indicating that there are one primary and two standby DNs in a shard.

Parameter	Description
Shards	This parameter is available only for distributed instances. It indicates the number of shards in an instance. A shard contains multiple DNs. The number of DNs in a shard depends on the value of **Replicas**, for example, if **Replicas** is set to **3**, there are three DNs (one primary and two standby DNs) in a shard.
Coordinator Nodes	This parameter is available only for distributed instances. It indicates the number of CNs in an instance. A CN provides the following functions: • It receives access requests from applications and returns execution results to clients. • It breaks down tasks and distributes task fragments to different DNs for parallel processing. **NOTICE** It is recommended that at least two CNs be configured for an instance. If only one CN is configured, the instance's service reliability is low, making it suitable only for testing scenarios and not suitable for use in a production environment.
AZ	An AZ is a physical region where resources have their own independent power supply and networks. AZs are physically isolated but interconnected through an internal network. A DB instance can be deployed in one AZ or three AZs.
Time Zone	Select a time zone according to the region hosting your DB instance when you buy the instance.

Figure 2-3 Specifications and storage

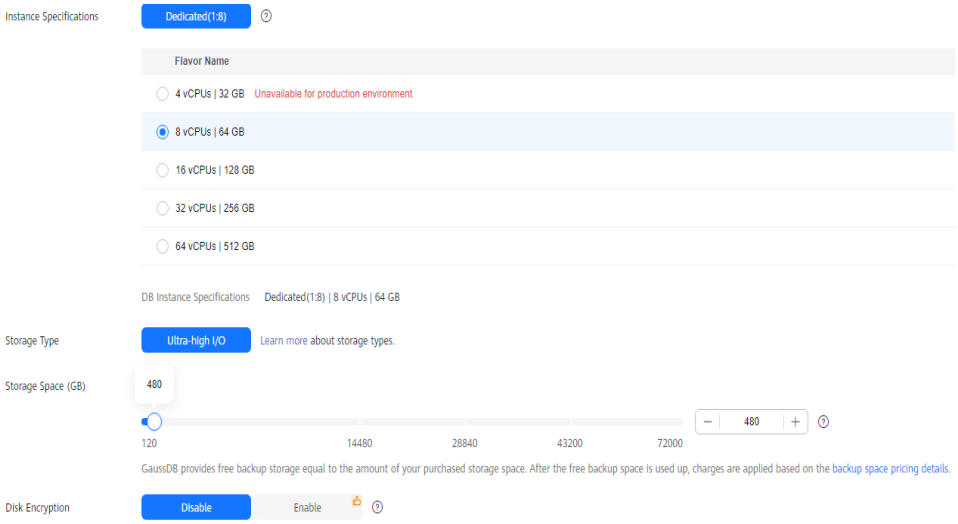


Table 2-2 Specifications and storage

Parameter	Description
Instance Specifications	CPU and memory specifications of the instance. Different instance specifications have different numbers of database connections. For details, see **Instance Specifications**.
Dedicated Cloud	M6. **NOTE** This option is available only when you have purchased Dedicated Computing Cluster (DCC).
Resource Type	EVS. **NOTE** This option is available only when you have purchased Dedicated Computing Cluster (DCC).
Storage Type	The storage type determines the read/write speed of an instance. The higher the maximum throughput is, the higher the instance read/write speed can be. GaussDB supports the ultra-high I/O and extreme SSD storage types. • Ultra-high I/O: Ultra-high performance cloud disks excellent for enterprise mission-critical services as well as workloads demanding high throughput and low latency When striped across multiple disks, they can achieve up to 800 MB/s throughput. • Extreme SSD: Superfast disks ideal for workloads demanding ultra-high bandwidth and ultra-low latency. When striped across multiple disks, they can achieve up to 2,500 MB/s throughput. **NOTE** To apply for the permissions needed for using the extreme SSD storage type, submit a service ticket to request it at **Service Tickets > Create Service Ticket** in the upper right corner of the management console.
Storage Space (GB)	The storage space contains the file system overhead required for inodes, reserved blocks, and database operations. After buying an instance, you can scale up its storage space. For details, see **Scaling Up Storage Space**. **NOTE** When you create a DB instance, the storage space for a single shard starts from 40 GB and can be increased at a step of 4 GB.
Free Backup Space	GaussDB provides free backup storage equal to the amount of your purchased storage space. After the free backup space is used up, you will be billed for the additional space used.

Parameter	Description
Disk Encryption	• **Disable**: Encryption is disabled. • **Enable**: Encryption is enabled, which improves data security but affects system performance. **Key Name**: If disk encryption is enabled, you need to select or create a key, which is used by tenants. **NOTE** If a shared KMS key is used, the corresponding CTS event is **createGrant**. Only the key owner can receive this event.

Figure 2-4 Network and database configuration

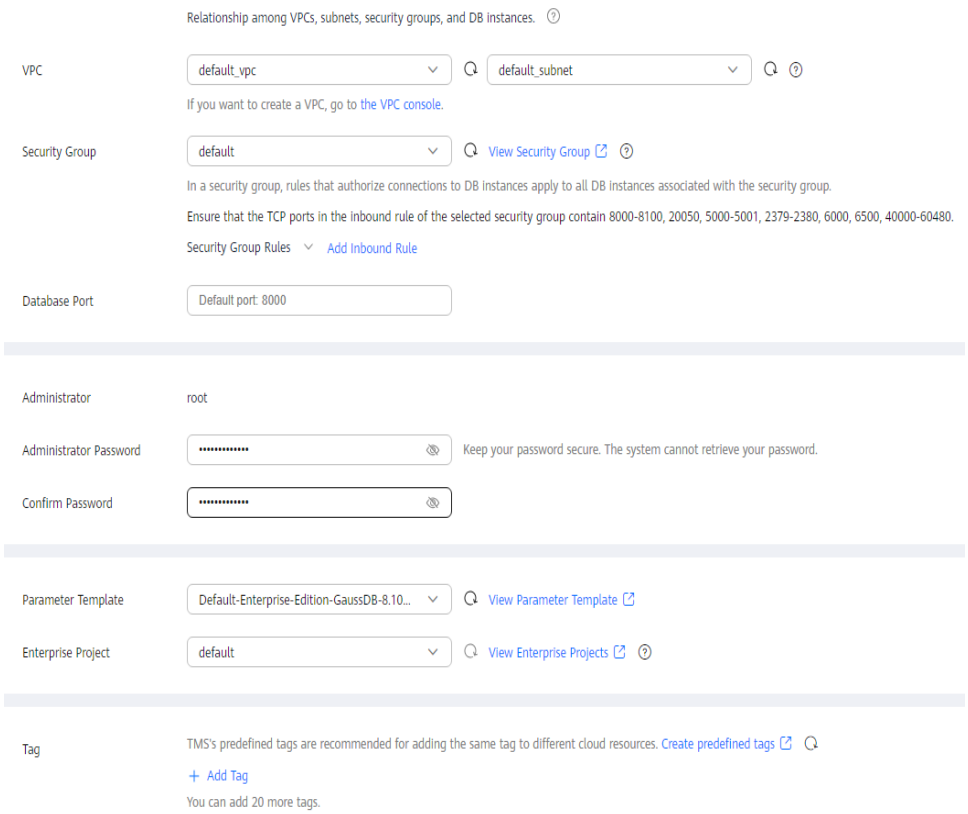


Table 2-3 Network

Parameter	Description
VPC	A virtual network where your GaussDB instances are located. A VPC isolates networks for different workloads. You need to create or select the required VPC. For details about how to create a VPC, see **Creating a VPC**. With VPC sharing, you can also use a VPC and subnet shared by another account. VPC owners can share the subnets in a VPC with one or multiple accounts through Resource Access Manager (RAM). This allows for more efficient use of network resources and reduces O&M costs. For more information about VPC subnet sharing, see **VPC Sharing** in the *Virtual Private Cloud User Guide*. If no VPC is available, GaussDB allocates a default VPC for you. **NOTICE** After the GaussDB instance is created, the VPC cannot be changed.
Subnet	A subnet provides dedicated network resources that are logically isolated from other networks for network security. Subnets take effect only within a specific AZ. Dynamic Host Configuration Protocol (DHCP) is enabled by default for subnets in which you plan to create GaussDB instances and cannot be disabled. GaussDB supports automatic IP address allocation during instance creation. **NOTE** • By default, a subnet supports up to 256 IP addresses. A distributed instance can require up to 1,286 IP addresses. You are advised to use a subnet that can provide 2,048 IP addresses.

Parameter	Description
Security Group	A security group controls the access that traffic has in and out of a GaussDB instance. By default, the security group associated with the instance is authorized. • If you need to change the security group when buying a distributed instance, ensure that the TCP ports in the inbound rule include the following: 40000-60480, 20050, 5000-5001, 2379-2380, 6000, 6500, and *<database port>-(<database port> + 100)*. (For example, if the database port is 8000, the TCP ports for the security group must include 8000-8100.) • If you need to change the security group when buying a centralized instance, ensure that the TCP ports in the inbound rule include the following: 20050, 5000-5001, 2379-2380, 6000, 6500, and *<database port>-(<database port> + 100)*. (For example, if the database port is 8000, the TCP ports for the security group must include 8000-8100.) The security group enhances security by controlling access to GaussDB from other services. When you select a security group, you must ensure that it allows the client to access your DB instances. If you do not need to specify a security group when creating a DB instance, you can submit a service ticket to request it at **Service Tickets > Create Service Ticket** in the upper right corner of the management console. If no security group is available, GaussDB allocates a default security group for you.
Database Port	The port is used by applications to access the database. Value range: 1024 to 39989. Default value: **8000**. The following ports are used by the system and cannot be used: 2378 to 2380, 2400, 4999 to 5001, 5100, 5500, 5999 to 6001, 6009 to 6010, 6500, 8015, 8097, 8098, 8181, 9090, 9100, 9180, 9187, 9200, 12016, 12017, 20049, 20050, 21731, 21732, 32122 to 32126, and 39001.
Single Floating IP Address	Specifies whether to enable the single floating IP address policy. If this policy is enabled, only one floating IP address is assigned to an instance and is bound to the primary node. The floating IP address does not change after a primary/standby switchover. If this policy is disabled, each node is bound to a floating IP address, and the floating IP address changes after a primary/standby switchover. The constraints on the single floating IP address policy are as follows: • This policy is only available for centralized instances of version V2.0-3.206 or later. • This policy is configurable only during instance creation and cannot be modified afterwards.

Table 2-4 Database configuration

Parameter	Description
Administrator	DB administrator. The default username is **root**.
Administrator Password	Enter a strong password and periodically change it to improve security, preventing security risks such as brute force cracking. **NOTICE** The password must contain: • 8 to 32 characters. • At least three types of the following: uppercase letters, lowercase letters, digits, and special characters. Supported characters: ~!@#%^*-_=+?, Keep your password secure because you cannot retrieve it from the system. After a DB instance is created, you can reset this password. For details, see **Resetting the Administrator Password**.
Confirm Password	Enter the administrator password again.

Table 2-5 Parameter templates

Parameter	Description
Parameter Template	A template of parameters for creating an instance. The template contains engine configuration values that are applied to one or more instances. You can modify the instance parameters as required after the DB instance is created. You can modify the instance parameters as required after the instance is created. For details, see **Modifying Parameters in a Parameter Template**.
Enterprise Project	If the instance has been associated with an enterprise project, select the target project from the **Enterprise Project** drop-down list. You can also go to the Enterprise Project Management console to create a project. For details, see ***Enterprise Management User Guide***.

Table 2-6 Tags

Parameter	Description
Tag	This parameter is optional. Adding tags helps you better identify and manage your DB instances. Each instance can have up to 20 tags. If your organization has configured tag policies for GaussDB, add tags to instances based on the policies. If a tag does not comply with the policies, instance creation may fail. Contact your organization administrator to learn more about tag policies.

If you have any questions about the price, click **Pricing details** at the bottom of the page.

NOTE

The performance of your GaussDB instance depends on its settings. Hardware items include the instance specifications, storage type, and storage space.

Step 6 Confirm the displayed details.

Confirm your specifications for pay-per-use instances.

- If you need to modify your settings, click **Previous**.
- If you do not need to modify your settings, click **Submit**.

Confirm your order for yearly/monthly instances.

- If you need to modify your settings, click **Previous**.
- If you do not need to modify your settings, click **Pay Now** to go to the payment page. On the displayed page, select a payment method and click **Pay**.

Step 7 To check the GaussDB instance information and manage it after the creation task is submitted, go to the **Instances** page.

- When a GaussDB instance is being created, its status is **Creating**. This process takes about 10 to 20 minutes.
- To refresh the instance list, click  in the upper right corner of the list. When the creation process is complete, the instance status will be **Available**.
- An automated full backup is immediately triggered after once your instance is created.
- The default database port is 8000. You can change it during instance creation or after an instance is created.

----End

Related Operations

- **Creating a DB Instance Using an API**

- **Modifying Instance Parameters**

3 GaussDB Instance Connection

3.1 Connecting to a GaussDB Instance

GaussDB instances can be connected using gsql, DBeaver, Navicat, or Data Admin Service (DAS).

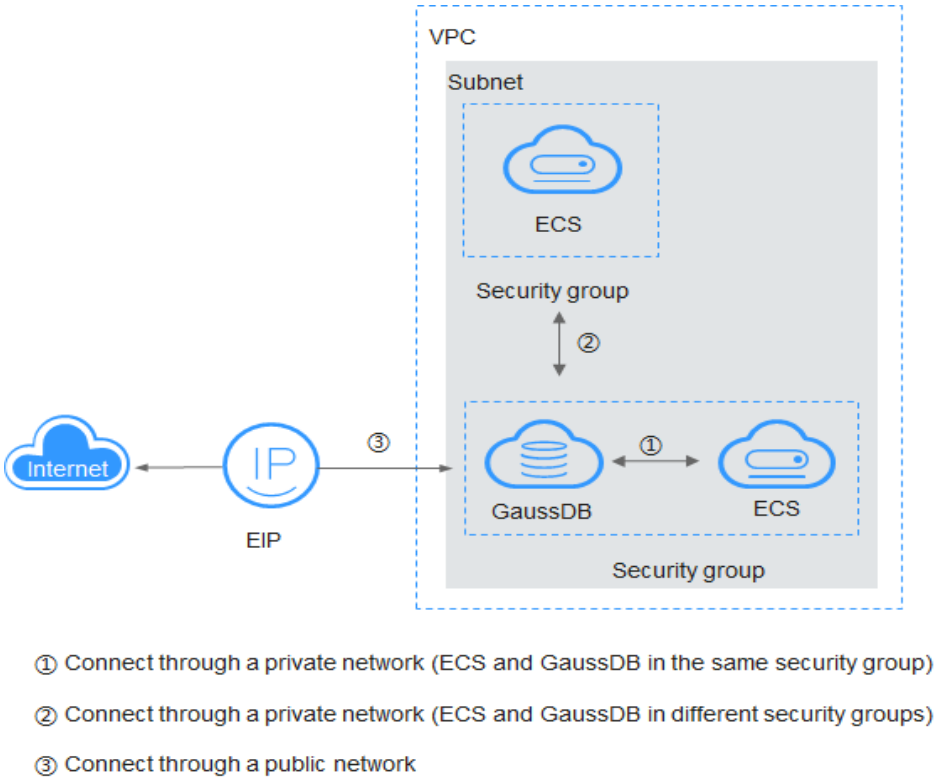
Table 3-1 GaussDB instance connection modes

Connect Through	IP Address	Description	Comments
DAS	Not required	Huawei Cloud DAS enables you to manage databases on a web-based console. It supports SQL execution, advanced database management, and intelligent O&M, simplifying database management and improving both efficiency and data security. The permissions required for connecting to a GaussDB instance through DAS are enabled by default.	Easy to use, secure, advanced, and intelligent

Connect Through	IP Address	Description	Comments
gsql	Private IP address/EIP	gsql is a client tool provided by GaussDB. You can use gsql to connect to the database and then enter, edit, and execute SQL statements in an interactive manner.	To achieve a higher data transmission rate and security level, migrate your applications to a server that is in the same subnet as your GaussDB instance and use a private IP address to access the instance. The bandwidth is not limited for private network connections.
DBeaver	EIP	DBeaver is a GUI-based database management tool. You can use this tool to view database schemas, execute SQL queries and scripts, browse and export data, process BLOB/CLOB data, and modify database schemas.	Open-source and easy-to-use
Navicat	EIP	Navicat is a database management tool. You can easily view and edit data on its graphical interface. For example, you can insert, delete, update, and query data, process SQL statements or scripts, use functions, and generate data.	Stable and easy to use

Figure 3-1 shows how an instance is connected.

Figure 3-1 Connecting to an instance through a private network and an EIP



NOTE

- If the ECS and GaussDB instance are in the same VPC and security group, they can communicate with each other through the private network by default. In this case, you can connect to the instance through a private IP address.
- If the ECS and GaussDB instance are in the same VPC but different security groups, you need to set security group rules for both the GaussDB instance and ECS, and then connect to the instance through a private IP address.
  - GaussDB instance: Configure an **inbound** rule for the security group with which the GaussDB instance is associated. For details, see **Configuring Security Group Rules for a GaussDB Instance**.
  - ECS: The default security group rule allows all outgoing data packets. In this case, you do not need to configure a security group rule for the ECS. If **not all outbound traffic is allowed** in the security group, you need to configure an **outbound** rule for the ECS to allow all outbound packets.
- If the ECS and GaussDB instance are in different VPCs, you can bind an EIP to the ECS and use the EIP to connect to the instance. Ensure that both the ECS and GaussDB instance have EIPs.
  - For details about how to bind an EIP to an ECS, see **Binding an EIP**.
  - For details about how to bind an EIP to a GaussDB instance, see **Binding an EIP**.

3.2 Connecting to an Instance Through DAS

Scenarios

DAS enables you to manage your databases from a web-based console. It supports SQL execution, advanced database management, and intelligent O&M,

simplifying database management and improving both efficiency and data security.

Procedure

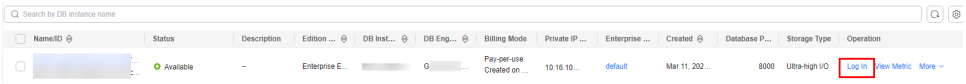
Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, locate the DB instance you want to log in to and click **Log In** in the **Operation** column.

Figure 3-2 Logging in to an instance



Alternatively, click the DB instance name on the **Instances** page. On the displayed **Basic Information** page, click **Log In** in the upper right corner of the page.

Figure 3-3 Logging in to an instance

Step 5 On the **Custom Login** page, select the node to be logged in to. Enter the correct database username and password, and click **Test Connection**. After the connection test is successful, click **Log In**.

Figure 3-4 Login page

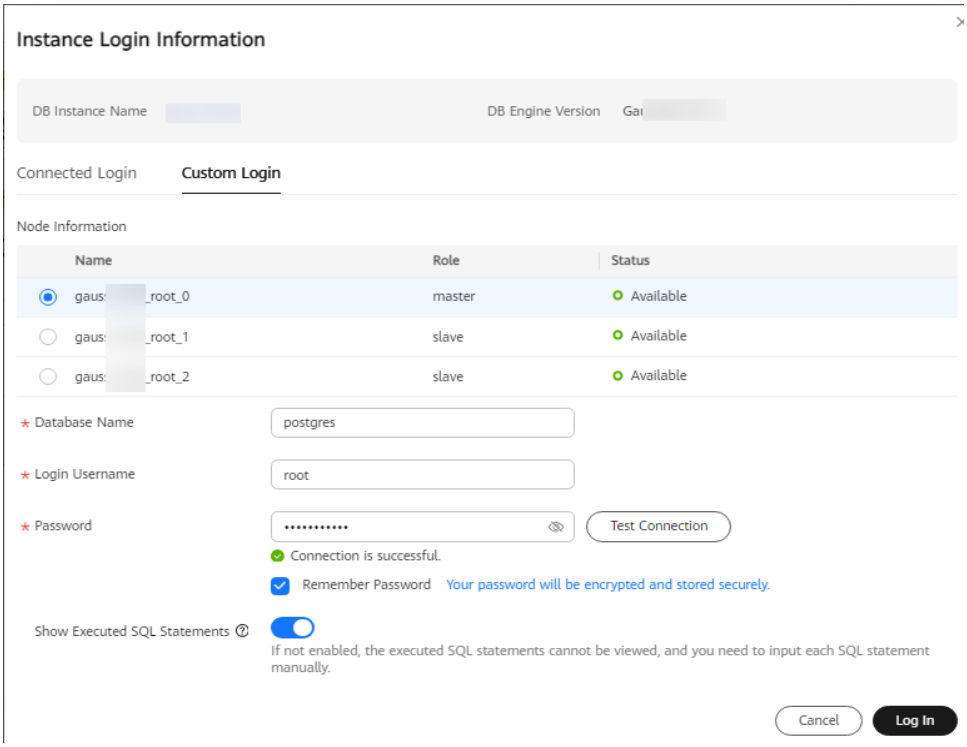


Table 3-2 Parameter description

Parameter	Description
Login Username	Username of the GaussDB database account. The default administrator is **root**.
Database Name	Name of the database to be connected. The default management database is **postgres**.
Password	Password of the database user.
Show Executed SQL Statements	You are advised to enable **Show Executed SQL Statements**. With it enabled, you can view the executed SQL statements under **SQL Operations** > **SQL History** and execute them again without entering the SQL statements.

For details about how to use DAS to manage databases, see **GaussDB Management**.

----End

Follow-up Operations

After logging in to the instance, you can create databases, create database users, and migrate databases.

- **Creating a Database**
- **Migrating a Database**
- **Creating a Database Through DAS**
- **Creating a Database User Through DAS**

FAQ

Question: What can I do if the DAS console is not displayed after I click **Log In** in the **Operation** column of an instance on the **Instances** page?

Solution: Set your browser to allow pop-ups and try again.

3.3 Using gsql to Connect to an Instance

This section describes how to use the gsql client to connect to a GaussDB instance you have bought on the GaussDB management console.

- **Step 1: Buy an ECS**
- **Step 2: Query the IP Address and Port Number of the Instance to Be Connected**
- **Step 3: Test the Connectivity**
- **Step 4: Obtain the Driver Package**
- **Step 5: Connect to the Database**

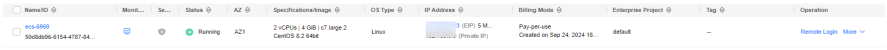
- **Non-SSL connection**
- **SSL connection**

Buying an ECS

If you want to connect to a database using the command-line interface (CLI), like gsql, you need to create an ECS and install gsql on it.

1. **Log in to the management console** and check whether there is an available ECS.
   - If there is, go to **3**.
   - If there is not, go to **2**.

   Figure 3-5 ECS instances

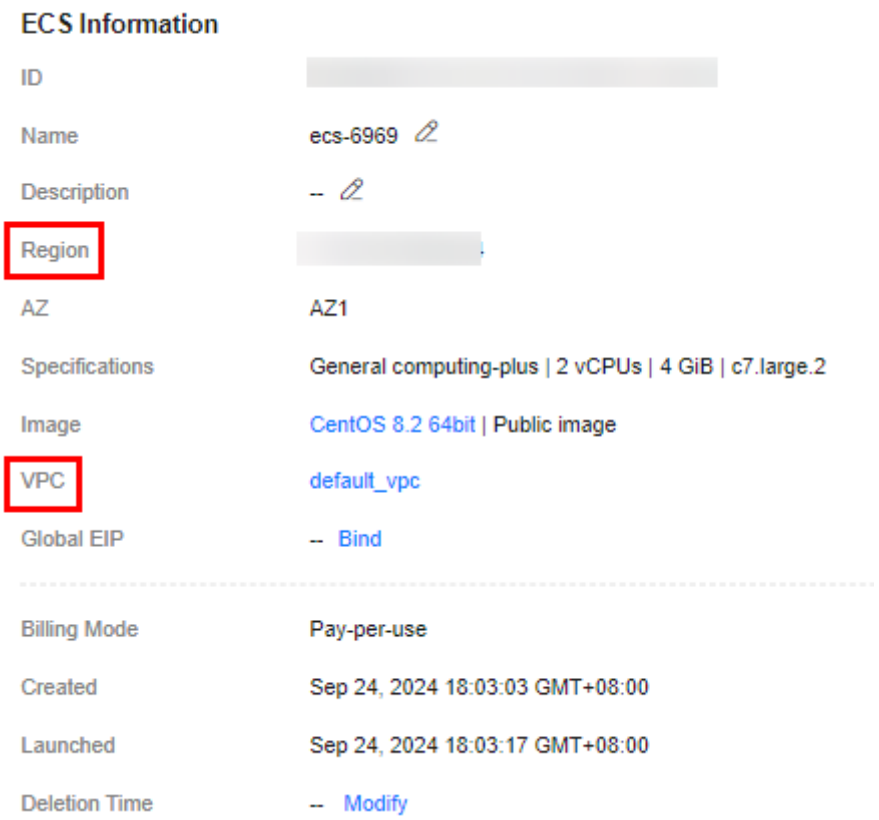
   

2. Buy an ECS that runs EulerOS.

   For details about how to buy a Linux ECS, see **Purchasing an ECS** in *Elastic Cloud Server Getting Started*.

3. On the **ECS Information** page of the target ECS, view the region and VPC of the ECS.

   Figure 3-6 ECS basic information

NOTICE

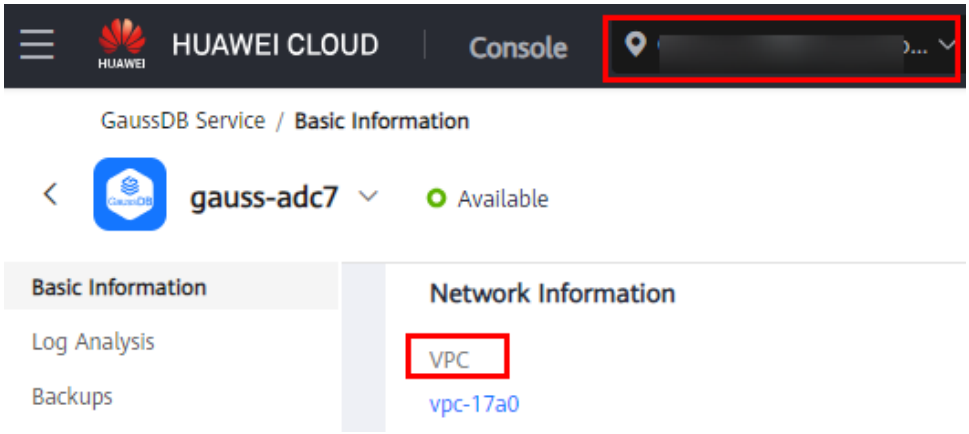
The ECS must run EulerOS. gsql supports the following versions:

For x86 servers: EulerOS V2.0SP5 and Kylin V10 SP2

For Kunpeng servers: EulerOS V2.0SP8 and Kylin V10 SP1

4. On the **Basic Information** page of your GaussDB instance, view the region and VPC of the instance.

Figure 3-7 Basic information about a GaussDB instance



5. Check whether the ECS and GaussDB instance are in the same region and VPC.
   - If the ECS and GaussDB instance are in the same region and VPC, the DB instance can be connected through a private network. For details about how to obtain the private IP address, see **Querying the IP Address of the Instance to Be Connected**.
   - If the ECS and DB instance are in different VPCs, the DB instance must be connected over a public network. For details about how to obtain the public IP address, see **Querying the IP Address of the Instance to Be Connected**. Ensure that both the ECS and GaussDB instance have EIPs.
     - For details about how to bind an EIP to an ECS, see **Binding an EIP**.
     - For details about how to bind an EIP to a GaussDB instance, see **Binding an EIP**.

Querying the IP Address and Port Number of the Instance to Be Connected

1. **Log in to the management console**.
2. Click in the upper left corner and select a region and project.
3. Click in the upper left corner of the page and choose **Databases** > **GaussDB**.
4. On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

5. In the **Node List** area and **Network Information** area, view the IP address and port number.
   - If the ECS and GaussDB instance are in the same VPC, obtain the private IP address and database port number.
   - If the ECS and GaussDB instance are in different VPCs, obtain the EIP and database port number.

Testing Connectivity

1. Log in to the ECS. For details, see **Logging In to a Linux ECS Using VNC** in *Elastic Cloud Server User Guide*.
2. On the ECS, check whether it can connect to the target GaussDB instance using the IP address and port number obtained in **Querying the IP Address and Port Number of the Instance to Be Connected**.

   telnet *IP address Port number*

   Example:

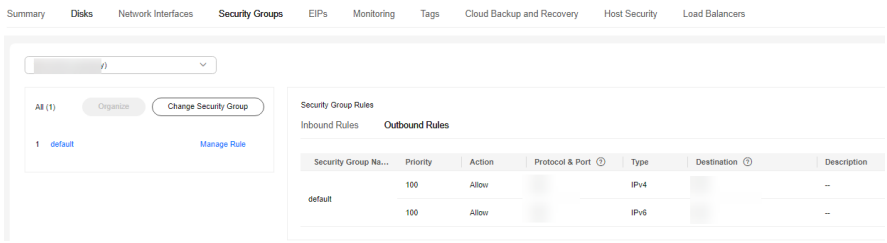
   telnet 192.168.0.16 8000

   NOTE

   If the message "command not found" is displayed, install a Telnet client that matches the OS of the ECS.

   - If the ECS can connect to the DB instance, no further action is required.
   - If the communication fails, check the security group rules.
     - On the **Outbound Rules** page of the ECS, add the IP address and port of the GaussDB instance to the outbound rules.
       - If the ECS and GaussDB instance are in the same VPC, add the private IP address and port of the GaussDB instance to the outbound rules.
       - If the ECS and GaussDB instance are in different VPCs, add the EIP address and port of the GaussDB instance to the outbound rules.

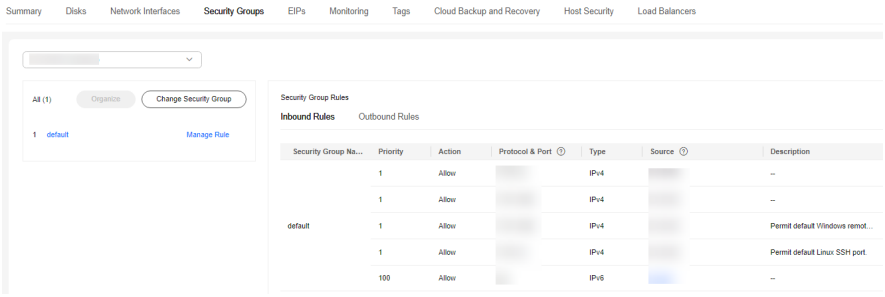
     Figure 3-8 ECS security group

     - On the **Inbound Rules** page of the GaussDB instance, add the IP address and port of the ECS to the inbound rules.
       - If the ECS and GaussDB instance are in the same VPC, add the private IP address and port of the ECS to the inbound rules.
       - If the ECS and GaussDB instance are in different VPCs, add the EIP address and port of the ECS to the inbound rules.

For details, see **Configuring Security Group Rules**.

Figure 3-9 GaussDB security group



Obtaining the Driver Package

Download particular packages listed in **Table 3-3** based on the version of your instance.

Table 3-3 Driver package download list

Version	Download Address
V2.0-8.x	**Driver package** **Verification package for the driver package**
V2.0-3.x	**Driver package** **Verification package for the driver package**
V2.0-2.x	**Driver package** **Verification package for the driver package**

To prevent a software package from being tampered with during transmission or storage, download the corresponding verification package and perform the following steps to verify the software package:

1. Upload the software package and verification package to the same directory on a Linux VM.
2. Run the following command to verify the integrity of the software package:

   cat *GaussDB_driver.zip.sha256* **| sha256sum --check**

   If **OK** is displayed in the command output, the verification is successful.

   ```
   GaussDB_driver.zip: OK
   ```

Connecting to a Database

- **Non-SSL connection**
  a. Log in as the **root** user to the ECS you have created.

b. Upload the client tool package and configure gsql environment variables.

i. Run the following command to create the **/tmp/tools** directory for storing the client tool package:

```
mkdir /tmp/tools
```

ii. Download the **GaussDB_driver.zip** driver package of the required version by referring to **Obtaining the Driver Package**, and upload it to the **/tmp/tools** directory of the created ECS.

iii. Run the following commands to decompress the **GaussDB_driver.zip** driver package:

```
cd /tmp/tools
unzip GaussDB_driver.zip
```

iv. Run the following commands to copy the decompressed **GaussDB-Kernel_***_EULER_64bit-Gsql.tar.gz** client tool package to the **/tmp/tools** directory:

NOTE

This section uses the gsql tool package suitable for the centralized instances running on Euler2.5_x86_64 as an example. The relative path of the tool package varies depending on where you decompressed it.

```
cd /tmp/tools/GaussDB_driver/Centralized/Euler2.5_X86_64/
cp GaussDB-Kernel_***_EULER_64bit-Gsql.tar.gz /tmp/tools
```

v. Run the following commands to decompress the package:

```
cd /tmp/tools
tar -zxvf GaussDB-Kernel_***_EULER_64bit-Gsql.tar.gz
```

vi. Configure environment variables.

Run the following command to open the **~/.bashrc** file:

```
vim ~/.bashrc
```

Press **G** to move the cursor to the last line, press **i** to enter Insert mode, and type the following information. Then, press **Esc** to exit Insert mode, and run **:wq** to save the settings and exit.

```
export PATH=/tmp/tools/bin:$PATH
export LD_LIBRARY_PATH=/tmp/tools/lib:$LD_LIBRARY_PATH
```

Run the following command to make the environment variables take effect permanently:

```
source ~/.bashrc
```

c. Enter the password when prompted to connect to the database.

After an instance is created, a **postgres** database is generated by default. Database **postgres** is used as an example.

```
gsql -d postgres -h 10.0.0.0 -U root -p 8000
Password for user root:
```

postgres is the name of the database you want to connect. **10.0.0.0** is the IP address of the instance obtained in **Querying the IP Address of the Instance to Be Connected**. **root** is the username for logging in to the database. **8000** is the database port obtained in **Querying the Port Number of the Instance to Be Connected**.

For more information about gsql commands, see **Tool Reference**.

- **SSL connection**

a. **Log in to the management console**.

b. Click  in the upper left corner and select a region and project.

c. Click ☰ in the upper left corner of the page and choose **Databases** > **GaussDB**.

d. On the **Instances** page, click the name of the target instance. In the **Configuration** area on the **Basic Information** page, click next to the **SSL** field to download the root certificate or certificate bundle.

e. Upload the root certificate to the ECS or save it to the device to be connected to the GaussDB instance.

Import the root certificate to the Linux ECS. For details, see **How Can I Import the Root Certificate to a Windows or Linux OS?**

f. Connect to a GaussDB instance.

A Linux ECS is used in this example. Run the following command to set environment variables on the ECS:

```
export PGSSLMODE=<sslmode>
export PGSSLROOTCERT=<ca-file-directory>
```

gsql -h *<host>* **-p** *<port>* **-d** *<database>* ***-U*** *<user>*

Table 3-4 Parameters

Parameter	Description
<host>	IP address of the DB instance. To obtain the IP address, click the instance name on the **Instances** page to go to the **Basic Information** page of the instance. The IP address can be found in the **Private IP Address** column of the **Node List** area.
<port>	Database port in use. The default value is **8000**. To obtain this parameter, go to the **Basic Information** page of the DB instance. The port number can be found in the **Database Port** field in the **Network Information** area.
<database>	Name of the database to be connected. The default management database is **postgres**.
<user>	Username of the GaussDB database account. The default administrator is **root**.
<ca-file-directory>	Path of the CA certificate for SSL connection.
<sslmode>	SSL connection mode. Set it to **verify-ca** to use a CA to check whether the service is trusted.

For example, to connect to a **postgres** database through an SSL connection as user **root**, run the following commands on the ECS:

```
export PGSSLMODE="verify-ca"
export PGSSLROOTCERT="/home/Ruby/ca.pem"
```

gsql -d *postgres* **-h** *10.0.0.0* **-U** *root* **-p** *8000*

```
Password for user root:
```

For more information about gsql commands, see **Tool Reference**.

g. Check the command output after you log in to the database. If information similar to the following is displayed, the SSL connection has been established.

```
SSL connection (cipher: DHE-RSA-AES256-GCM-SHA384, bits: 256)
```

3.4 Using Navicat to Connect to an Instance

Navicat Premium 16.2.8 for Windows PC now supports GaussDB management and development. This section describes how to use Navicat to connect to a GaussDB instance.

NOTE

This section uses Navicat 16.3.3 as an example. Navicat's graphical user interfaces (GUIs) differ depending on the version you are using. To learn more about how to perform operations in different versions, refer to the product documentation specific to that version.

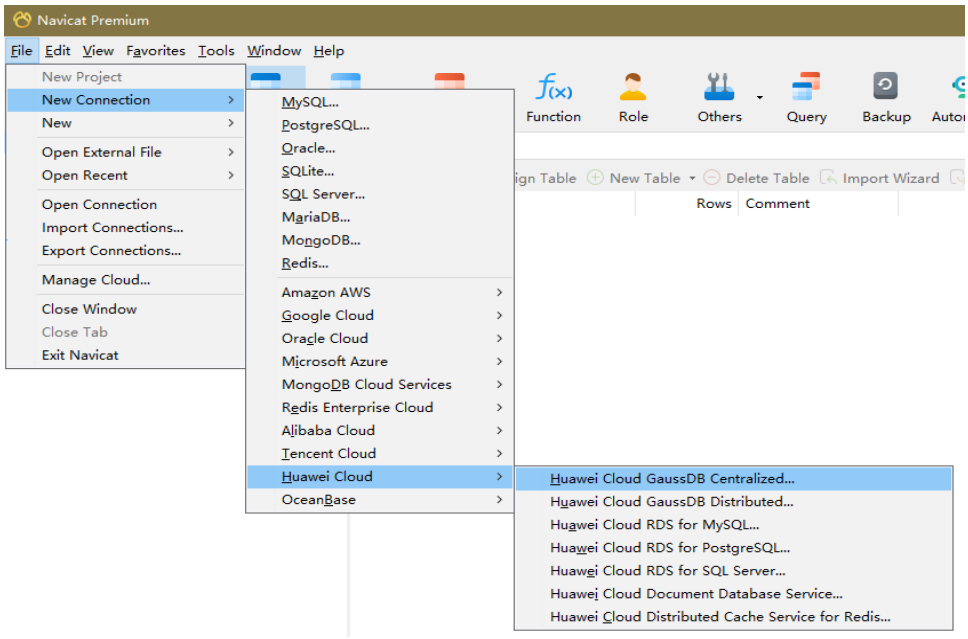
Prerequisites

You have **downloaded** or bought Navicat Premium and installed it on the local PC.

Procedure

Step 1 Start the Navicat Premium client and choose **File** > **New Connection** > **Huawei Cloud** > **Huawei Cloud GaussDB Centralized** or **Huawei Cloud GaussDB Distributed**.

Figure 3-10 Creating a connection



Step 2 In the **New Connection** window, enter the correct connection name, host, port, initial database, user name, and password.

Figure 3-11 Setting information for connecting to a centralized instance

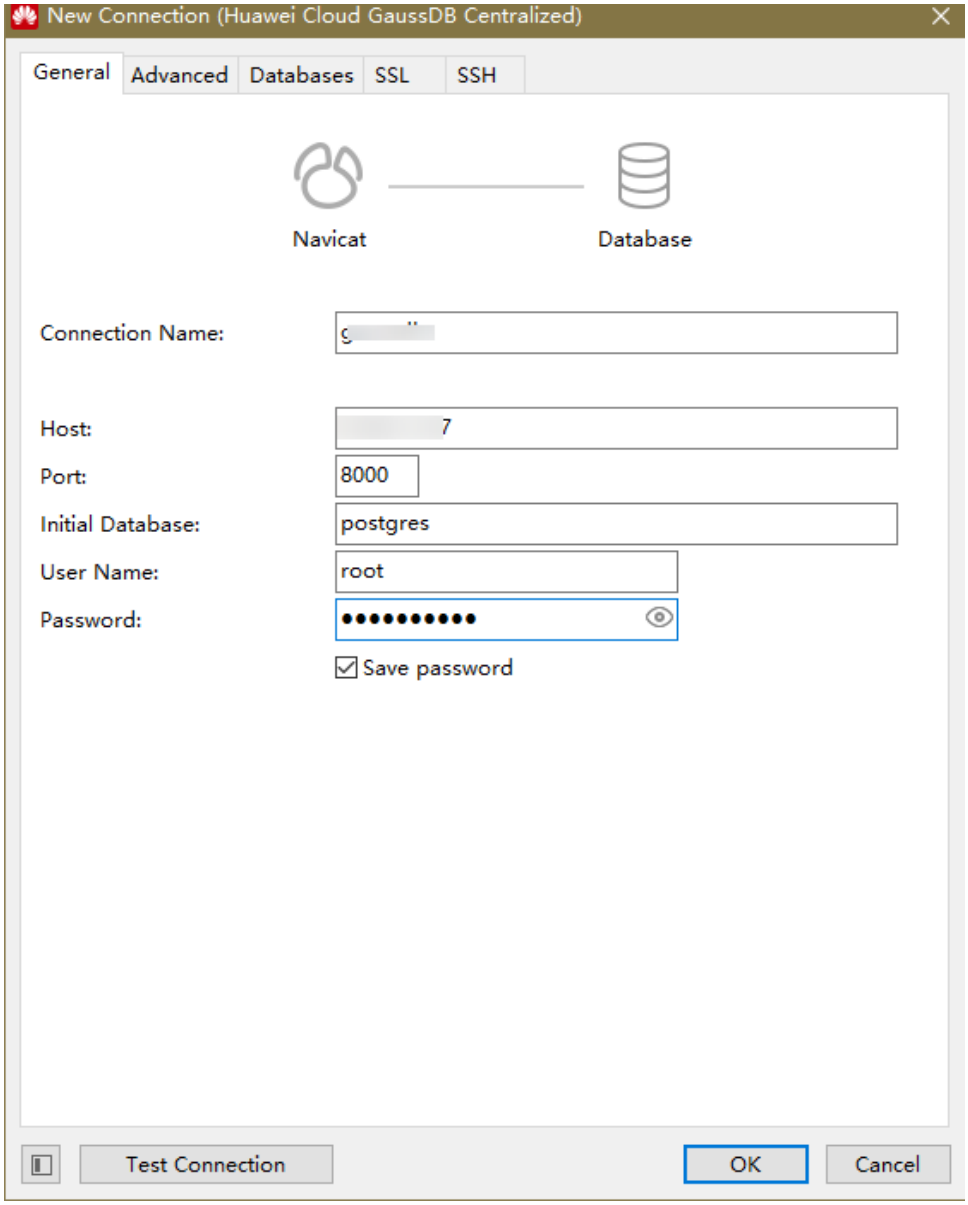


Figure 3-12 Setting information for connecting to a distributed instance

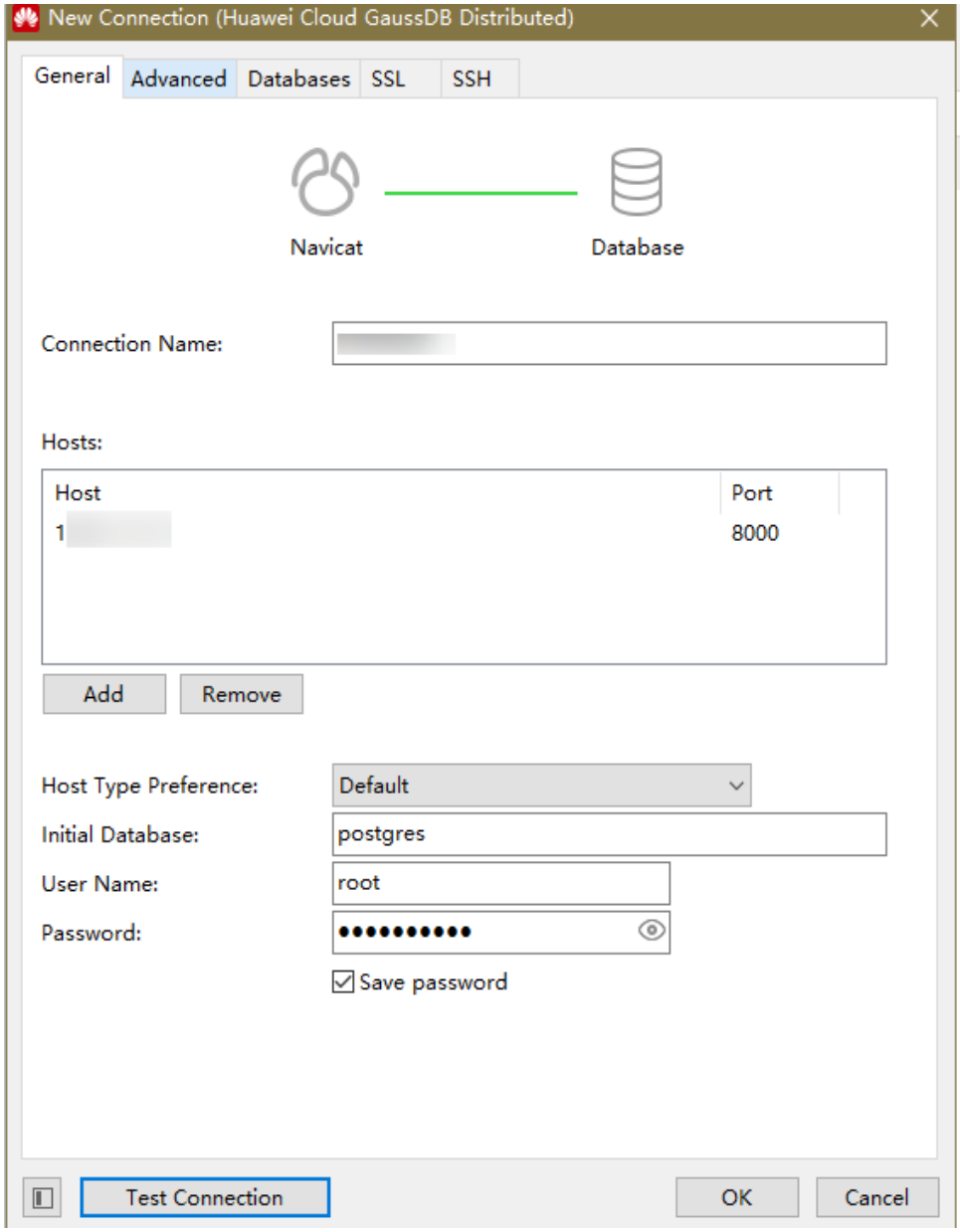


Table 3-5 Parameters

Parameter	Description
Connection Name	Use a name that is easy to identify.

Parameter	Description
Host	Private IP address of the DB instance to be connected. To obtain the IP address, perform the following steps: 1. Log in to the GaussDB management console. 2. Select the region in which the target instance is located. 3. Click the name of the target instance to enter the **Basic Information** page. 4. In the **Node List** area, view the EIP of the instance. If no EIP is bound to the instance, bind one to the instance first. For details, see **Binding an EIP**.
Port	Port of your DB instance specified during instance creation. The default port of a GaussDB instance is 8000.
Initial Database	Name of the database to be connected. After a DB instance is created, a database named **postgres** is generated by default.
User Name	Name of the user who will access the GaussDB instance. The default user is **root**.
Password	Password of the user who will access the GaussDB instance.

Step 3 Click **Test Connection**. If **Connection Successful** is displayed in the dialog box, the connection is normal. Click **OK** to close the dialog box.

Step 4 Click **OK**. The connection is disabled by default after being created.

Step 5 Right-click the connection name and choose **Open Connection** from the shortcut menu.

Step 6 Right-click the database name and choose **Open Database** from the shortcut menu.

----End

3.5 Using DBeaver to Connect to an Instance

DBeaver is a multi-platform database client for you to connect to different databases using particular drivers. This section describes how to use DBeaver to connect to a GaussDB instance.

NOTE

This section uses DBeaver 24.1.4 as an example. DBeaver's GUIs differ depending on the version you are using. To learn more about how to perform operations in different versions, refer to the product documentation specific to that version.

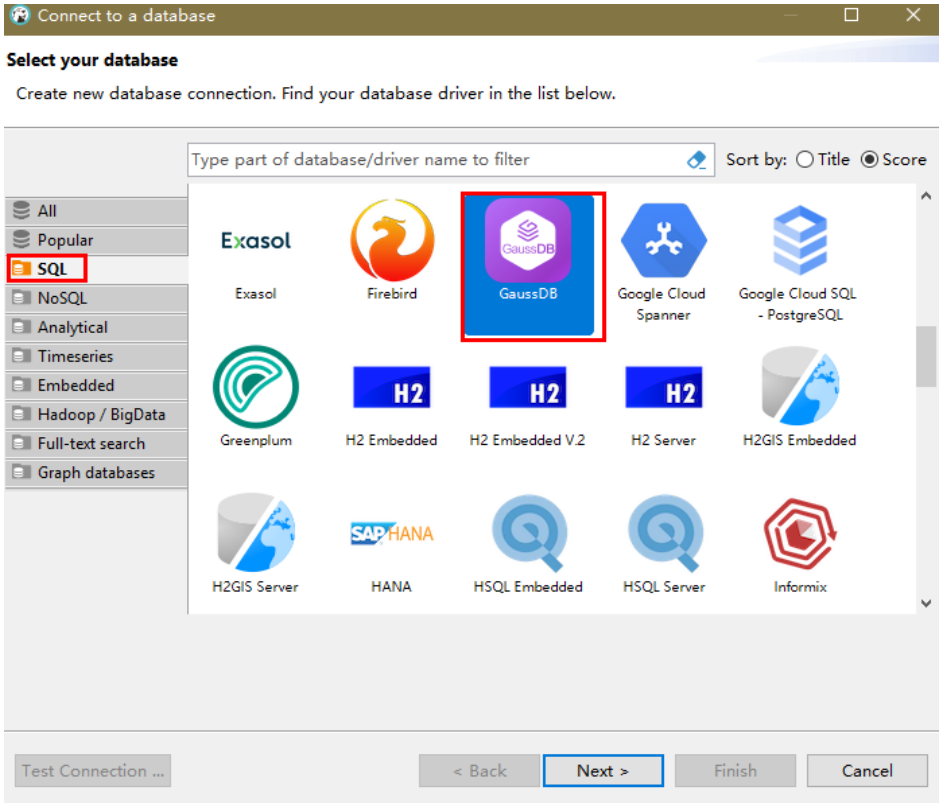
Step 1: Obtain the DBeaver Client Installation Package

The DBeaver official website provides client installation packages for different OSs. **Download** the required DBeaver client installation package, and install it on the local PC.

Step 2: Connect to the Database

1. On the DBeaver client, choose  to create a connection.
2. On the **SQL** tab, select **GaussDB** and click **Next**.

Figure 3-13 Selecting a database



3. Enter the host IP address, port number, database name, username, and password.

Figure 3-14 Configuring the connection

Table 3-6 Parameters

Parameter	Description
Host	Private IP address of the DB instance to be connected. To obtain the IP address, perform the following steps: 1. Log in to the GaussDB management console. 2. Select the region in which the target instance is located. 3. Click the name of the target instance to enter the **Basic Information** page. 4. In the **Node List** area, view the EIP of the instance. If no EIP is bound to the instance, bind one to the instance first. For details, see **Binding an EIP**.

Parameter	Description
Port	Port of your DB instance specified during instance creation. The default port of a GaussDB instance is 8000.
Database	Name of the database to be connected. After a DB instance is created, a database named **postgres** is generated by default.
Username	Name of the user who will access the GaussDB instance. The default user is **root**.
Password	Password of the user who will access the GaussDB instance.

4. Click **Test Connection**. In the **Driver settings** dialog box, click **Download** to enable the server to automatically download driver files. If **Connected** is displayed in the dialog box, the connection is successful. Click **OK**.

   Alternatively, **obtain driver files** by yourself and click **Edit Driver**. Then, click the **Libraries** tab and add the local driver files to the driver editor.

Figure 3-15 Automatically downloading driver files

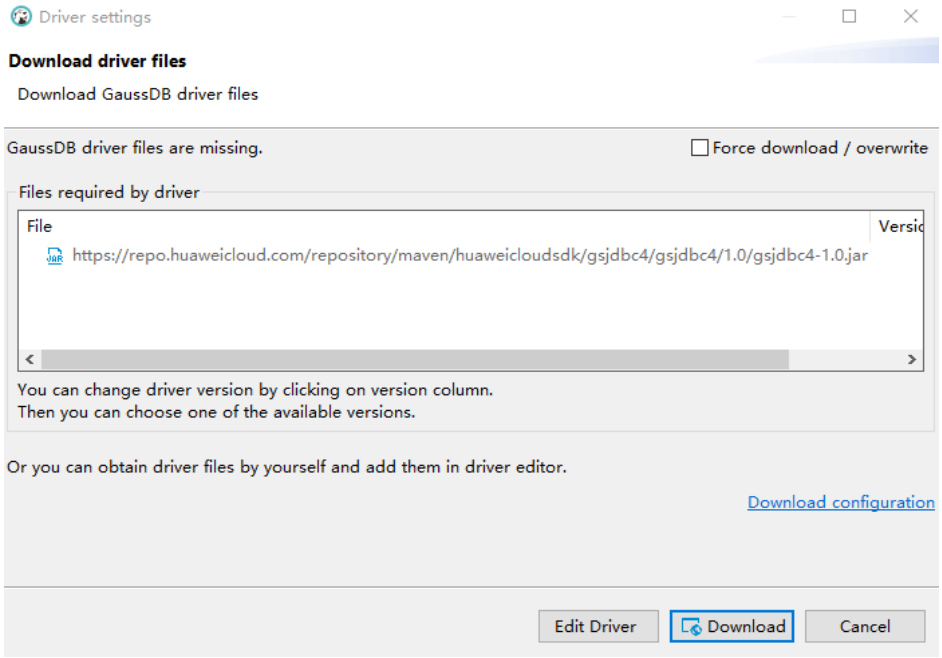
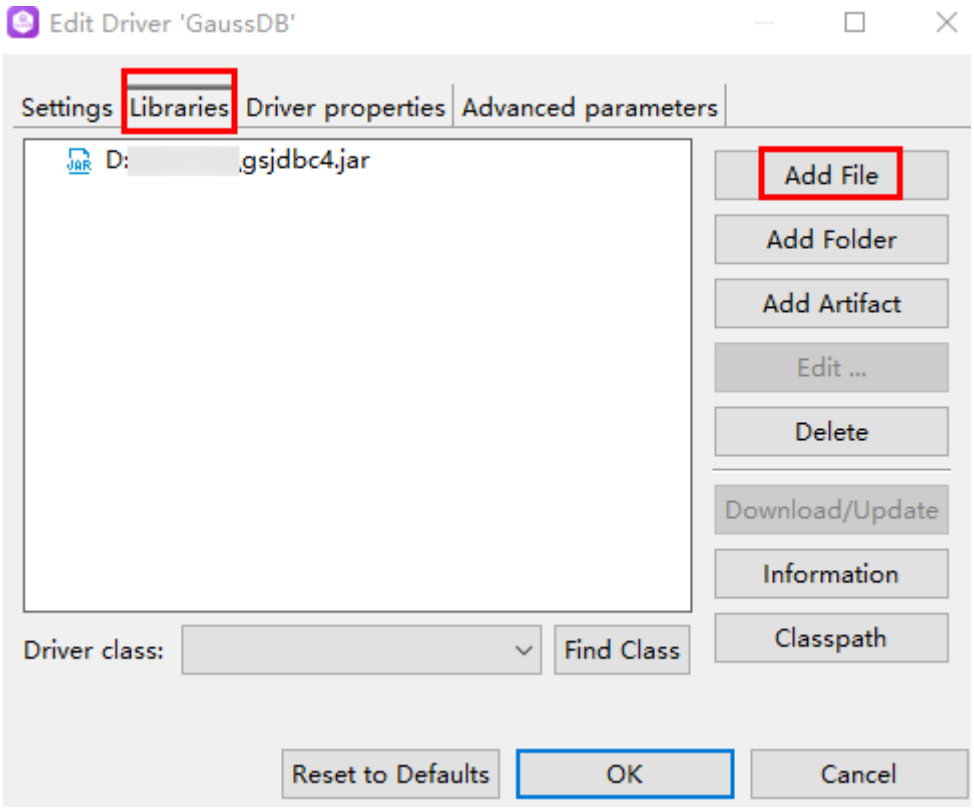


Figure 3-16 Manually adding local driver files



5. Click **Finish** to connect to the database. You can view information about the connected database in the **Database Navigator** area.

Related Operations

1. Obtain the driver package and its verification package.

   Download the driver package and its verification package of the relevant version to any local directory. **Table 3-7** lists the download list.

   Table 3-7 Driver package download list

Version	Download Address
V2.0-8.x	**Driver package** **Verification package for the driver package**
V2.0-3.x	**Driver package** **Verification package for the driver package**
V2.0-2.x	**Driver package** **Verification package for the driver package**

2. Verify the driver package.

To prevent the driver package from being maliciously tampered during transfer or storage, perform the following steps to verify the driver package:

a. Press **Win+R** to open the **Run** text box. Type **cmd** in the **Open** field and press **Enter** to open the **Command Prompt** window.

b. Run the following command to obtain the hash value of the driver package:

certutil -hashfile *{Local directory of the driver package}**{Driver package name}* **sha256**

- Replace *{Local directory of the driver package}* with the actual download path, for example, **C:\Users**.
- Replace *{Driver package name}* with the name of the downloaded driver package, for example, **GaussDB_driver.zip**.

Example: **certutil -hashfile C:\Users\GaussDB_driver.zip sha256**

c. Compare the hash value obtained in **2.b** with the hash value of the verification package obtained in **1**.

- If they are consistent, the verification is successful.
- If they are inconsistent, download the driver package again and repeat **2.a** to **2.c** to verify the driver package.

3. Extract the **gsjdbc4.jar** package from the driver package.

Decompress the driver package obtained in **1** to the local PC. Then, go to any OS directory in the directory of the driver package corresponding to the type of the instance to be connected, extract the **gsjdbc4.jar** package from the **GaussDB-Kernel**_*Database version*_*OS version*_**64bit_Jdbc.tar.gz** package, and save it to any local directory. The following is an example:

To connect to a distributed instance, go to the **GaussDB_driver\Distributed \Euler2.5_X86_64** directory, find the **GaussDB-Kernel_503.1.0.SPC2300_Euler_64bit_Jdbc.tar.gz** package, and extract the **gsjdbc4.jar** package from it.

NOTE

The same JDBC driver package is used across different operating systems and CPU architectures. You only need to focus on the instance type when obtaining the required **gsjdbc4.jar** package.

4 Database Migration

4.1 Overview of GaussDB Migration Solutions

You can migrate data from MySQL, PostgreSQL, Oracle, Db2 for LUW, RDS for SQL Server, or Microsoft SQL Server databases to GaussDB, or from one GaussDB instance to another GaussDB instance.

Data migration tools include DRS, DAS, and gs_loader. You are advised to use DRS because it is easy to use and can complete a migration task in minutes. GaussDB migration service helps you reduce DBA labor costs, hardware costs, and data transmission costs.

Data migration tools include gs_dump, gs_dumpall, gs_restore, **copy** commands, and CopyManager.

Table 4-1 GaussDB migration solutions

Solution	Data Source	Description	Reference
Using DRS to import data to GaussDB	MySQL	Real-time data synchronization of DRS allows you to copy data from a data source to GaussDB to implement real-time data flow of key services. It focuses on the synchronous import of tables and data.	**Using DRS to Migrate Data from MySQL Database to GaussDB**
	Oracle		**Using DRS to Migrate Data from Oracle Database to GaussDB**
	GaussDB (distributed deployment)		• **From GaussDB Distributed to GaussDB Distributed** • **From GaussDB Distributed to GaussDB Centralized**

Solution	Data Source	Description	Reference
	GaussDB (centralized deployment)		• From GaussDB Centralized to GaussDB Distributed • From GaussDB Centralized to GaussDB Centralized
	Db2 for LUW		• From Db2 for LUW to GaussDB Centralized • From Db2 for LUW to GaussDB Distributed
	PostgreSQL		• From PostgreSQL to GaussDB Centralized • From PostgreSQL to GaussDB Distributed
	SQL Server		• From Microsoft SQL Server to GaussDB Centralized • From Microsoft SQL Server to GaussDB Distributed
Using DAS to export and import data	SQL/CSV files	You can use DAS to export data from the source database first and then import the data from your local PC or OBS bucket to the destination database.	Migrating Data to GaussDB Using the Export and Import Functions of DAS

Solution	Data Source	Description	Reference
Using the **copy to/from** command to export and import data	CSV files	The gsql tool provides the **\copy** meta-command to import or export data. **\copy** applies only to small-scale data import in good format. It does not preprocess invalid characters or provide error tolerance. Therefore, **\copy** cannot be used in scenarios where abnormal data exists.	**Using the copy to/from Command to Export and Import Data**
Using CopyManager in JDBC to export and import data	Other files or databases	When you use Java to develop applications, the CopyManager API of the JDBC driver is called to write data from files or other databases to GaussDB.	**Using CopyManager in JDBC to Export and Import Data**

Solution	Data Source	Description	Reference
Using gs_dump and gs_dumpall to export data	• Plain-text archives • Custom-format archives • Directory-format archives • TAR-format archives	gs_dump can export a single database or its objects. gs_dumpall can export all databases or global objects in a cluster. You can use a tool to import the exported metadata to a destination database for database migration.	**Using gs_dump and gs_dumpall to Export Data**

Solution	Data Source	Description	Reference
Using gs_restore to import data	SQL/TMP/TAR files	During database migration, you can use gs_restore to import files exported by gs_dump to GaussDB. In this way, metadata, such as table definitions and database object definitions, can be imported. The imported data includes: • Object definitions of all databases • Object definitions of a single database • Definitions of a single schema • Definitions of a single table	**Using gs_restore to Import Data**

Solution	Data Source	Description	Reference
Using gs_loader to import data	CSV files	You can use gs_loader to import the files exported by using the **copy to** command. The gs_loader tool converts the syntax supported by control files into **\copy** syntax, then leverages the existing **\copy** function to import main data. At the same time, gs_loader logs the results of the **\copy** operations to a log file.	**Using gs_loader to Import Data**

4.2 Using DRS to Migrate Data from Oracle Database to GaussDB

Scenarios

This section describes how to use real-time synchronization of DRS to migrate data from an on-premises Oracle database to Huawei Cloud GaussDB in real time. Full+incremental synchronization can ensure that data is always in sync between the source Oracle database and the destination GaussDB instance. Full synchronization is used to synchronize data. Incremental synchronization is used to synchronize data between the source and destination databases in real time.

Step 1: Create a VPC and Security Group

Step 2: Create a GaussDB Instance

Step 3: Construct Data Before Migration

Step 4: Migrating the Database

Step 5: Verify Data After Migration

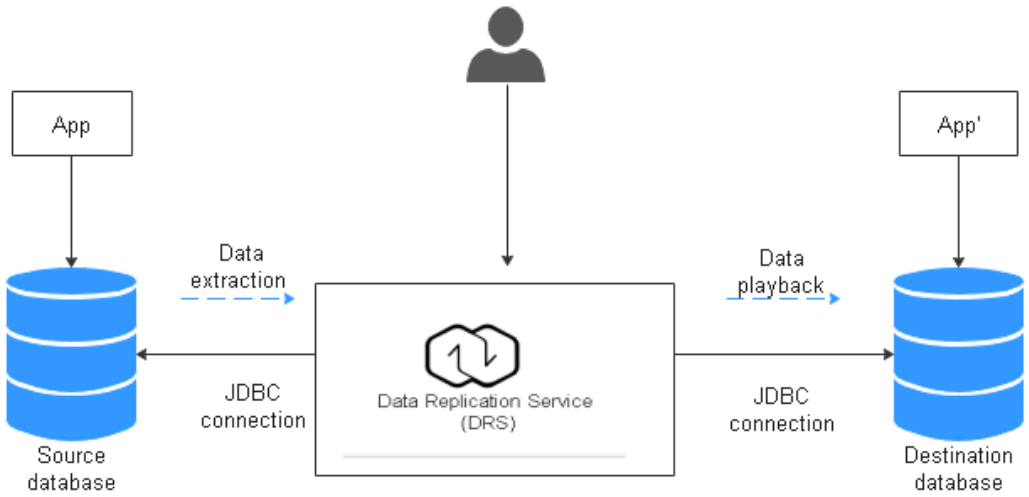
Problems to Resolve

- Enterprise workloads have been growing and evolving fast, and traditional databases lack the scalability needed to keep up. Enterprises need distributed databases.
- Building a traditional database means purchasing and installing servers, systems, databases, and other software. The O&M is expensive and difficult.
- Traditional databases have poor performance when it comes to handling complex queries.
- It is hard for traditional databases to smoothly synchronize data with no downtime.

Prerequisites

- You have registered with Huawei Cloud and completed account authentication.
- Your account balance is greater than or equal to $0 USD.
- In a testing scenario, you have set up an on-premises Oracle database.
- You have obtained the IP address, port number, username, and password of the Oracle database to be migrated.

Service Architecture



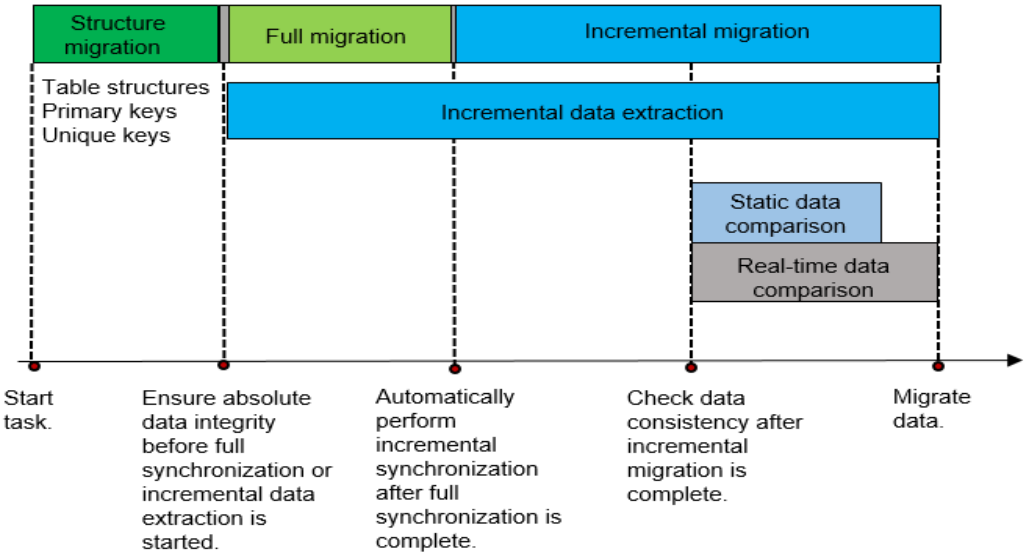
How Data Migration Works

The data migration process is completed using full and incremental synchronization, which includes the following operations:

1. In the full synchronization phase, schemas, including tables, primary keys, and unique keys, are synchronized first.
2. After schemas are synchronized, incremental data extraction is started to ensure that the incremental data generated during full data synchronization is completely extracted to the DRS instance.

3. A full migration task is started.
4. An incremental synchronization is automatically started after the full migration is complete. The replay starts from the position where the full synchronization starts.
5. A comparison task is started after the incremental replay is complete to check the data consistency. Real-time comparison is supported.
6. Workloads synchronization is started if the data is consistent between the source and destination databases.

Figure 4-1 Migration principle



Resource Planning

The resource planning in this section is just an example. You need to adjust it as needed.

Table 4-2 Resource planning

Category	Item	Planned Value	Remarks
VPC	VPC name	vpc-src-172	Specify a name that is easy to identify.
	Region	Test region	To achieve lower network latency, select the region nearest to you.
	AZ	AZ 3	-
	Subnet CIDR block	172.16.0.0/16	Select a subnet with sufficient network resources.

Category	Item	Planned Value	Remarks
	Subnet name	subnet-src-172	Specify a name that is easy to identify.
On-premises Oracle database	Name	orcl	Specify a name that is easy to identify.
	Specifications	16 vCPUs | 32 GB	-
	Database version	11.2.0.1	-
	Database user	test_info	Specify a username. The user must have the following permissions during migration: CREATE SESSION, SELECT ANY TRANSACTION, SELECT ANY TABLE, SELECT ANY DICTIONARY, and EXECUTE_CATALOG_ROLE.
GaussDB	Instance name	Auto-drs-gaussdbv5-tar-1	Specify a name that is easy to identify.
	Database version	GaussDB V2.0-8.103 Enterprise edition	-
	Instance type	Distributed (3 CNs, 3 DN shards, and 3 replicas)	In this example, a distributed instance will be created.
	Deployment model	Independent	-
	Transaction consistency	Strong consistency	-
	Shards	3	-
	Coordinator nodes	3	-
	Storage type	Ultra-high I/O	-

Category	Item	Planned Value	Remarks
	AZ	AZ 2	In this example, a single AZ is select. You are advised to select multiple AZs to improve instance availability in actual use.
	Instance Specifications	Dedicated (1:8); 8 vCPUs | 64 GB	Small specifications are selected for this test instance. You are advised to configure specifications based on service requirements in actual use.
	Storage space	480 GB	A small storage space is selected for this test instance. You are advised to configure the storage space based on service requirements in actual use.
	Disk encryption	Disable	In this example, disk encryption is disabled. Enabling disk encryption improves the security of data, but may slightly affect the database read/write performance.
Logging in to the database through DAS	Database engine	GaussDB	-
	Database source	GaussDB	Select the GaussDB instance created in this example.
	Database name	postgres	-
	Username	root	-
	Password	-	Enter the password of the **root** user of the GaussDB instance created in this example.
DRS migration task	Migration task name	DRS-test-info	Specify a name that is easy to identify.

Category	Item	Planned Value	Remarks
	Destination database name	test_database_info	Specify a name that is easy to identify. The name must be compatible with the Oracle database name.
	Source database engine	Oracle	-
	Destination database engine	GaussDB	-
	Network type	Public network	In this example, a public network is used.

Step 1: Create a VPC and Security Group

Create a VPC and security group for the GaussDB instance.

Creating a VPC

1. Log in to the **Huawei Cloud console**.
2. Click  in the upper left corner and select a region.
3. Click the service list icon on the left and choose **Networking** > **Virtual Private Cloud**. The VPC console is displayed.
4. Click **Create VPC**.

Figure 4-2 Basic information

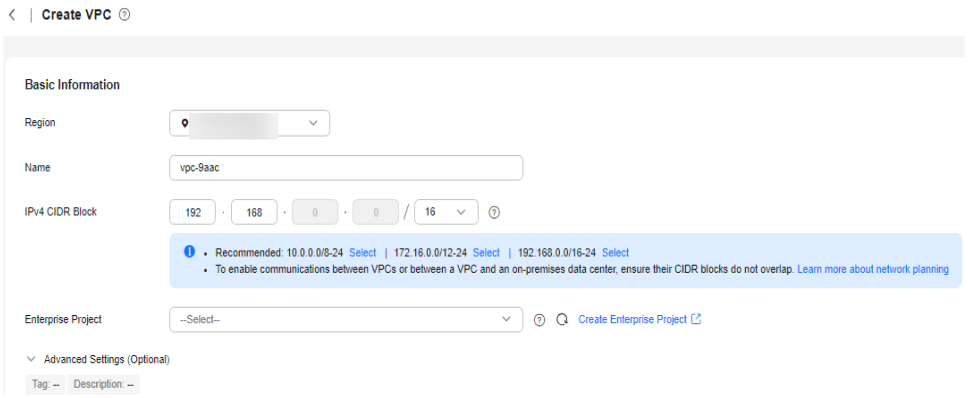
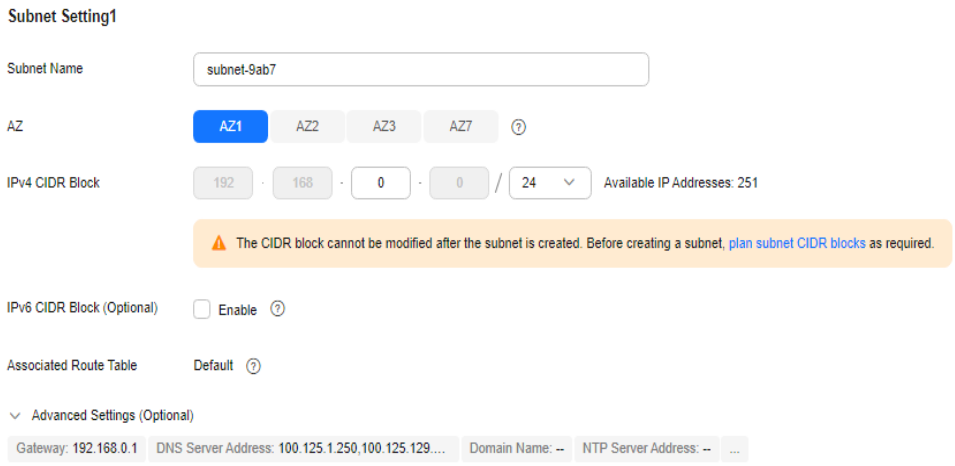


Figure 4-3 Setting a subnet

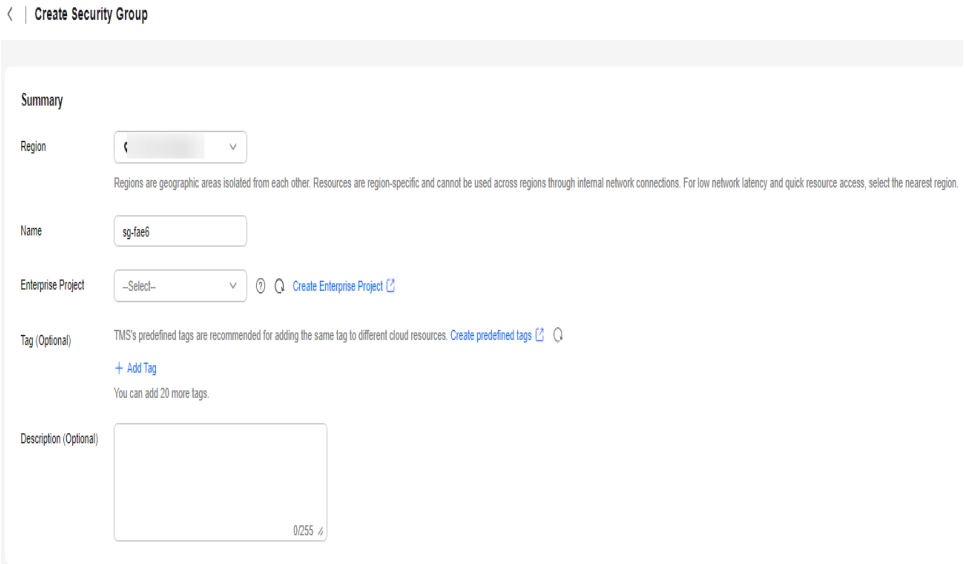


5. Configure parameters as needed and click **Create Now**.
6. Return to the VPC list and check whether the VPC is created.
   If the VPC status becomes available, the VPC has been created.

Creating a Security Group

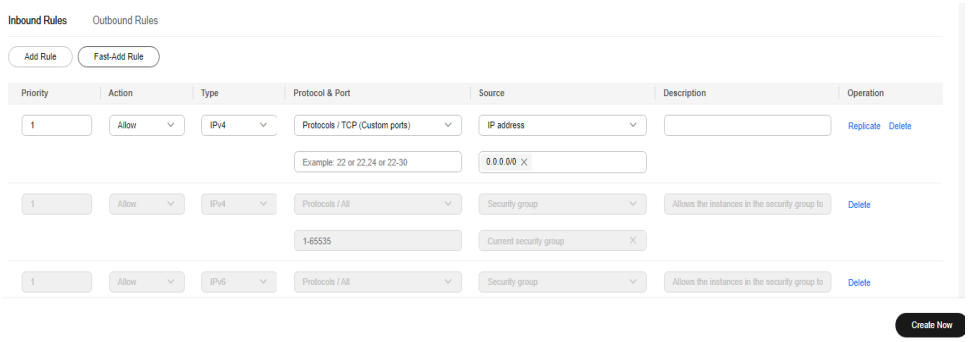
1. Log in to the **Huawei Cloud console**.
2. Click  in the upper left corner and select a region.
3. Click the service list icon on the left and choose **Networking** > **Virtual Private Cloud**.
   The VPC console is displayed.
4. In the navigation pane, choose **Access Control** > **Security Groups**.
5. Click **Create Security Group**.
6. Specify a security group name and other information.

Figure 4-4 Basic information



7. Click the **Inbound Rules** tab and then click **Add Rule**.
8. Configure an inbound rule, add the IP address of the source database, and click **OK**.

Figure 4-5 Configuring an inbound rule



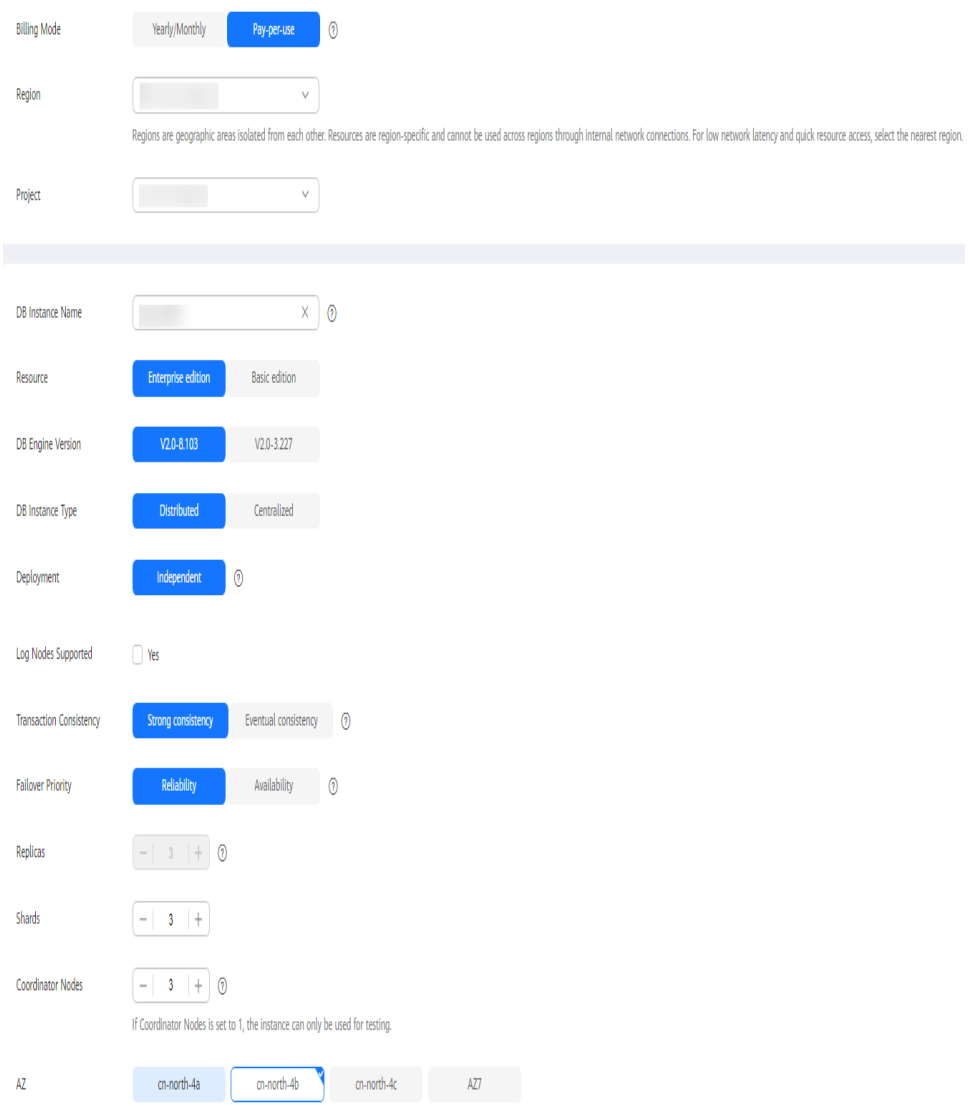
9. Configure parameters as needed and click **Create Now**.

Step 2: Create a GaussDB Instance

Create a GaussDB instance as the destination database of the migration task.

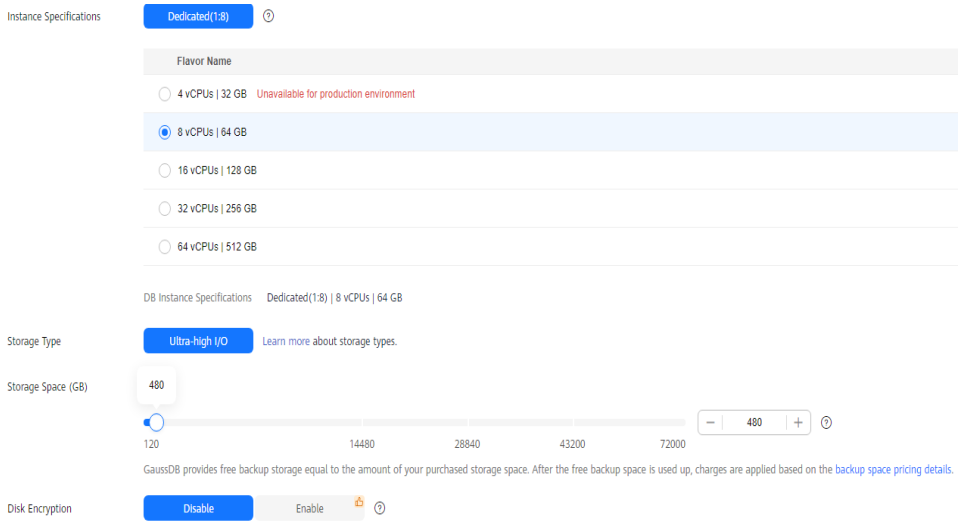
1. Log in to the **Huawei Cloud console**.
2. Click  in the upper left corner and select a region.
3. Click the service list icon on the left and choose **Databases** > **GaussDB**.
4. In the navigation pane on the left, choose **GaussDB** > **Instances**.
5. Click **Buy DB Instance**.
6. On the page shown in **Figure 4-6**, configure basic information about the instance, including the instance name, billing mode, edition type, DB engine version, instance type, transaction consistency, number of shards, number of coordinator nodes, and deployment AZ.

Figure 4-6 Basic information



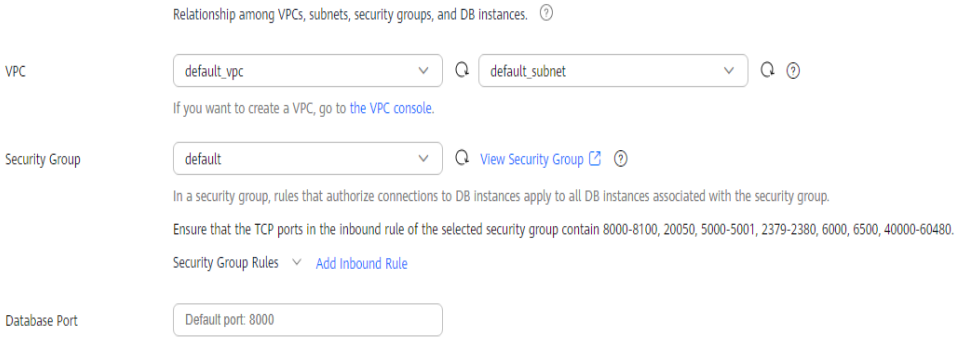
7. Select the instance specifications and storage space.

Figure 4-7 Instance specifications



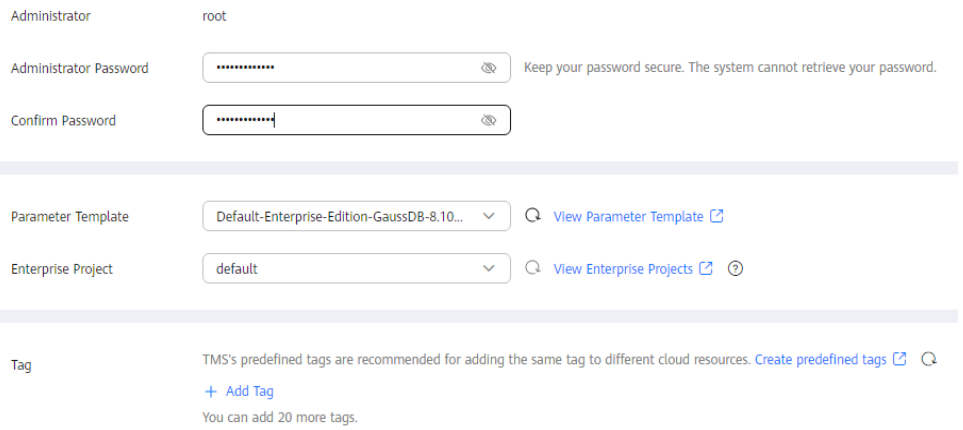
8. Select the VPC created in **Creating a VPC** and security group created in **Creating a Security Group** for the instance and configure the database port.

Figure 4-8 Selecting a VPC and security group



9. Configure the password and other information.

Figure 4-9 Configuring the password and other information



10. Click **Next**, confirm the information, and click **Submit**.
11. Go to the instance list.

    If status of the instance becomes **Available**, the instance has been created.

Step 3: Construct Data Before Migration

Before the migration, prepare some data types in the source database for verification after the migration is complete. The end-to-end test data in this section is for reference only.

The following table lists data types supported by DRS.

Table 4-3 Data type mapping

Source Data Type	Destination Data Type	Sync (Source Data Type as Primary Key)	Sync (Source Data Type as Non-Primary Key)	Comparison (Source Data Type as Primary Key)	Comparison (Source Data Type as Non-Primary Key)	Remarks
CHAR	character	Supported	Supported	Supported. The spaces before and after a character are ignored.	Supported. The spaces before and after a character are ignored.	-
VARCHAR	character varying	Supported	Supported	Supported	Supported	The precision ranges of the source and destination databases are different, causing precision loss.
VARCHAR2	character varying	Supported	Supported	Supported	Supported	-

Source Data Type	Destination Data Type	Sync (Source Data Type as Primary Key)	Sync (Source Data Type as Non-Primary Key)	Comparison (Source Data Type as Primary Key)	Comparison (Source Data Type as Non-Primary Key)	Remarks
NCHAR	character	Supported	Supported	Supported. The spaces before and after a character are ignored.	Supported. The spaces before and after a character are ignored.	-
NVARCHAR2	nvarchar2	Supported	Supported	Supported	Supported	-
NUMBER	numeric	Supported	Supported	Supported	Supported	-
NUMBER (6,3)	numeric(6,3)	Supported	Supported	Supported	Supported	-
NUMBER (6,0)	Integer	Supported	Supported	Supported	Supported	-
NUMBER (3)	smallint	Supported	Supported	Supported	Supported	-
NUMBER (6,-2)	integer	Supported	Supported	Supported	Supported	-
BINARY_FLOAT	real	Not supported (The destination database does not support creating tables using the primary key.)	Supported	Not supported	Supported	The precision ranges of the source and destination databases are different, causing precision loss.

Source Data Type	Destination Data Type	Sync (Source Data Type as Primary Key)	Sync (Source Data Type as Non-Primary Key)	Comparison (Source Data Type as Primary Key)	Comparison (Source Data Type as Non-Primary Key)	Remarks
BINARY_DOUBLE	double precision	Not supported (The destination database does not support creating tables using the primary key.)	Supported	Not supported	Supported	-
FLOAT	real	Not supported (The destination database does not support creating tables using the primary key.)	Supported	Not supported	Supported	The precision ranges of the source and destination databases are different, causing precision loss.
INT	numeric	Supported	Supported	Supported	Supported	-
INTEGER	numeric	Supported	Supported	Supported	Supported	-

Source Data Type	Destination Data Type	Sync (Source Data Type as Primary Key)	Sync (Source Data Type as Non-Primary Key)	Comparison (Source Data Type as Primary Key)	Comparison (Source Data Type as Non-Primary Key)	Remarks
DATE	date	Supported	Supported	Not supported	Supported	If a table with the date type is created in the destination database, the data type precision range in the source database is different from that in the destination database, causing precision loss. Therefore, comparison is not supported.
TIMESTAMP	timestamp(6) without time zone	Supported	Supported	Not supported	The value is accurate to six decimal places.	Restrictions on the source database: The maximum precision supported by the source database is 6.

Source Data Type	Destination Data Type	Sync (Source Data Type as Primary Key)	Sync (Source Data Type as Non-Primary Key)	Comparison (Source Data Type as Primary Key)	Comparison (Source Data Type as Non-Primary Key)	Remarks
TIMESTAMP_TZ	timestamp(6) with time zone	Not supported (The source database does not support creating tables using the primary key.)	Supported	Not supported	Filter out this column.	-
TIMESTAMP_LTZ	timestamp(6) with time zone	Not supported (The destination database does not support creating tables using the primary key.)	Supported	Not supported	Filter out this column.	-

Source Data Type	Destination Data Type	Sync (Source Data Type as Primary Key)	Sync (Source Data Type as Non-Primary Key)	Comparison (Source Data Type as Primary Key)	Comparison (Source Data Type as Non-Primary Key)	Remarks
INTERVAL_YM	interval year to month	Supported	Supported	Not supported	Not supported	Incremental synchronization does not support this type.
INTERVAL_DS	interval day to second	Supported	Supported	Not supported	Not supported	Incremental synchronization does not support this type. Restrictions on the source database: The maximum precision supported by the source database is 6.

Source Data Type	Destination Data Type	Sync (Source Data Type as Primary Key)	Sync (Source Data Type as Non-Primary Key)	Comparison (Source Data Type as Primary Key)	Comparison (Source Data Type as Non-Primary Key)	Remarks
BLOB	bytea	Not supported (The source database does not support creating tables using the primary key.)	Supported	Not supported	Filter out this column.	-
CLOB	text	Not supported (The source database does not support creating tables using the primary key.)	Supported	Not supported	Filter out this column.	-

Source Data Type	Destination Data Type	Sync (Source Data Type as Primary Key)	Sync (Source Data Type as Non-Primary Key)	Comparison (Source Data Type as Primary Key)	Comparison (Source Data Type as Non-Primary Key)	Remarks
NCLOB	text	Not supported (The source database does not support creating tables using the primary key.)	Supported	Not supported	Filter out this column.	-
LONG	text	Not supported (The source database does not support creating tables using the primary key.)	Supported	Not supported	Filter out this column.	-

Source Data Type	Destination Data Type	Sync (Source Data Type as Primary Key)	Sync (Source Data Type as Non-Primary Key)	Comparison (Source Data Type as Primary Key)	Comparison (Source Data Type as Non-Primary Key)	Remarks
LONG_RAW	bytea	Not supported (The source database does not support creating tables using the primary key.)	Supported	Not supported	Filter out this column.	-
RAW	bytea	Not supported (The destination database does not support creating tables using the primary key.)	Supported	Not supported	Supported	-
RowID	character varying(18)	Supported	Supported	Supported	Supported	-

Source Data Type	Destination Data Type	Sync (Source Data Type as Primary Key)	Sync (Source Data Type as Non-Primary Key)	Comparison (Source Data Type as Primary Key)	Comparison (Source Data Type as Non-Primary Key)	Remarks
BFILE	-	Not supported	Not supported	Not supported	Not supported	Restrictions on the source database: The BFILE type is not supported.
XMLTYPE	-	Not supported	Not supported	Not supported	Not supported	Restrictions on the source database: The XMLTYPE type is not supported.
UROWID	-	Not supported	Not supported	Not supported	Not supported	Full and incremental synchronizations are not supported.
sdo_geometry	-	Not supported	Not supported	Not supported	Not supported	Restrictions on the source database: The sdo_geometry type is not supported.
NUMBER(*, 0)	numeric	Supported	Supported	Supported	Supported	-

Perform the following steps to construct data in the source database:

1. Use a database connection tool to connect to the source Oracle database based on its IP address.
2. Construct data in the source database based on data types supported by DRS.

a. Create a test user.

create user *test_info* **identified by** *xxx*;

test_info indicates the user created for the migration, and *xxx* indicates the password of the user.

b. Assign permissions to the user.

grant dba to *test_info*;

c. Create a data table for the user.

```
CREATE TABLE test_info.DATATYPELIST(
ID INT,
COL_01_CHAR_____E CHAR(100),
COL_02_NCHAR____E NCHAR(100),
COL_03_VARCHAR__E VARCHAR(1000),
COL_04_VARCHAR2_E VARCHAR2(1000),
COL_05_NVARCHAR2_E NVARCHAR2(1000),
COL_06_NUMBER___E NUMBER(38,0),
COL_07_FLOAT____E FLOAT(126),
COL_08_BFLOAT___E BINARY_FLOAT,
COL_09_BDOUBLE__E BINARY_DOUBLE,
COL_10_DATE_____E DATE DEFAULT SYSTIMESTAMP,
COL_11_TS_______E TIMESTAMP(6),
COL_12_TSTZ_____E TIMESTAMP(6) WITH TIME ZONE,
COL_13_TSLTZ____E TIMESTAMP(6) WITH LOCAL TIME ZONE,
COL_14_CLOB_____E CLOB DEFAULT EMPTY_CLOB(),
COL_15_BLOB_____E BLOB DEFAULT EMPTY_BLOB(),
COL_16_NCLOB____E NCLOB DEFAULT EMPTY_CLOB(),
COL_17_RAW______E RAW(1000),
COL_19_LONGRAW__E LONG RAW,
COL_24_ROWID____E ROWID,
PRIMARY KEY(ID)
);
```

d. Insert two rows of data.

```
insert into test_info.DATATYPELIST
values(4,'huawei','xian','shanxi','zhongguo','shijie',
666,12.321,1.123,2.123,sysdate,sysdate,sysdate,sysdate,'hw','cb','df','F
F','FF','AAAYEVAAJAAAACrAAA');

insert into test_info.DATATYPELIST values(2,'Migrate-
test','test1','test2','test3','test4',
666,12.321,1.123,2.123,sysdate,sysdate,sysdate,sysdate,'hw','cb','df','F
F','FF','AAAYEVAAJAAAACrAAA');
```

e. Commit the changes to the database.

```
commit;
```

3. Create a database in the destination GaussDB instance.

a. Log in to the **Huawei Cloud console**.

b. Click in the upper left corner and select a region.

c. Click the service list icon on the left and choose **Databases** > **Data Admin Service**.

d. In the navigation pane on the left, choose **Development Tool** to go to the login list page.

e. Click **Add Login**.

f. On the displayed page, select the DB engine, source database, and target instance, enter the login username, password, and description (optional), and enable **Show Executed SQL Statements**.

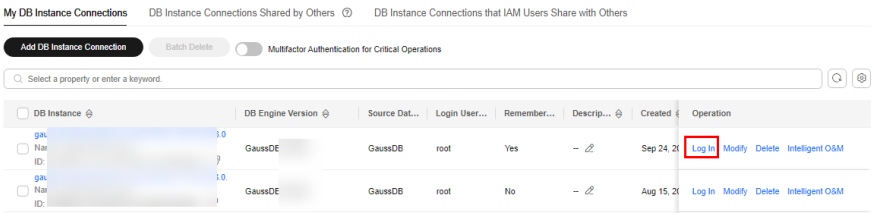
g. Click **Test Connection** to check whether the connection is successful.

If a message is displayed indicating the connection is successful, continue with the operation. If a message is displayed indicating the connection failed and the failure cause is provided, make modifications according to the error message.

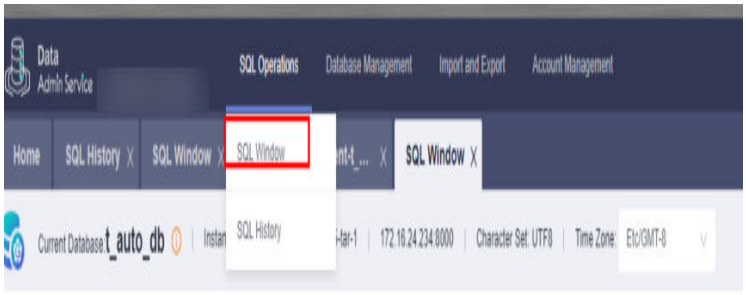
h. Click **OK**.

i. Locate the added record and click **Log In** in the **Operation** column.

Figure 4-10 Logging in to an instance



j. Choose **SQL Operations** > **SQL Window** on the top menu bar.



k. Run the following statement to create an Oracle-compatible database:

test_database_info indicates the database name. Replace it as required.

```
CREATE DATABASE test_database_info DBCOMPATIBILITY 'ORA';
```

Step 4: Migrating the Database

Create a DRS instance and migrate data from the **test_info** database in the on-premises Oracle database to the **test_database_info** database in the GaussDB instance.

Performing a Pre-migration Check

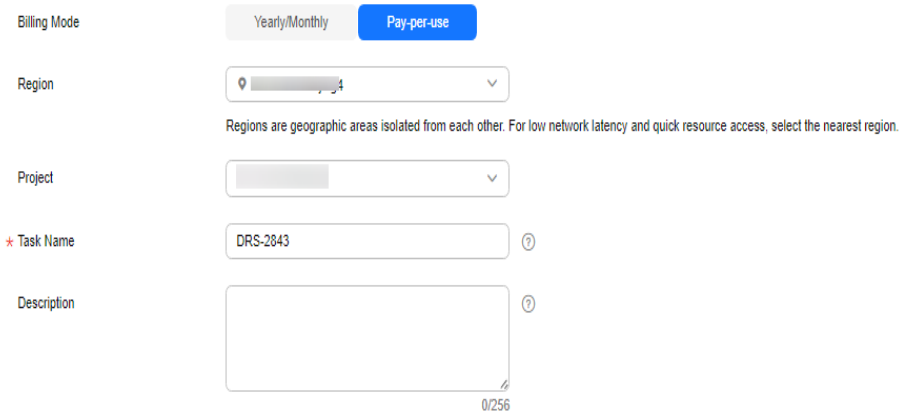
Before creating a migration task, check the migration conditions to ensure smooth migration.

Before the migration, you need to obtain the **notes on migration to the cloud**.

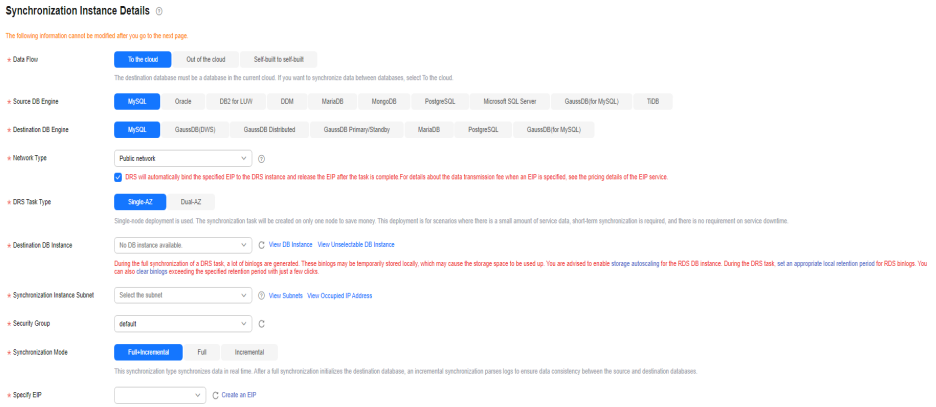
Creating a Migration Task

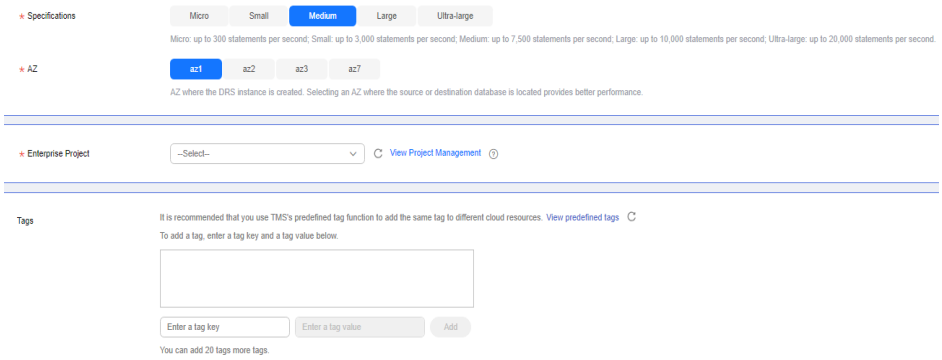
1. Log in to the **Huawei Cloud console**.
2. Click  in the upper left corner and select a region.

   Select the region where the destination instance is deployed.
3. Click the service list icon on the left and choose **Databases** > **Data Replication Service**.
4. In the navigation pane on the left, choose **Data Synchronization Management**. On the displayed page, click **Create Synchronization Task**.
5. Configure synchronization instance information.
   a. Select a region, billing mode, and project, and enter a task name.



   b. Configure synchronization instance details. Specifically, specify **Data Flow**, **Source DB Engine**, **Destination DB Engine**, **Network Type**, **DRS Task Type**, **Destination DB Instance**, **Synchronization Instance Subnet**, **Synchronization Mode**, **Specify EIP** (mandatory when **Network Type** is set to **Public Network**), **Specifications**, **AZ**, **Enterprise Project**, and **Tags** (optional).



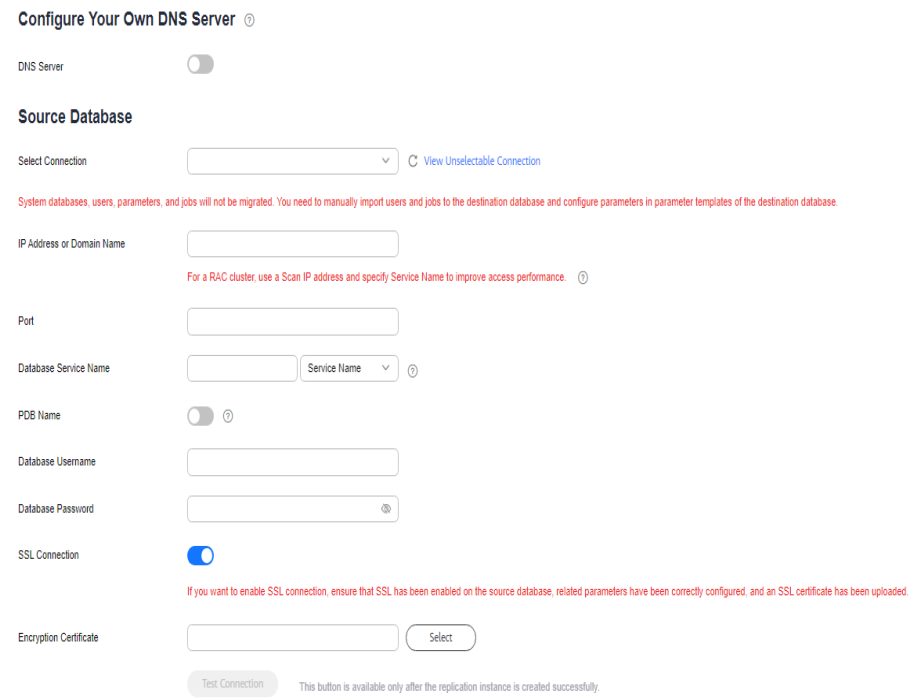


c. Click **Create Now**.

6. Configure the source and destination database information.

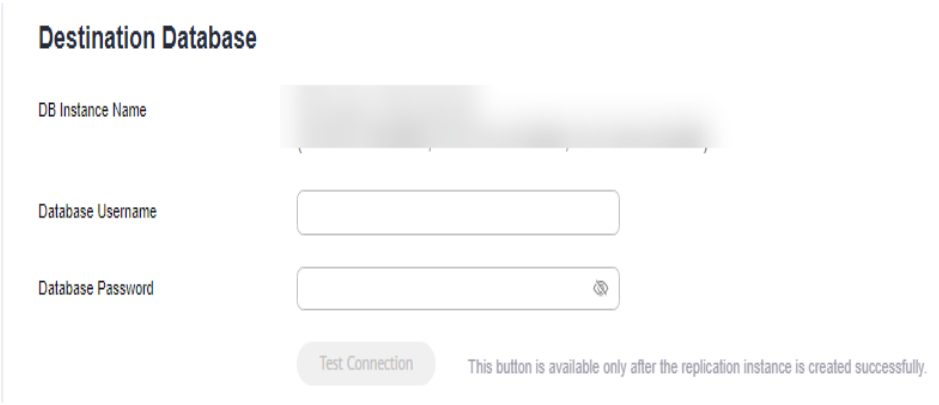
   a. Configure **DNS Server** as required. Specify connection information about the source database, including the IP address, port, username, and password.

      Click **Test Connection**.



   b. Enter the username and password of the destination database.

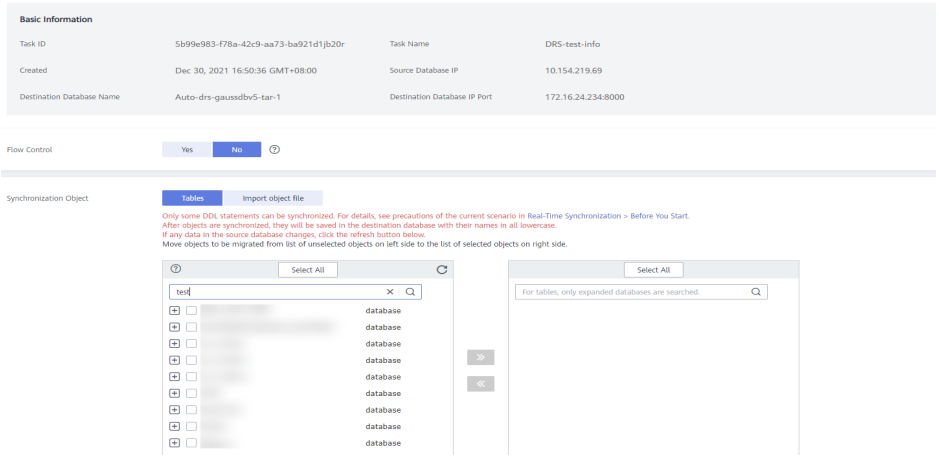
      Click **Test Connection**.



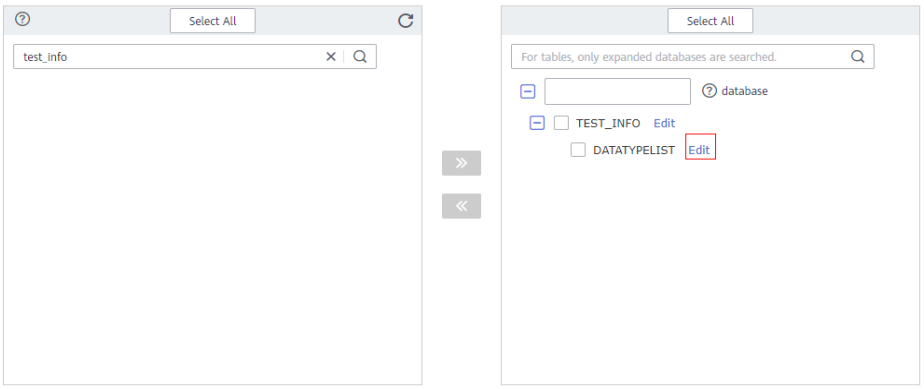
c. Click **Next**. In the displayed box, read the message carefully and click **Agree**.

7. Configure the synchronization task.

a. Select the databases and tables of the source database to be migrated. In this example, select the **DATATYPELIST** table from the **test_info** database.

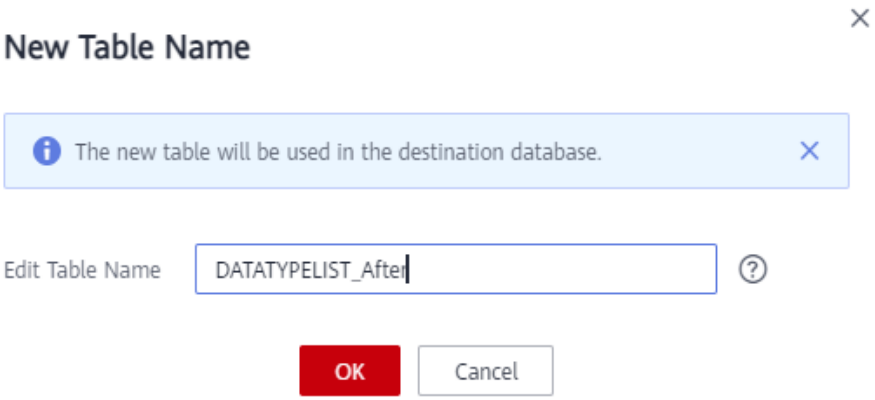


b. Locate the database and table, respectively, and click **Edit** to change the database name and table name as needed.

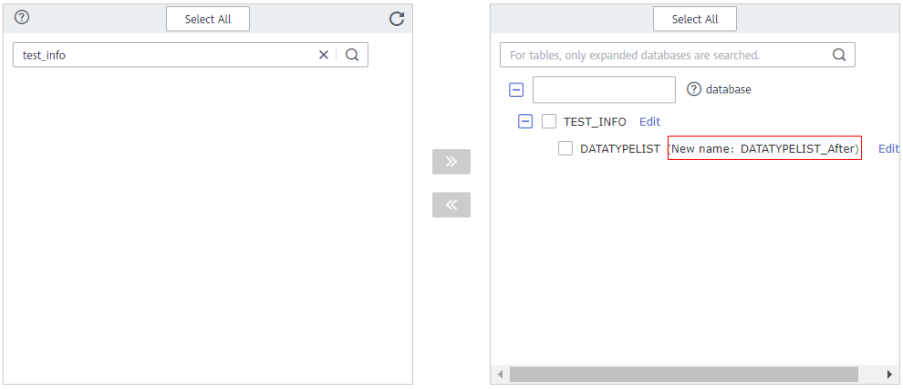


c. On the displayed dialog box, enter a new name, for example, **DATATYPELIST_After**.

The name cannot include special characters. Otherwise, an error will be reported during SQL statement execution after the migration.

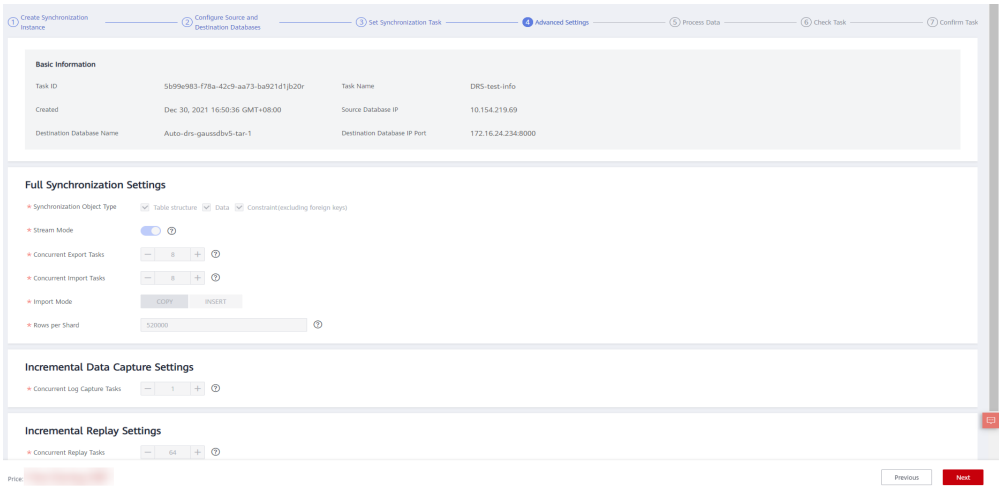


d. Confirm the settings and click **Next**.



8. Confirm advanced settings.

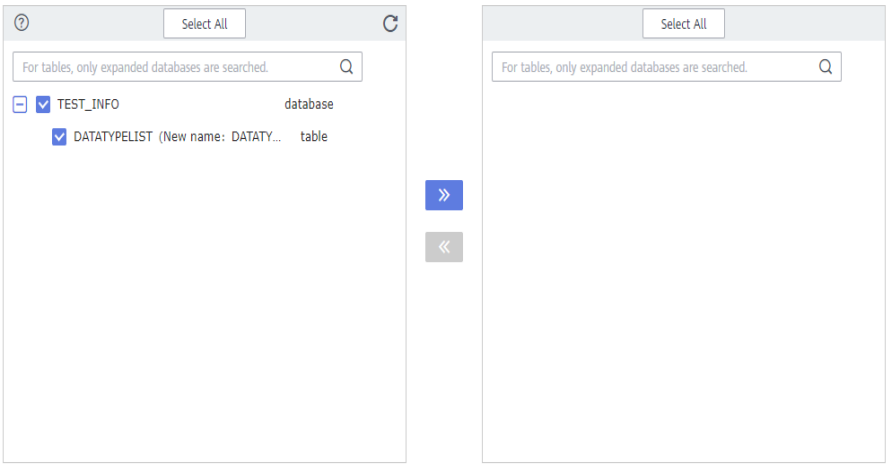
   The information on the **Advanced Settings** page is for confirmation only and cannot be modified. After confirming the information, click **Next**.



9. Process data.

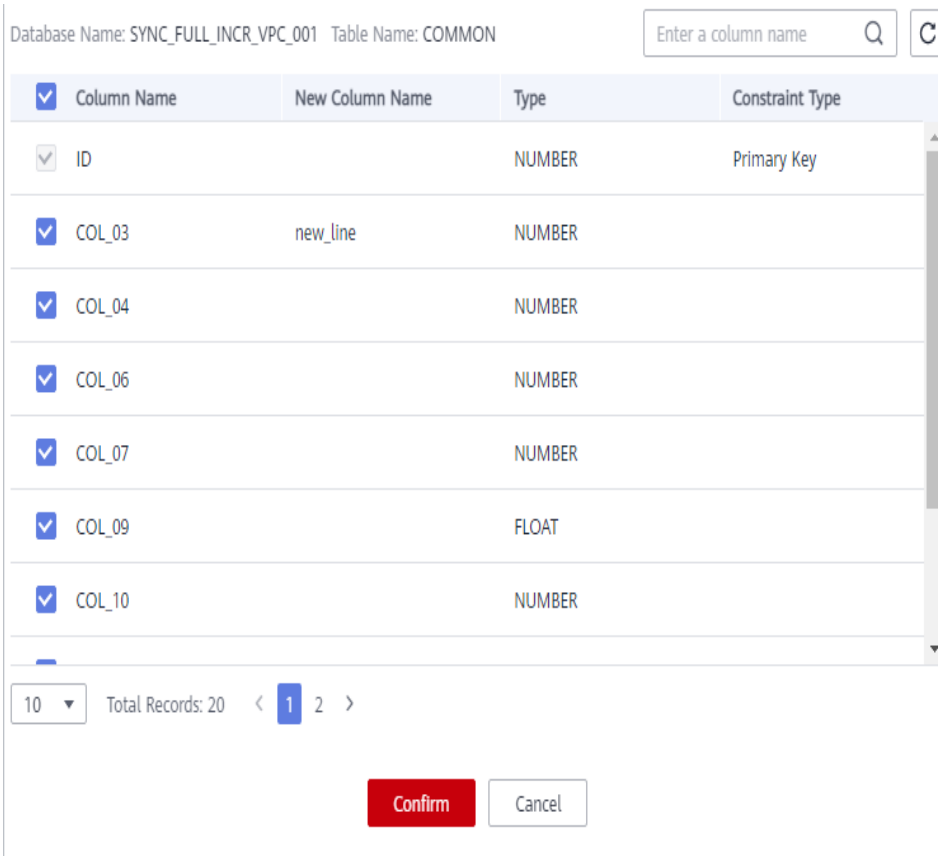
   On this page, you can process the table to be migrated. For example, you can select the column to be migrated and change its name. In this example, change the column name **COL_01_CHAR_____E** to **new-line**.

   a. Click **Edit** next to the table to be processed.



b. Edit the **COL_01_CHAR_____E** column.

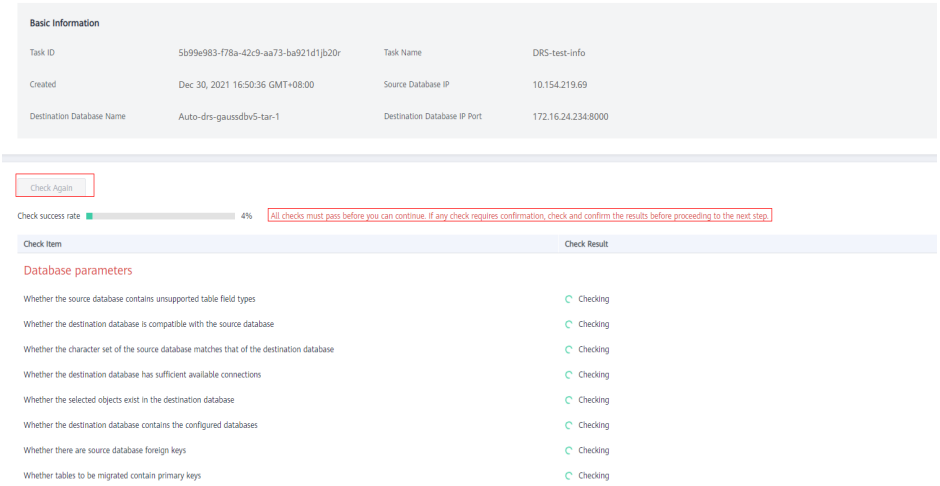
c. Enter the new name **new-line** and click **Confirm**.



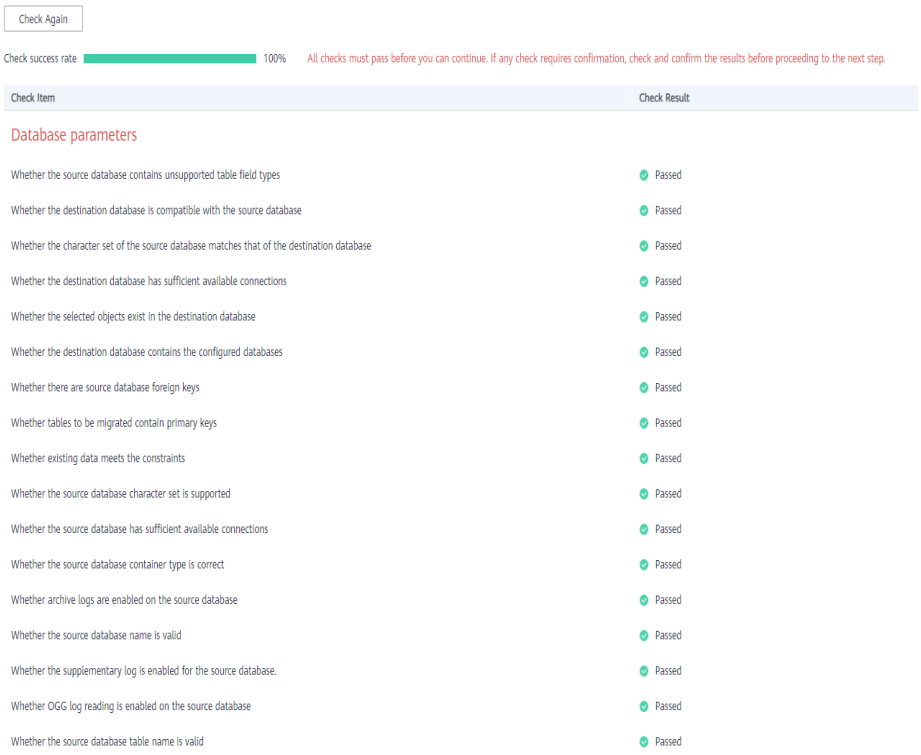
d. Click **Next**.

10. Perform a pre-check.

a. After all settings are complete, perform a pre-check to ensure that the migration is successful.



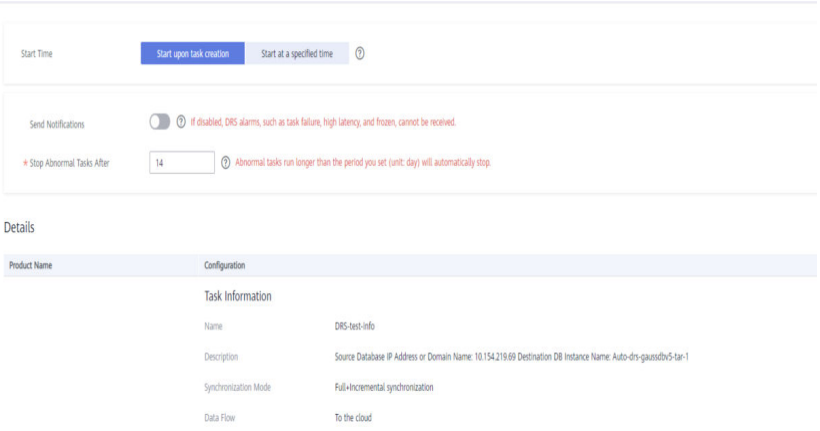
b. If any check item fails, review the cause and rectify the fault. Then, click **Check Again**.



c. After all check items pass the pre-check, click **Next**.

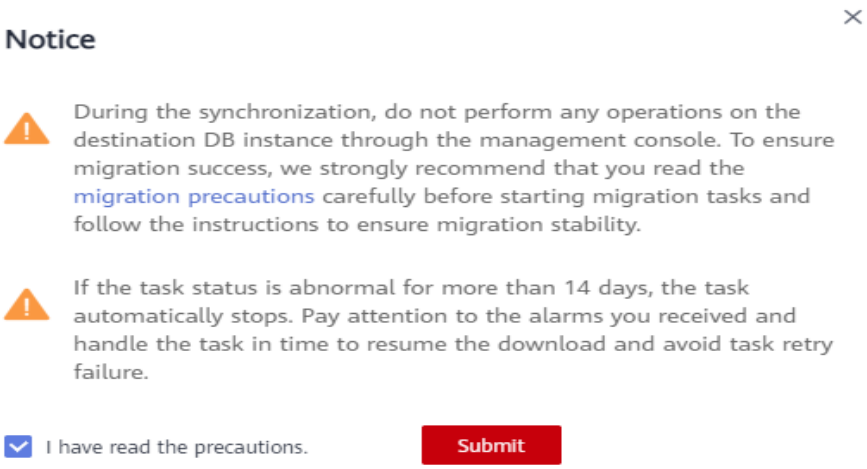
11. Confirm the task.

a. Check that all configured information is correct.



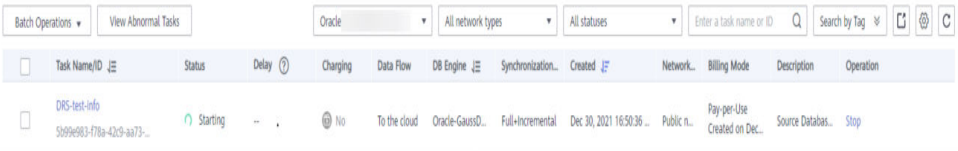
b. Click **Submit**. In the display box, select **I have read the precautions**.

c. Click **Submit**.



12. After the task is submitted, view and manage it.

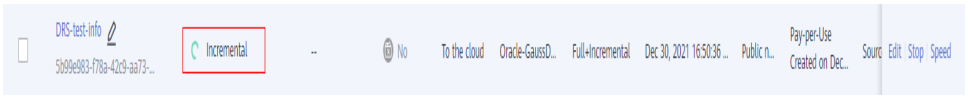
    After the task is created, return to the task list to view the status of the created task.



Step 5: Verify Data After Migration

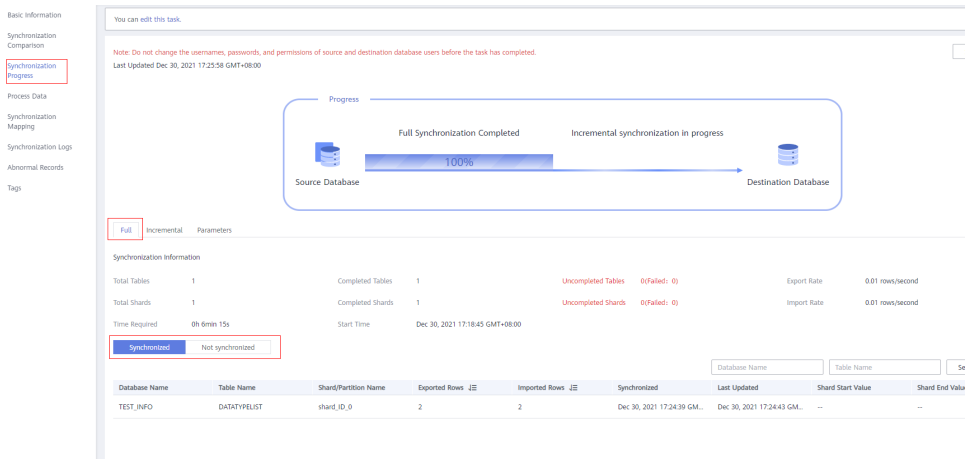
When the task status changes to **Incremental**, the full synchronization is complete. You can log in to GaussDB and view the data migration result.

1. Wait until the migration task status becomes **Incremental**.



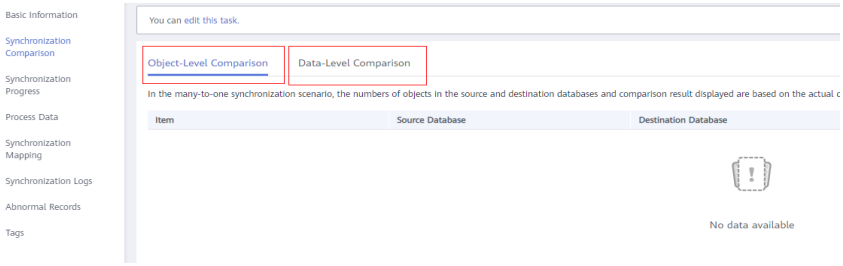
2. Click the task name to go to the **Basic Information** page.
3. On the **Synchronization Progress** page, view the full synchronization result.

   As shown in the following figure, the **DATATYPELIST** table in the **TEST_INFO** database has been migrated to **shard_0**. Two rows of data were migrated successfully.

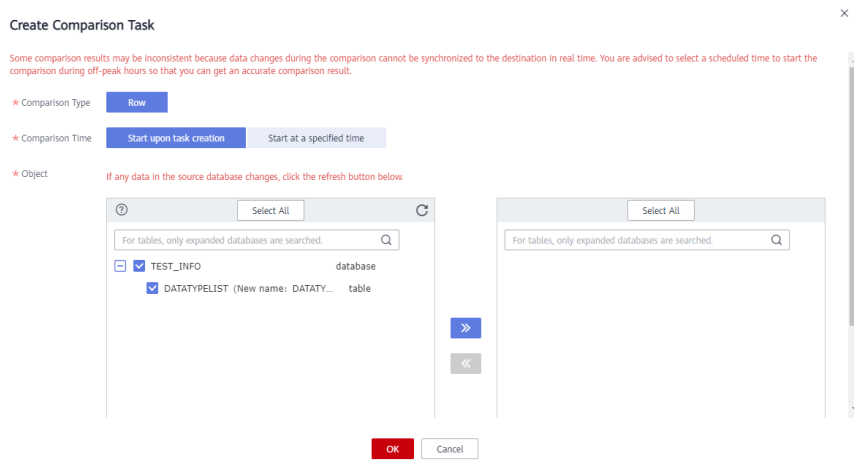


4. Verify data consistency.

   a. Choose **Synchronization Comparison** > **Object-Level Comparison** to view the database and table migration results.



   b. Choose **Synchronization Comparison** > **Data-Level Comparison**, click **Create Comparison Task**, and view the migration results of the rows in the table.

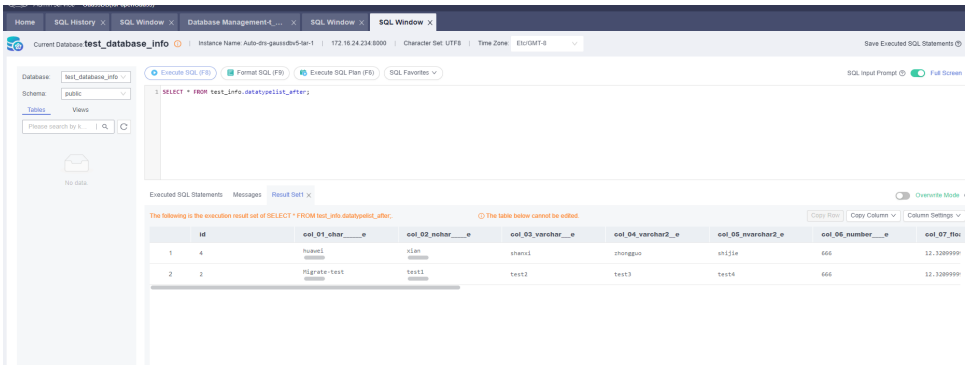


5. Connect to the **test_database_info** database in GaussDB using DAS.
6. Run the following statement to query the full synchronization result:

```
SELECT * FROM test_info.datatypelist_after;
```

   After the schema in Oracle is migrated, it will be used as the schema in GaussDB. Therefore, it is required to add the schema in the query statement for exact query.

   As shown in the following figure, all data types in the table were successfully migrated and the data is correct.



7. Verify incremental synchronization.

   In full+incremental synchronization, after the full synchronization is complete, the data that is written to the source database after the task is created can still be synchronized to the destination database until the task is stopped. The following describes how to synchronize incremental data from the source database to the destination database:

   a. Use a database connection tool to connect to the source Oracle database based on its IP address.

   b. Run the following statement to insert a data record into the source database:

      Insert a data record whose ID is 1.

      ```
      insert into test_info.DATATYPELIST values(1,'Migrate-test','test1','test2','test3','test4',
      666,12.321,1.123,2.123,sysdate,sysdate,sysdate,sysdate,'hw','cb','df','FF','FF','AAAYEVAAJAAAACrA
      AA');
      commit;
      ```

   c. Run the following statement in the destination database to query the result:

      ```
      SELECT * FROM test_info.datatypelist_after;
      ```

      As shown in the following figure, the new data inserted in the source database has been synchronized to the destination database in real time.

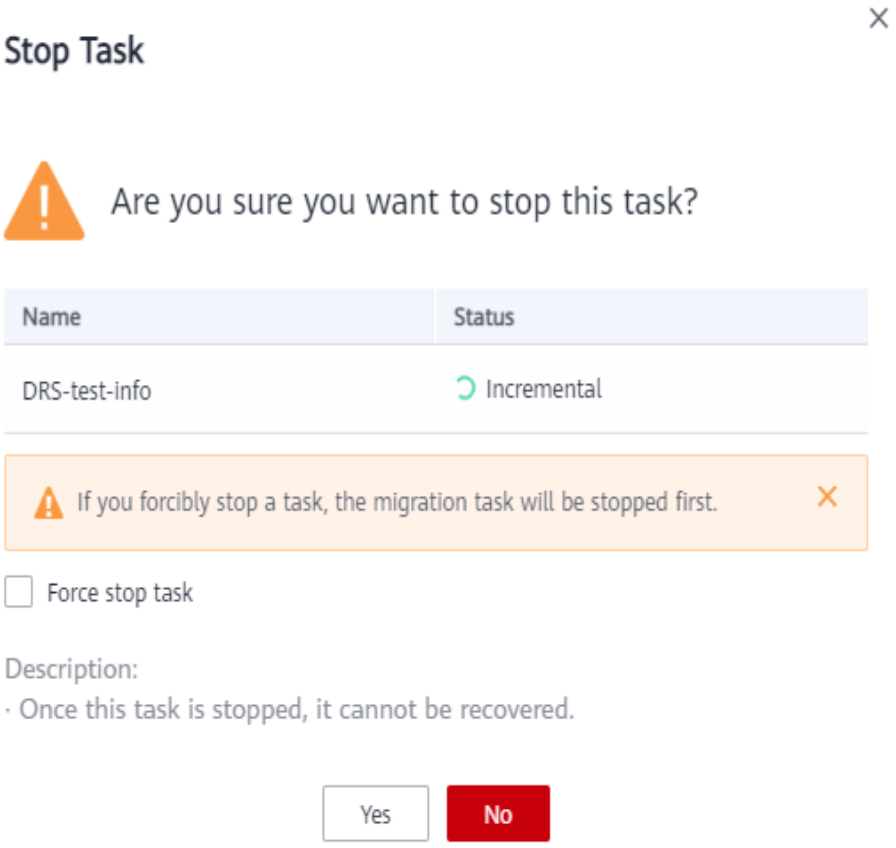
8. Stop the migration task.

   After data is completely migrated to the destination database, stop the synchronization task.

   a. Locate the task and click **Stop** in the **Operation** column.

   b. In the display box, click **Yes**.



9. After the migration is complete, test the GaussDB performance.
   For details, see ***Performance White Paper***.

4.3 Using DRS to Migrate Data from MySQL Database to GaussDB

Scenarios

This section describes how to use DRS to migrate data from an on-premises MySQL database to Huawei Cloud GaussDB in real time. Full+incremental synchronization can ensure that data is always in sync between the source MySQL database and the destination GaussDB instance.

Step 1: Create a VPC and Security Group

Step 2: Create a GaussDB Instance

Step 3: Construct Data Before Migration

Step 4: Migrating the Database

Step 5: Verify Data After Migration

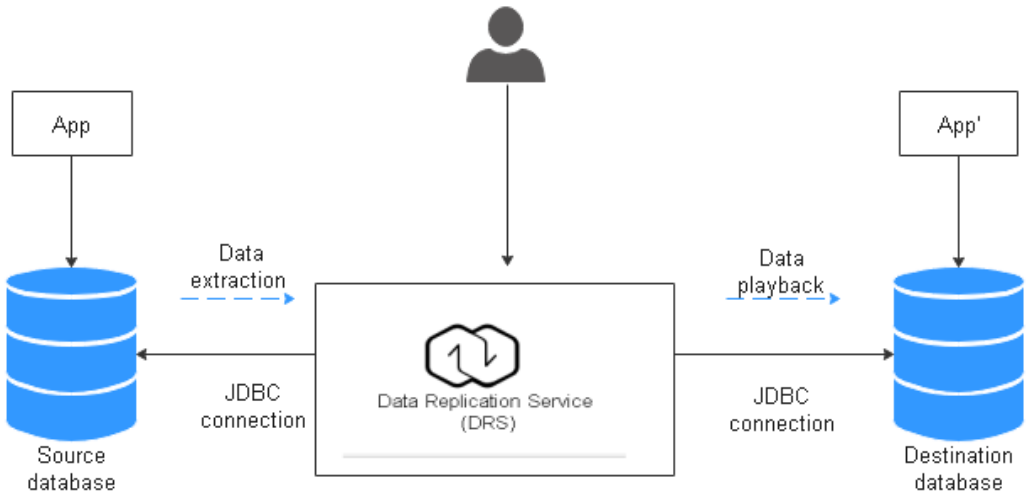
Problems to Resolve

- Enterprise workloads have been growing and evolving fast, and traditional databases lack the scalability needed to keep up. Enterprises need distributed databases.
- Building a traditional database means purchasing and installing servers, systems, databases, and other software. The O&M is expensive and difficult.
- Traditional databases have poor performance when it comes to handling complex queries.
- It is hard for traditional databases to smoothly synchronize data with no downtime.

Prerequisites

- You have registered with Huawei Cloud and completed account authentication.
- Your account balance is greater than or equal to $0 USD.
- In a testing scenario, you have set up an on-premises MySQL database.
- You have obtained the IP address, port number, username, and password of the MySQL database to be migrated.

Service Architecture



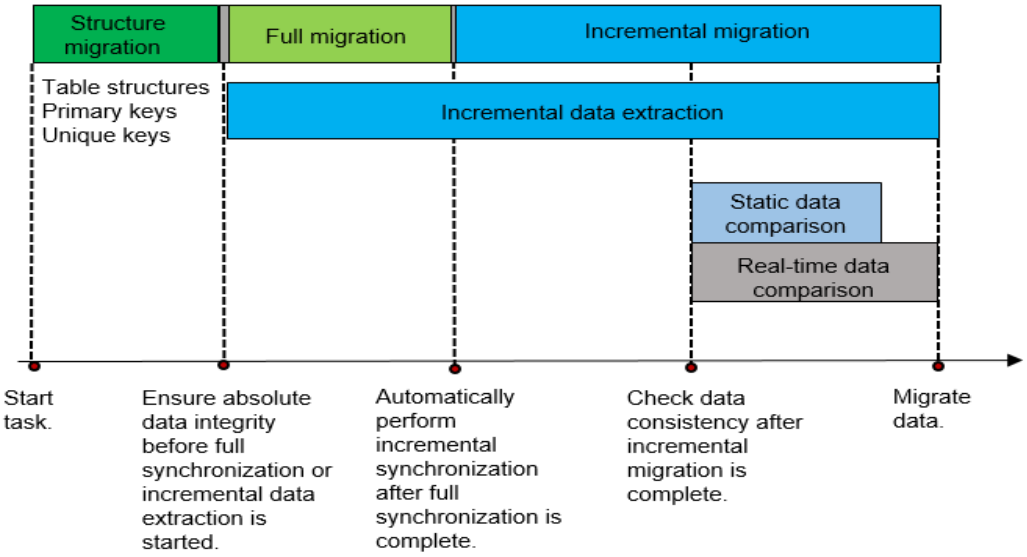
How Data Migration Works

The data migration process is completed using full and incremental synchronization, which includes the following operations:

1. In the full synchronization phase, schemas, including tables, primary keys, and unique keys, are synchronized first.
2. After schemas are synchronized, incremental data extraction is started to ensure that the incremental data generated during full data synchronization is completely extracted to the DRS instance.

3. A full migration task is started.
4. An incremental synchronization is automatically started after the full migration is complete. The replay starts from the position where the full synchronization starts.
5. A comparison task is started after the incremental replay is complete to check the data consistency. Real-time comparison is supported.
6. Workloads synchronization is started if the data is consistent between the source and destination databases.

Figure 4-11 Migration principle



Resource Planning

The resource planning in this section is just an example. You need to adjust it as needed.

Table 4-4 Resource planning

Category	Item	Planned Value	Remarks
VPC	VPC name	vpc-src-172	Specify a name that is easy to identify.
	Region	Test region	To achieve lower network latency, select the region nearest to you.
	AZ	AZ 3	-
	Subnet CIDR block	172.16.0.0/16	Select a subnet with sufficient network resources.

Category	Item	Planned Value	Remarks
	Subnet name	subnet-src-172	Specify a name that is easy to identify.
On-premises MySQL database	Database version	5.7.38	-
	Database user	test_info	Specify a database user. The user must at least have the following privileges: SELECT, LOCK TABLES, REPLICATION SLAVE, and REPLICATION CLIENT.
GaussDB	Instance name	Auto-drs-gaussdbv5-tar-1	Specify a name that is easy to identify.
	Database version	GaussDB V2.0-8.103 Enterprise edition	-
	Instance type	Distributed (3 CNs, 3 DN shards, and 3 replicas)	In this example, a distributed instance will be created.
	Deployment model	Independent	-
	Transaction consistency	Strong consistency	-
	Shards	3	-
	Coordinator nodes	3	-
	Storage type	Ultra-high I/O	-
	AZ	AZ 2	In this example, a single AZ is select. You are advised to select multiple AZs to improve instance availability in actual use.

Category	Item	Planned Value	Remarks
	Instance Specifications	Dedicated (1:8); 8 vCPUs | 64 GB	Small specifications are selected for this test instance. You are advised to configure specifications based on service requirements in actual use.
	Storage space	480 GB	A small storage space is selected for this test instance. You are advised to configure the storage space based on service requirements in actual use.
	Disk encryption	Disable	In this example, disk encryption is disabled. Enabling disk encryption improves the security of data, but may slightly affect the database read/write performance.
Logging in to the database through DAS	Database engine	GaussDB	-
	Database source	GaussDB	Select the GaussDB instance created in this example.
	Database name	postgres	-
	Username	root	-
	Password	-	Enter the password of the **root** user of the GaussDB instance created in this example.
DRS migration task	Migration task name	DRS-test-info	Specify a name that is easy to identify.
	Destination database name	test_database_info	Specify a name that is easy to identify. The name must be compatible with the MySQL database name.
	Source database engine	MySQL	-

Category	Item	Planned Value	Remarks
	Destination database engine	GaussDB	-
	Network type	Public network	In this example, a public network is used.

Step 1: Create a VPC and Security Group

Create a VPC and security group for the GaussDB instance.

Creating a VPC

1. Log in to the **Huawei Cloud console**.
2. Click  in the upper left corner and select a region.
3. Click the service list icon on the left and choose **Networking** > **Virtual Private Cloud**. The VPC console is displayed.
4. Click **Create VPC**.

Figure 4-12 Basic information

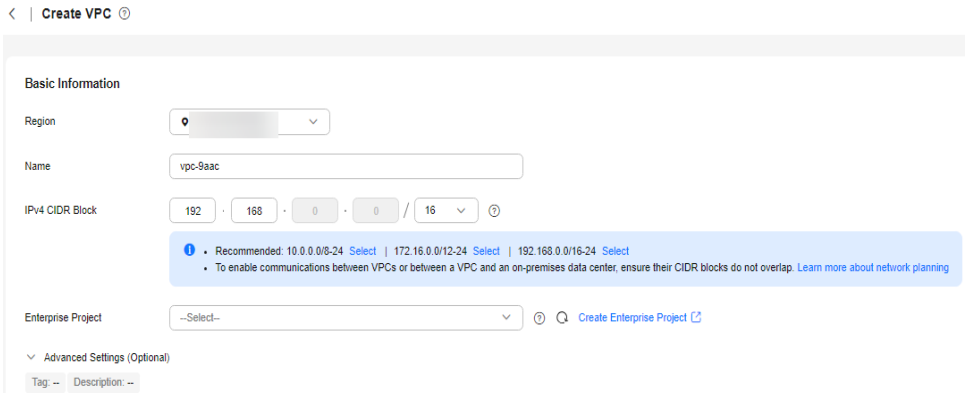
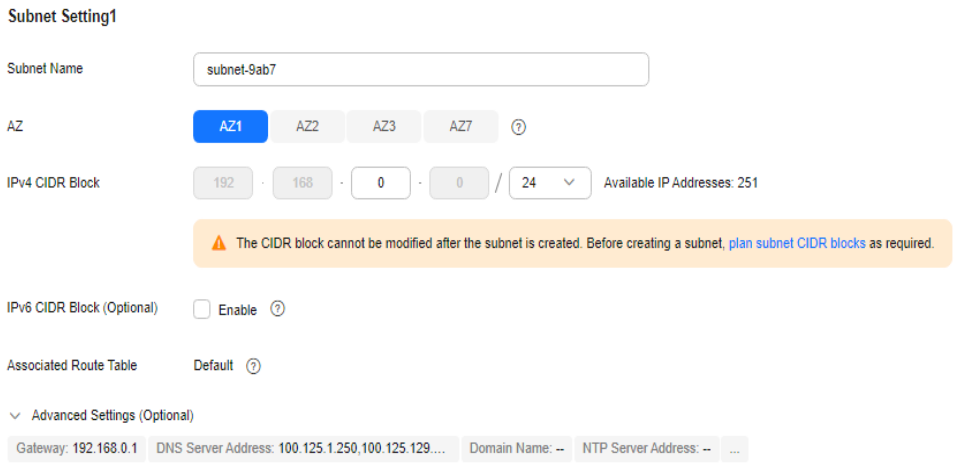


Figure 4-13 Setting a subnet

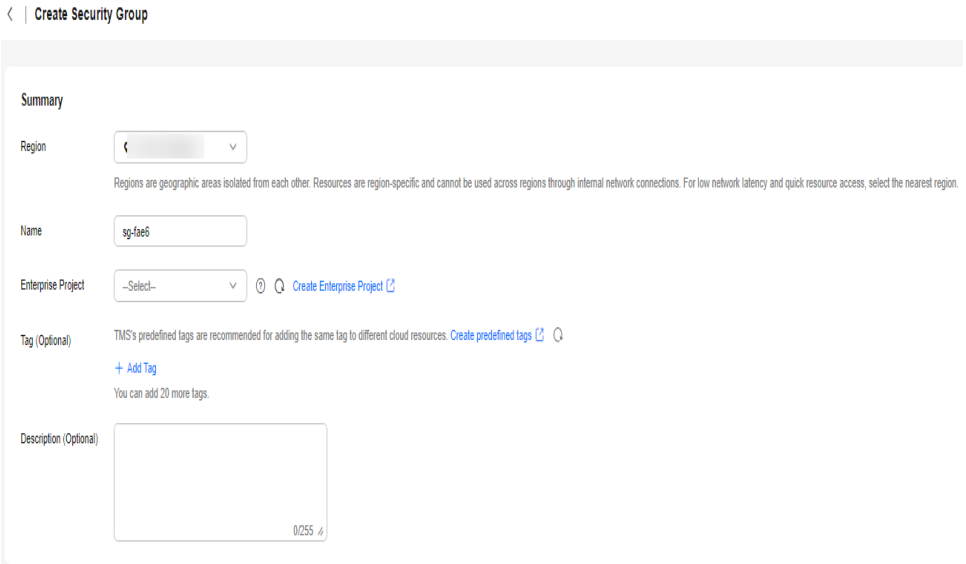


5. Configure parameters as needed and click **Create Now**.
6. Return to the VPC list and check whether the VPC is created.
   If the VPC status becomes available, the VPC has been created.

Creating a Security Group

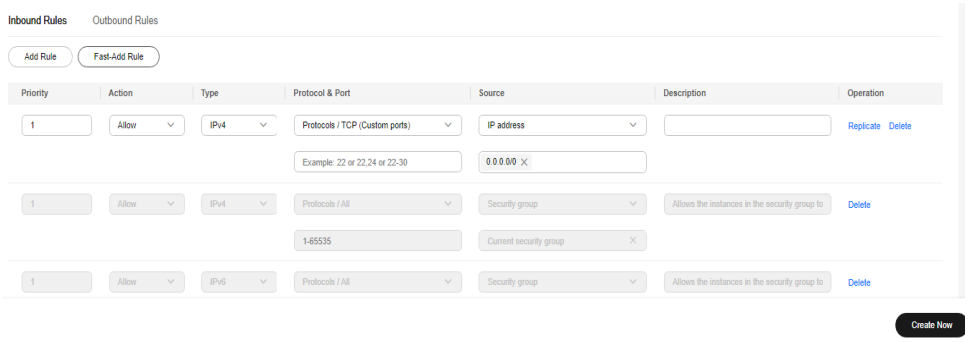
1. Log in to the **Huawei Cloud console**.
2. Click  in the upper left corner and select a region.
3. Click the service list icon on the left and choose **Networking** > **Virtual Private Cloud**.
   The VPC console is displayed.
4. In the navigation pane, choose **Access Control** > **Security Groups**.
5. Click **Create Security Group**.
6. Specify a security group name and other information.

Figure 4-14 Basic information



7. Click the **Inbound Rules** tab and then click **Add Rule**.
8. Configure an inbound rule, add the IP address of the source database, and click **OK**.

Figure 4-15 Configuring an inbound rule



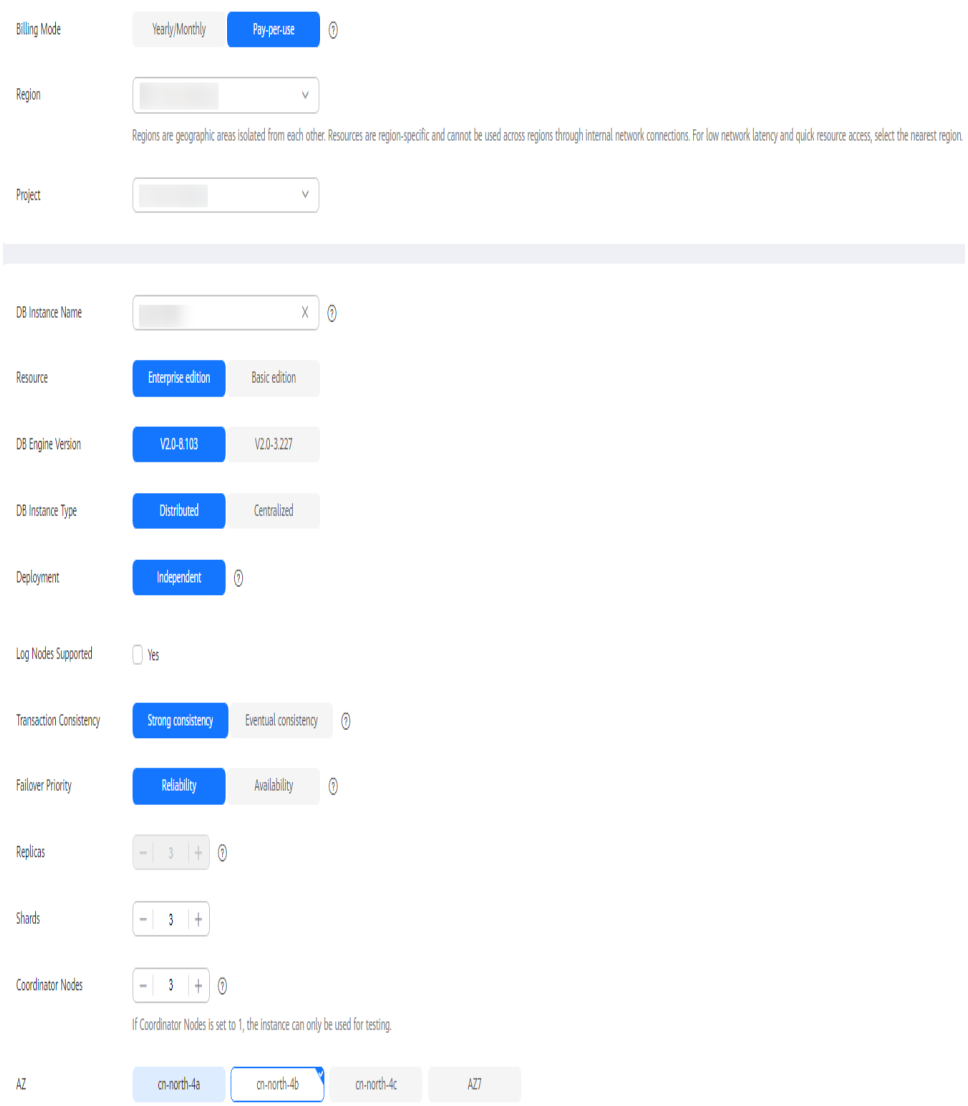
9. Configure parameters as needed and click **Create Now**.

Step 2: Create a GaussDB Instance

Create a GaussDB instance as the destination database of the migration task.

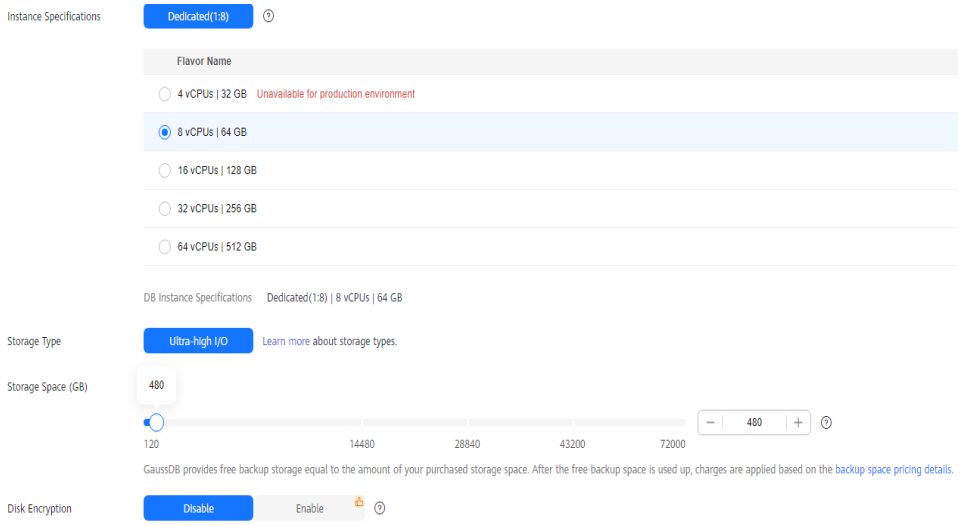
1. Log in to the **Huawei Cloud console**.
2. Click  in the upper left corner and select a region.
3. Click the service list icon on the left and choose **Databases** > **GaussDB**.
4. In the navigation pane on the left, choose **GaussDB** > **Instances**.
5. Click **Buy DB Instance**.
6. On the page shown in **Figure 4-16**, configure basic information about the instance, including the instance name, billing mode, edition type, DB engine version, instance type, transaction consistency, number of shards, number of coordinator nodes, and deployment AZ.

Figure 4-16 Basic information



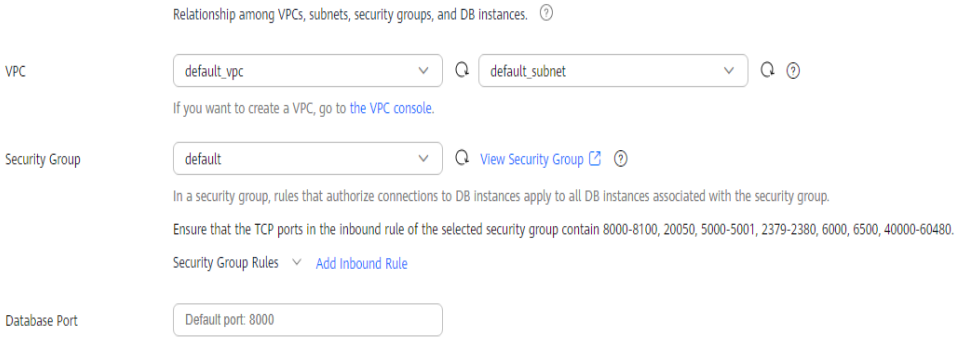
7. Select the instance specifications and storage space.

Figure 4-17 Instance specifications



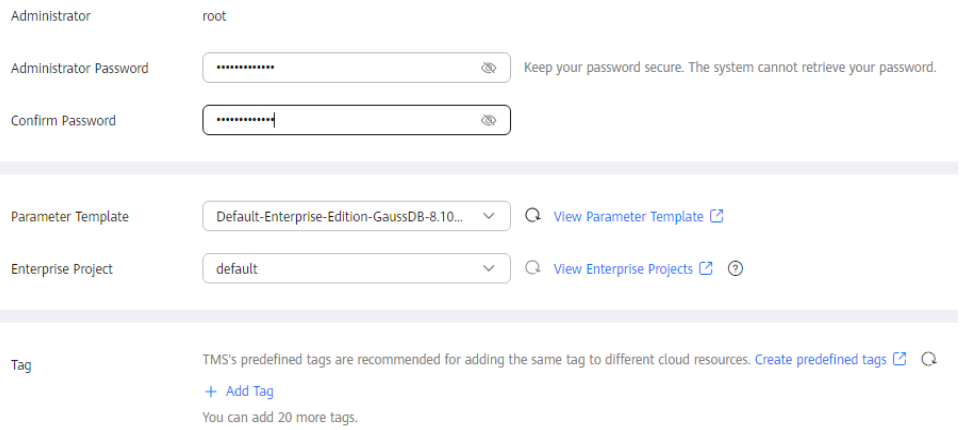
8. Select the VPC created in **Creating a VPC** and security group created in **Creating a Security Group** for the instance and configure the database port.

Figure 4-18 Selecting a VPC and security group



9. Configure the password and other information.

Figure 4-19 Configuring the password and other information



10. Click **Next**, confirm the information, and click **Submit**.
11. Go to the instance list.

    If status of the instance becomes **Available**, the instance has been created.

Step 3: Construct Data Before Migration

Before the migration, prepare some data types in the source database for verification after the migration is complete. The end-to-end test data in this section is for reference only.

For details about the data types supported by DRS, see **MySQL->GaussDB**.

Perform the following steps to construct data in the source database:

1. Use a database connection tool to connect to the source MySQL database based on its IP address.
2. Construct data in the source database based on data types supported by DRS.
   a. Create a test user.

      create user *test_info* **identified by** *xxx*;

      test_info indicates the user created for the migration, and *xxx* indicates the password of the user.

   b. Create a database named **test_info** under the user.

      CREATE DATABASE test_info;

   c. Create a table in the **test_info** database.

      CREATE TABLE `test_info`.`test_table` **(**
      `id` int NOT NULL,
      `c1` char(10) DEFAULT NULL,
      `c2` varchar(10) DEFAULT NULL,
      `c3` binary(10) DEFAULT NULL,
      `c4` varbinary(10) DEFAULT NULL,
      `c5` tinyblob,
      `c6` mediumblob,
      `c7` longblob,
      `c8` tinytext,
      `c9` text,
      `c10` mediumtext,
      `c11` longtext,
      `c12` enum('1','2','3') DEFAULT NULL,
      `c13` set('1','2','3') DEFAULT NULL,
      `c14` tinyint DEFAULT NULL,
      `c15` smallint DEFAULT NULL,
      `c16` mediumint DEFAULT NULL,
      `c17` bigint DEFAULT NULL,
      `c18` float DEFAULT NULL,
      `c19` double DEFAULT NULL,


```
`c20` date DEFAULT NULL,
`c21` datetime DEFAULT NULL,
`c22` timestamp,
`c23` time DEFAULT NULL,
`c24` year DEFAULT NULL,
`c25` bit(10) DEFAULT NULL,
`c26` json DEFAULT NULL,
`c27` decimal(10,0) DEFAULT NULL,
`c28` decimal(10,0) DEFAULT NULL,
PRIMARY KEY (`id`)
);
```

d. Assign privileges to the user.

GRANT SELECT,LOCK TABLES ON <database>.<table> to test_info**;**

GRANT REPLICATION SLAVE,REPLICATION CLIENT ON *.* to test_info**;**

In the preceding commands, *test_info* indicates the user created for this migration task, *<database>* indicates the name of the database to be synchronized, and *<table>* indicates the name of the table to be synchronized. Replace them as required.

e. Insert two rows of data into the table.

```
insert into test_info.test_table values
(1,'a','b','111','111','tinyblob','mediumblob','longblob','tinytext','text','mediumtext','longtext','1','3',1,2,3,4,1.123,1.1234,'2024-03-08','2024-03-08 08:00:00','2024-03-08 08:00:00','08:00:00','2024','1010','{"a":"b"}',1.23,1.234);
insert into test_info.test_table values
(2,'a','b','111','111','tinyblob','mediumblob','longblob','tinytext','text','mediumtext','longtext','1','3',1,2,3,4,1.123,1.1234,'2024-03-08','2024-03-08 08:00:00','2024-03-08 08:00:00','08:00:00','2024','1010','{"a":"b"}',1.23,1.234);
```

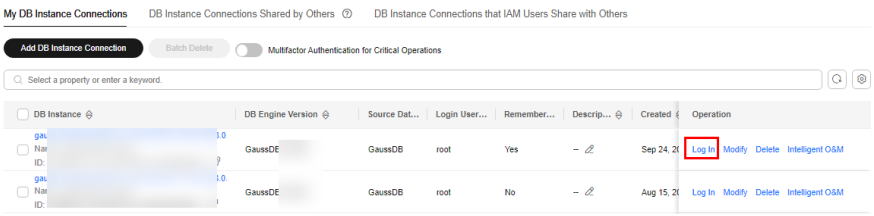
3. Create a database in the destination GaussDB instance.
   a. Log in to the **management console**.
   b. Click  in the upper left corner and select a region.
   c. Click the service list icon on the left and choose **Databases** > **Data Admin Service**.
   d. In the navigation pane on the left, choose **Development Tool** to go to the login list page.
   e. Click **Add Login**.
   f. On the displayed page, select the DB engine and source database, enter the login username, password, and description (optional), and enable **Show Executed SQL Statements**.
   g. Click **Test Connection** to check whether the connection is successful.

      If a message is displayed indicating the connection is successful, continue with the operation. If a message is displayed indicating the connection failed and the failure cause is provided, make modifications according to the error message.

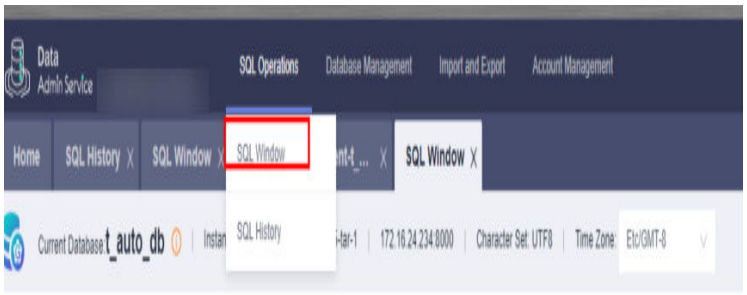
h. Click **OK**.

i. Locate the added record and click **Log In** in the **Operation** column.

Figure 4-20 Logging in to an instance



j. Choose **SQL Operations** > **SQL Window** on the top menu bar.



k. Run the following statement to create a database compatible with MySQL:

test_database_info indicates the database name. Replace it as required.

```
CREATE DATABASE test_database_info DBCOMPATIBILITY 'mysql';
```

Step 4: Migrating the Database

Create a DRS instance and synchronize data from the **test_info** database in the on-premises MySQL database to the **test_database_info** database in the GaussDB instance.

Performing a Pre-migration Check

Before creating a migration task, check the migration conditions to ensure smooth migration.

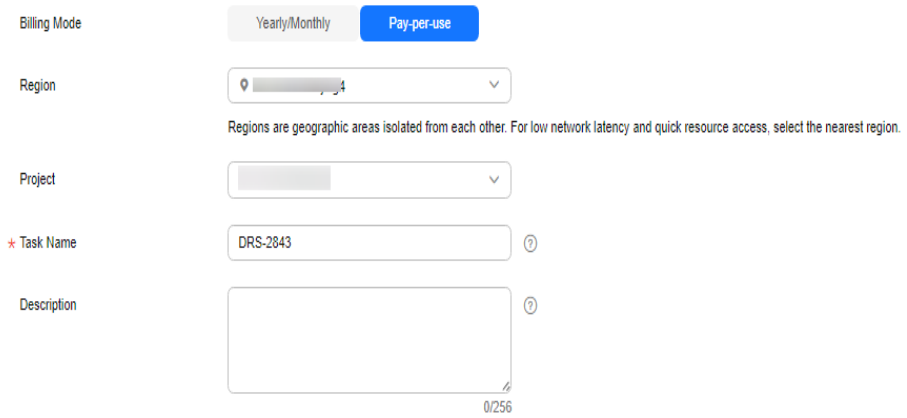
Before the migration, you need to obtain the **notes on migration to the cloud**.

Creating a Migration Task

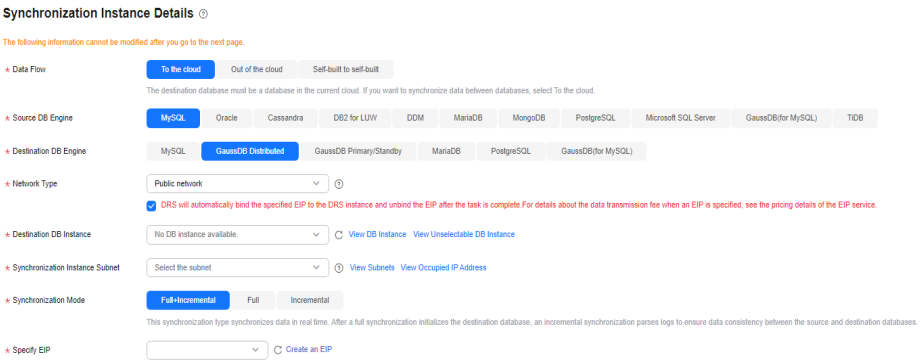
1. Log in to the **management console**.
2. Click  in the upper left corner and select a region.

   Select the region where the destination instance is deployed.
3. Click the service list icon on the left and choose **Databases** > **Data Replication Service**.
4. In the navigation pane on the left, choose **Data Synchronization Management**. On the displayed page, click **Create Synchronization Task**.
5. Configure synchronization instance information.

a. Select a region, billing mode, and project, and enter a task name.



b. Specify **Data Flow**, **Source DB Engine**, **Destination DB Engine**, **Network Type**, **DRS Task Type**, **Destination DB Instance**, **Synchronization Instance Subnet** (optional), **Synchronization Mode**, **Specifications**, **AZ**, and **Tags** (optional).

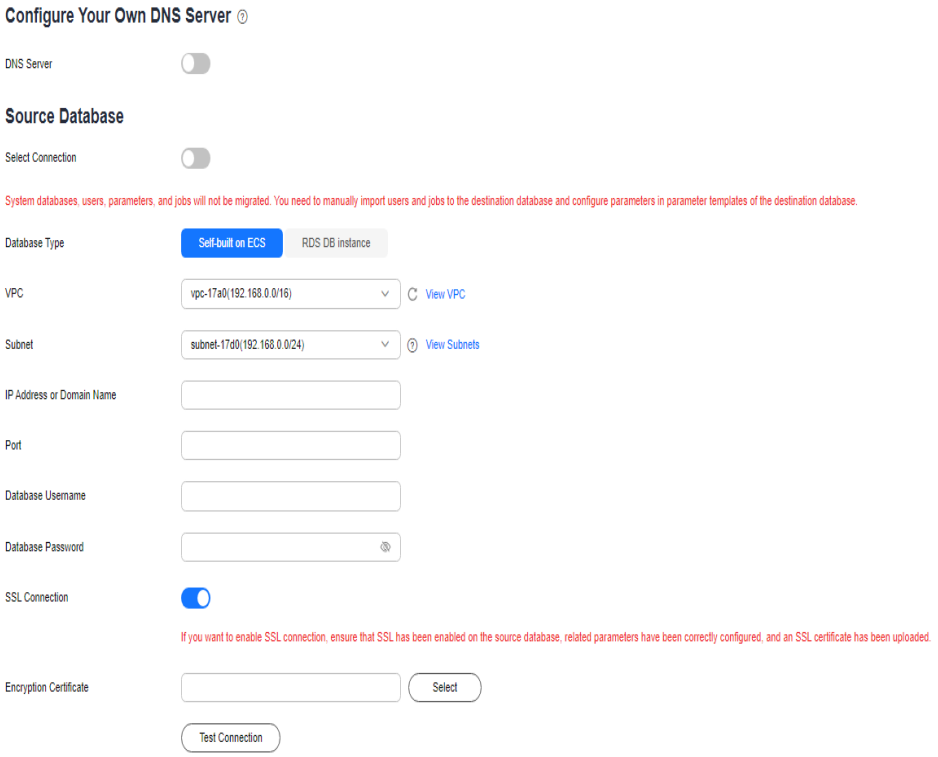


c. Click **Create Now**.

6. Configure the source and destination database information.

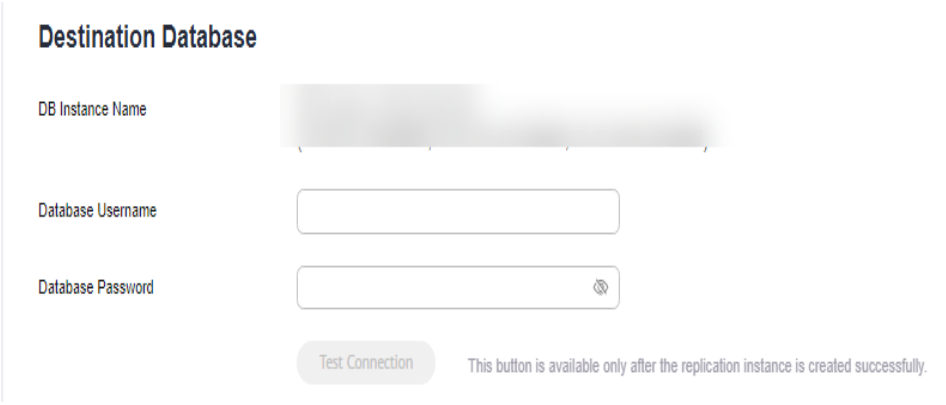
a. Enter the IP address, port number, username, and password of the source database.

Click **Test Connection**.



b. Enter the username and password of the destination database.

Click **Test Connection**.



c. Click **Next**. In the displayed box, read the message carefully and click **Agree**.

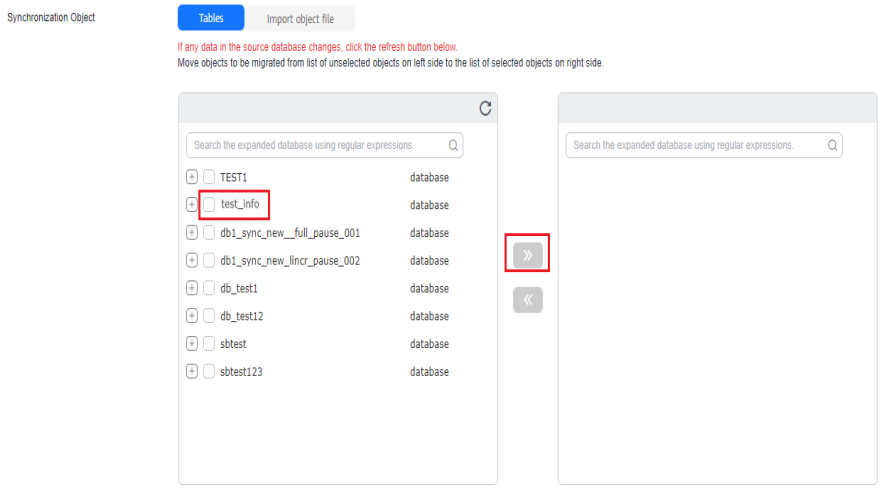
7. Configure the synchronization task.

a. Select the object type for full synchronization. If the table structure to be synchronized has not been created in the destination database, select **Table structure** (the table structure contains primary keys and unique keys) for **Synchronization Object Type**. Otherwise, deselect **Table structure**. Select **Index** for **Synchronization Object Type** as needed.

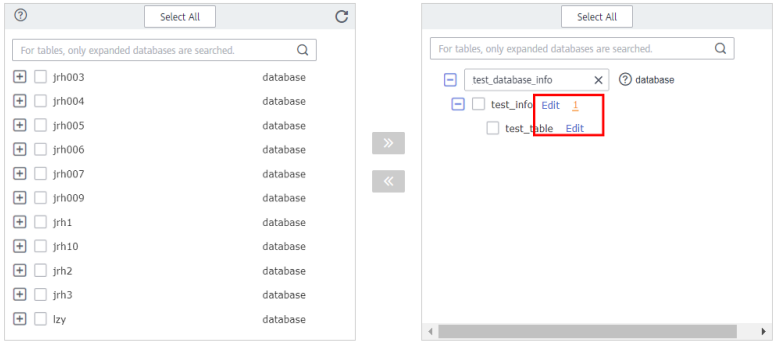
b. Specify **Incremental Conflict Policy**. This option determines how the system reacts when there is a data conflict (for example, duplicate primary or unique keys) between the source and destination databases.

- **Ignore**: The system will ignore the conflicting data in the source database and continue the subsequent synchronization process. If you select **Ignore**, data in the source database may be inconsistent with that in the destination database.
- **Report error**: The synchronization task will be stopped and fail. You can view the details in synchronization logs.
- **Overwrite**: Conflicting data in the destination database will be overwritten.

c. Select the databases and tables of the source database to be migrated. In this example, select the **test_table** table from the **test_info** database.

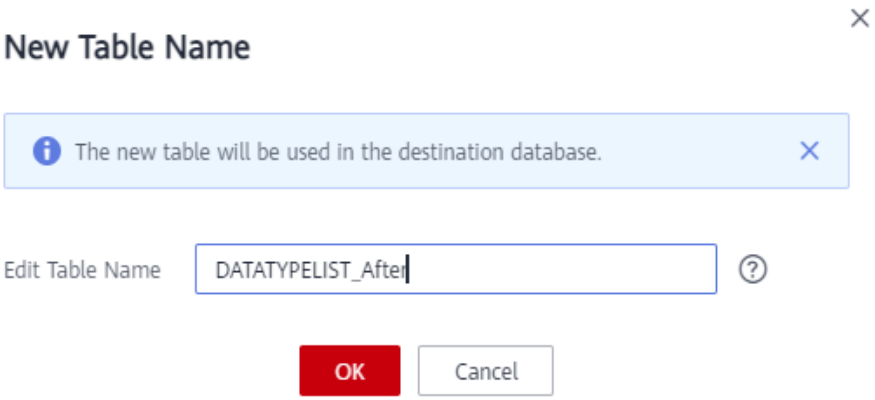


d. Locate the database and table, respectively, and click **Edit** to change the database name and table name as needed.

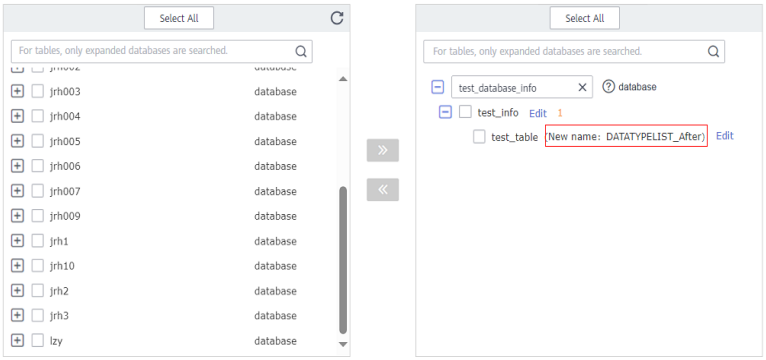


e. On the displayed dialog box, enter a new name, for example, **DATATYPELIST_After**.

The name cannot include special characters. Otherwise, an error will be reported during SQL statement execution after the migration.

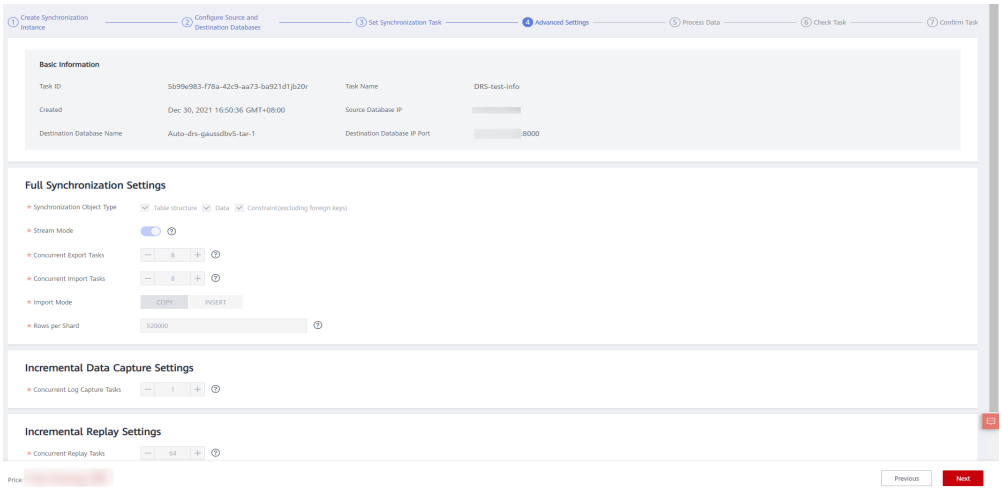


f. Confirm the settings and click **Next**.



8. Confirm advanced settings.

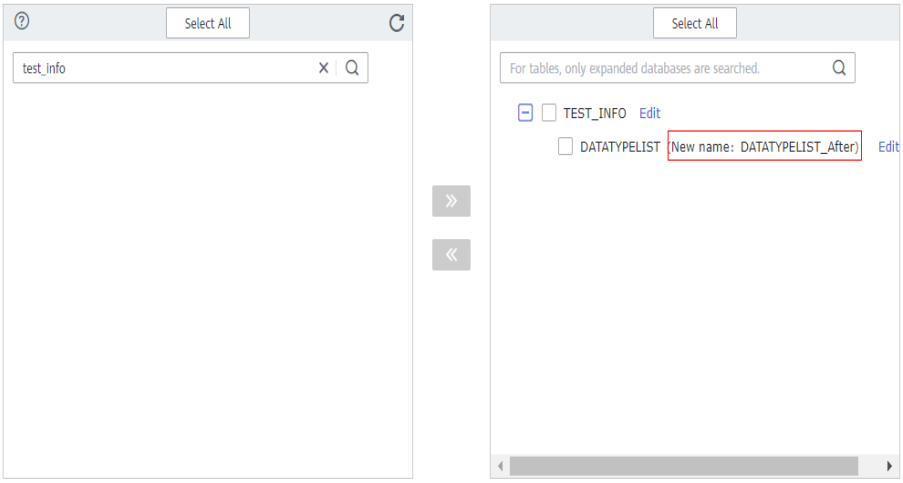
   The information on the **Advanced Settings** page is for confirmation only and cannot be modified. After confirming the information, click **Next**.



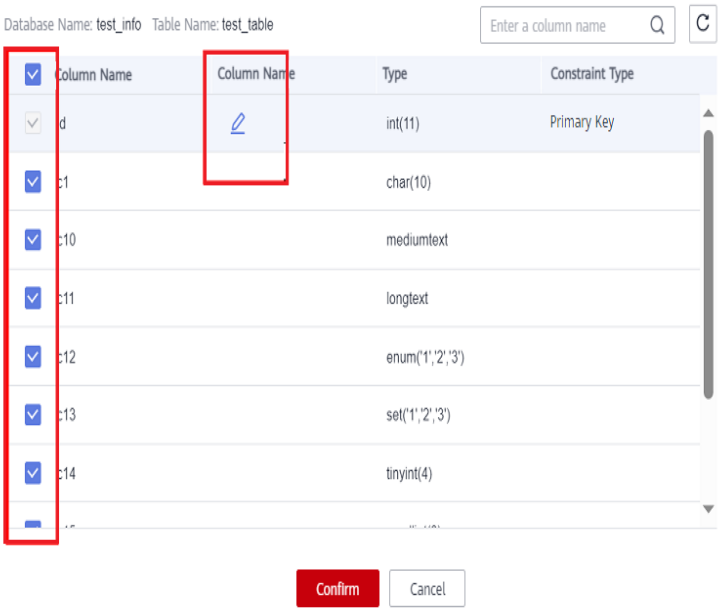
9. Process data.

   On this page, you can process the table to be migrated. For example, you can select the column to be migrated and change its name. In this example, change the column name **c1** to **new-line**.

   a. Click **Edit** next to the table to be processed.



b. Edit the **c1** column.



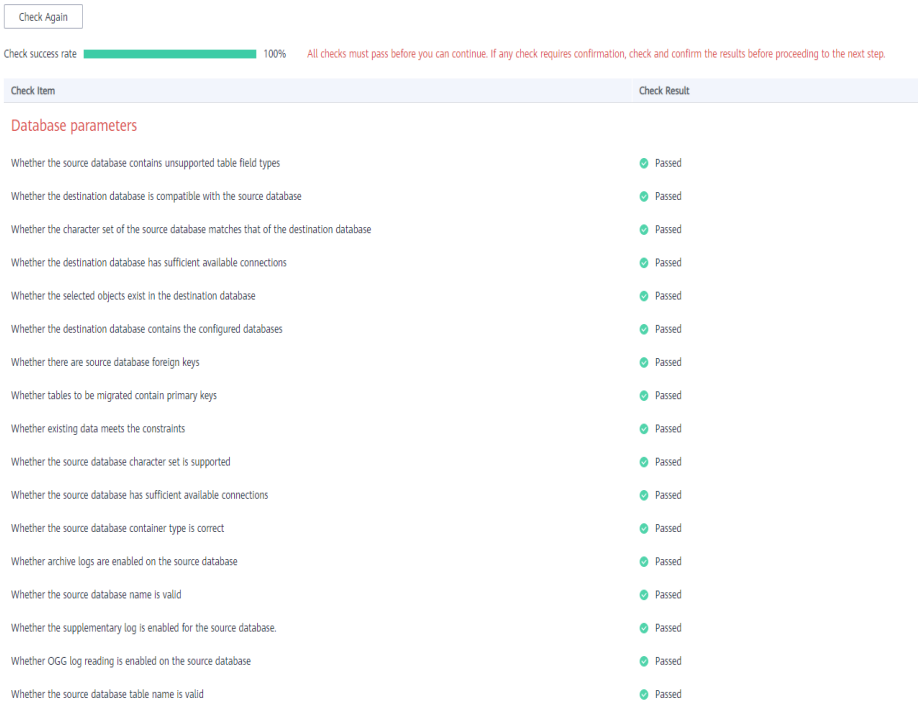
c. Enter the new name **new-line** and click **Confirm**.

d. Click **Next**.

10. Perform a pre-check.

a. After all settings are complete, perform a pre-check to ensure that the migration is successful.

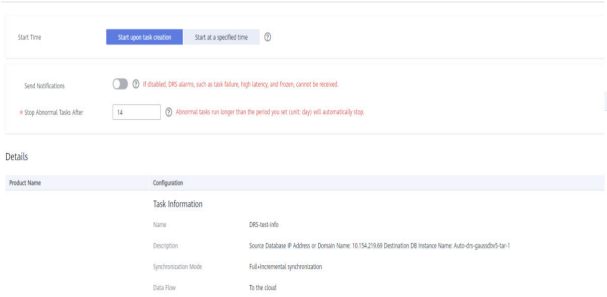
b. If any check item fails, review the cause and rectify the fault. Then, click **Check Again**.



c. After all check items pass the pre-check, click **Next**.

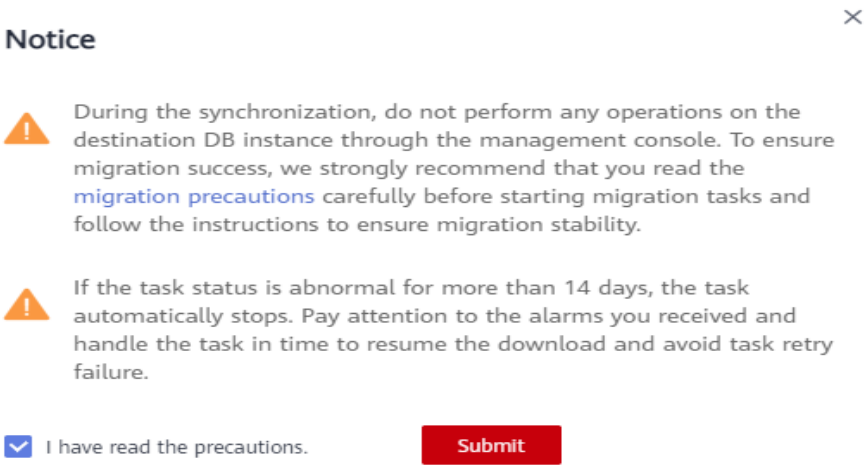
11. Confirm the task.

a. Check that all configured information is correct.



b. Click **Submit**. In the display box, select **I have read the precautions**.

c. Click **Submit**.



12. After the task is submitted, view and manage it.

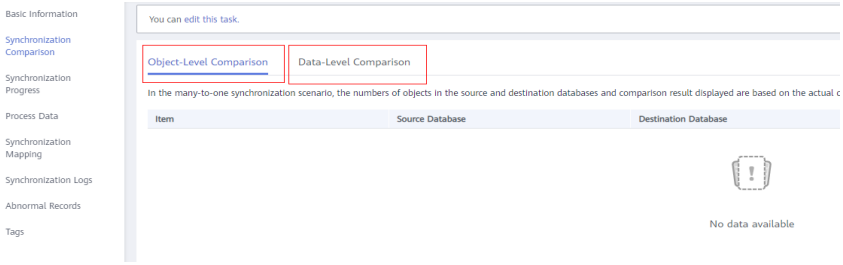
    After the task is created, return to the task list to view the status of the created task.

Step 5: Verify Data After Migration

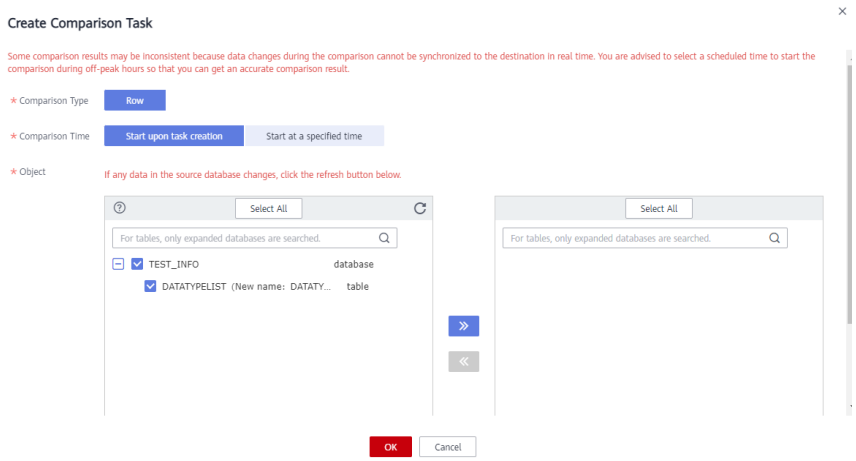
When the task status changes to **Incremental**, the full synchronization is complete. You can log in to GaussDB and view the data migration result.

1. Wait until the migration task status becomes **Incremental**.

2. Click the task name to go to the **Basic Information** page.
3. Verify data consistency.
   a. Choose **Synchronization Comparison** > **Object-Level Comparison** to view the database and table migration results.



   b. Choose **Synchronization Comparison** > **Data-Level Comparison**, click **Create Comparison Task**, and view the migration results of the rows in the table.



4. Connect to the **test_database_info** database in GaussDB using DAS.

   For details about how to connect to an instance through DAS, see **Adding DB Instance Login Information**.

5. Run the following statement to query the full synchronization result:

   ```
   SELECT * FROM test_info.datatypelist_after;
   ```

   After the schema in the MySQL database is migrated, it will be used as the schema in GaussDB. Therefore, it is required to add the schema in the query statement for exact query.

   The query result shows that all data types in the table were successfully synchronized and the data is correct.

6. Verify incremental synchronization.

   In full+incremental synchronization, after the full synchronization is complete, the data that is written to the source database after the task is created can still be synchronized to the destination database until the task is stopped. The following describes how to synchronize incremental data from the source database to the destination database:

   a. Use a database connection tool to connect to the source MySQL database based on its IP address.

   b. Run the following statement to insert a data record into the source database:

      Insert a data record whose ID is 3.

      ```
      insert into test_info.test_table values
      (3,'a','b','111','111','tinyblob','mediumblob','longblob','tinytext','text','mediumtext','longtext','1','3',1
      ,2,3,4,1.123,1.1234,'2024-03-08','2024-03-08 08:00:00','2024-03-08
      08:00:00','08:00:00','2024','1010','{"a":"b"}',1.23,1.234);
      ```

   c. Run the following statement in the destination database to query the result:

      ```
      SELECT * FROM test_info.datatypelist_after;
      ```

      The query result shows that new data in the source database has been synchronized to the destination database in real time.

7. Stop the migration task.

   After data is completely migrated to the destination database, stop the synchronization task.

   a. Locate the task and click **Stop** in the **Operation** column.

b. In the display box, click **Yes**.

4.4 Migrating Data to GaussDB Using the Export and Import Functions of DAS

Scenarios

Data Admin Service (DAS) is a one-stop management platform that allows you to manage Huawei Cloud databases on a web console. It offers database development, O&M, and intelligent diagnosis, making it easy for you to use and maintain databases.

To back up or migrate data, you can use DAS to export data from the source database first and then import the data from your local PC or OBS bucket to the destination database.

For more information, see **Data Import and Export**.

Constraints

- The file to be imported should be no larger than 1 GB.
- Only data files in the CSV or SQL format can be imported.
- Binary fields such as BINARY, VARBINARY, TINYBLOB, BLOB, MEDIUMBLOB, and LONGBLOB are not supported.
- Data cannot be exported or imported using cross-region OBS buckets.

Exporting Data

Step 1 **Log in to the management console**.

Step 2 Click in the upper left corner and select the desired region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, locate the DB instance you want to log in to and click **Log In** in the **Operation** column.

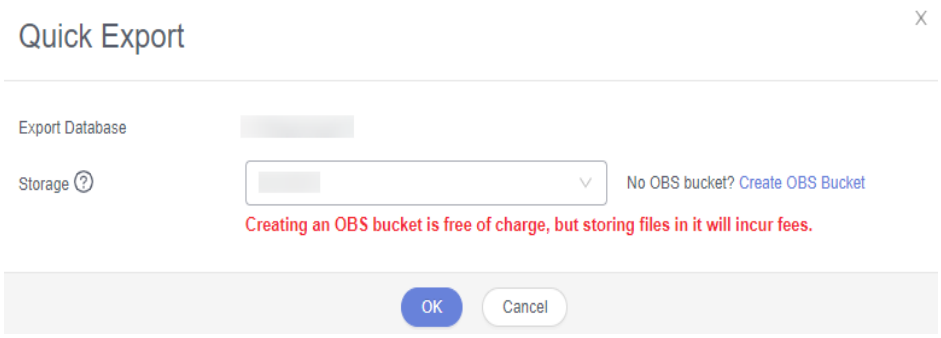
Step 5 On the displayed login page, enter the username and password and click **Log In**.

Step 6 On the top menu bar, choose **Import and Export** > **Export**.

Step 7 On the displayed page, click **Create Task** and choose **Export Database** or **Export SQL Result** as required. The following takes database export as an example.

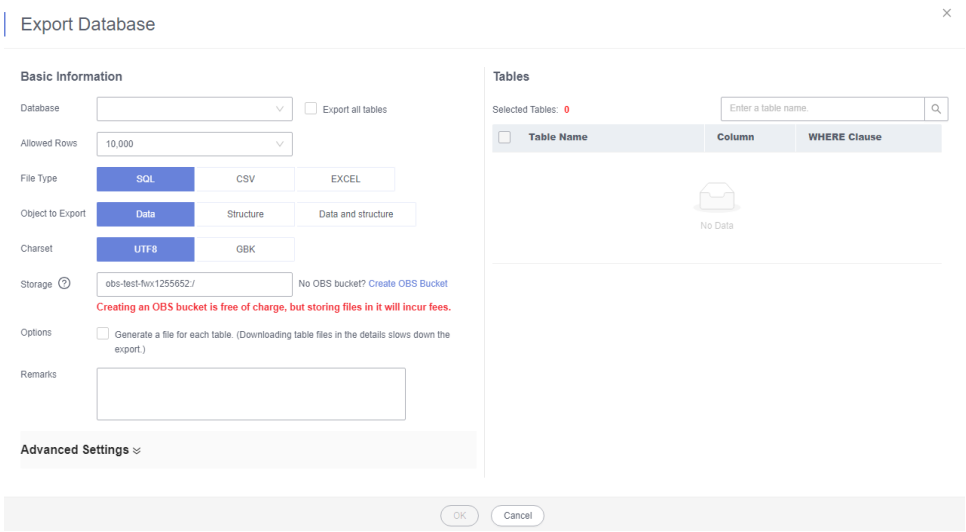
Alternatively, click **Quick Export** and select the target database. On the displayed page, select a storage path and click **OK**.

Figure 4-21 Quick export



Step 8 On the displayed page, set parameters as required in areas **Basic Information** and **Advanced Settings**. Then, select the tables to be exported on the right.

Figure 4-22 Creating an export task



NOTE

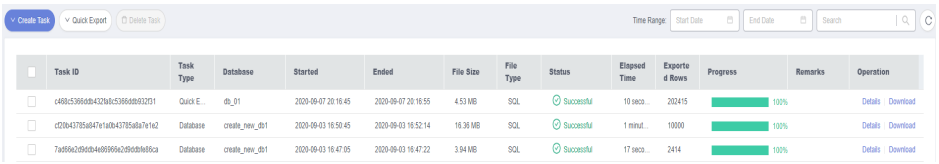
- In a SQL result export task, the executed SQL statements cannot exceed 5 MB.
- Databases are classified as user databases or system databases. System databases cannot be exported. If system database data is required, deploy system database services in a created user database, so that you can export the system database data from the user database.
- DAS connects to your standby database to export data. This prevents the primary database from being affected by data export. However, if the standby database has a high replication delay, the exported data may not be the latest.

Step 9 After settings are complete, click **OK**.

Step 10 In the task list, view the task ID, type, status, and progress.

Step 11 Click **Details** in the **Operation** column to view task details.

Figure 4-23 Task list



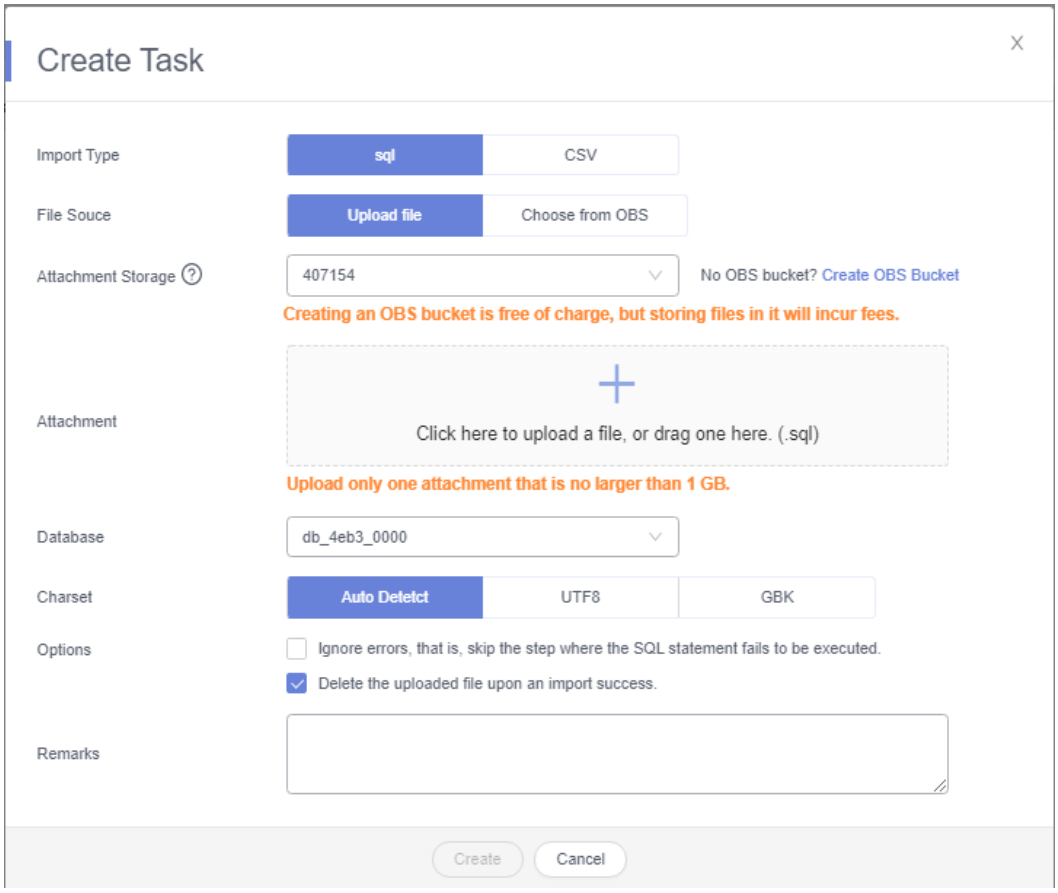
----End

Importing Data

Step 1 On the top menu bar, choose **Import and Export** > **Import**.

Step 2 Import a file from your local PC or an OBS bucket.

Figure 4-24 Creating an import task



- From your local PC

  In the upper left corner, click **Create Task**. On the displayed page, select an import type, select **Upload file** for **File Source**, set the attachment storage, and upload the file. Then, set other parameters as required.

  For security purposes, imported files are stored in OBS buckets.

NOTE

- To keep your data secure, provide your own OBS bucket to store the attachments you upload. In this way, DAS automatically connects to your OBS bucket for in-memory reading.
- If you select **Delete the uploaded file upon an import success**, the file you uploaded will be automatically deleted from the OBS bucket after being imported to the destination database.

- From an OBS bucket

  In the upper left corner, click **Create Task**. On the displayed page, select an import type, select **Choose from OBS** for **File Source**, and select a file from the bucket. Then, set other parameters as required.

  NOTE

  The file uploaded from an OBS bucket will not be deleted upon an import success.

Step 3 After setting the import parameters, click **Create**. Confirm the information again before you click **OK** because original data may be overwritten after data import.

Step 4 View the import progress in the task list or check task details.

----End

4.5 Using the copy to/from Command to Export and Import Data

Scenarios

The gsql tool provides the **\copy** meta-command to import or export data. **\copy** applies only to small-scale data import in good format. It does not preprocess invalid characters or provide error tolerance. Therefore, **\copy** cannot be used in scenarios where abnormal data exists.

Preparing for Data Migration

1. Prepare an ECS or a device that can access the GaussDB instance over EIP.
   - To connect to a GaussDB instance through an ECS, you must first create an ECS.

     For details on how to create and log in to an ECS, see **Purchasing an ECS** and **Logging In to an ECS** in *Elastic Cloud Server Getting Started*.
   - To connect to a GaussDB instance through a device that can access the GaussDB instance over EIP, you must:
     i. Bind an EIP to the GaussDB instance. For details, see **Binding an EIP**.
     ii. Ensure that the local device can access the EIP that has been bound to the GaussDB instance.
2. Install the gsql client on the prepared ECS or device that can access the GaussDB database, and connect it to the GaussDB database instance. For details, see **Using gsql to Connect to a Database**.

Exporting Data

On the prepared ECS or device that can access GaussDB, connect to the GaussDB instance and export the content of the **copy_example** table.

- Method 1: Export the content of the **copy_example** table to **stdout** in CSV format. Use quotation marks (") as the quotes, and use the quotes to enclose the fourth and fifth columns.

```
\copy copy_example to stdout CSV quote as '"' force quote col_4,col_5;
1,iamtext,iamvarchar,"2006-07-07","12:00:00"
2,sometext,somevarchar,"2006-07-07","12:00:00"
3,sometext,somevarchar,"2006-07-07","12:00:00"
4,sometext,somevarchar,"2022-07-07","19:00:02"
5,sometext,somevarchar,"2006-07-07",
6,sometext,somevarchar,"2022-07-07","19:00:02"
```

- Method 2: Export the content of the **copy_example** table to the **copy_example.csv** file under the local path **/tmp/data/**. Use vertical bars (|) as the delimiters and quotation marks (") as the quotes.

```
\copy copy_example to '/tmp/data/copy_example.csv' csv delimiter '|' quote '"';
```

  Check the **/tmp/data/copy_example.csv** file to ensure that the data has been exported.

```
1|iamtext|iamvarchar|2006-07-07|12:00:00
2|sometext|somevarchar|2006-07-07|12:00:00
3|sometext|somevarchar|2006-07-07|12:00:00
4|sometext|somevarchar|2022-07-07|19:00:02
5|sometext|somevarchar|2006-07-07|
6|sometext|somevarchar|2022-07-07|19:00:02
```

- Method 3: Export the query result set of the **copy_example** table to the **copy_example2.csv** file under the local path **/tmp/data/** . Use commas (,) as the delimiters and quotation marks (") as the quotes.

```
\copy (select * from copy_example where col_1 = 1) to '/tmp/data/copy_example2.csv' csv delimiter ',' quote '"';
```

  Check the **/tmp/data/copy_example2.csv** file to ensure that the data has been exported.

```
1,iamtext,iamvarchar,2006-07-07,12:00:00
```

Importing Data

Import data to a GaussDB instance. For example, import data to the target table **copy_example**, the schema is as follows:

```
create table copy_example
(
 col_1 integer,
 col_2 text,
 col_3 varchar(12),
 col_4 date,
 col_5 time
);
```

On the prepared ECS or device that can access GaussDB, connect to the GaussDB instance and import data to the target table **copy_example**.

- Method 1: Copy data from **stdin** and import data to the target table **copy_example**.

```
\copy copy_example from stdin csv;
```

  When **>>** is displayed, enter data. To end your input, enter a backslash and a period (\.).

```
Enter data to be copied followed by a newline.
End with a backslash and a period on a line by itself.
```



```
>> 1,"iamtext","iamvarchar",2006-07-07,12:00:00
>> 2,"sometext","somevarchar",2006-07-07,12:00:00
>> \.
```

View the imported data.

```
select * from copy_example;
 col_1 |  col_2   |    col_3    |        col_4        |  col_5
-------+----------+-------------+---------------------+----------
     1 | iamtext  | iamvarchar  | 2006-07-07 00:00:00 | 12:00:00
     2 | sometext | somevarchar | 2006-07-07 00:00:00 | 12:00:00
(2 rows)
```

- Method 2: The **example.csv** file exists in the local **/tmp/data/** directory. The file contains the header row. Use vertical bars (|) as the delimiters and quotation marks (") as the quotes. The content is as follows:

  ```
  header
  3|"sometext"|"somevarchar"|2006-07-07|12:00:00
  4|"sometext"|"somevarchar"|2022-07-07|19:00:02
  ```

  Import data from the local file **example.csv** to the target table **copy_example**. If **header** is set to **on**, the first row is automatically ignored. Use quotation marks (") as the quotes by default.

  ```
  \copy copy_example from  '/tmp/data/example.csv' with(header 'on', format 'csv', delimiter '|',
  date_format 'yyyy-mm-dd',  time_format 'hh24:mi:ss');
  ```

  View the imported data.

  ```
  select * from copy_example;
   col_1 |  col_2   |    col_3    |        col_4        |  col_5
  -------+----------+-------------+---------------------+----------
       1 | iamtext  | iamvarchar  | 2006-07-07 00:00:00 | 12:00:00
       2 | sometext | somevarchar | 2006-07-07 00:00:00 | 12:00:00
       3 | sometext | somevarchar | 2006-07-07 00:00:00 | 12:00:00
       4 | sometext | somevarchar | 2022-07-07 00:00:00 | 19:00:02
  (4 rows)
  ```

- Method 3: The **example2.csv** file exists in the local directory **/tmp/data/**. Use commas (,) as the delimiters and quotation marks (") as the quotes. The last field is missing in the first line, and the last field is added in the second line. The content is as follows:

  ```
  5,"sometext","somevarchar",2006-07-07
  6,"sometext","somevarchar",2022-07-07,19:00:02,12:00:00
  ```

  Import data from the local file **example2.csv** to the target table **copy_example**. The default delimiters are commas (,). Because the error tolerance parameters IGNORE_EXTRA_DATA and FILL_MISSING_FIELDS are specified, the missing fields are replaced with NULL, and the extra fields are ignored.

  ```
  \copy copy_example from  '/tmp/data/example2.csv' with( format 'csv', date_format 'yyyy-mm-dd',
  time_format 'hh24:mi:ss', IGNORE_EXTRA_DATA 'true', FILL_MISSING_FIELDS 'true');
  ```

  View the imported data.

  ```
  select * from copy_example;
   col_1 |  col_2   |    col_3    |        col_4        |  col_5
  -------+----------+-------------+---------------------+----------
       1 | iamtext  | iamvarchar  | 2006-07-07 00:00:00 | 12:00:00
       2 | sometext | somevarchar | 2006-07-07 00:00:00 | 12:00:00
       3 | sometext | somevarchar | 2006-07-07 00:00:00 | 12:00:00
       4 | sometext | somevarchar | 2022-07-07 00:00:00 | 19:00:02
       5 | sometext | somevarchar | 2006-07-07 00:00:00 |
       6 | sometext | somevarchar | 2022-07-07 00:00:00 | 19:00:02
  (6 rows)
  ```

Helpful Links

For more information, see:

- **COPY (distributed)**
- **COPY (centralized)**

4.6 Using CopyManager in JDBC to Export and Import Data

Scenarios

When you use Java to develop applications, the CopyManager API of the JDBC driver is called to write data from files or other source databases to GaussDB.

Example 1: Importing and Exporting Data Through Local Files

Call the CopyManager API of the JDBC driver to export data from the database to a local file or import a local file to the database in streaming mode. The file format can be CSV or TEXT.

The sample program is as follows. Load the GaussDB JDBC driver before executing it.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.io.IOException;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.sql.SQLException;
import org.postgresql.copy.CopyManager;
import org.postgresql.core.BaseConnection;

public class Copy{

    public static void main(String[] args)
    {
     String urls = new String("jdbc:postgresql://localhost:8000/postgres"); // URL of the database
     String username = System.getenv("EXAMPLE_USERNAME_ENV");     // Username
     String password = System.getenv("EXAMPLE_PASSWORD_ENV");     // Password
     String tablename = new String("migration_table"); // Define table information.
     String tablename1 = new String("migration_table_1"); // Define table information.
     String driver = "org.postgresql.Driver";
     Connection conn = null;

     try {
         Class.forName(driver);
         conn = DriverManager.getConnection(urls, username, password);
     } catch (ClassNotFoundException e) {
          e.printStackTrace(System.out);
     } catch (SQLException e) {
          e.printStackTrace(System.out);
     }

     // Export data from the migration_table table to the local d:/data.txt file.
     try {
         copyToFile(conn, "d:/data.txt", "(SELECT * FROM migration_table)");
     } catch (SQLException e) {
         // TODO Auto-generated catch block
         e.printStackTrace();
     } catch (IOException e) {
         // TODO Auto-generated catch block
         e.printStackTrace();
     }
     // Import data from the d:/data.txt file to the migration_table_1 table.
```



```
    try {
        copyFromFile(conn, "d:/data.txt", tablename1);
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    // Export data from the migration_table_1 table to the local d:/data1.txt file.
    try {
        copyToFile(conn, "d:/data1.txt", tablename1);
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

public static void copyFromFile(Connection connection, String filePath, String tableName)
        throws SQLException, IOException {

    FileInputStream fileInputStream = null;

    try {
        CopyManager copyManager = new CopyManager((BaseConnection)connection);
        fileInputStream = new FileInputStream(filePath);
        copyManager.copyIn("COPY " + tableName + " FROM STDIN ", fileInputStream);
    } finally {
        if (fileInputStream != null) {
            try {
                fileInputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
    public static void copyToFile(Connection connection, String filePath, String tableOrQuery)
            throws SQLException, IOException {

     FileOutputStream fileOutputStream = null;

     try {
        CopyManager copyManager = new CopyManager((BaseConnection)connection);
        fileOutputStream = new FileOutputStream(filePath);
        copyManager.copyOut("COPY " + tableOrQuery + " TO STDOUT", fileOutputStream);
     } finally {
        if (fileOutputStream != null) {
            try {
                fileOutputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
     }
  }
}
```

Example 2: Migrating Data from Database B

The following example shows how to use CopyManager to migrate data from database B to GaussDB.

```
import java.io.StringReader;
import java.sql.Connection;
```



```
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import org.postgresql.copy.CopyManager;
import org.postgresql.core.BaseConnection;

public class Migration{

    public static void main(String[] args) {
        String url = new String("jdbc:postgresql://localhost:8000/postgres"); //URL of the database
        String user = new String("username");             // GaussDB database username
        String pass = new String("passwd");              // GaussDB database password
        String tablename = new String("migration_table_1"); // Define table information.
       String delimiter = new String(",");               // Define a delimiter.
        String encoding = new String("UTF8");             // Define a character set.
        String driver = "org.postgresql.Driver";
        StringBuffer buffer = new StringBuffer();       // Define a buffer to store formatted data.

        try {
            // Obtain the query result set of the source database.
            ResultSet rs = getDataSet();

            // Traverse the result set and obtain records row by row.
            // The values of columns in each record are separated by the specified delimiter and end with a
newline character to form strings.
            // Add the strings to the buffer.
            while (rs.next()) {
                buffer.append(rs.getString(1) + delimiter
                        + rs.getString(2) + delimiter
                        + rs.getString(3) + delimiter
                        + rs.getString(4)
                        + "\n");
            }
            rs.close();

            try {
                // Connect to the destination database.
                Class.forName(driver);
                Connection conn = DriverManager.getConnection(url, user, pass);
                BaseConnection baseConn = (BaseConnection) conn;
                baseConn.setAutoCommit(false);

                // Initialize table information.
                String sql = "Copy " + tablename + " from STDIN with (DELIMITER " + "'" + delimiter + "'" +","+ "
ENCODING " + "'" + encoding + "')";

                // Commit data in the buffer.
                CopyManager cp = new CopyManager(baseConn);
                StringReader reader = new StringReader(buffer.toString());
                cp.copyIn(sql, reader);
                baseConn.commit();
                reader.close();
                baseConn.close();
            } catch (ClassNotFoundException e) {
                e.printStackTrace(System.out);
            } catch (SQLException e) {
                e.printStackTrace(System.out);
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    //**********************************
    // Return the query result set from the source database.
    //**********************************
```



```
    private static ResultSet getDataSet() {
        ResultSet rs = null;
        try {
            Class.forName("com.B.jdbc.Driver").newInstance();
            Connection conn = DriverManager.getConnection("jdbc:MY://10.119.179.227:3306/jack?
useSSL=false&allowPublicKeyRetrieval=true", "jack", "xxxxxxxxx");
            Statement stmt = conn.createStatement();
            rs = stmt.executeQuery("select * from migration_table");
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return rs;
    }
}
```

Helpful Links

For more information, see:

- **CopyManager (distributed)**
- **CopyManager (centralized)**

4.7 Using gs_dump and gs_dumpall to Export Data

Scenarios

GaussDB provides gs_dump and gs_dumpall to export required database objects and related information. You can use a tool to import the exported data to a destination database for database migration. gs_dump can export a single database or its objects. gs_dumpall can export all databases or global objects in the database. For details, see **Table 4-5**.

NOTE

In the multitenancy scenario, gs_dump can be used to export a single PDB or its objects, but gs_dumpall cannot support this scenario.

Table 4-5 Scenarios

Scenario	Export Granularity	Export Format	Import Method
Exporting a single database	Database-level export (see **Exporting a Database**). • Export full information of a database. You can use the exported information to create the same database containing the same data. • Export all object definitions of a database, including the definitions of the database, functions, schemas, tables, indexes, and stored procedures. You can use the exported object definitions to quickly create the same database, without data. • Export data of a database.	• Plain-text • Custom • Directory • TAR	• Use gsql to import plain-text data files. For details, see "Client Tools" > "gsql" in *Tool Reference*. • For details about how to import data files in .tar, directory, or custom format, see **Using gs_restore to Import Data**.
	Schema-level export (see **Exporting a Schema**). • Export full information of a schema. • Export data of a schema. • Export all object definitions of a schema, including the definitions of tables, stored procedures, and indexes.		
	Table-level export (see **Exporting a Table**). • Export full information of a table. • Export data of a table. • Export the definition of a table.		

Scenario	Export Granularity	Export Format	Import Method
Exporting all databases	Database-level export (see **Exporting All Databases**). • Export full information of databases. You can use the exported full information to create the same host environment containing the same databases and public global objects, with the same data. • Export all object definitions of databases, including the definitions of tablespaces, databases, functions, schemas, tables, indexes, and stored procedures. You can use the exported object definitions to quickly create the same host environment containing the same databases and tablespaces but without data. • Export data of databases.	Plain-text	For details about how to import data files, see **Using copy from to Import Data**.
	Public global object export (see **Exporting Global Objects**). • Export tablespaces. • Export roles. • Export tablespaces and roles.		

gs_dump and gs_dumpall use **-U** to specify the user that performs the export. If the specified user does not have the required permissions, data cannot be exported. For details about the scenarios where this function can be used, see **Table 4-5**.

Precautions

gs_dump and gs_dumpall encrypt the exported data files. These files are decrypted before being imported to prevent data disclosure for higher database security. Note that gsql cannot decrypt and import stored procedures and functions for plain-text files encrypted using gs_dump. Therefore, if the exported database contains stored procedures or functions, use the other three modes to export the database and use gs_restore to restore the database.

When gs_dump or gs_dumpall is used to export data, other users can still operate (read or write) the database.

gs_dump and gs_dumpall can export complete, consistent data. For example, if gs_dump exports data from database A or gs_dumpall exports data from GaussDB database at T1, the exported data is the data status of database A or GaussDB database at T1. Modified data of database A or GaussDB database after T1 will not be exported.

When gs_dump or gs_dumpall is used to export data, generated columns are not dumped.

- Do not modify the files and contents exported using the **-F c/d/t** format. Otherwise, the restoration may fail. For files exported using the **-F p** format, edit the exported files with caution if necessary.
- If the number of objects (data tables, views, and indexes) in the database exceeds 500,000, you are advised to contact technical support to improve performance and avoid memory problems.
- To ensure data consistency and integrity, the export tools will set a shared lock for the tables to be dumped. If a shared lock has been set for the table in other transactions, gs_dump and gs_dumpall lock the table after it is released. If the table cannot be locked within the specified time, the dump fails. You can customize the timeout duration to wait for lock release by specifying the **--lock-wait-timeout** parameter.
- During an export, gs_dumpall reads tables in all databases. Therefore, you need to connect to the databases as a database administrator to export a complete file. When you use gsql to execute scripts, administrator permissions are also required so as to add users and user groups, and create databases.

Preparing for Data Migration

1. Prepare an ECS or a device that can access the GaussDB instance over EIP.
   - To connect to a GaussDB instance through an ECS, you must first create an ECS.

     For details on how to create and log in to an ECS, see **Purchasing an ECS** and **Logging In to an ECS** in *Elastic Cloud Server Getting Started*.
   - To connect to a GaussDB instance through a device that can access the GaussDB instance over EIP, you must:
     i. Bind an EIP to the GaussDB instance. For details, see **Binding an EIP**.
     ii. Ensure that the local device can access the EIP that has been bound to the GaussDB instance.
2. Install the gsql client on the prepared ECS or device that can access the GaussDB database, and connect it to the GaussDB database instance. For details, see **Using gsql to Connect to a Database**.

Exporting a Database

1. Create the database and table to be exported and insert data into them.

```
create database gs_example;

\c gs_example
password:

create schema gs_sch_example;
set search_path to gs_sch_example;
create table gs_table_example
```



```
(
 col_1 integer,
 col_2 text,
 col_3 varchar(12),
 col_4 date,
 col_5 time
);
insert into gs_table_example values(1,'iamtext','iamvarchar','2006-07-07','12:00:00');
insert into gs_table_example values(2,'sometext','somevarchar','2006-07-07','12:00:00');
insert into gs_table_example values(3,'sometext','somevarchar','2006-07-07','12:00:00');
insert into gs_table_example values(4,'sometext','somevarchar','2006-07-07','19:00:02');
insert into gs_table_example values(5,'sometext','somevarchar','2006-07-07', null);
insert into gs_table_example values(6,'sometext','somevarchar','2006-07-07','19:00:02');
```

2. Use gs_dump to export data of the **gs_example** database.
   - Example 1: Use gs_dump to export full information of the **gs_example** database by specifying the database IP address. The exported files are in .sql format.

     ```
     gs_dump -U root -f /tmp/data/gs_example_dump.sql -p 8000 gs_example -F p -h 192.*.*.139;
     Password:
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:04:20]: The total objects number is 458.
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:04:20]: [100.00%] 458 objects have been dumped.
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:04:20]: dump database gs_example successfully
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:04:20]: total time: 8779  ms
     ```

   - Example 2: Use gs_dump to export full information of the **gs_example** database by specifying the database IP address. The exported information is archived to the **/tmp/data/gs_example_dump.tar** file in .tar format.

     ```
     gs_dump -U root -f /tmp/data/gs_example_dump.tar -p 8000 gs_example -F t -h 192.*.*.139;
     Password:
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 14:58:49]: The total objects number is 458.
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 14:58:49]: [100.00%] 458 objects have been dumped.
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 14:58:49]: dump database gs_example successfully
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 14:58:49]: total time: 8201  ms
     ```

   - Example 3: Use gs_dump to export data of the **gs_example** database by specifying the database IP address. The exported data does not contain object definitions of the database. The exported files are in custom format.

     ```
     gs_dump -U root -f /tmp/data/gs_example_dump.dmp -p 8000 gs_example -a -F c -h 192.*.*.139;
     Password:
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:07:23]: dump database gs_example successfully
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:07:23]: total time: 8369  ms
     ```

   - Example 4: Use gs_dump to export all object definitions of the **gs_example** database by specifying the database IP address. The exported files are in .sql format.

     ```
     gs_dump -U root -f /tmp/data/gs_example_dump_s.sql -p 8000 gs_example -s -F p -h 192.*.*.139;
     Password:
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:09:37]: The total objects number is 457.
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:09:37]: [100.00%] 457 objects have been dumped.
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:09:37]: dump database gs_example successfully
     ```



```
gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:09:37]: total time:
8523  ms
```

– Example 5: Use gs_dump to export all object definitions of the **gs_example** database by specifying the database IP address. The exported files are encrypted in .txt format.

```
gs_dump -U root -f /tmp/data/gs_example_dump_s_key.sql -p 8000 gs_example --with-
encryption AES128 --with-key abcdefg_?1234567 -s -F p -h 192.*.*.139;
Password:
gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:10:38]: The total
objects number is 457.
gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:10:38]: [100.00%]
457 objects have been dumped.
gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:10:38]: dump
database gs_example successfully
gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:10:38]: total time:
9101  ms
```

Table 4-6 Common parameters

Parameter	Description	Example
-U	Username for database connection. **NOTE** If the username for connecting to the database is not specified, the initial system administrator created during installation is used for connection by default.	-U jack
-W	User password for database connection. • This parameter is not required for database administrators if the trust policy is used for authentication. • If you connect to the database without specifying this parameter and you are not a database administrator, you will be prompted to enter the password.	-W ********
-f	Folder to store exported files. If this parameter is not specified, the exported files are stored in the standard output. If the output format is **(-F c/-F d/-F t)**, the **-f** parameter must be specified.	-f /home/omm/backup/*MPPDB*_backup.tar

Parameter	Description	Example
-p	TCP port or local Unix-domain socket file name extension on which the server is listening for connections.	-p 8000
dbname	Name of the database to be exported.	testdb
-F	Format of exported files. The values are as follows: • **p**: plain-text • **c**: custom • **d**: directory • **t**: TAR	-F t

Exporting a Schema

1. Create a schema to be exported and insert data into it.

```
create database gs_example;

\c gs_example
password:

create schema gs_sch_example;
create schema gs_sch_1_example;
create table gs_sch_example.gs_table_example
(
 col_1 integer,
 col_2 text,
 col_3 varchar(12),
 col_4 date,
 col_5 time
);
create table gs_sch_1_example.gs_table_example
(
 col_1 integer,
 col_2 text,
 col_3 varchar(12),
 col_4 date,
 col_5 time
);
insert into gs_sch_example.gs_table_example values(1,'iamtext','iamvarchar','2006-07-07','12:00:00');
insert into gs_sch_example.gs_table_example
values(2,'sometext','somevarchar','2006-07-07','12:00:00');
insert into gs_sch_example.gs_table_example
values(3,'sometext','somevarchar','2006-07-07','12:00:00');
insert into gs_sch_example.gs_table_example
values(4,'sometext','somevarchar','2006-07-07','19:00:02');
insert into gs_sch_example.gs_table_example values(5,'sometext','somevarchar','2006-07-07', null);
insert into gs_sch_example.gs_table_example
values(6,'sometext','somevarchar','2006-07-07','19:00:02');
insert into gs_sch_1_example.gs_table_example values(7,'iamtext','iamvarchar','2006-07-07','12:00:00');
insert into gs_sch_1_example.gs_table_example
values(8,'sometext','somevarchar','2006-07-07','12:00:00');
insert into gs_sch_1_example.gs_table_example
values(9,'sometext','somevarchar','2006-07-07','12:00:00');
insert into gs_sch_1_example.gs_table_example
values(10,'sometext','somevarchar','2006-07-07','19:00:02');
insert into gs_sch_1_example.gs_table_example values(11,'sometext','somevarchar','2006-07-07', null);
```



```
insert into gs_sch_1_example.gs_table_example
values(12,'sometext','somevarchar','2006-07-07','19:00:02');
```

2. Use gs_dump to export schemas from the **gs_example** database at the same time.
   - Example 1: Use gs_dump to export the **gs_sch_example** and **gs_sch_1_example** schemas at the same time by specifying the database IP address. The exported files are in directory format.

     ```
     gs_dump -U root -f /tmp/data/gs_sch_dump -p 8000 gs_example -n gs_sch_example -n gs_sch_1_example -F d -h 192.*.*.139;
     Password:
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:37:11]: The total objects number is 460.
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:37:11]: [100.00%] 460 objects have been dumped.
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:37:11]: dump schema gs_sch_example gs_sch_1_example successfully
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:37:11]: dump database gs_example successfully
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:37:11]: total time: 9602  ms
     ```

   - Example 2: Use gs_dump to export full information of the **gs_sch_example** schema by specifying the database IP address. The exported files are in .txt format.

     ```
     gs_dump -U root -f /tmp/data/gs_sch_dump.sql -p 8000 gs_example -n gs_sch_example -F p -h 192.*.*.139;
     Password:
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:39:00]: The total objects number is 457.
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:39:00]: [100.00%] 457 objects have been dumped.
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:39:00]: dump schema gs_sch_example successfully
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:39:00]: dump database gs_example successfully
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:39:00]: total time: 8582  ms
     ```

   - Example 3: Use gs_dump to export data from the **gs_example** database by specifying the database IP address. The exported data does not contain the **gs_sch_example** schemas. The exported files are in custom format.

     ```
     gs_dump -U root -f /tmp/data/gs_sch_dump.dmp -p 8000 gs_example -N gs_sch_example -F c -h 192.*.*.139;
     Password:
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:41:14]: The total objects number is 458.
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:41:14]: [100.00%] 458 objects have been dumped.
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:41:14]: dump database gs_example successfully
     gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:41:14]: total time: 8323  ms
     ```

Table 4-7 Common parameters

Parameter	Description	Example
-U	Username for database connection.	-U jack

Parameter	Description	Example
-W	User password for database connection. ● This parameter is not required for database administrators if the trust policy is used for authentication. ● If you connect to the database without specifying this parameter and you are not a database administrator, you will be prompted to enter the password.	-W ********
-f	Folder to store exported files. If this parameter is not specified, the exported files are stored in the standard output.	-f /home/omm/backup/MPPDB_schema_backup
-p	TCP port or local Unix-domain socket file name extension on which the server is listening for connections.	-p 8000
dbname	Name of the database to be exported.	human_resource
-n	Names of schemas to be exported. This option contains the schema and all its contained objects. ● Single schema: Enter **-n** *schemaname*. ● Multiple schemas: Enter **-n** *schemaname* for each schema.	● Single schemas: **-n hr** ● Multiple schemas: **-n hr -n public**
-F	Format of exported files. The values are as follows: ● **p**: plain-text ● **c**: custom ● **d**: directory ● **t**: TAR	-F d

Exporting a Table

1. Create a schema to be exported and insert data into it.

```
create database gs_example;
```



```
\c gs_example
password:

create schema gs_sch_example;
create table gs_sch_example.gs_table_example
(
 col_1 integer,
 col_2 text,
 col_3 varchar(12),
 col_4 date,
 col_5 time
);
create table gs_sch_example.gs_table_example_2
(
 col_1 integer,
 col_2 text,
 col_3 varchar(12),
 col_4 date,
 col_5 time
);
insert into gs_sch_example.gs_table_example values(1,'iamtext','iamvarchar','2006-07-07','12:00:00');
insert into gs_sch_example.gs_table_example
values(2,'sometext','somevarchar','2006-07-07','12:00:00');
insert into gs_sch_example.gs_table_example
values(3,'sometext','somevarchar','2006-07-07','12:00:00');
insert into gs_sch_example.gs_table_example
values(4,'sometext','somevarchar','2006-07-07','19:00:02');
insert into gs_sch_example.gs_table_example values(5,'sometext','somevarchar','2006-07-07', null);
insert into gs_sch_example.gs_table_example
values(6,'sometext','somevarchar','2006-07-07','19:00:02');
insert into gs_sch_example.gs_table_example_2 values(7,'iamtext','iamvarchar','2006-07-07','12:00:00');
insert into gs_sch_example.gs_table_example_2
values(8,'sometext','somevarchar','2006-07-07','12:00:00');
insert into gs_sch_example.gs_table_example_2
values(9,'sometext','somevarchar','2006-07-07','12:00:00');
insert into gs_sch_example.gs_table_example_2
values(10,'sometext','somevarchar','2006-07-07','19:00:02');
insert into gs_sch_example.gs_table_example_2 values(11,'sometext','somevarchar','2006-07-07', null);
insert into gs_sch_example.gs_table_example_2
values(12,'sometext','somevarchar','2006-07-07','19:00:02');
```

2. Use gs_dump to export the **gs_sch_example.gs_table_example** and **gs_sch_example.gs_table_example_2** tables at the same time.

   NOTE

   1. In the following example, after the export, ensure that the schema to which the exported table belongs exists before the import.

   – Example 1: Use gs_dump to export the **gs_sch_example.gs_table_example** and **gs_sch_example.gs_table_example_2** tables at the same time by specifying the database IP address. The exported files are in directory format.

```
gs_dump -U root -f /tmp/data/gs_table_dump -p 8000 gs_example -t
gs_sch_example.gs_table_example -t gs_sch_example.gs_table_example_2 -F d -h 192.*.*.139;
Password:
gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:49:06]: The total
objects number is 458.
gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:49:06]: [100.00%]
458 objects have been dumped.
gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:49:06]: dump table
gs_sch_example.gs_table_example gs_sch_example.gs_table_example_2 successfully
gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:49:06]: dump
database gs_example successfully
gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:49:06]: total time:
7694  ms
```


– Example 2: Use gs_dump to export the tables excluding the **gs_sch_example.gs_table_example_2** table by specifying the database IP address. The exported files are in custom format.

```
gs_dump -U root -f /tmp/data/gs_table_dump.dmp -p 8000 gs_example -T
gs_sch_example.gs_table_example_2 -F c -h 192.*.*.139;
Password:
gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:52:07]: The total
objects number is 461.
gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:52:07]: [100.00%]
461 objects have been dumped.
gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:52:07]: dump
database gs_example successfully
gs_dump[user='root'][localhost][port='8000'][gs_example][2024-07-26 15:52:07]: total time:
8203  ms
```

Table 4-8 Common parameters

Parameter	Description	Example
-U	Username for database connection.	-U jack
-W	User password for database connection. ● This parameter is not required for database administrators if the trust policy is used for authentication. ● If you connect to the database without specifying this parameter and you are not a database administrator, you will be prompted to enter the password.	-W ********
-f	Folder to store exported files. If this parameter is not specified, the exported files are stored in the standard output.	-f /home/omm/backup/MPPDB_table_backup
-p	TCP port or local Unix-domain socket file name extension on which the server is listening for connections.	-p 8000
dbname	Name of the database to be exported.	human_resource

Parameter	Description	Example
-t	Tables (or views, sequences, foreign tables) to export. You can specify multiple tables by listing them or using wildcard characters. When you use wildcard characters, quote the pattern to prevent the shell from expanding the wildcard characters. • Single table: Enter **-t** *schema.table*. • Multiple tables: Enter **-t** *schema.table* for each table.	• Single table: **-t hr.staffs** • Multiple tables: **-t hr.staffs -t hr.employments**
-F	Format of exported files. The values are as follows: • **p**: plain-text • **c**: custom • **d**: directory • **t**: TAR	-F d
-T	A list of tables, views, sequences, or foreign tables not to be dumped. You can use multiple **-t** parameters or wildcard characters to specify tables. When **-t** and **-T** are input, the object will be stored in **-t** list not **-T** table object.	-T *table1*

Exporting All Databases

Use gs_dumpall to export all database information at a time.

- Example 1: Use gs_dumpall to export all database information by specifying the database IP address. The exported file is in .sql format. After the command is executed, a large amount of output information will be displayed. **total time** will be displayed at the end of the information, indicating that the command is executed successfully. In this example, only relative output information is included.

```
gs_dumpall -U root -f /tmp/data/dumpall.sql -p 8000 -h 192.*.*.139;
Password:
gs_dumpall[user='root'][localhost][port='8000'][2024-07-26 16:02:15]: dumpall operation successful
gs_dumpall[user='root'][localhost][port='8000'][2024-07-26 16:02:15]: total time: 35133  ms
```

- Example 2: Use gs_dumpall to export all database definitions by specifying the database IP address. The exported file is in .sql format. After the command is executed, a large amount of output information will be displayed. **total time** will be displayed at the end of the information,

indicating that the command is executed successfully. In this example, only relative output information is included.

```
gs_dumpall -U root -f /tmp/data/dumpall_def.sql -p 8000 -s -h 192.*.*.139;
Password:
gs_dumpall[user='root'][localhost][port='8000'][2024-07-26 16:07:50]: dumpall operation successful
gs_dumpall[user='root'][localhost][port='8000'][2024-07-26 16:07:50]: total time: 21239  ms
```

Table 4-9 Common parameters

Parameter	Description	Example
-U	Username for database connection. The user must be a database administrator.	-U omm
-W	User password for database connection. ● This parameter is not required for database administrators if the trust policy is used for authentication. ● If you connect to the database without specifying this parameter and you are not a database administrator, you will be prompted to enter the password.	-W ********
-f	Folder to store exported files. If this parameter is not specified, the exported files are stored in the standard output.	-f /home/omm/backup/MPPDB_backup.sql
-p	TCP port or local Unix-domain socket file name extension on which the server is listening for connections.	-p 8000

Exporting Global Objects

Use gs_dumpall to export tablespace object information.

- Example 1: Use gs_dumpall to export the global tablespace and user information of all databases by specifying the database IP address. The exported files are in .sql format. In this example, only relative output information is included.

  ```
  gs_dumpall -U root -f /tmp/data/dumpall_tablespace.sql -p 8000 -t -h 192.*.*.139;
  Password:
  gs_dumpall[user='root'][localhost][port='8000'][2024-07-26 16:10:42]: dumpall operation successful
  gs_dumpall[user='root'][localhost][port='8000'][2024-07-26 16:10:42]: total time: 1800  ms
  ```

- Example 2: Use gs_dumpall to export the global user information of all databases by specifying the database IP address. The exported files are in .txt format. In this example, only relative output information is included.

  ```
  gs_dumpall -U root -f /tmp/data/dumpall_user.sql -p 8000 -r -h 192.*.*.139;
  Password:
  ```



```
gs_dumpall[user='root'][localhost][port='8000'][2024-07-26 16:12:15]: dumpall operation successful
gs_dumpall[user='root'][localhost][port='8000'][2024-07-26 16:12:15]: total time: 1269  ms
```

Table 4-10 Common parameters

Parameter	Description	Example
-U	Username for database connection. The user must be a database administrator.	-U omm
-W	User password for database connection. • This parameter is not required for database administrators if the trust policy is used for authentication. • If you connect to the database without specifying this parameter and you are not a database administrator, you will be prompted to enter the password.	-W ********
-f	Folder to store exported files. If this parameter is not specified, the exported files are stored in the standard output.	-f /home/omm/backup/*MPPDB_tablespace*.sql
-p	TCP port or local Unix-domain socket file name extension on which the server is listening for connections.	-p 8000
-t	Dumping only tablespaces. You can also use **--tablespaces-only** alternatively.	-t

Helpful Links

For more information, see:

- **gs_dump (distributed)**
- **gs_dump (centralized)**
- **gs_dumpall (distributed)**
- **gs_dumpall (centralized)**

4.8 Using gs_restore to Import Data

Scenarios

gs_restore is an import tool provided by GaussDB and works together with gs_dump. You can use gs_restore to import the files exported by gs_dump to a database. gs_restore can import the files in .tar, custom, or directory format.

gs_restore can:

- Import data to a database.

  If a database is specified, data is imported to the database. If parallel import is enabled, the password for connecting to the database also needs to be specified. During data import, the generated columns are automatically updated and saved as common columns.

- Import data to a script file.

  If no database is specified, a script containing the SQL statement to rebuild the database is created and written to a file or standard output. This script file is equivalent to the plain-text output of gs_dump.

You can specify and sort the data to import.

Precautions

gs_restore incrementally imports data by default. To prevent data exception caused by consecutive imports, use the **-c** and **-e** parameters for each import. **-c** indicates that the database objects that already exist in the database to be restored are cleared (deleted) before the database objects are rebuilt. **-e** indicates that if an error occurred when an SQL statement is sent to the database, the system exits. By default, the system continues to import data and displays a series of error information after the import is complete.

Preparing for Data Migration

1. Prepare an ECS or a device that can access the GaussDB instance over EIP.
   - To connect to a GaussDB instance through an ECS, you must first create an ECS.

     For details on how to create and log in to an ECS, see **Purchasing an ECS** and **Logging In to an ECS** in *Elastic Cloud Server Getting Started*.
   - To connect to a GaussDB instance through a device that can access the GaussDB instance over EIP, you must:
     i. Bind an EIP to the GaussDB instance. For details, see **Binding an EIP**.
     ii. Ensure that the local device can access the EIP that has been bound to the GaussDB instance.
2. Install the gsql client on the prepared ECS or device that can access the GaussDB database, and connect it to the GaussDB database instance. For details, see **Using gsql to Connect to a Database**.

Procedure

Step 1 Upload the file exported by gs_dump to the device. For details about the file exported by gs_dump, see **Using gs_dump and gs_dumpall to Export Data**.

Step 2 Use gs_restore to import the definitions of all database objects from the exported file to the destination database.

- Example 1: Use gs_restore to import the data and object definitions of the **gs_example** database from the **gs_example_dump.tar** file (in .tar format) by specifying the database IP address and an existing database (for example, **gs_example_restore**). In this example, only relative output information is included.

```
gs_restore -U root /tmp/data/gs_example_dump.tar -p 8000 -d gs_example_restore -e -h 192.*.*.139;
Password:
restore operation successful
total time: 1430  ms
```

- Example 2: Use gs_restore to import the data and object definitions of the **gs_example** database from the **gs_example_dump.tar** file (in .tar format) by specifying the database IP address and an existing database (for example, **gs_example_restore**). In addition, the database objects that already exist in the database to be imported are cleared (deleted). In this example, only relative output information is included.

```
gs_restore -U root /tmp/data/gs_example_dump.tar -p 8000 -d gs_example_restore -e -c -h 192.*.*.139;
Password:
restore operation successful
total time: 1621  ms
```

Table 4-11 Common parameters

Parameter	Description	Example
-U	Username for database connection.	-U jack
-W	User password for database connection. • This parameter is not required for database administrators if the trust policy is used for authentication. • If you connect to the database without specifying this parameter and you are not a database administrator, you will be prompted to enter the password.	-W ********
-d	Name of a database to which data will be imported.	-d backupdb
-p	TCP port or local Unix-domain socket file name extension on which the server is listening for connections.	-p 8000

Parameter	Description	Example
-e	If an error occurs when you send the SQL statement to the database, the system exits. Error messages are displayed after the import process is complete.	-e
-c	Before re-creating database objects, clear (delete) the database objects that exist in the database to be imported.	-c
-s	Only schema definitions are imported. Sequence values and data will not be imported.	-s

----End

Helpful Links

For more information, see:

- **gs_restore (distributed)**
- **gs_restore (centralized)**

4.9 Using gs_loader to Import Data

Scenarios

You can use gs_loader to import the files exported by using the **copy to** command. The gs_loader tool converts the syntax supported by control files into **\copy** syntax, then leverages the existing **\copy** function to import main data. At the same time, gs_loader logs the results of the **\copy** operations to a log file.

Precautions

- gs_loader does not support M-compatible databases.
- Before using gs_loader, ensure that the gs_loader version is consistent with the gsql version and database version.
- Currently, gs_loader is only available for centralized instances.
- When you use gs_loader to import data, if transcoding is not required, the size of a single row of data (including tuple metadata, same as mentioned below) is less than 1 GB to 1 B. If transcoding is required, the size of a single row of data is less than 256 MB to 1 B. Special handling has been applied to the following transcoding scenarios: the size for UTF-8 -> GB18030/GB18030_2022 transcoding is less than 512 MB to 1 B, and the size for UTF-8 -> GBK transcoding is less than 1 GB to 1 B.
- It is recommended that the size of a single file to be imported be less than or equal to 1 GB. gs_loader has no limit on the size of a single file to be

imported. However, importing a large file is time-consuming. Therefore, you are advised to split a large file, and start multiple gs_loader processes to write data to the table in append mode. (If there is a need to truncate data, it should be done by performing a separate TRUNCATE operation, rather than writing the TRUNCATE statement into the control file.) When the CPU resources are sufficient, this method can effectively improve the import speed.

Preparing for Data Migration

1. Prepare an ECS or a device that can access the GaussDB instance over EIP.
   - To connect to a GaussDB instance through an ECS, you must first create an ECS.

     For details on how to create and log in to an ECS, see **Purchasing an ECS** and **Logging In to an ECS** in *Elastic Cloud Server Getting Started*.
   - To connect to a GaussDB instance through a device that can access the GaussDB instance over EIP, you must:
     i. Bind an EIP to the GaussDB instance. For details, see **Binding an EIP**.
     ii. Ensure that the local device can access the EIP that has been bound to the GaussDB instance.
2. Install the gsql client on the prepared ECS or device that can access the GaussDB database, and connect it to the GaussDB database instance. For details, see **Using gsql to Connect to a Database**.

Procedure

Step 1 Create a control file and prepare a data file.

1. Create a control file, for example, **/tmp/data/loader.ctl**, and import data to the **loader_tbl** table. WHEN requires that the second character in each row be a comma (,).

```
LOAD DATA
truncate into table loader_tbl
WHEN (2:2) = ','
fields terminated by ','
trailing nullcols
(
    id integer external,
    name char(32),
    con ":id || '-' || :name",
    dt date
)
```

2. Create a GUC parameter file, for example, **/tmp/data/guc.txt**.

```
set a_format_copy_version='s1';
```

3. Create a data file, for example, **/tmp/data/data.csv**.

```
1,OK,,2007-07-8
2,OK,,2008-07-8
3,OK,,2009-07-8
4,OK,,2007-07-8
43,DISCARD,,2007-07-8
,,,
32,DISCARD,,2007-07-8
a,ERROR int,,2007-07-8
8,ERROR date,,2007-37-8
,,,,
 ,
8,ERROR fields,,2007-37-8
,,,
5,OK,,2021-07-30
```


Step 2 Create a user and grant privileges to the user.

```
CREATE USER load_user WITH PASSWORD '************';
GRANT ALL ON SCHEMA public TO load_user;
SELECT copy_summary_create() WHERE NOT EXISTS(SELECT * FROM pg_tables WHERE
schemaname='public' AND tablename='gs_copy_summary');
GRANT ALL PRIVILEGES ON  public.gs_copy_summary To load_user;
SELECT copy_error_log_create() WHERE NOT EXISTS(SELECT * FROM pg_tables WHERE
schemaname='public' AND tablename='pgxc_copy_error_log');
GRANT ALL PRIVILEGES ON  public.pgxc_copy_error_log To load_user;
\c - load_user
Password for user load_user:
```

Step 3 Create a target table to import data. For example, you can create the **loader_tbl** table in the **gs_example** database.

```
\c gs_example
Password for user load_user:
CREATE TABLE loader_tbl
(
    ID   NUMBER,
    NAME VARCHAR2(20),
    CON  VARCHAR2(20),
    DT   DATE
);
```

Step 4 Import the data.

Exit the current login connection.

```
\q
```

Before importing data, ensure that gs_loader has the required privileges. Ensure that the current directory has write privileges (gs_loader generates some temporary files when importing data. The files are automatically deleted after the import is completed).

```
gs_loader control=/tmp/data/loader.ctl data=/tmp/data/data.csv db=gs_example bad=/tmp/data/loader.bad
guc_param=/tmp/data/guc.txt errors=5 port=8000 passwd=************ user=load_user -h 192.*.*.139;
```

The output is shown as follows:

```
gs_loader: version 0.1

 5 Rows successfully loaded.

log file is:
 /tmp/data/loader.log
```

In the **/tmp/data/data.csv** file, the first four rows and the last row are imported. Rows 5 and 7 were ignored because the second character was not a comma (,). Rows 6, 10, and 13 were skipped because all fields were empty. Rows 8, 9, and 12 were not imported due to incorrect field values. The file defined as **bad=/tmp/data/loader.bad** can be used to view the erroneous lines, and the execution result is in **/tmp/data/loader.log**, which records the imported log information.

----End

Table 4-12 gs_loader parameters

Parameter	Description	Parameter Type/Value Range
help	Help information.	-

Parameter	Description	Parameter Type/Value Range
user	Database connection user (equivalent to **-U**).	Character string
-U	Database connection user (equivalent to **user**).	Character string
passwd	User password (equivalent to **-W**).	Character string
-W	User password (equivalent to **passwd**).	Character string
db	(Required) Database name. This parameter is equivalent to **-d**.	Character string
-d	(Required) Database name. This parameter is equivalent to **db**.	Character string
host	Host name of the running server, the path of the Unix domain socket, or the domain name. You can specify multiple host addresses by using character strings separated by commas (,). This parameter is equivalent to **-h**. If multiple host addresses are specified, the primary node is connected by default.	See the **gsql --host** parameter.
-h	Host name of the running server, the path of the Unix domain socket, or the domain name. You can specify multiple host addresses by using character strings separated by commas (,). This parameter is equivalent to **host**. If multiple host addresses are specified, the primary node is connected by default.	See the **gsql --host** parameter.
port	Port number of the database server. One or more port numbers can be configured. When one port number is configured, all IP addresses use the same port for connection. When multiple port numbers are configured, the sequence is the same as the IP address sequence, and the number of port numbers must be the same as the number of IP addresses. If they are different, an error is reported. This parameter is equivalent to **-p**.	See the **gsql --port** parameter.

Parameter	Description	Parameter Type/Value Range
-p	Port number of the database server. One or more port numbers can be configured. When one port number is configured, all IP addresses use the same port for connection. When multiple port numbers are configured, the sequence is the same as the IP address sequence, and the number of port numbers must be the same as the number of IP addresses. If they are different, an error is reported. This parameter is equivalent to **port**.	See the **gsql --port** parameter.
create	Specifies whether to create the **pgxc_copy_error_log** and **gs_copy_summary** tables. In the current version, the two tables are created by default. Therefore, this parameter is meaningless. This parameter is reserved only for compatibility.	The value can be **true** or **false**. The default value is **true**.
data	(Required) Data file. You can specify multiple data files or use wildcards (*) and question marks (?) to represent multiple data files.	Character string
control	(Required) Name of a control file.	Character string
log	Name of a log file.	Character string
bad	Name of the file that records the error rows and details. You can also specify a directory. If you do not specify a directory, the file is generated based on the data file name.	Character string
errors	Maximum number of error rows in a data file.	Integer. The default value is **0**.
limit	Maximum number of rows that can be imported.	Integer. By default, the value is infinite.

CAUTION

- All parameters are in lowercase and are compatible with the gsql login mode, including **-p** port number, **-h** host, **-d** database, **-U** username, and **-W** password.
- When the **rows** parameter is specified, the number of commit times cannot exceed 1,000. Otherwise, the performance will be affected. The number of commit times is approximately equal to the number of data rows in the data file divided by the value of **rows**. If the **rows** parameter is not specified, there is no default value for **rows**. In this case, the transaction is committed only once after all data is imported to the table.
- Frequent commit of a small amount of data affects the data import performance. You are advised to set the **rows** parameter properly to ensure that the amount of data committed each time is greater than 5 MB. For common servers with 16 vCPUs | 128 GB specifications, in the scenario where one primary node and two standby nodes are deployed and 13 GB of data is imported to a table with five columns, the rate of multiple commits is about 10 MB/s, which is basically same as that of a single commit (5 MB data is committed each time; network impacts are not considered).
- Currently, gs_loader are only compatible with NUL characters in data files. It does not support NUL characters in CTL files. If the .ctl file contains the nul character, unexpected problems may occur.

For details about other parameters and control file syntax, see **gs_loader** in *Tool Reference for Centralized Instances*.

5 Database Use

5.1 Overview of Database Usage

After creating a GaussDB instance, you can use DAS or SQL statements to perform basic database operations such as creating a database, creating a database user, creating tables, inserting data into tables, and deleting data from tables based on your requirements. For details, see **Table 5-1**.

Basic operations should comply with relevant design specifications. For details, see **Development and Design Proposal**.

You can also develop applications based on the JDBC, ODBC, libpq, Psycopg, ecpg, and Go drivers. For details, see **Application Development Guide**.

Table 5-1 Database operations

Database Operation		Description
Creating a database	• **Creating a Database Through DAS** • **Creating a Database Using SQL Statements**	These sections describe how to create a database using DAS and the CREATE DATABASE command.
Creating a database user	• **Creating a Database User Through DAS** • **Creating a Database Account Using SQL Statements**	These sections describe how to use the CREATE USER command to create a database user. Data is not shared between users.

Database Operation		Description
Creating a table	**Creating a Table Using SQL Statements**	A table is created in a database and can be stored in different databases. Tables under different schemas in a database can have the same name. This section describes how to use the CREATE TABLE command to create a table.
Inserting data to a table	**Inserting Data to Tables Using SQL Statements**	A new table contains no data. You need to insert data to the table before using it. This section describes how to insert one or more rows of data using the INSERT command.
Updating data in a table	**Updating Data in a Table Using SQL Statements**	You can update one row, all rows, or specified rows of data. You can also update data in a single column without affecting the data in the other columns. This section describes how to use the UPDATE command to update data in a table.
Viewing data in a table	• **Viewing Data in a Table Using SQL Statements** • **Opening a Table and Viewing Table Details Through DAS**	These sections describe how to use SQL statements to view data in tables and how to open tables and view table details on the DAS console.
Deleting data from a table	**Deleting Data from a Table Using SQL Statements**	Outdated data in existing tables may need to be deleted. This section describes how to use SQL statements to delete tables or table data.
Creating and managing views	• **Creating and Managing Views Using SQL Statements** • **Opening a View and Viewing Details of a View Through DAS**	These sections describe how to create, query, and delete views using SQL statements, and how to open a view and view details of a view through DAS.
Creating and managing tablespaces	**Creating and Managing Tablespaces Using SQL Statements**	This section describes how to create tablespaces, create objects in tablespaces, query tablespaces, query tablespace usage, change tablespace names, and delete tablespaces.

Database Operation		Description
Querying system catalogs	**Querying System Catalogs Using SQL Statements**	This section describes how to query system catalogs.
Creating and managing schemas	**Creating and Managing Schemas Using SQL Statements**	This section describes how to create a schema, use a schema, view the search path of a schema, control schema privileges, and delete a schema.
Creating and managing partitioned tables	**Creating and Managing Partitioned Tables Using SQL Statements**	This section describes how to create a partitioned table, insert data into a table, modify the row migration attributes of a partitioned table, delete a partition, add a partition, rename a partition, modify the tablespace of a partition, query a partition, and delete a partitioned table and tablespace.
Creating and managing indexes	**Creating and Managing Indexes Using SQL Statements**	This section describes how to create an index, modify the tablespace of an index partition, rename an index partition, query an index, and delete an index.
Creating and managing sequences	**Creating and Managing Sequences Using SQL Statements**	This section describes how to use a sequence to make a column a unique identifier.
Creating and managing scheduled jobs	**Creating and Managing Scheduled Jobs Using SQL Statements**	This section describes how to create a job, view job details, start a job, stop a job, modify job attributes, and delete a job.

5.2 Creating a GaussDB Database

Scenarios

After creating a GaussDB instance, you can create more databases as required. Database creation should comply with relevant design specifications. For details, see **Development and Design Proposal**.

You can create a database by using either of the following methods:

- **Creating a Database Through DAS**
- **Creating a Database Using SQL Statements**

Creating a Database Through DAS

Step 1 **Log in to the management console**.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, locate the DB instance you want to log in to and click **Log In** in the **Operation** column.

Alternatively, click the DB instance name on the **Instances** page. On the displayed **Basic Information** page, click **Log In** in the upper right corner of the page.

Step 5 On the **Custom Login** page, select the node to be logged in to. Enter the correct database username and password, and click **Test Connection**. After the connection test is successful, click **Log In**.

Step 6 Create a database.

- In the database list of the homepage, click **Create Database**. In the displayed dialog box, enter a database name and specify a character set, template, and other required parameters. Click **OK**.
- Alternatively, on the top menu bar, choose **SQL Operations** > **SQL Query**. Run the following command to create a database:

  create database *database_name*;

----End

Creating a Database Using SQL Statements

You can also use the **CREATE DATABASE** command to create a database.

5.3 Creating a GaussDB Database User

Scenarios

When you create a GaussDB instance, the **root** user is created at the same time by default. You can add other users as required.

You can create a database user by one of the following methods:

- **Creating a Database User Through DAS**
- **Creating a Database User Through SQL Statements**

Constraints

Only centralized instances of version V2.0-8.100.0 or later are supported.

Creating a Database User Through DAS

Step 1 **Log in to the management console**.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, locate the DB instance you want to log in to and click **Log In** in the **Operation** column.

Alternatively, click the DB instance name on the **Instances** page. On the displayed **Basic Information** page, click **Log In** in the upper right corner of the page.

Step 5 On the displayed login page, enter the username and password and click **Log In**.

Step 6 On the top menu bar, choose **SQL Operations** > **SQL Query**.

Step 7 Create a database user.

create user *username*;

----End

Creating a Database User Through SQL Statements

You can also use **CREATE USER** to create a database user.

6 Instance Management

6.1 Viewing GaussDB Instance Overview Data

Scenarios

You can view information about created DB instances, including instance statuses and alarm statistics.

NOTE

To apply for the permissions needed, submit an application by choosing **Service Tickets** > **Create Service Ticket** in the upper right corner of the management console.

Procedure

Step 1 **Log in to the management console**.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 In the navigation pane, choose **Overview**.

NOTE

If there are no DB instances, no DB instance information will be displayed on the **Overview** page. Instead, you can only create a DB instance on this page.

If you are a new user, create a DB instance as prompted. If there are existing instances, you can view instance information on this page.

- Instances by Status

  You can also view instance status statistics on the **Instances** page.

  Figure 6-1 Instances by Status

Table 6-1 Parameters for instance status statistics

Item	Description	Operation	Handling Suggestion
Total instances	Number of instances installed in the current GaussDB system	Click **Total instances** to go to the instance list and view all instances.	-
Abnormal	Number of instances that are in an abnormal state on the **Instances** page If the storage space of an instance is full, the instance is also considered abnormal.	Click **Abnormal** to go to the instance list and view abnormal instances.	Contact customer service.
Creation failed	Number of instances that are in the creation failed state on the **Instances** page	Click **Creation failed** to go to the instance list and view instances that fail to be created.	Contact customer service.
Frozen	Number of instances that are in the frozen state on the **Instances** page	Click **Frozen** to go to the instance list and view frozen instances.	For details, see **Resource Freezing, Unfreezing, Release, Deletion, and Unsubscription**.
Creating	Number of instances that are in the creating state on the **Instances** page	Click **Creating** to go to the instance list and view instances that are being created.	-
Running	Number of instances that are in the available state on the **Instances** page	Click **Running** to go to the instance list and view running instances.	-
Stopped	Number of instances that are in the stopped state on the **Instances** page	Click **Stopped** to go to the instance list and view stopped instances.	-

- Alarm Statistics

You can view alarm statistics in the last 1 hour, last 3 hours, last 12 hours, last 1 day, last 3 days, or last week. Instance alarm details are displayed from multiple dimensions. For details, see **Table 6-2** and **Table 6-3**.

Table 6-2 Description of the Top 5 Instances by Total Number of Alarms area

Item	Description
Alarm Severity	Number of all uncleared alarms of different severities in the specified period The alarm severity can be critical, major, minor, or warning.
Top 5 Instances by Total Number of Alarms	Number of alarms of the top 5 instances with the largest number of uncleared alarms in a specified period

Table 6-3 Alarm list description

Item	Description
Alarm Name/ID	Name and ID of an alarm
Instance Name/ID	Name and ID of the instance for which an alarm is generated
Alarm Severity	Severity of an alarm **NOTICE** Critical alarms are generated for faults that affect system-provided services. You need to take countermeasures immediately. For example, if a device or resource is unavailable, fix it immediately.
Alarm Type	Type of an alarm, which can be **Metric** or **Event**
First Reported	Time when an alarm was reported for the first time
Last Reported	Lasted time when an alarm was reported

----End

6.2 Configuring Security Group Rules for a GaussDB Instance

Scenarios

A security group is a collection of access control rules for ECSs and GaussDB instances that are within the same VPC, have the same security requirements, and are mutually trusted.

If you have applied for the whitelist of not specifying a security group when creating an instance, skip this section. The security group information will not be displayed in the DB instance information area.

To ensure database security and reliability, you need to configure security group rules to allow specific IP addresses and ports to access the GaussDB instances.

- When you attempt to connect to a GaussDB instance through a private network, check whether the ECS and GaussDB instance are in the same security group.
  - If they are in the same security group, they can communicate with each other by default. No security group rule needs to be configured.
  - If they are in different security groups, you need to configure security group rules for the ECS and GaussDB instance, respectively.
    - GaussDB instance: Configure an **inbound rule** for the security group with which the GaussDB instance is associated.
    - ECS: The default security group rule allows all outgoing data packets. In this case, you do not need to configure a security group rule for the ECS. If **not all outbound traffic is allowed** in the security group, you need to configure an **outbound** rule for the ECS to allow all outbound packets.
- When you attempt to connect to a GaussDB instance using an EIP, you need to configure an **inbound rule** for the security group associated with the instance.

This section describes how to configure an inbound rule for a GaussDB instance.

For details about the requirements of security group rules, see **Adding a Security Group Rule** in the *Virtual Private Cloud User Guide*.

Precautions

The default security group rule allows all outbound data packets. This means that ECSs and GaussDB instances associated with the same security group can access each other by default. After a security group is created, you can add security group rules to control the access from and to the GaussDB instance.

- By default, you can create up to 500 security group rules.
- Ensure that each security group has no more than 50 rules.
- To access a GaussDB instance from resources outside the security group, configure an **inbound rule** for the security group associated with the instance.

- All Kunpeng ECS flavors do not support inconsecutive ports.

  If you use inconsecutive port numbers in a security group rule of a Kunpeng ECS, this rule and rules configured after this one do not take effect.

  For example, if you configure security group rule A with inconsecutive ports **22, 24** and then configure security group rule B with port 9096, both rule A and rule B do not take effect.
- Outbound rules typically do not apply to DB instances. The rules are used only when a DB instance acts as a client.
- If a DB instance resides in a VPC but is not publicly accessible, you can also use a VPN connection to connect to it.
- If you need to change the security group when creating a distributed instance, ensure that the TCP ports in the inbound rule include the following: 40000-60480, 20050, 5000-5001, 2379-2380, 6000, 6500, and *<database port>-(<database port> + 100)*. (For example, if the database port is 8000, the TCP ports for the security group must include 8000-8100.)
- If you need to change the security group when creating a centralized instance, ensure that the TCP ports in the inbound rule include the following: 20050, 5000-5001, 2379-2380, 6000, 6500, and *<database port>-(<database port> + 100)*. (For example, if the database port is 8000, the TCP ports for the security group must include 8000-8100.)

NOTE

To ensure data and instance security, use permissions properly. You are advised to use the principle of least privilege for database access. Set the accessible IP address to the remote server's address or the remote server's smallest subnet address to control the access scope of the remote server.

The default value of **Source** is **0.0.0.0/0**, indicating that all IP addresses can access the GaussDB instance as long as they are associated with the same security group as the instance.

For details about the requirements of security group rules, see **Adding a Security Group Rule** in the *Virtual Private Cloud User Guide*.

Procedure

Step 1 **Log in to the management console**.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 Configure security group rules.

In the **Network Information** area, click the security group name.

Figure 6-2 Network information

Step 6 On the **Inbound Rules** tab, click **Add Rule**. In the displayed dialog box, configure the required parameters and click **OK**.

You can click **+** to add more inbound rules.

Table 6-4 Inbound rule parameter description

Parameter	Description	Example Value
Protocol & Port	Network protocol. Currently, the value can be **All**, **TCP**, **UDP**, **ICMP**, **GRE**, or others.	TCP (Custom ports)
	Port: port or port range over which the traffic can reach your ECS. The value ranges from 1 to 65535.	When connecting to your instance through a private network, enter the port of the ECS used to connect to your instance.
Type	IP address type. • IPv4 • IPv6	IPv4
Source	Source of the security group rule. The value can be a security group or an IP address. Examples: • xxx.xxx.xxx.xxx/32 (IPv4 address) • xxx.xxx.xxx.0/24 (subnet) • 0.0.0.0/0 (any IP address)	0.0.0.0/0
Description	Provides supplementary information about the security group rule. This parameter is optional. The description can contain up to 255 characters and cannot contain angle brackets (<) or (>).	-

----End

6.3 Binding and Unbinding an EIP for a GaussDB Instance

Scenarios

You can bind an EIP to a GaussDB instance for public access and can unbind the EIP from an instance as required.

NOTICE

To ensure that the database can be accessed, the security group used by the database must allow access to the database port. For example, if the database port is **1611**, ensure that the security group allows access to the port **1611**.

Precautions

- If a DB instance has already been bound with an EIP, you must unbind the EIP from the instance first before binding a new EIP to it.
- An EIP can be bound to only one node IP address of a DB instance.
- Security risks may arise after an EIP is bound to an instance. To enhance database security, set strict inbound and outbound rules for the security group. For details about how to configure a security group, see **Configuring Security Group Rules**.

Binding an EIP

Step 1 **Log in to the management console**.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the **Node List** area, click **Bind the EIP** in the **Operation** column.

For a distributed instance, switch to **CN** in the filter area first and then bind an EIP.

Figure 6-3 Binding an EIP (distributed instance)

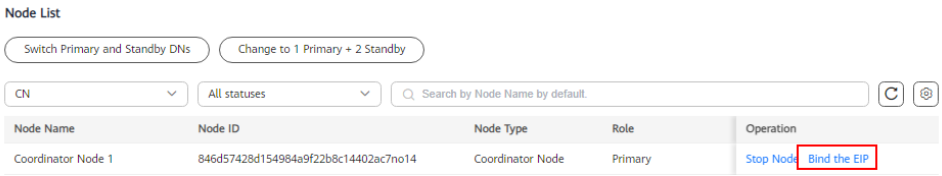
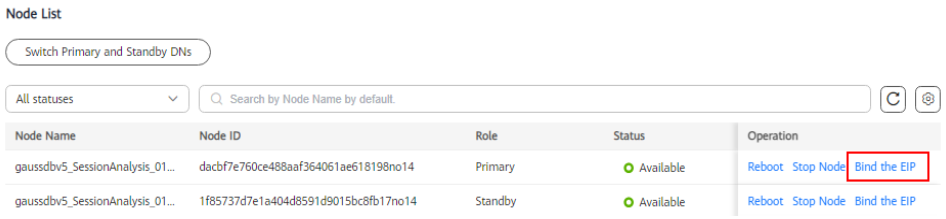


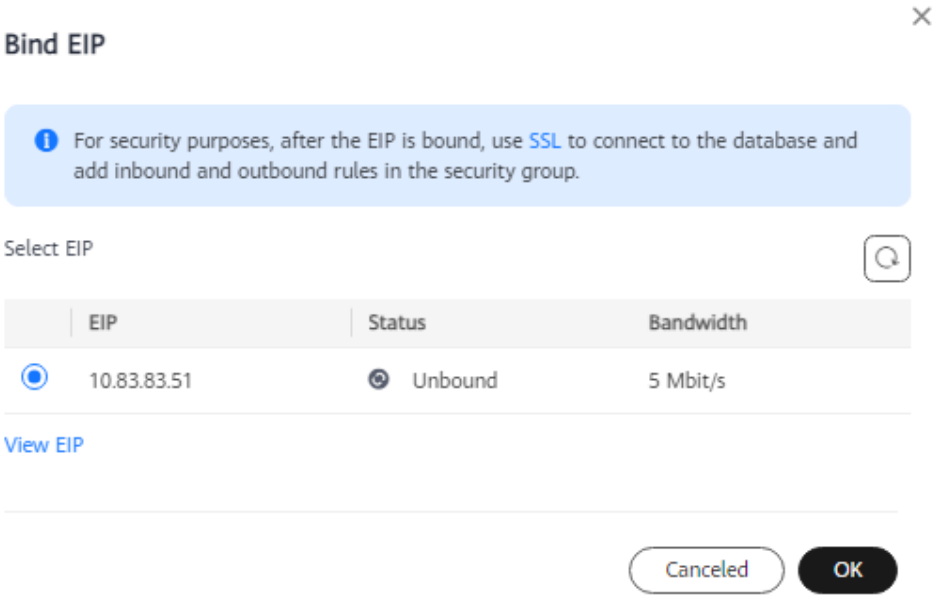
Figure 6-4 Binding an EIP (centralized instance)



Step 6 In the displayed dialog box, all available unbound EIPs are listed. Select the required EIP and click **OK**.

If no available EIPs are displayed, click **View EIP** and obtain an EIP.

Figure 6-5 Binding an EIP



Step 7 In the **Node List** area, check the operation result in the **EIP** column.

Figure 6-6 Checking the binding result

To unbind the EIP from the instance, see **Unbinding an EIP**.

NOTE

After the EIP is bound, you can click next to the private IP address to view its details.

----End

Unbinding an EIP

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the instance that has been bound with an EIP.

Step 5 In the **Node List** area, click **Unbind the EIP** in the **Operation** column.

- For a distributed instance, switch to **CN** in the filter area first and then click **Unbind the EIP** in the **Operation** column.
- For a centralized instance, choose **More** > **Unbind the EIP** in the **Operation** column.

Figure 6-7 Unbinding an EIP (distributed instance)

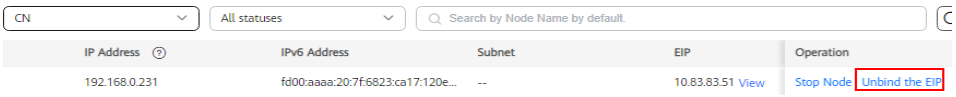
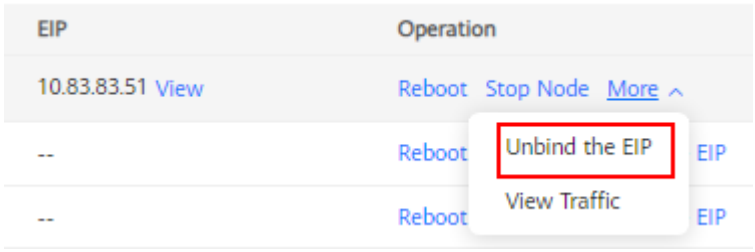


Figure 6-8 Unbinding an EIP (centralized instance)



Step 6 In the displayed dialog box, click **OK** to unbind the EIP.

Step 7 If you have enabled operation protection, click **Send Code** in the displayed **Identity Verification** dialog box and enter the obtained verification code. Then, click **OK**.

Two-factor authentication improves the security of your account. For details about how to enable operation protection, see ***Identity and Access Management User Guide***.

Step 8 In the **Node List** area, check the operation result in the **EIP** column.

To bind an EIP to the instance again, see **Binding an EIP**.

----End

6.4 Modifying the Recycle Bin Policy for a GaussDB Instance

You can recycle deleted GaussDB instances within the configured retention period and **rebuild instances** from the recycle bin as needed.

The recycle bin is enabled by default and cannot be disabled. The deleted instances can be retained for 7 days by default.

Procedure

NOTICE

- You can modify the retention period, and the changes only apply to the DB instances deleted after the changes, so exercise caution when performing this operation.
- DB instances to be rebuilt in the recycle bin will not incur charges.

Step 1 **Log in to the management console**.

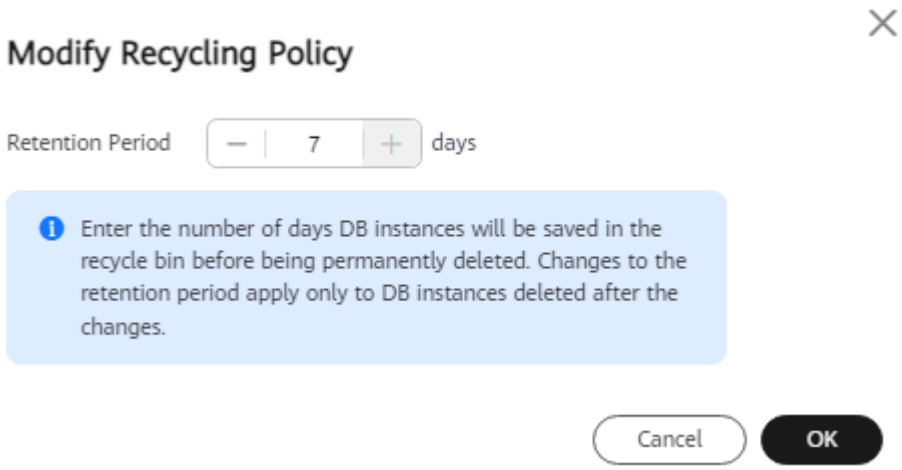
Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 In the navigation pane on the left, choose **Recycle Bin**.

Step 5 Click **Modify Recycling Policy**. In the displayed dialog box, set the retention period for the deleted DB instances from 1 day to 7 days.

Figure 6-9 Modifying the recycling policy



Step 6 Click **OK**.

----End

6.5 Exporting Information About All GaussDB Instances

Scenarios

You can export information about all instances in the list for review and analysis.

Constraints

A tenant can export a maximum of 3,000 instances at a time. The time required for the export depends on the number of instances.

Exporting All Instance Information

Step 1 **Log in to the management console**.

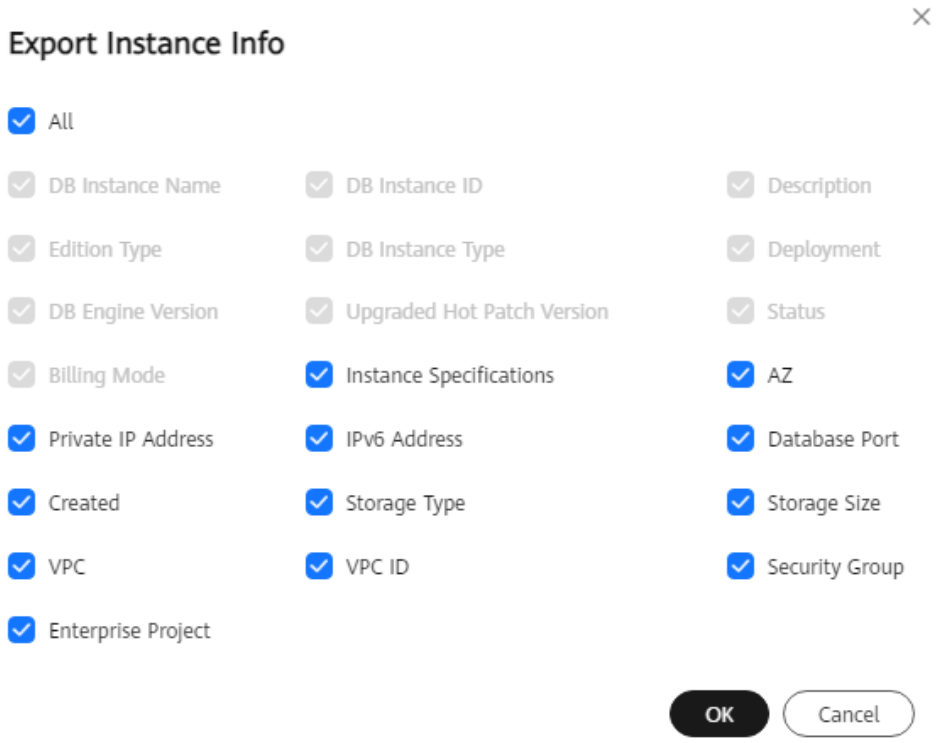
Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click **Export Instance Info**. By default, all instance information is exported.

Step 5 In the displayed dialog box, select the items to be exported and click **OK**.

Figure 6-10 Exporting information about all instances



Step 6 After the export task is complete, a CSV file is generated on the local PC.

----End

6.6 Unsubscribing a Yearly/Monthly GaussDB Instance

Scenarios

To delete a DB instance billed on a yearly/monthly basis, you need to unsubscribe the order. Currently, DB instances cannot be unsubscribed in batches. You can unsubscribe only one instance at a time. For details, see **Unsubscribing a Single Instance**. For unsubscription fees, see **Unsubscription Rules**.

For pay-per-use DB instances, you need to delete them on the **Instances** page. For details, see **Deleting a Pay-per-Use GaussDB Instance**.

Unsubscribing a Single Instance (Method 1)

Unsubscribe a yearly/monthly DB instance on the **Instances** page.

Step 1 **Log in to the management console**.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, locate the instance and choose **More** > **Unsubscribe** in the **Operation** column.

Step 5 On the displayed page, confirm the order to be unsubscribed and select a reason. Then, click **Confirm**.

For unsubscription details, see **Unsubscription Rules**.

Step 6 In the displayed dialog box, click **Yes**.

NOTICE

- After an unsubscription request is submitted, resources and data will be deleted and cannot be retrieved.
- If you want to retain data, complete a manual backup before submitting the unsubscription request.

Step 7 View the unsubscription result. After the DB instance order is successfully unsubscribed, the DB instance is no longer displayed in the instance list on the **Instances** page.

----End

Unsubscribing a Single Instance (Method 2)

Unsubscribe a yearly/monthly instance on the **Billing Center** page.

Step 1 **Log in to the management console**.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

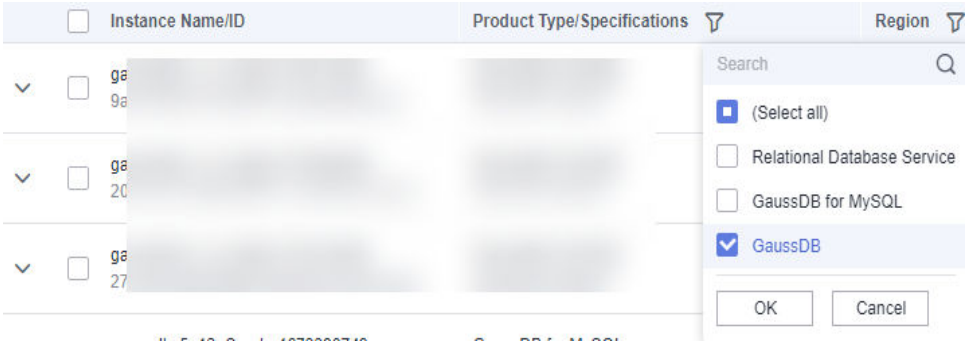
Step 4 In the upper right corner, click **Billing**.

Step 5 In the navigation pane, choose **Orders** > **Unsubscriptions**.

Step 6 On the displayed page, select the order to be unsubscribed and click **Unsubscribe from Resource** in the **Operation** column.

- You can select **GaussDB** in the **Product Type/Specifications** filter box to filter all GaussDB orders.

  Figure 6-11 Filtering all GaussDB orders

- Alternatively, you can search for orders by name, order No, or ID in the search box above the order list.

CAUTION

A maximum of 100 resources can be unsubscribed at a time.

Step 7 On the displayed page, confirm the order to be unsubscribed and select a reason. Then, click **Confirm**.

For details, see **Unsubscription Rules**.

Step 8 In the displayed dialog box, click **Yes**.

NOTICE

1. Unsubscribed DB instances will be moved to the recycle bin, but will be permanently deleted after a length of time determined by the recycling policy. Automated backups are deleted, but manual backups are retained and still billed. To delete the manual backups, go to the **Backups** page on the console.
2. If you want to retain data, complete a manual backup before submitting the unsubscription request.

Step 9 View the unsubscription result. After the instance order is successfully unsubscribed, the instance will be deleted.

----End

6.7 Stopping a GaussDB Instance

Scenarios

You can manually stop a GaussDB instance. This operation stops the component processes on all nodes of the instance, but does not stop the VM or physical server.

Constraints

- The following operations cannot be performed when an instance is being stopped and after it is stopped: scaling up storage, changing specifications, backing up data, resetting passwords, rebooting the instance, and deleting the instance.
- After an instance is stopped, its component information cannot be queried.
- After an instance is stopped, it cannot provide services. Exercise caution when performing this operation.
- Instance parameters cannot be modified after an instance is stopped.
- Stopping an instance will stop all component processes in it and workloads will be interrupted. Exercise caution when performing this operation.
- After an instance is stopped, all resources will still be billed. To stop incurring charges, delete any resources you no longer need. For details, see **Billing Termination**.

Procedure

Step 1 **Log in to the management console**.

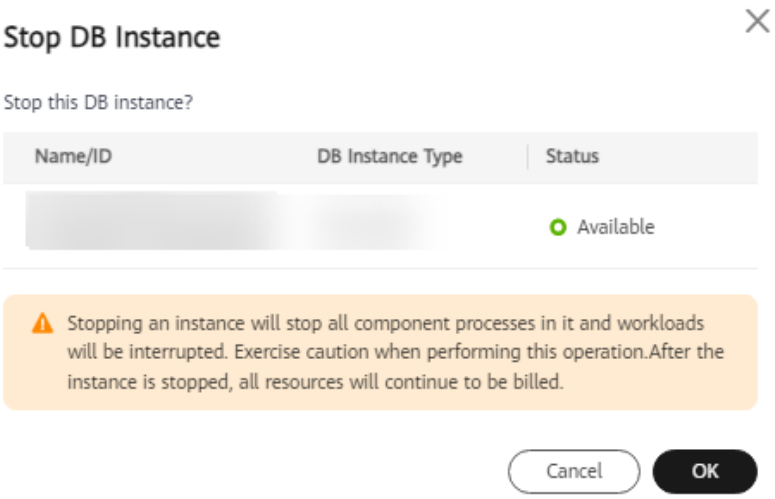
Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click **More** in the **Operation** column of the target instance and choose **Stop**.

Step 5 In the displayed dialog box, click **OK**.

Figure 6-12 Stopping an instance



Step 6 Refresh the instance list and view the status of the instance. If its status is **Stopped**, it has been stopped successfully.

----End

6.8 Starting a GaussDB Instance

Scenarios

GaussDB allows you to manually start a stopped DB instance.

Precautions

- Only instances in the **Stopped** state can be started.
- During instance startup, the following operations cannot be performed: scaling up storage, changing specifications, backing up data, resetting passwords, rebooting the instance, and deleting the instance.
- If the number of stopped nodes in a DN shard exceeds half of the replicas of the shard, the instance may be abnormal. You are advised to stop all nodes and then start the instance.

Procedure

Step 1 Log in to the management console.

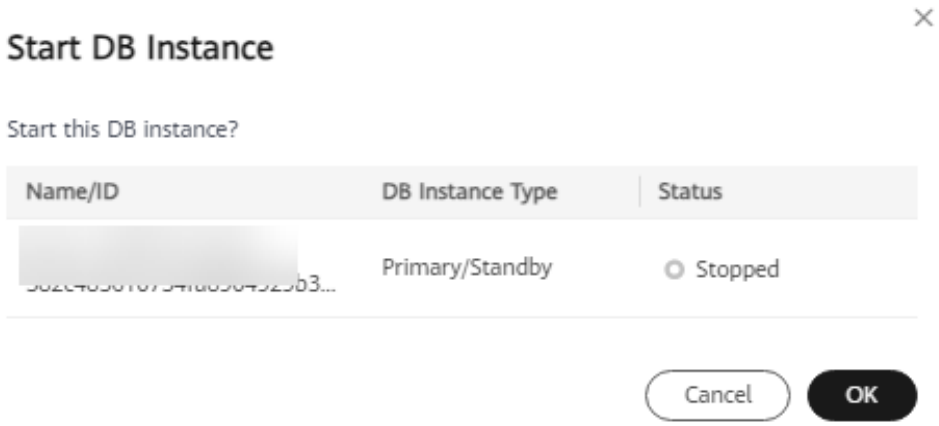
Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click **More** in the **Operation** column of the target instance and choose **Start**.

Step 5 In the displayed dialog box, click **OK**.

Figure 6-13 Starting an instance



Step 6 Refresh the instance list and view the status of the instance. If its status is **Available**, it has been started successfully.

----End

6.9 Rebooting a GaussDB Instance

Scenarios

You can reboot a DB instance for the modifications to take effect.

NOTICE

- You can reboot a DB instance only when its status is **Available**. Your database may be unavailable in some cases, for example, when some modifications are being made.
- Rebooting a DB instance will cause service interruptions. During this period, the DB instance status is **Rebooting**.
- An instance is not available when it is being rebooted. After the reboot completes, the cached memory will be automatically cleared. You are advised to reboot the instance during off-peak hours.
- To quickly reboot a DB instance, perform fewer operations on the DB instance.
- If there are a large number of slow SQL statements or sessions, or if the thread pool is full, the reboot process may take a longer time than usual.

Procedure

Step 1 **Log in to the management console**.

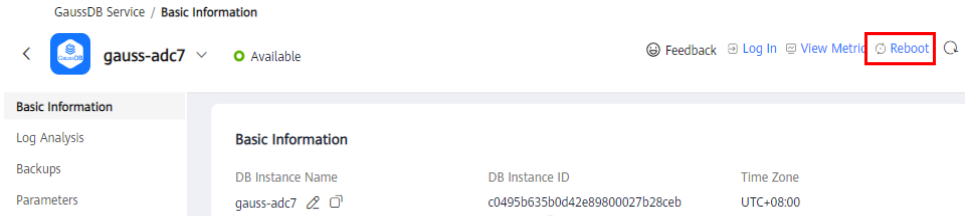
Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, locate the instance you want to reboot and choose **More** > **Reboot** in the **Operation** column.

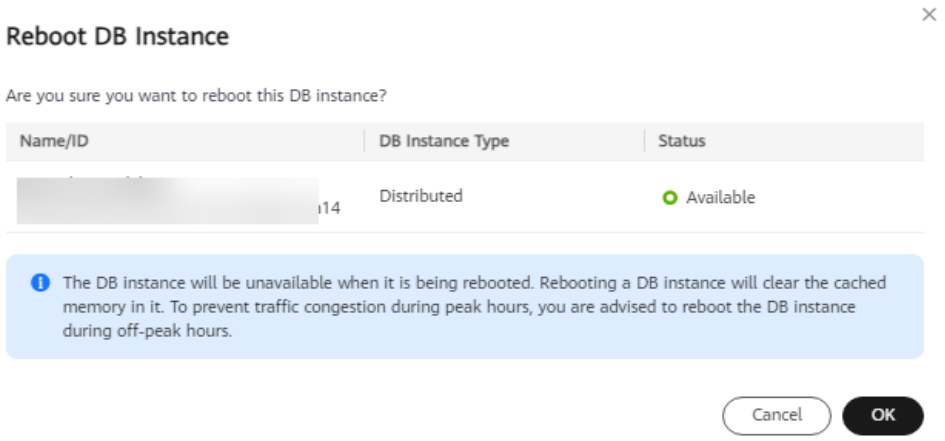
Alternatively, click the instance name to go to the **Basic Information** page. Click **Reboot** in the upper right corner of the page.

Figure 6-14 Basic instance information



Step 5 In the displayed dialog box, click **OK**.

Figure 6-15 Rebooting an instance



The instance status becomes **Rebooting**.

Step 6 If you have enabled operation protection, click **Start Verification** in the displayed dialog box. On the displayed page, click **Send Code**, enter the obtained verification code, and click **Verify** to close the page.

Two-factor authentication improves the security of your account. For details about how to view and enable high-risk operation protection, see ***Identity and Access Management User Guide***.

Step 7 Refresh the DB instance list and view the status of the DB instance. If its status is **Available**, it has been rebooted.

----End

6.10 Deleting a Pay-per-Use GaussDB Instance

Scenarios

- You need to delete unnecessary DB instances.
- You need to delete the DB instance that fails to be created.

NOTICE

- Deleted DB instances cannot be recovered. Exercise caution when performing this operation. To retain data, back up the data before deleting a DB instance.
- DB instances cannot be deleted when operations are being performed on them.
- You can restore a DB instance that was deleted up to 7 days ago from the recycle bin. For details, see **Modifying the Recycle Bin Policy for a GaussDB Instance**.
- If you delete a stopped instance, it cannot be rebuilt from the recycle bin. Exercise caution when performing this operation. If you might want to rebuild it later, start the instance first and make sure it is running properly before deleting it.
- DB instances to be rebuilt in the recycle bin will not incur charges.
- When **pay-per-use** instances are deleted, manual backups are retained.
- Instances cannot be deleted while an operation is being performed on them. They can only be deleted once the operations are complete.

Procedure

Step 1 **Log in to the management console**.

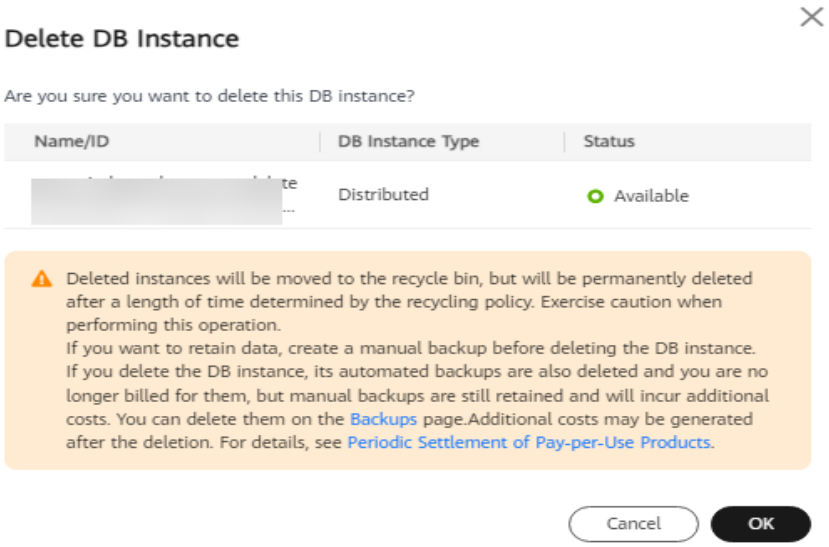
Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, locate the instance you want to delete and click **More** > **Delete** in the **Operation** column.

Step 5 In the displayed dialog box, click **Yes**. Refresh the **Instances** page later to check that the deletion is successful.

Figure 6-16 Deleting an instance



Step 6 If you have enabled the operation protection function, click **Start Verification** in the **Delete DB Instance** dialog box. On the displayed page, click **Send Code**, enter the obtained verification code, and click **Verify** to close the page.

Two-factor authentication improves the security of your account. For details about how to view and enable high-risk operation protection, see ***Identity and Access Management User Guide***.

----End

6.11 Rebuilding a GaussDB Instance

You can rebuild instances in the recycle bin within the retention period.

Procedure

Step 1 **Log in to the management console**.

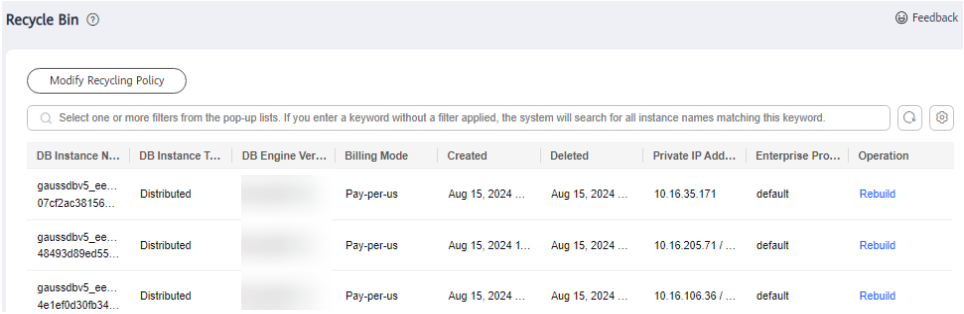
Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 In the navigation pane on the left, choose **Recycle Bin**.

Step 5 Locate the instance to be rebuilt and click **Rebuild** in the **Operation** column.

Figure 6-17 Rebuilding an instance



Step 6 On the displayed page, configure required parameters and submit the task.

Rebuilding an instance indicates that you restore data to a new instance using backup files. To configure parameters of the new instance, see **Buying an Instance**.

----End

6.12 Stopping a GaussDB Node

Scenarios

You can stop a node for your GaussDB instance. This operation will stop all database component processes on the node, but will not directly stop the VM or physical server where the node is located.

Constraints

- Stopping a node may expose the instance to higher risk of downtime. Exercise caution when performing this operation.
- The following operations cannot be performed when a node is being stopped: changing specifications, backing up data, resetting passwords, rebooting an instance, and deleting an instance. When you perform these operations, nodes cannot be stopped.
- HA monitoring will be disabled for a node before it is stopped. HA monitoring will be enabled when you start the node. You can also manually enable HA monitoring for a node. After HA monitoring is disabled, the node status will not be monitored.
- If a stopped node contains at least half of the replicas of the corresponding DN shard, the instance may be abnormal and a single node may fail to be started.
- After a node is stopped, you can still log in to the node using CLI commands, but all database-related operations cannot be performed.
- For a distributed instance, at least one CN in the instance must be available, or the entire instance will be unavailable.
- Instance parameters cannot be modified if there are stopped nodes in an instance.

- If a node is stopped for too long, an alarm is triggered. You can **start the node** to rectify the fault and clear the alarm.
- After the node is stopped, all resources will continue to be billed.

Procedure

Step 1 **Log in to the management console**.

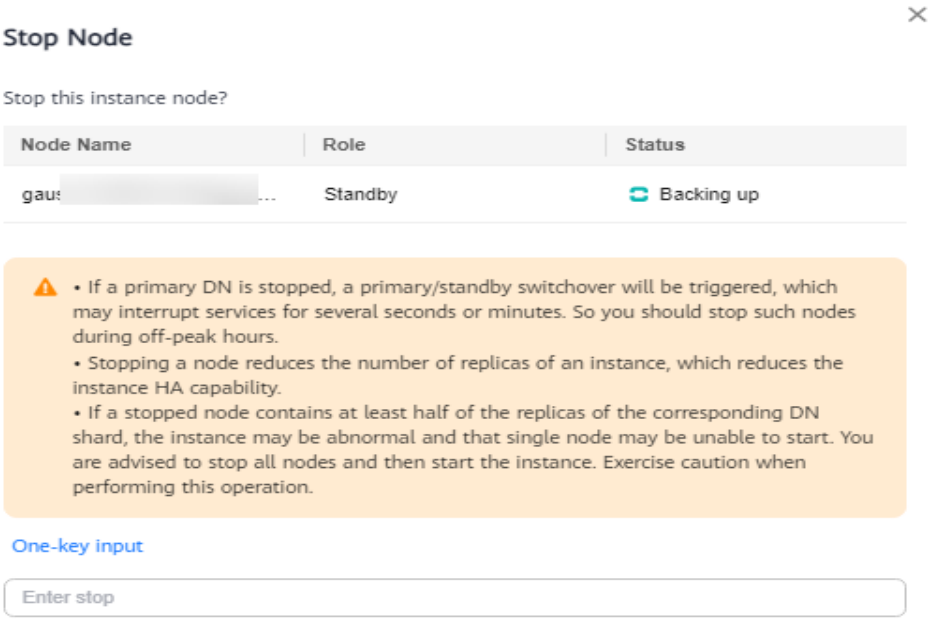
Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the **Node List** area, click **Stop Node** in the **Operation** column.

Figure 6-18 Stopping a node



Step 6 In the **Stop Node** dialog box, enter **stop**, and click **OK**.

Step 7 Refresh the instance and view the status of the node. If its status is **Stopped**, it has been stopped successfully.

----End

6.13 Starting a GaussDB Node

Scenarios

GaussDB allows you to manually start a stopped node.

Precautions

- Only nodes in the **Stopped** state can be started.
- The following operations cannot be performed when a node is being started: scaling up storage, changing specifications, backing up data, resetting passwords, rebooting an instance, and deleting an instance. When you perform these operations, nodes cannot be started.
- After a node is started, the distribution of primary and standby nodes for the instance may be unbalanced. Contact O&M personnel to determine whether to balance the primary and standby statuses for the instance.
- If the number of stopped nodes in a DN shard exceeds half of the replicas of the shard, a single node may fail to be started.

Procedure

Step 1 Log in to the management console.

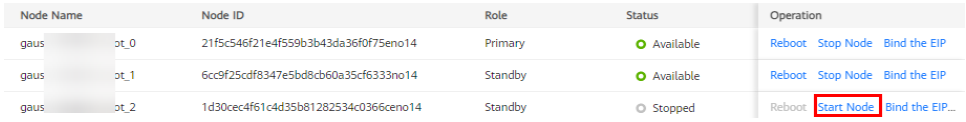
Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the **Node List** area, click **Start Node** in the **Operation** column.

Figure 6-19 Starting a node



Step 6 In the displayed dialog box, enter **start**, and click **OK**.

Step 7 Refresh the instance and view the status of the node. If its status is **Available**, it has been started successfully.

----End

6.14 Rebooting a GaussDB Node

Scenarios

If the status of a GaussDB instance node is abnormal, you can reboot the node to restore the node status. You can also reboot a node when it is in the **Available** state. A node is not available when it is being rebooted.

Precautions

- You can reboot a node when the DB instance is in the following state or performing the following operations:

  - Backup and restoration failed
  - Changing the billing mode from pay-per-use to yearly/monthly
  - DR in progress for the primary instance in a streaming DR task
  - Caching logs for the primary instance in a streaming DR task
  - DR simulation in progress for the DR instance in a streaming DR task
  - DR in progress for the DR instance in a streaming DR task
  - DR instance promoted to primary in a streaming DR task
- Rebooting nodes will clear the cached memory in them. To prevent traffic congestion during peak hours, you are advised to reboot nodes during off-peak hours.
- Only nodes of centralized instances can be rebooted.
- A primary/standby switchover will be triggered if a primary node is rebooted.

Procedure

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 Click the target instance name to go to the **Basic Information** page.

Step 5 In the **Node List** area, click **Reboot** in the **Operation** column of a node. Confirm information about the node to be restarted, enter **reboot**, and click **OK**.

Figure 6-20 Rebooting a node

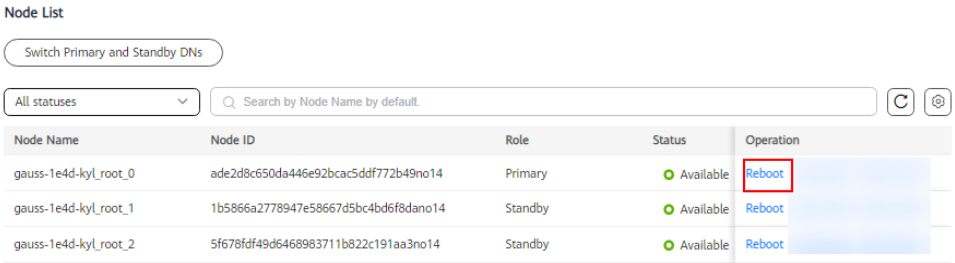
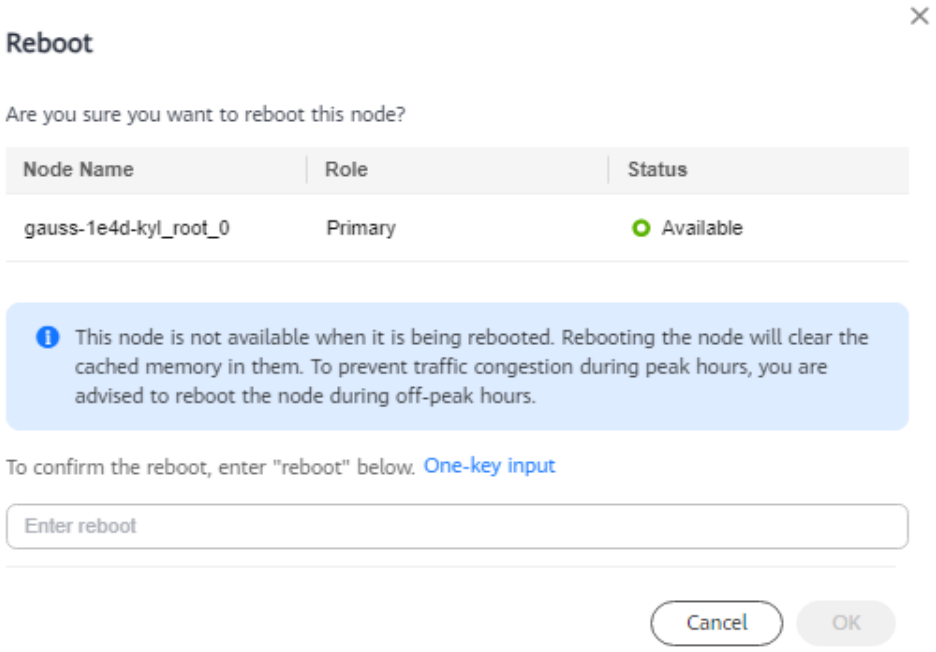


Figure 6-21 Confirming the reboot



The node status becomes **Rebooting node**.

Step 6 Refresh the instance basic information and check the reboot result. If the node status is **Available**, the reboot is successful.

----End

7 Instance Modifications

7.1 Changing the Name of a GaussDB Instance

Scenarios

You can change the name of an instance.

Constraints

You cannot perform the following operations when the instance name is being changed:

- Binding an EIP
- Deleting the instance
- Creating a backup for the instance

Precautions

- The new name of an instance can be the same as an existing instance name.
- Changing the name of a DB instance does not disassociate the associated tags from the instance.
- If a DB instance is renamed, backups of the DB instance are still retained.

Procedure

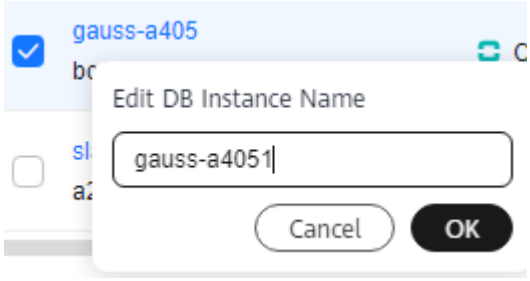
Step 1 **Log in to the management console**.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, locate the instance whose name you want to edit and click next to the instance name. Then, edit the name and click **OK**.

Figure 7-1 Changing the name of an instance



Alternatively, click the instance name to go to the **Basic Information** page. In the **Basic Information** area, click next to the **DB Instance Name** field to edit the instance name.

The name must start with a letter and consist of 4 to 64 characters. It can contain only uppercase letters, lowercase letters, digits, hyphens (-), and underscores (_).

- To submit the change, click .
- To cancel the change, click .

Step 5 View the new instance name.

----End

7.2 Changing the Database Port of a GaussDB Instance

Scenarios

You can change the database port of your GaussDB instance.

Precautions

- The database port can be changed only for instances of version V2.0-2.0 or later.
- Changing the port of an instance will reboot all nodes of the instance, during which services will be intermittently interrupted.

Procedure

Step 1 **Log in to the management console**.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance.

Step 5 In the **Network Information** area on the **Basic Information** page, click in the **Database Port** field.

Figure 7-2 Changing the database port

- To submit the change, click ✓.
  - In the displayed dialog box, click **Yes** to submit the change.
  - In the displayed dialog box, click **No** to cancel the change.
- To cancel the change, click ✗.

NOTE

- The GaussDB port ranges from 1024 to 39989, but the following ports that are reserved for system use cannot be used: 2378 to 2380, 2400, 4999 to 5001, 5100, 5500, 5999 to 6001, 6009 to 6010, 6500, 8015, 8097, 8098, 8181, 9090, 9100, 9180, 9187, 9200, 12016, 12017, 20049, 20050, 21731, 21732, 32122 to 32126, and 39001.
- The port cannot be a number in the range [*Database port*, *Database port* + 10].

Step 6 View the result of the change on the **Basic Information** page.

----End

7.3 Changing the M Compatibility Port

Scenarios

GaussDB allows you to use the **templatem** template to create an M-compatible database. You can enable, disable, or change the M compatibility port for a centralized instance on the console for better compatibility.

Constraints

- M-compatible databases can be created only for centralized instances of version V2.0-8.100 or later and distributed instances of version V2.0-8.200 or later. The distributed instances must be newly created, instead of those upgraded from an earlier version.
- Only one M-compatible database can be created for an instance, and database- and table-level backup and restoration are not supported.
- The M compatibility port cannot be changed for GaussDB instances that contain a database named **templatem**.
- This operation is not allowed for DR instances. To enable the M compatibility port for a DR instance, delete its DR relationship first.
- Enabling the M compatibility port will reboot the DB instance, during which services are interrupted. Perform this operation during off-peak hours.
- The port must have been enabled in the inbound rule of the security group.

Procedure

Step 1 Log in to the management console.

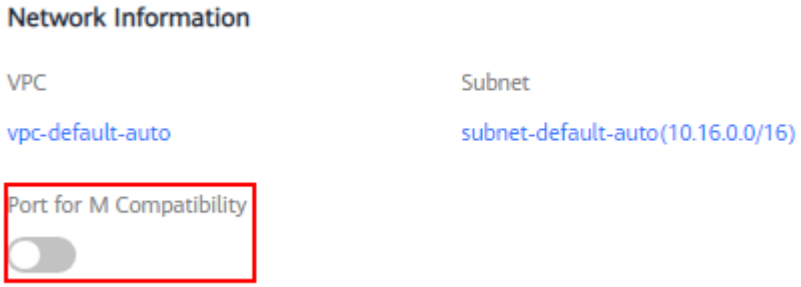
Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the **Network Information** area, click in the **Port for M Compatibility** field.

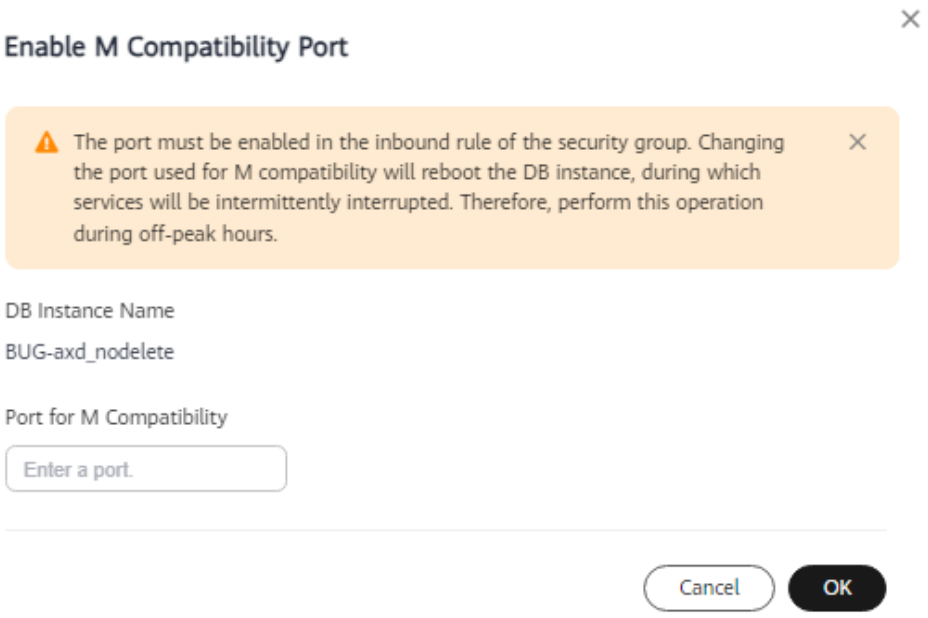
Figure 7-3 Setting the M compatibility port



Step 6 In the displayed dialog box, set the M compatibility port, and click **OK**.

- The port ranges from 1024 to 39989, but the following ports that are reserved for system use cannot be used: 2378 to 2380, 2400, 4999 to 5001, 5100, 5500, 5999 to 6001, 6009, 6010, 6500, 8015, 8097, 8098, 8181, 9090, 9100, 9180, 9187, 9200, 12016, 12017, 20049, 20050, 21731, 21732, 32122 to 32126, and 39001
- The port cannot be a number in the range [*Database port*, *Database port* + 10].

Figure 7-4 Enabling M compatibility port



----End

7.4 Changing the CPU and Memory Specifications of a GaussDB Instance

Scenarios

You can change the instance specifications (CPU and memory) as required. Once the change is complete, the status of an instance changes from **Changing instance specifications** to **Available**.

Precautions

- You can scale up or down the CPU and memory specifications of your GaussDB instances as needed.
- Before you change the instance specifications, ensure that the instance is available. If the instance or node is abnormal, or the storage space is full, you cannot perform this operation.
- During the specification change for an HA (1 primary + 2 standby) instance, a primary/standby failover is triggered. During the failover, services are interrupted for about 1 minute.
- For a single-replica instance, changing instance specifications will reboot the instance and interrupt services for 5 to 10 minutes.
- After you change instance specifications, the DB instances will be rebooted and services will be interrupted. You are advised to perform this operation during off-peak hours.
- If the instance load is heavy, it takes a longer time to change its instance specifications.

- If there is only one coordinator node in an instance, services will be interrupted during the specification change. Exercise caution when performing this operation.

Procedure

Step 1 Log in to the management console.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, locate the instance and choose **More** > **Change Instance Specifications** in the **Operation** column.

Alternatively, click the instance name to go to the **Basic Information** page. In the **Configuration** area, click **Change** in the **Instance Specifications** field.

Step 5 On the displayed page, specify the new instance specifications and click **Next**.

Step 6 Confirm the specifications and click **Submit**.

Step 7 View the new instance specifications.

After the task is submitted, click **Go to Instance List**. On the **Instances** page, the DB instance status is **Changing instance specifications**. After a few minutes, view the new instance specifications on the **Basic Information** page.

----End

7.5 Configuring Read Replicas

Scenarios

You can add and delete read replicas and change their specifications for GaussDB instances to meet read service requirements.

Precautions

- Read replica-related operations are supported for centralized (1 primary + 2 standby) instances of version V2.0-2.7.1 or later.
- Read replica-related operations are supported for centralized (1 primary + 1 standby + 1 log) instances of version V2.0-8.0.0 or later.
- After snapshot-based backup is enabled, read replicas cannot be added.

Adding a Read Replica

Step 1 Log in to the management console.

Step 2 Click  in the upper left corner and select a region and project.

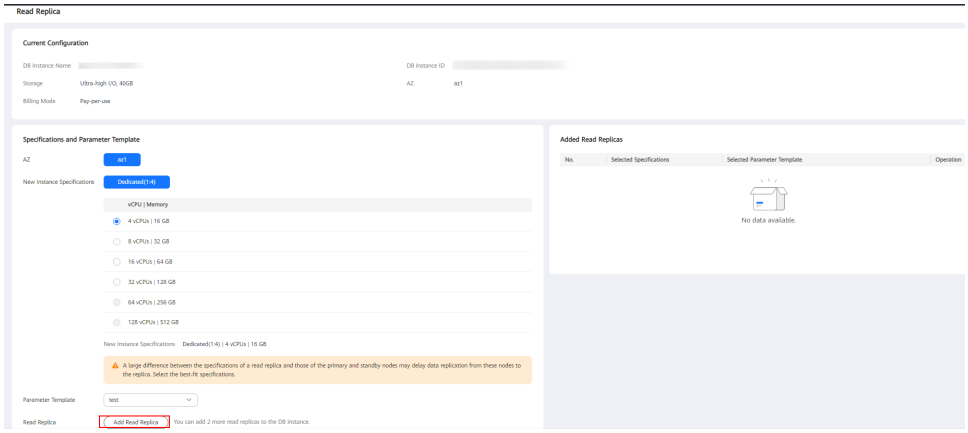
Step 3 Click ☰ in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the **Configuration** area, click **Add** in the **Read Replicas** field.

Step 6 Select the specifications and parameter template of the read replica, and click **Add Read Replica**.

Figure 7-5 Adding a read replica



Step 7 Click **Next**, confirm the information, and click **Submit**.

Step 8 Check the result of adding a read replica.

On the **Instances** page, the instance status is **Creating read replicas**. After the instance status becomes **Available**, click the instance name. On the **Basic Information** page, check the number of read replicas.

----End

Deleting a Read Replica

Step 1 **Log in to the management console**.

Step 2 Click 📍 in the upper left corner and select a region and project.

Step 3 Click ☰ in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the **Configuration** area, click **Delete** in the **Read Replicas** field.

Step 6 Select the read replica to be deleted and click **Next**.

Step 7 Confirm the information and click **Submit**.

Step 8 Check the result of deleting a read replica.

On the **Instances** page, the instance status is **Deleting a read replica**. After the instance status becomes **Available**, click the instance name. On the **Basic Information** page, check the number of read replicas.

----End

Changing the Specifications of Read Replicas

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the **Configuration** area, click **Change** in the **Read Replicas** field.

Step 6 On the displayed page, select the target read replica and new specifications, and click **Next**.

Step 7 Confirm the specifications and click **Submit**.

Step 8 Check the specification change result.

After the task is submitted, click **Go to Instance List**. On the **Instances** page, the instance status is **Changing instance specifications**. After a few minutes, check the new instance specifications on the **Basic Information** page.

----End

7.6 Synchronizing Data to a Single-Replica Instance

Scenarios

GaussDB allows you to synchronize data from a three-replica instance to a single-replica instance.

Precautions

- If the instance or any of its nodes is abnormal, you cannot perform this operation.
- If you are performing other operations on a single-replica instance, this operation is unavailable.
- After this operation is performed, all data on the target single-replica instance, including its password, will be overwritten.

Procedure

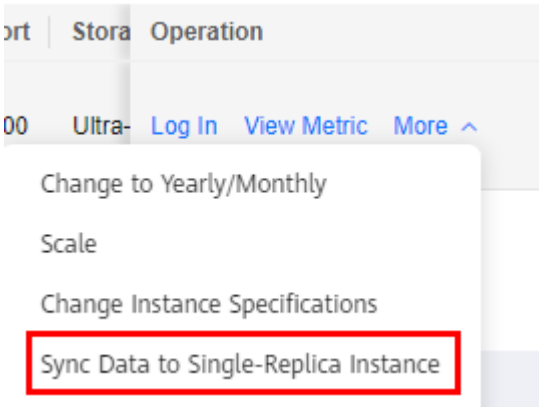
Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click ☰ in the upper left corner of the page and choose **Databases** > **GaussDB**.

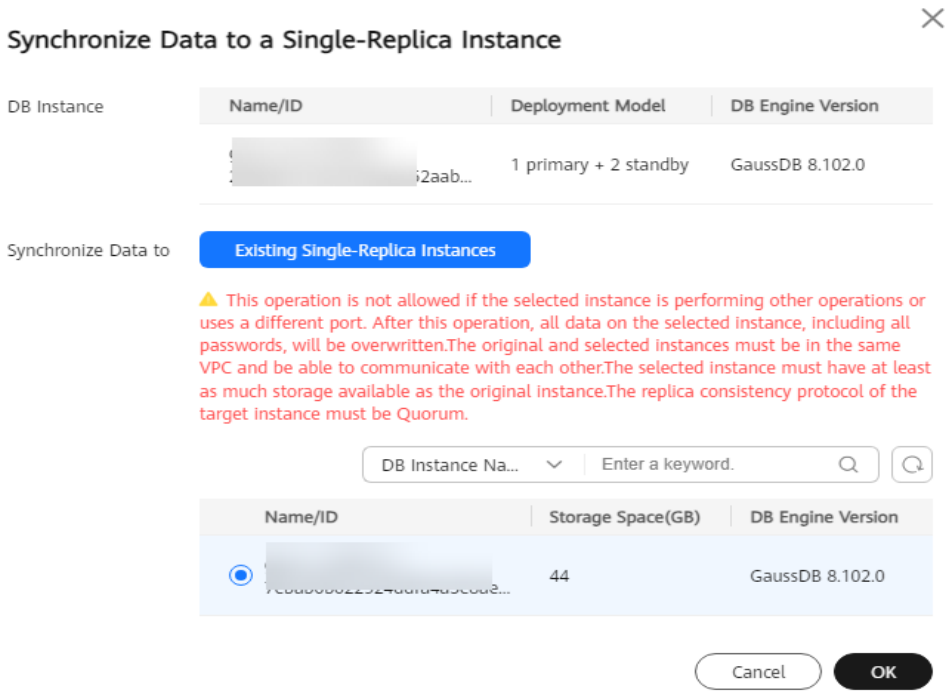
Step 4 On the **Instances** page, click **More** in the **Operation** column of the target instance and choose **Synchronize Data to a Single-Replica Instance**.

Figure 7-6 Choosing Synchronize Data to a Single-Replica Instance



Step 5 In the displayed dialog box, select the target instance and click **OK**.

Figure 7-7 Synchronizing data to a single-replica instance



----End

7.7 Viewing and Modifying Advanced Features

Scenarios

You can enable advanced features such as advanced compression, dynamic data masking, and anti-tampering for your instance on the instance details page.

Precautions

- Advanced compression status can only be changed for instances of the enterprise or basic edition whose version is V2.0-8.1.0 or later.
- Dynamic data masking and anti-tampering are only supported for instances of the enterprise edition whose version is V2.0-8.0.0 or later.

Procedure

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the **Configuration** area, click **View and Modify** in the **Advanced Features** field.

Step 6 In the feature list, change the feature statuses or values, enter **Yes**, and click **OK**.

Figure 7-8 Modifying advanced features for an instance of the enterprise edition

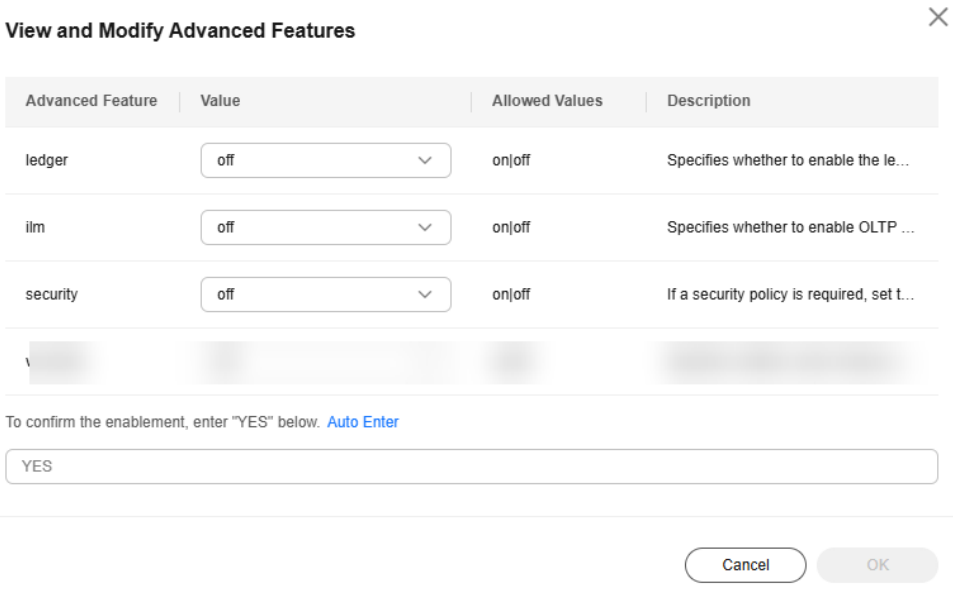
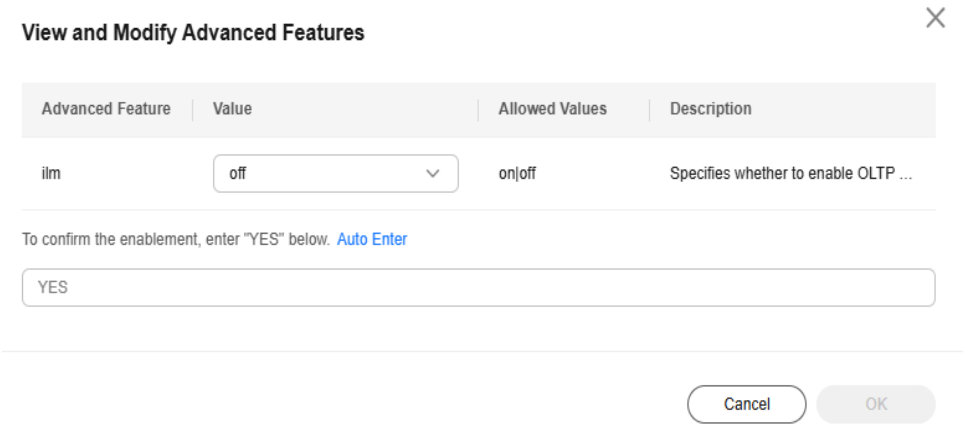


Figure 7-9 Modifying the advanced feature for an instance of the basic edition



Step 7 Wait until the modification is complete.

----End

7.8 Scaling In and Out an Instance

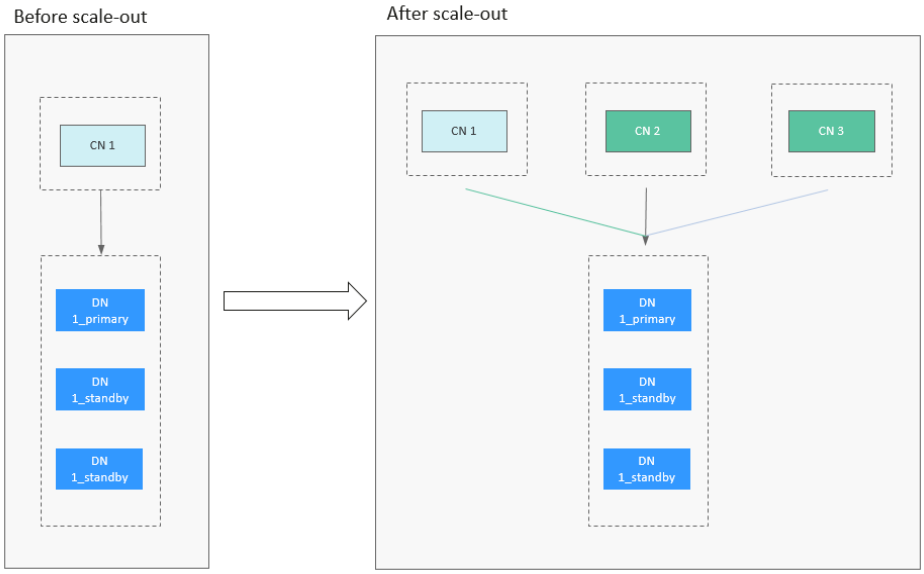
7.8.1 Overview of Scaling In and Out an Instance

After you purchase a GaussDB instance, resource requirements may change with service changes. In this scenario, GaussDB distributed instances that are independently deployed can be scaled in or out. Users can adjust resources as required. Currently, instances can be scaled in or out in the following ways:

Adding CNs for an Instance

Instances can be scaled out by adding CNs. For example, if the original instance is configured with 1 CN, 3 replicas, and 1 shard, the instance will have 3 CNs, 3 replicas, and 1 shard after 2 CNs is added. For details, see **Adding Coordinator Nodes for an Instance (Distributed)**.

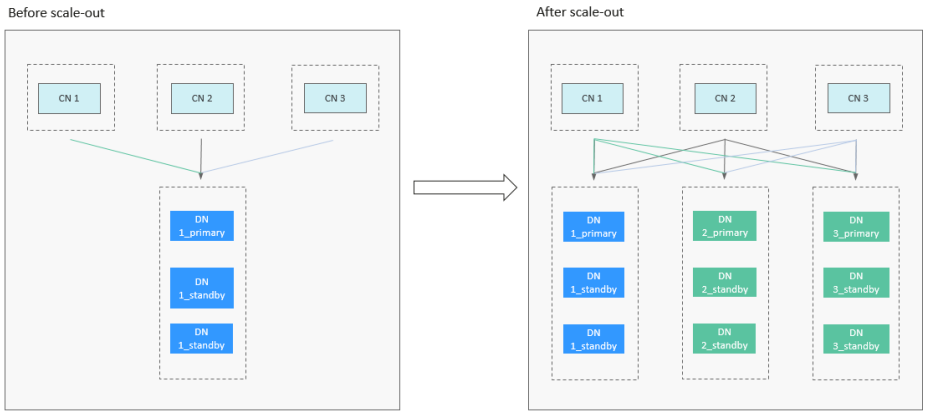
Figure 7-10 Adding CNs for an instance



Adding Shards for an Instance

Instances can be scaled out by adding shards. For example, if the original instance is configured with 3 CNs, 3 replicas, and 1 shard, the instance will have 3 CNs, 3 replicas, and 3 shards after 2 shards is added. For details, see **Adding Shards for an Instance (Distributed)**.

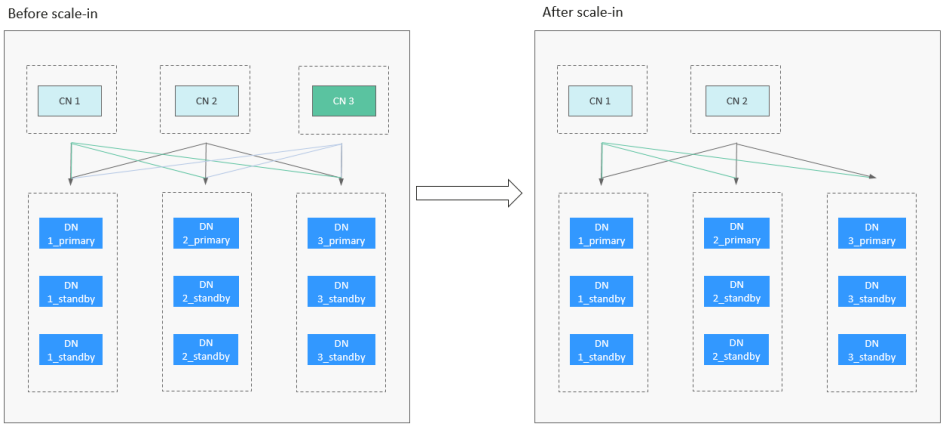
Figure 7-11 Adding shards for an instance



Deleting CNs for an Instance

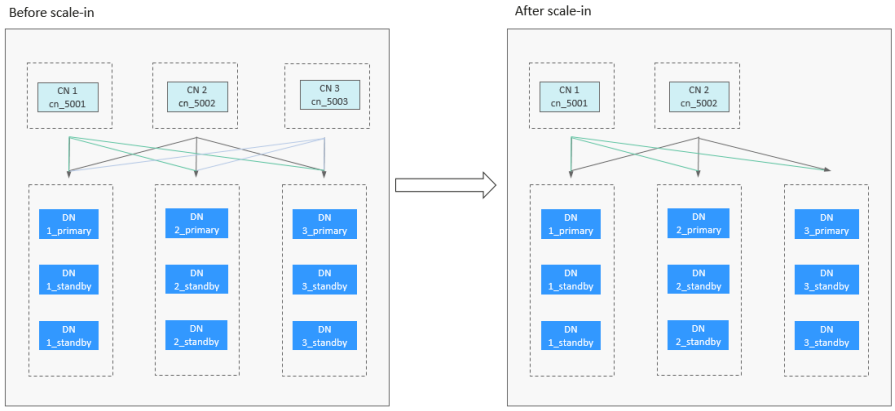
Instances can be scaled in by deleting CNs. For example, if the original instance is configured with 3 CNs, 3 replicas, and 3 shards, the instance will have 2 CNs, 3 replicas, and 3 shards after 1 CN is deleted. For details, see **Deleting Coordinator Nodes for an Instance (Distributed)**.

Figure 7-12 Deleting CNs for an instance



Main processes are running on the main CN (that is, the CN whose component ID is cn_5001), so this CN cannot be deleted for scale-in. You can call the **Querying the Components of a DB Instance** API to query cn_5001. To scale in the instance, another CN will be deleted. In the example shown in **Figure 7-13**, the CN whose component ID is cn_5003 is deleted instead.

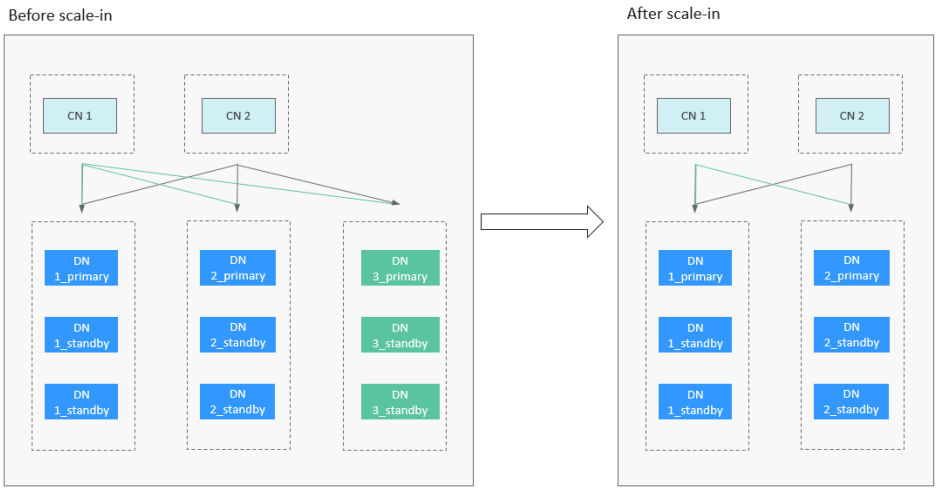
Figure 7-13 Deleting CNs for an instance



Deleting Shards for an Instance

Instances can be scaled in by deleting shards. For example, if the original instance is configured with 2 CNs, 3 replicas, and 3 shards, the instance will have 2 CNs, 3 replicas, and 2 shards after 1 shard is deleted. For details, see **Deleting Shards for an Instance (Distributed)**.

Figure 7-14 Deleting shards for an instance



7.8.2 Adding Coordinator Nodes for an Instance (Distributed)

Scenarios

As the instance deployment time and data increase, the database performance and storage will gradually reach the bottleneck. Adding nodes can improve the instance performance and storage capacity. You can only add nodes for distributed GaussDB instances that are deployed independently.

NOTICE

- The scaling duration depends on the amount of data. The default timeout period is seven days. When nodes are being added, the instance is available, but you are not allowed to perform other operations on the console. If you need to perform any operations, contact customer service.
- You can flexibly add CNs or shards as needed. It is recommended that the number of CNs of a DB instance do not exceed twice the number of shards.
- Instances can be scaled out only when they are in the **Available** state.

Procedure

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

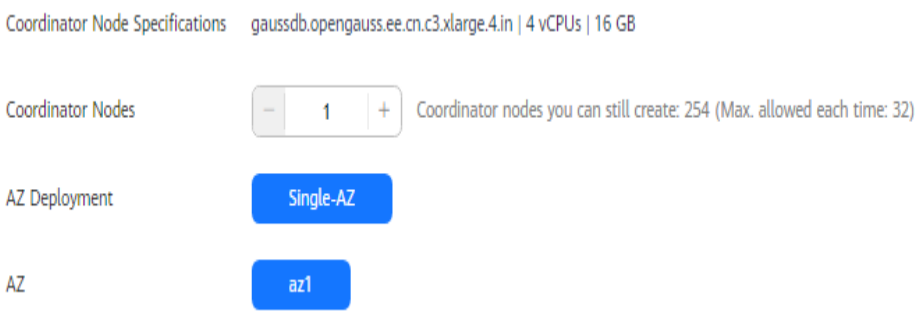
Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the instance for which you want to add nodes.

Step 5 On the **Basic Information** page, click **Add** in the **Coordinator Nodes** field.

Step 6 Specify the number of coordinator nodes to be added and the AZ.

Figure 7-15 Adding coordinator nodes



If single-AZ deployment is specified during the instance creation, CNs are only added to the AZ you specified.

Step 7 Click **Next**.

Step 8 Confirm the information and click **Submit**.

----End

7.8.3 Adding Shards for an Instance (Distributed)

Scenarios

As the instance deployment time and data increase, the database performance and storage will gradually reach the bottleneck. In this case, you need to add hosts to improve the instance performance and storage capability. This function is available only for distributed GaussDB instances that are deployed independently.

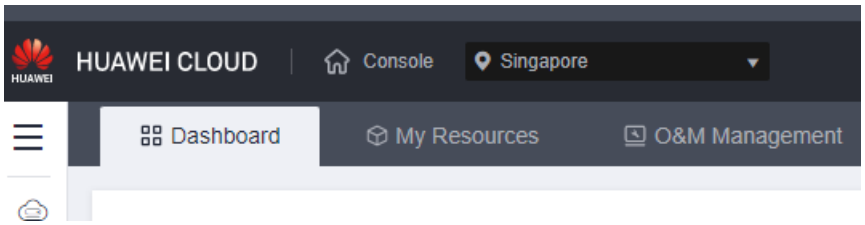
NOTICE

- The scaling duration depends on the amount of data. The default timeout period is seven days. When nodes are being added, the instance is available, but you are not allowed to perform other operations on the console. If you need to perform any operations, contact customer service.
- Instances can be scaled out only when they are in the **Available** state. During shard scale-out, you can still query and insert data, query services are not interrupted, and the data insertion performance is not affected. The performance of join queries on local tables across node groups during redistribution may be affected.

Procedure

Step 1 **Log in to the management console**.

Step 2 Click in the upper left corner and select a region and project.



Step 3 Click ☰ in the upper left corner of the page and choose **Databases** > **GaussDB**.

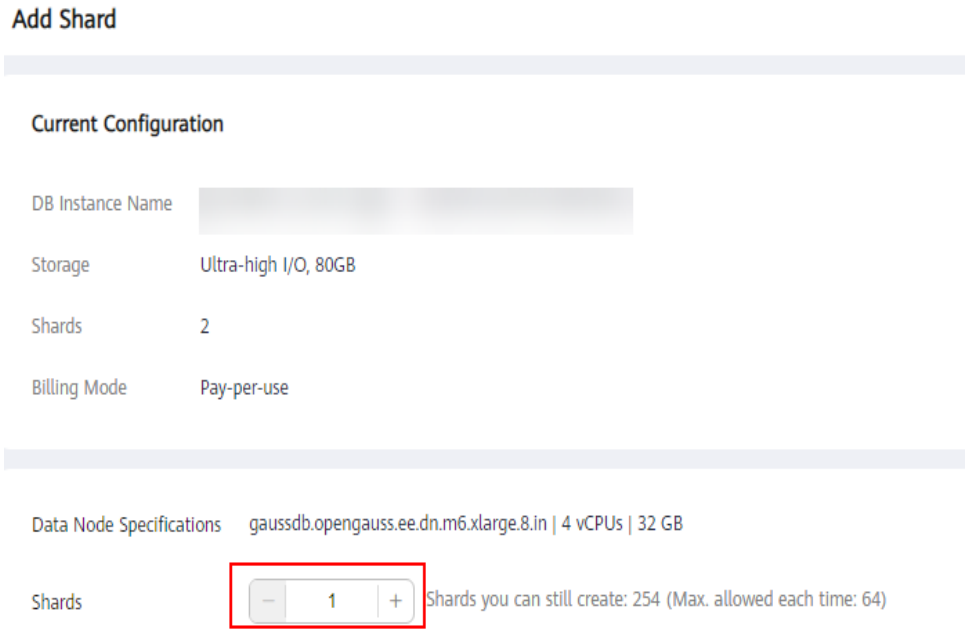
Step 4 On the **Instances** page, click the name of the target instance.

Step 5 On the **Basic Information** page, click **Add** in the **Shards** field.

Figure 7-16 Adding shards

Step 6 Specify the number of shards to be added. Click **Next**.

Figure 7-17 Adding shards



Step 7 Confirm the information and then click **Submit**.

NOTE

By default, a shard contains three replicas (a primary DN and two standby DNs). Each time you add a shard, three replicas will be added.

----End

7.8.4 Deleting Coordinator Nodes for an Instance (Distributed)

Scenarios

As service demand decreases, some CNs are left idle. To improve resource utilization, you can delete unnecessary coordinator nodes. This function is available only for distributed GaussDB instances that are deployed independently.

Precautions

- Deleting CNs does not interrupt ongoing services.
- You can only delete the CNs of instances that were deployed independently.
- At least one CN needs to be reserved for each DB instance.
- Before deleting a CN, ensure that the CN is not in a JDBC connection configuration, or the high availability of the JDBC connection may be affected.
- DDL operations will be rolled back when CNs are being deleted.
- PITR backup is suspended during the deletion and is automatically restored after deletion is complete.
- After the deletion is complete, a full backup is performed automatically.
- Before you delete CNs, you need to ensure that the instance status and all CNs are normal.
- Main processes are running on the main CN (that is, the CN whose component ID is cn_5001), so this CN cannot be deleted for scale-in. You can call the **Querying the Components of a DB Instance** API to query cn_5001. If the CN to be deleted is cn_5001, the system will randomly select another CN to delete.

Procedure

Step 1 **Log in to the management console**.

Step 2 Click  in the upper left corner and select a region and project.

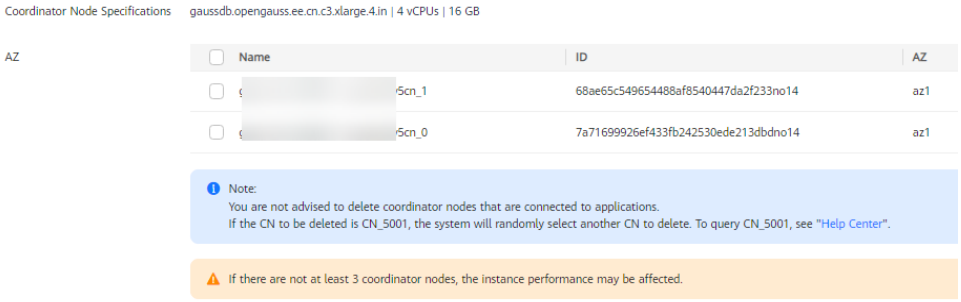
Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the instance for which you want to delete CNs.

Step 5 In the **DB Information** area of the **Basic Information** page, delete CNs.

1. Click **Delete** next to **Coordinator Nodes**.
2. Select the coordinator nodes to be deleted.

Figure 7-18 Deleting CNs



3. Click **Next**.
4. Confirm the information about the CNs to be deleted and click **Submit**.

----End

7.8.5 Deleting Shards for an Instance (Distributed)

Scenarios

There may be more than enough DNs in your DB instance after read/write splitting is enabled or redundant business data is cleared. You can delete shards as needed to avoid cost waste. This function is available only for distributed GaussDB instances that are deployed independently.

Precautions

- The scaling duration depends on the amount of data. The default timeout period is seven days. When shards are being deleted, the instance is available, but you are not allowed to perform other operations on the console. If you need to perform any operations, contact customer service.
- When shards are being deleted, existing sessions on the DNs in the deleted shards will be cleared, and some services will be affected. Therefore, delete shards during off-peak hours.
- There must be at least one shard in a DB instance. The storage space of the instance after the deletion must meet and following condition: Used space of the current instance/Number of DNs after the deletion + Maximum capacity of a table/Number of DNs after the deletion) < Read-only threshold (85%) x Disk capacity
- PITR backup is suspended during the deletion and is automatically restored after deletion is complete.
- After the deletion is complete, a full backup is performed automatically.
- DB instances can be scaled in only when they are in the **Available** state. When shards are being deleted, you can still query and insert data, query services are not interrupted, and the data insertion performance is not affected. The performance of join queries on local tables across node groups during redistribution may be affected.

Procedure

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance.

Step 5 On the **Basic Information** page, click **Delete** in the **Shards** field.

Figure 7-19 Deleting shards

Step 6 Select the number of shards to be deleted and click **Next**.

Figure 7-20 Deleting shards

Step 7 Confirm the information and then click **Submit**.

NOTE

By default, a shard contains three replicas (a primary DN and two standby DNs). Each time you delete a shard, three replicas will be deleted.

----End

7.9 Scaling Up Storage Space

7.9.1 Overview of Scaling Up Storage Space

As more data is added, you may start to run out of space. This section describes how to scale up the storage space of a DB instance. **Table 7-1** describes the scale-up methods provided by GaussDB.

Table 7-1 Scale-up methods

Scale-up Method	Supported Instance Type	Scope	Description
Manually Scaling Up Storage Space for an Instance	• Distributed • Centralized	All shards	Shard-level scale-up is performed based on the selected storage space. All shards are scaled up at a time. The added storage space must be a multiple of 4 GB x *Number of shards*.
Manually Scaling Up Storage Space of Specified Shards	Distributed	• Specified shards • All shards	Shard-level scale-up is performed based on the selected storage space. One or more shards are scaled up at a time. The added storage space must be a multiple of 4 GB.

Scale-up Method	Supported Instance Type	Scope	Description
Configuring Storage Autoscaling for an Instance	• Distributed • Centralized	• Specified shards • All shards	Shard-level scale-up is performed based on the user-defined scaling increment (specified by the **Scale Up By** parameter). When the available storage of a DN component drops to or below the specified threshold, an automatic scale-up task is triggered to scale up the shard where the DN component with insufficient available storage is located. Storage can be scaled by percentage or a fixed amount. • Percentage: The storage is scaled up by a percentage you specify. The added storage space is the storage space of the shard multiplied by the scaling increment. • Fixed amount: The storage is scaled up by a fixed amount you specify. The added storage space is a fixed value.

Manually Scaling Up Storage Space for an Instance

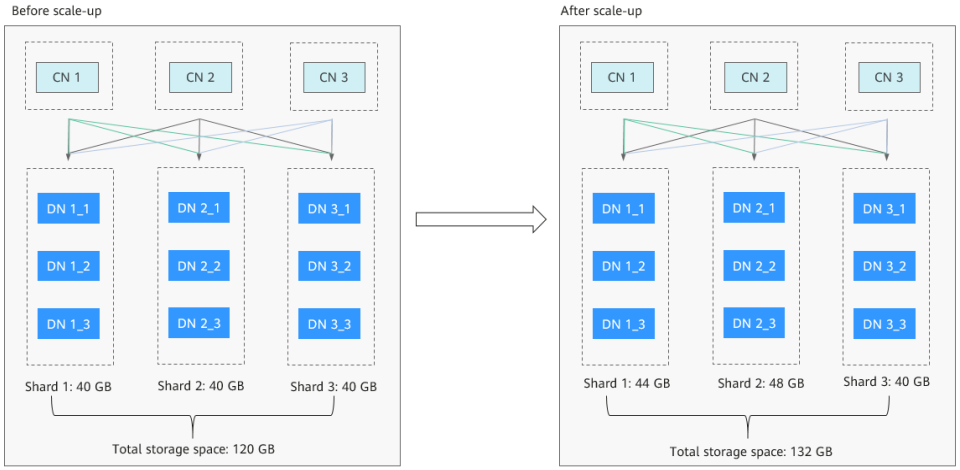
Let's take an independently deployed distributed instance as an example. This instance has three shards, three replicas, and three CNs, which are the default configurations, with a total storage space of 120 GB before the scale-up. This means that each shard has a storage space of 40 GB. If 12 GB of storage is added, the total storage space is 132 GB after scale-up, including 4 GB for each shard. After the scale-up, each shard has a storage space of 44 GB.

Figure 7-21 Scaling up storage space

Manually Scaling Up Storage Space of Specified Shards

Let's take an independently deployed distributed instance as an example. This instance has three shards, three replicas, and three CNs, which are the default configurations, with a total storage space of 120 GB before the scale-up. This means that each shard has a storage space of 40 GB. If you increase the storage space of shard 1 and shard 2 by 4 GB and 8 GB, respectively, the total storage space after scale-up is 132 GB. The storage spaces of shard 1 and shard 2 are 44 GB and 48 GB, respectively.

Figure 7-22 Scaling up storage space



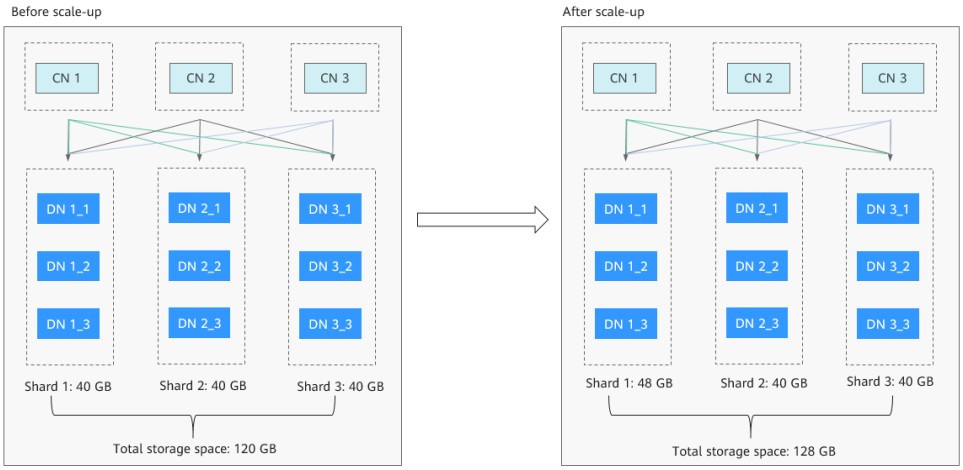
Configuring Storage Autoscaling for an Instance

- By percentage

  Take the independently deployed distributed instance with three shards, three replicas, and three CNs as an example. Its total storage space before the scale-up is 120 GB, with each shard holding 40 GB. In the autoscaling policy, both the **Trigger If Available Storage Drops To** and **Scale Up By** parameters are set to **20%**. If the available storage usage of shard 1 drops to 20% or lower, a scale-up action is automatically triggered. The added storage space

of shard 1 is 8 GB (40 x 20%). After the scale-up, the storage space of shard 1 is 48 GB, and the total storage space of the instance is 128 GB.

Figure 7-23 Scaling up storage space



- By fixed amount

  Take the independently deployed distributed instance with three shards, three replicas, and three CNs as an example. Its total storage space is 120 GB before scale-up, and each shard has a storage space of 40 GB. In the autoscaling policy, the **Trigger If Available Storage Drops To** parameter is set to **20%**, and the **Scale Up By** parameter is set to **40 GB**. If the available storage usage of the DN 1_1 and DN 3_3 components drops to or below 20%, a scale-up action is automatically triggered for the two shards where DN 1_1 and DN 3_3 are located. A 40 GB storage increment is added to both shards. After the scale-up, their storage space is both 80 GB, and the total storage space of the instance is 200 GB.

Figure 7-24 Scaling up storage space

7.9.2 Manually Scaling Up Storage Space for an Instance

Scenarios

As more data is added, you may start to run out of space. If the kernel system detects that the storage usage exceeds the predefined threshold, the instance is set to read-only and no data can be written to it. (The default threshold is 85%. You can **modify the instance parameter cms:datastorage_threshold_value_check** to change the usage threshold.) This section describes how to scale up the storage space of a DB instance. Services will not be interrupted during storage scale-up.

Precautions

- Within the maximum allowed range, the storage usage cannot exceed the predefined threshold (85% by default) after the scale-up.
- If any node becomes faulty, contact the O&M engineers for troubleshooting before the scale-up.
- The storage space must be a multiple of (Number of shards x 4 GB).
- A single shard can hold up to 24 TB
- If a DB instance is unavailable because the storage space is used up, you can scale up the storage space.
- If a DR relationship has been established for the instance and the instance functions as the primary one, ensure that the storage of its DR instances is greater than or equal to that of the primary instance during storage scale-up. Otherwise, disaster recovery may fail if the data volume of the primary instance is greater than that of the DR instance.

Constraints

- You can scale up storage space only when your account balance is greater than or equal to $0 USD.
- The maximum allowed storage for a single shard is 24 TB by default. There is no limit on the number of scale-ups.
- The DB instance is in the **Scaling up** state when its storage space is being scaled up and the backup services are not affected.
- Do not reboot or delete the instance whose storage is being scaled up.
- Storage space can only be scaled up, not down.

Procedure

Step 1 **Log in to the management console**.

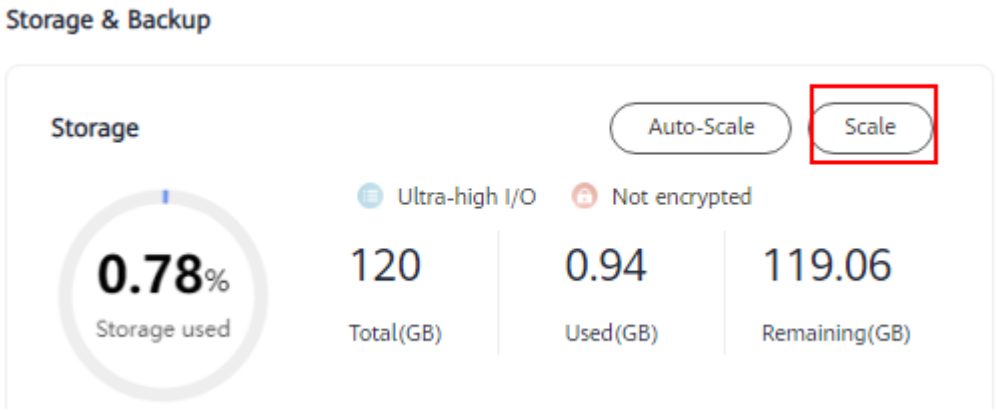
Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, locate the instance you want to scale up and click **More** > **Scale Storage Space** in the **Operation** column.

Alternatively, click the instance name to go to the **Basic Information** page. In the **Storage** section of the **Storage & Backup** area, click **Scale**.

Figure 7-25 Scaling up storage



Step 5 On the displayed page, specify the new storage space and click **Next**.

Figure 7-26 Setting the space size

When you scale up the storage space, ensure that the usage of the new storage space is less than the predefined threshold (85% by default). Once the storage usage of an instance reaches or exceeds the specified threshold, the instance cannot process write operations and becomes read-only.

Step 6 Confirm settings.

- If you need to modify your settings, click **Previous**.
- If the settings are correct, click **Submit**.

Step 7 View the storage scale-up results.

During the scale-up, the status of the instance on the **Instances** page is **Scaling up**. This process may take 3 to 5 minutes. Once the scale-up is complete, click the instance name to go the **Basic Information** page and you can see the new storage space.

----End

7.9.3 Manually Scaling Up Storage Space of Specified Shards

Scenarios

As more data is added, you may start to run out of space. If the kernel system detects that the storage usage exceeds the predefined threshold, the instance is set to read-only and no data can be written to it. (The default threshold is 85%. You can modify the instance parameter **cms:datastorage_threshold_value_check** to change the usage threshold.) You can scale up the storage space of one or more specified shards. Services are not interrupted during storage scale-up.

Precautions

- Within the maximum allowed range, the storage usage cannot exceed the predefined threshold (85% by default) after the scale-up.
- If any node becomes faulty, contact the O&M engineers for troubleshooting before the scale-up.
- The storage space must be a multiple of (Number of shards x 4 GB).
- Each shard can hold up to 24 TB, so 24 TB of storage can be added for each shard added.
- If a DB instance is unavailable because the storage space is used up, you can scale up the storage space.
- The disk size of all shards must be the same.
- When you restore backup data to a new instance, the number of disks of the new instance is the number of disks in the largest shard of the original instance multiplied by the number of shards.

Constraints

- You can scale up storage space only when your account balance is greater than or equal to $0 USD.
- The maximum allowed storage is 24 TB. There is no limit on the number of scale-ups.
- The DB instance is in the **Scaling up** state when its storage space is being scaled up and the backup services are not affected.
- Reboot is not required during instance scale-up.
- You cannot reboot or delete an instance that is being scaled up.
- Storage space can only be scaled up, not down.

Procedure

Step 1 **Log in to the management console**.

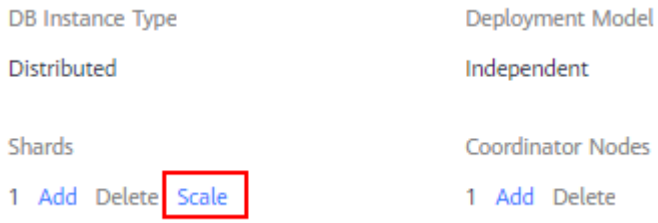
Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance.

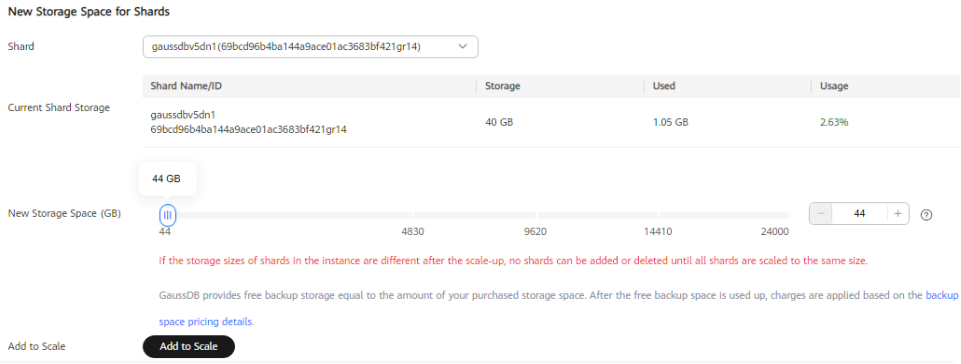
Step 5 On the **Basic Information** page, click **Scale** in the **Shards** field.

Figure 7-27 Basic information



Step 6 On the **Scale Storage Space** page, select target shards one by one, set the new storage space, and click **Add to Scale**. Then, click **Next**.

Figure 7-28 Scaling up storage for specified shards



Ensure that the usage of the new storage space is less than the predefined threshold (85% by default) after the scale-up. An instance can be restored from read-only to the read/write state only when the storage usage is lower than the specified threshold.

Step 7 Confirm settings.

- If you need to modify your settings, click **Previous**.
- If your settings are correct, click **Submit**.

Step 8 View the results.

During the scale-up, the status of the instance on the **Instances** page is **Scaling up**. Later, click the instance name to go the **Basic Information** page and view the new storage space. This process may take 3 to 5 minutes.

----End

7.9.4 Configuring Storage Autoscaling for an Instance

Scenarios

You can enable autoscaling for a GaussDB instance so that its storage can be automatically scaled up when the disk usage reaches the specified threshold.

Precautions

- DB instances of the basic edition do not support storage autoscaling. If autoscaling has been enabled for such an instance, the automatic scale-up task will be automatically stopped upon request and autoscaling will be disabled.
- All nodes in the target instance must be in an available state.
- Storage autoscaling is mutually exclusive with the following operations: manually scaling up storage space, adding nodes, changing the disk type, deleting an instance, checking snapshots, updating agents, and storage autoscaling. That is, storage autoscaling cannot be performed when any of the preceding operations is ongoing, even if the autoscaling policy is configured.
- Autoscaling for centralized instances is at the instance level.
- Autoscaling for distributed instances is at the shard level.
- If the storage sizes of shards in a distributed instance are different after the scale-up, no shards can be added or deleted until all shards are scaled to the same size.
- During storage autoscaling, the storage space is increased in increments of 40 GB.
  - If the space to increase exceeds the upper limit you have specified, only the space size equal to the upper limit will be increased.
  - If the space to increase exceeds the upper limit specified by the system, only the space size equal to the upper limit will be increased.
- An alarm will be generated when autoscaling fails. This alarm will be automatically cleared when the disk usage is lower than the specified threshold.
- If a yearly/monthly DB instance has pending orders, autoscaling will fail.
- If your account balance is insufficient, autoscaling will fail.

Procedure

Step 1 **Log in to the management console**.

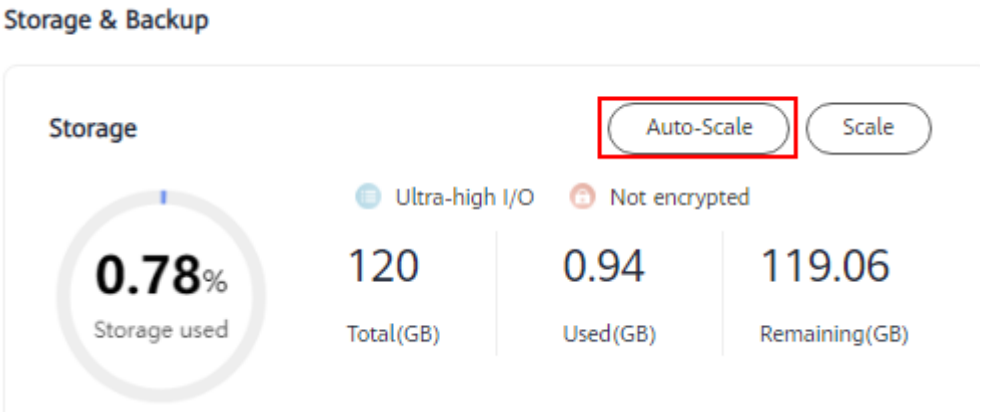
Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the **Storage & Backup** area, click **Auto-Scale**.

Figure 7-29 Enabling autoscaling



Step 6 In the displayed **Configure Storage Autoscaling** dialog box, set the following parameters:

Figure 7-30 Configuring autoscaling

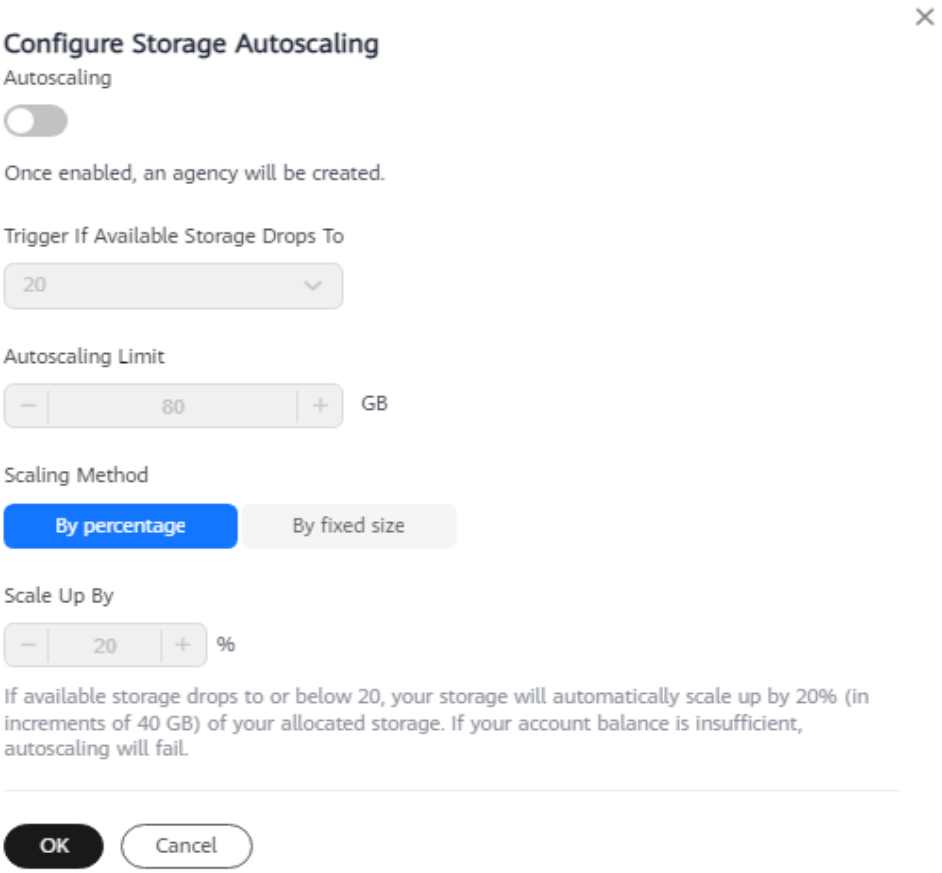


Table 7-2 Parameters

Parameter	Description
Autoscaling	Specifies whether to enable automatic scale-out. By default, automatic scale-out is disabled.
Trigger If Available Storage Drops To	The storage will be automatically scaled up if the available storage drops to or below the threshold specified by this parameter. The default value is **20%**. The value can be **20%**, **25%**, or **50%**.
Autoscaling Limit	Upper limit of the storage space in GB that can be automatically scaled to. The value of this parameter must be greater than the current storage of the instance. Value range: ● Centralized instances: [*Current storage* + 40 GB, 24,000 GB] ● Distributed instances: [*Current storage* + 40 GB, 24,000 GB x *Number of shards*]
Scaling Method	The value can be **By percentage** or **By fixed size**. ● If **By percentage** is selected, the storage space to be expanded increases each time. ● If **By fixed size** is selected, a fixed volume of storage will be expanded each time. Evaluate your workloads and costs and select a method as required.

Parameter	Description
Scale Up By	Size of the storage space to be expanded each time, which depends on the selected scaling method. Storage can be scaled by percentage or fixed size. • If **By percentage** is selected for **Scaling Method**, the default value of this parameter is **20%**, and the value range is [1%, 100%]. If the available storage drops to or below the specified threshold, the storage will be automatically scaled up by a percentage specified by this parameter (in increments of 40 GB). For example, if the current storage space of an instance is 40 GB and this parameter is set to **20%**, the volume to be expanded is 8 GB, which will be rounded up to 40 GB. • If **By fixed size** is selected for **Scaling Method**, the default value of this parameter is **40 GB**, and the value range is [40 GB, *Current storage space* + 40 GB]. If the available storage drops to or below the specified threshold, the storage will be automatically scaled up by a fixed size specified by this parameter.

Step 7 Click **OK**.

----End

7.10 Changing the Deployment Model

7.10.1 Overview of Changing the Deployment Model

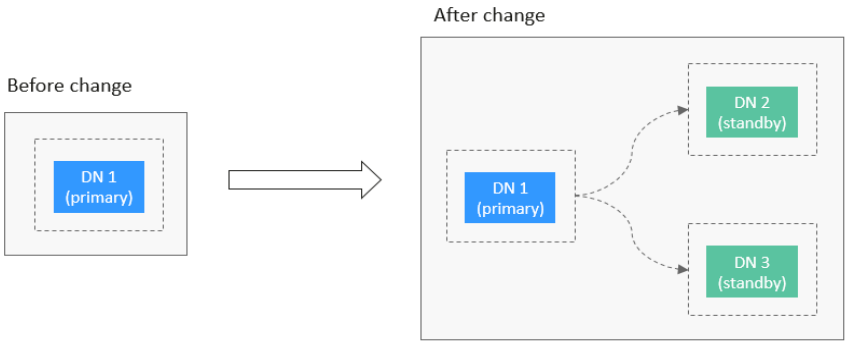
To meet various service requirements, GaussDB allows you to change the instance deployment model. Currently, the following deployment models are supported:

- For centralized instances, a single-replica instance can be changed into a 1 primary + 2 standby instance or a 1 primary + 1 standby + 1 log instance. For details, see **Changing the Deployment Model of a Single-Replica Instance (Centralized)**. Note that rollback is not supported after the deployment model of a single-replica instance is changed.
- For distributed instances, a standby DN can be changed into a log node. For details, see **Changing Standby DNs to Log Nodes (for a Distributed Instance)**. The log node is only used to store logs and does not incur fees, which reduces costs and resource consumption. During the change, services will be interrupted for about 1 minute. So, change the deployment model during off-peak hours.

Deployment Model Change Principle of a Single-Replica Centralized Instance

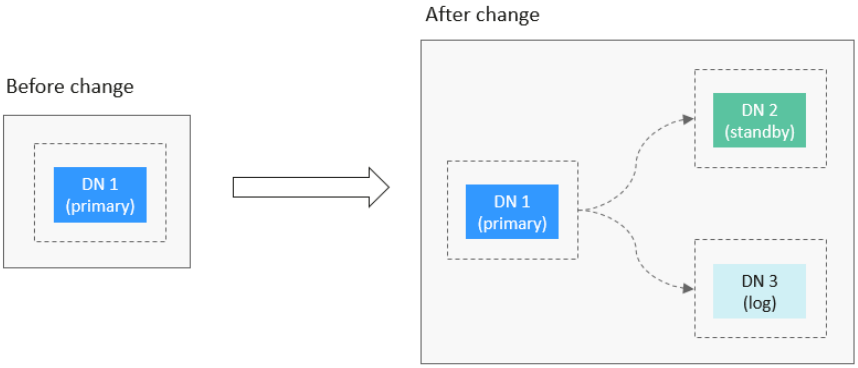
- Changing a single-replica instance into a 1 primary + 2 standby instance: In this case, two standby nodes will be added after the change.

  Figure 7-31 Changing a single-replica instance into a 1 primary + 2 standby instance



- Changing a single-replica instance into a 1 primary + 1 standby + 1 log instance: In this case, one standby node and one log node will be added after the change.

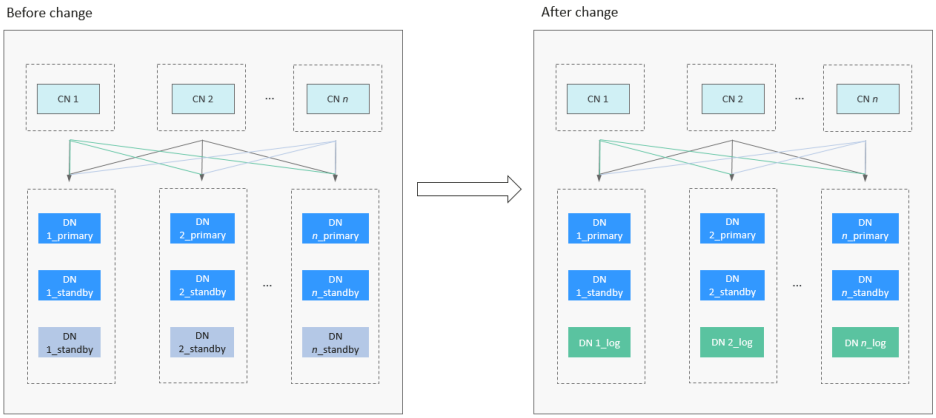
  Figure 7-32 Changing a single-replica instance into a 1 primary + 1 standby + 1 log instance



Deployment Model Change Principle of a Distributed Instance

- Changing standby DNs to log nodes: Before the change, each shard of the instance has one primary DN and two standby DNs. After the change, one of the standby DNs in each shard is changed to a log node.

Figure 7-33 Changing standby DNs into log nodes



7.10.2 Changing the Deployment Model of a Single-Replica Instance (Centralized)

Scenarios

You can change the deployment model of a single-replica GaussDB instance to an instance with multiple replicas.

NOTE

To apply for the permissions needed, submit an application by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.

Precautions

- Currently, the following deployment model change scenarios are supported:
  - Change a single-replica centralized instance to a 1 primary + 2 standby instance.
  - Change a single-replica centralized instance to a 1 primary + 1 standby + 1 log instance.
- Before changing the deployment model, ensure that the instance status is normal.
- The following operations cannot be performed when the deployment model of an instance is being changed: scaling up storage, changing specifications, backing up data, resetting passwords, rebooting the instance, and deleting the instance.
- Changing the deployment model will interrupt services. Therefore, perform this operation during off-peak hours.
- After the deployment model is changed, the specifications of the new nodes are the same as those of the original nodes. The specifications of the log node use the configured specifications for log nodes.
- After a single-replica centralized instance is changed to a 1 primary + 1 standby + 1 log instance, its replica consistency protocol changes to Paxos.
- After the deployment model is changed, an automated backup will be triggered and log archiving will be enabled.

- Only the deployment model of single-replica instances whose version is V2.0-8.0 or later can be changed.
- Currently, only the deployment model of pay-per-use instances can be changed.

Procedure

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

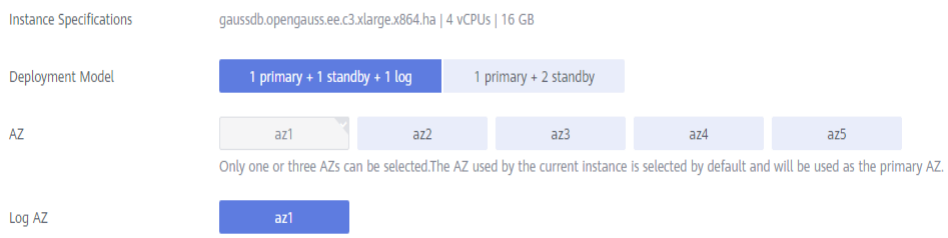
Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the **Configuration** area, click **Change** in the **Deployment Model** field. The **Change Deployment Model** page is displayed.

Figure 7-34 Changing the deployment model

Step 6 On the **Change Deployment Model** page, select the new deployment model and AZ, and click **Next**.

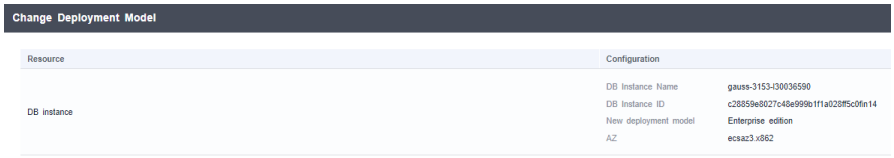
Figure 7-35 Changing the deployment model



Step 7 Confirm the displayed details.

- If you need to modify your settings, click **Previous**.
- If the information is correct, click **Submit** to submit the change request.

Figure 7-36 Submitting the request



After the task is submitted, the instance status will be **Changing deployment model**.

----End

Verification After the Change

After the change is complete, check the instance status, backup creation status, and instance connectivity, and whether you can add, delete, update, and query data in the instance.

Step 1 On the **Instances** page, check whether **Status** of the target instance is **Available**.

Step 2 Click the name of the target instance to go to the **Basic Information** page. In the **Nodes** area, check that the statuses of all nodes are normal.

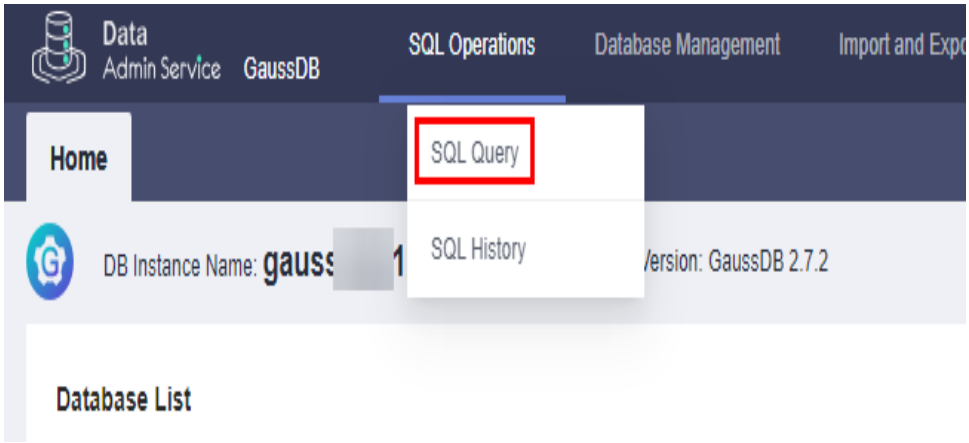
Step 3 Check that the automated backup triggered after the change is successfully created.

1. On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.
2. In the navigation pane, choose **Backups**. Check that a backup has been created and the backup status is **Completed**.

Step 4 Check that the instance is properly connected and you can add, delete, update, and query data in the instance.

1. Log in to the database. For details, see **Connecting to an Instance Through DAS**.
2. Go to the **SQL Query** page.

Figure 7-37 SQL query



3. Create a database.

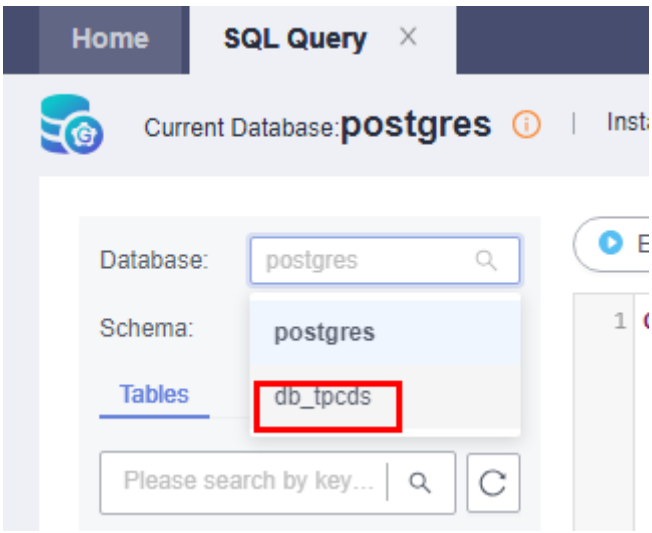
   CREATE DATABASE *database_name*;

In this example, run the following command to create a database named **db_tpcds**:

CREATE DATABASE db_tpcds;

Switch to the newly created database in the upper left corner.

Figure 7-38 Switching to the new database



4. Create a table and add, delete, update, and query data in the table.

   a. Create a schema.

      CREATE SCHEMA *myschema*;

   b. Create a table named **mytable** that has only one column. The column name is **firstcol** and the column type is integer.

      CREATE TABLE myschema.mytable *(firstcol int)*;

   c. Insert data into the table.

      INSERT INTO myschema.mytable values (100);

   d. View data in the table.

      SELECT * FROM myschema.mytable;

      ```
        | firstcol |
      ---+----------+
      1 |   100    |
      ```

   e. Update data in the table.

      UPDATE myschema.mytable SET firstcol = 200;

   f. View the data in the table again.

      SELECT * FROM myschema.mytable;

      ```
        | firstcol |
      ---+----------+
      1 |   200    |
      ```

   g. Delete the table.

      DROP TABLE myschema.mytable;

----End

7.10.3 Changing Standby DNs to Log Nodes (for a Distributed Instance)

Scenarios

Each shard of a distributed GaussDB instance consists of one primary DN and two standby DNs. However, this architecture may not meet your application requirements. GaussDB provides the flexibility to convert one standby DN in each shard into a log node.

Precautions

- This function is only available for distributed instances of V2.0-3.200.0 or later that have one primary DN and two standby DNs in each shard.
- PITR backup is suspended during the process of changing standby DNs to log nodes and automatically resumes after the operation is complete.
- After standby DNs are changed to log nodes, a full backup is automatically performed.

Procedure

Step 1 Log in to the management console.

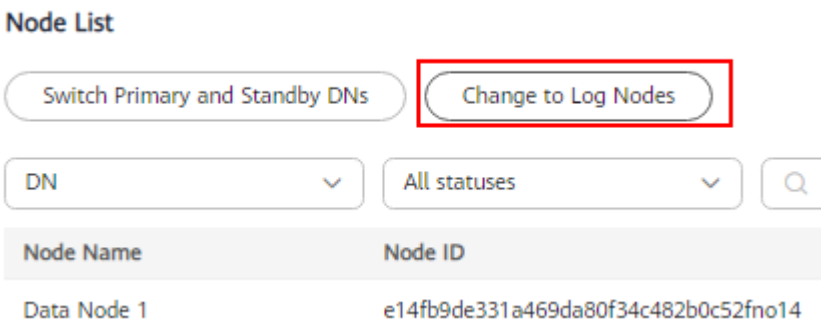
Step 2 Click in the upper left corner and select the desired region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the **Node List** area, click **Change to Log Nodes**.

Figure 7-39 Changing the deployment model to 1 primary + 1 standby + 1 log



Step 6 In the **Change Standby Data Nodes to Log Nodes** dialog box, select an AZ and click **OK**.

Step 7 Check the change result.

After the task is submitted, click **Back to DB Instance List**. On the **Instances** page, the instance status is **Changing to log node**. After the task is complete, go to the

Basic Information page of the instance and check that the deployment model is changed to 1 primary + 1 standby + 1 log.

----End

7.11 Performing a Primary/Standby DN Switchover

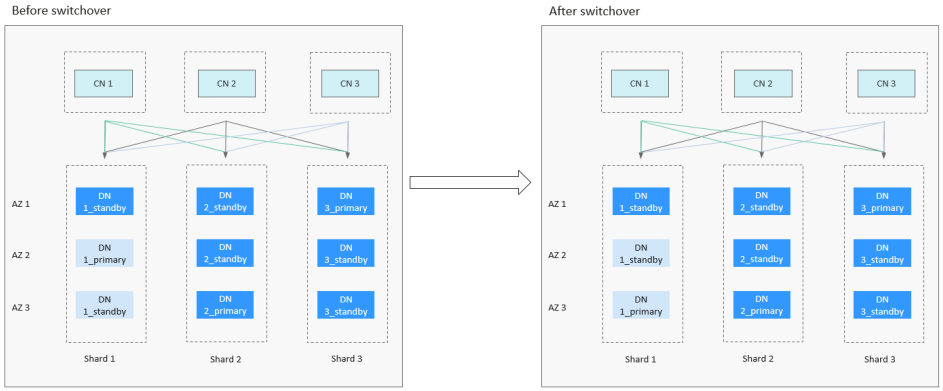
7.11.1 Overview of Performing a Primary/Standby DN Switchover

Performing a Primary/Standby DN Switchover

GaussDB supports primary/standby DN switchover in a shard in a distributed instance. You can promote a standby DN to the primary DN in a shard.

Take the distributed instances that are independently deployed as an example. The instance is configured with 3 shards, 3 replicas, and 3 AZs. If you perform a primary/standby switchover on a node in shard 1 and promote the standby node in AZ 3 to primary, the node in AZ 2 becomes the standby node and the node in AZ 3 becomes the primary node after the switchover.

Figure 7-40 Performing a primary/standby DN switchover



Changing the Failover Priority of DNs

GaussDB provides failover priority on reliability or availability.

- Reliability applies to scenarios that require high data consistency. In these scenarios, if the primary DN is faulty, services are provided only after log replay is complete to prevent data loss. During this period, connections cannot be established.
- Availability applies to scenarios that require uninterrupted online services. In these scenarios, if the primary DN is faulty and no new primary node is selected within 10 minutes, the standby node is forcibly started as the new primary node to provide services to ensure cluster availability. In this way, the cluster can be recovered as soon as possible at the cost of data loss.

7.11.2 Changing the DN Failover Priority

Scenarios

GaussDB provides failover priority on availability or reliability. You can change the failover priority of a GaussDB instance on the **Basic Information** page. Reliability applies to scenarios that require high data consistency, and availability applies to scenarios that require uninterrupted online services.

NOTE

To apply for the permissions needed, submit an application by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.

Precautions

This function is available only to distributed instances.

Procedure

Step 1 **Log in to the management console**.

Step 2 Click in the upper left corner and select a region and project.

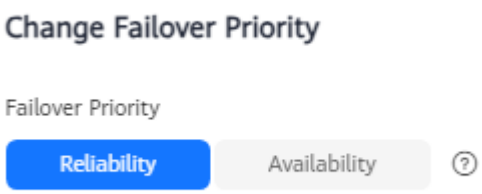
Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 Click the name of the target instance to go to the **Basic Information** page.

Step 5 Click **Change** in the **Failover Priority** field.

Step 6 In the displayed dialog box, select **Reliability** or **Availability** as required.

Figure 7-41 Changing the failover priority



- **Reliability**: Data consistency is given priority during a failover. This is recommended for applications with highest priority for data consistency.
- **Availability**: Database availability is given priority during a failover. This is recommended for applications that require their databases to provide uninterrupted online services.

NOTICE

In availability scenarios, exercise caution when changing the following database parameters:

- **recovery_time_target**: Specifies the time for the standby node to write and replay logs. The value ranges from **0** to **3600**, in seconds. The default value is **60**. **0** indicates that log flow control is disabled. A value from **1** to **3600** indicates that a standby node can write and replay logs within the period specified by this parameter, so that the standby node can quickly assume the primary role. If **recovery_time_target** is set to a small value, the performance of the primary node is affected. If it is set to a large value, the log flow is not effectively controlled. You are advised to retain the default value.
- **audit_system_object**: Specifies whether to audit the CREATE, DROP, and ALTER operations on GaussDB database objects. GaussDB database objects include databases, users, schemas, and tables. The value of this parameter ranges from **0** to **536,870,911**. The default value is **67121159**. You can change the value of this parameter to audit only the operations on required database objects. In the scenario where the leader node is forcibly selected, you are advised to set **audit_system_object** to the maximum value and audit all DDL objects.

Step 7 Click **OK**.

Step 8 After you change some parameters, manually reboot the instance for the changes to take effect. For details, see **Rebooting a GaussDB Instance**.

The failover priority cannot be changed when the DB instance is in the **Rebooting** state.

----End

7.11.3 Performing a Primary/Standby Switchover

Scenarios

GaussDB supports primary/standby DN switchover in a shard of an instance when the instance is available. You can promote a standby DN to the primary DN in a shard.

Constraints

- This operation cannot be performed when the node status is abnormal.
- Only one standby node can be specified as the primary node in a shard.
- Single-node instances do not support primary/standby DN shard switchovers.
- During a primary/standby switchover, the following operations cannot be performed:
  - Rebooting a DB instance
  - Switching AZs
  - Changing CPU and memory specifications of an instance
  - Repairing a node

  - Replacing a node
  - Adding nodes
  - Backing up and restoring an instance

Precautions

- Services may be interrupted for several seconds or minutes during the switchover. You are advised to perform this operation during off-peak hours.
- Switchovers will not change private IP addresses of an instance.

Procedure

Step 1 **Log in to the management console**.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

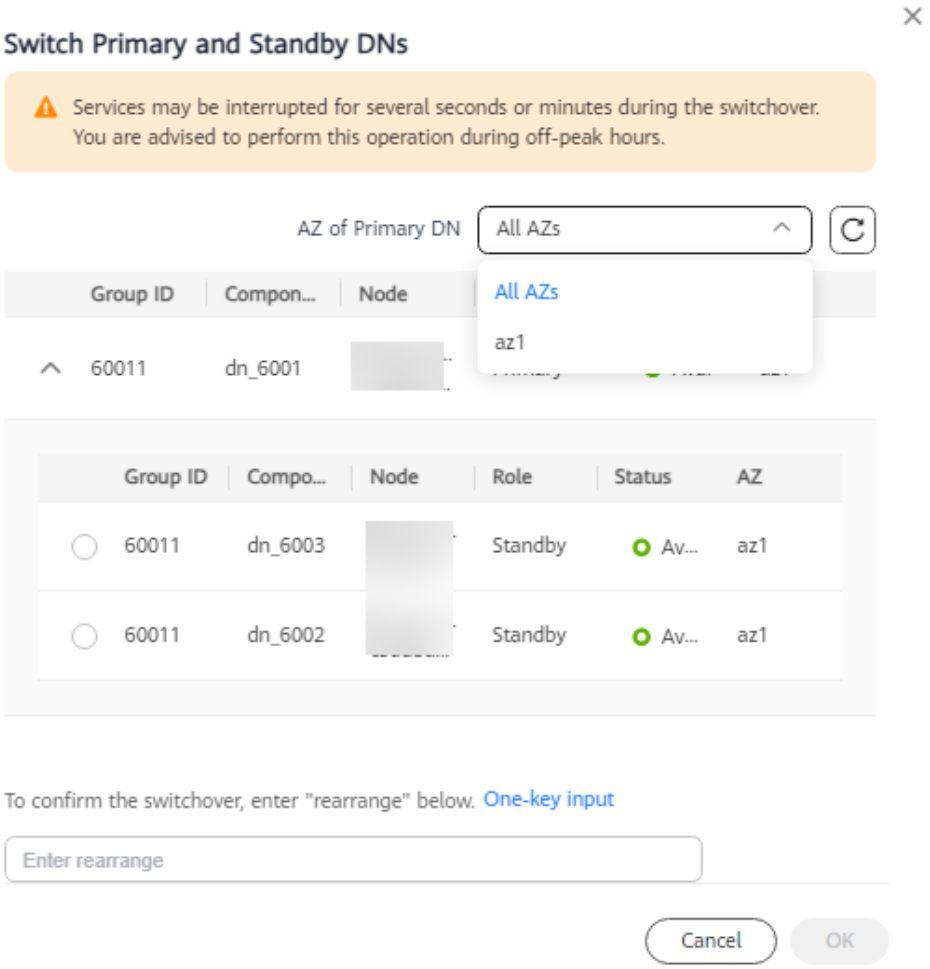
Step 4 Click the target instance name to go to the **Basic Information** page.

Step 5 In the **Node List** area, click **Switch Primary and Standby DNs**.

Step 6 Select an AZ to view the DN shard of the primary DN in the selected AZ. Select the standby DN to be promoted to primary, enter **rearrange**, and click **OK**.

- If there is no primary DN in the selected AZ, shard information is not displayed.
- Services may be interrupted for several seconds or minutes during the switchover. You are advised to perform this operation during off-peak hours.
- Primary/Standby switchovers can be performed on a maximum of 30 shards at a time.

Figure 7-42 Switching primary and standby DNs



----End

8 Instance Upgrade

8.1 Overview

You can manually upgrade the GaussDB kernel version of a single instance or multiple instances in batches using in-place upgrade, gray upgrade, or hot patch update to improve performance, add new functions, and fix bugs.

NOTE

To use this function, submit a service ticket by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.

Checking the Current Kernel Version

To check the version of an instance, go to the **Basic Information** page of the instance and check the value of **DB Engine Version** in the **Configuration** area.

Figure 8-1 Basic information

You can learn **details about kernel versions** and determine the target version to upgrade.

Upgrade Methods

The following table describes the upgrade methods supported by GaussDB.

Table 8-1 Upgrade methods

Upgrade Method	Action	Type	Scenario	Rollback Method	Impact on Services	Suggestions
Hot patch	Auto-commit	Online upgrade	Fix product issues.	• Automatic • Manual	No service is interrupted during the upgrade.	None
In-place upgrade	N/A	Offline upgrade	• Add new functions. • Fix product issues.	Automatic	Services are interrupted for about 30 minutes during the in-place upgrade.	Stop all workloads during the upgrade.

Upgrade Method	Action	Type	Scenario	Rollback Method	Impact on Services	Suggestions
Gray upgrade	Auto-commit	Online upgrade	• Add new functions. • Fix product issues.	Automatic	Services are interrupted for about 10s during the upgrade of primary DNs and during the upgrade of CNs. During upgrade commit, primary/standby distribution balancing may be performed. Services may be interrupted for different periods of time based on factors such as data volume.	Add the service reconnection mechanism. It is recommended that the retry interval be 1s. During low-pressure periods (less than 3,000 TPS + 4,000 QPS for each shard), the total retry duration is 25s. During high-pressure periods (less than 6,000 TPS + 10,000 QPS for each shard), the total retry duration is 100s. The upgrade is not recommended when the pressure is out of the acceptable range.
	Rolling upgrade	Online upgrade	• Add new functions. • Fix product issues.	• Automatic • Manual	If the AZ to be upgraded contains primary DNs, services will be interrupted for about 10s during the upgrade of each primary DN. If the AZ to be upgraded contains CNs, services will be interrupted for about 10s during the upgrade of each CN.	

8.2 Hot Patch Update

Scenarios

You can install a hot patch for your GaussDB instance to rectify product issues. A hot patch can be loaded without interrupting services and can be used to resolve some emergent database kernel problems online without affecting services. Hot patch update supports manual rollback.

Precautions

- During the update, hot patch packages will be downloaded and decompressed, which occupies certain disk space. It is recommended that the disk usage on the DN be less than or equal to the disk usage threshold minus 10%.

  NOTE

  To check the current DN disk usage, go to the metric monitoring page on the management console.

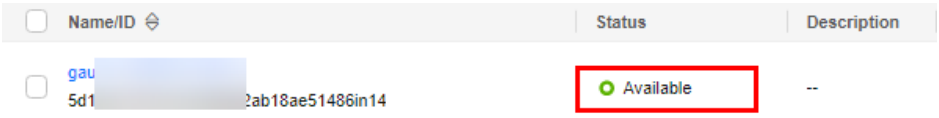
  To obtain the disk usage threshold, contact technical support.
- Version upgrade is unavailable if instance nodes are in an abnormal state.
- If a hot patch conflicts with the backup, the differential backup and full backup of the instance will be stopped during hot patch installation.
- During an upgrade or rollback, the following operations cannot be performed: scaling up storage, changing specifications, backing up data, resetting passwords, rebooting instances, and deleting instances.
- You are advised to perform an upgrade during off-peak hours because there are more idle CPU, disk, and memory resources.
- Hot patch update is available only when there is a hot patch for installation. If no hot patch is available, hot patch update is not displayed.
- Hot patch update and rollback can be performed in batches for different patch versions of a single instance. During the update, hot patches are installed in ascending order of version numbers. During the rollback, hot patches are rolled back in descending order of version numbers.
- If the upgrade fails, the system automatically rolls back the instance to the source version. You can contact Huawei Cloud technical support, and Huawei Cloud engineers will help you upgrade the instance if necessary.
- After the upgrade is complete, you can manually roll back the upgrade.
- A maximum of 30 instances can be selected at a time for batch upgrade.
- GaussDB can also automatically install hot patches for an instance after the instance is created or after a cold patch is installed for it. You can submit a service ticket to request this function at **Service Tickets > Create Service Ticket** in the upper right corner of the management console. Note that automatic hot patch update is not supported for new instances created using backups of existing instances.

Step 1: Perform a Pre-upgrade Check

Before an upgrade, check the instance status and whether monitoring metrics such as the CPU usage, memory usage, and disk usage of the instance are normal.

1. Check instance status.
   a. **Log in to the management console**.
   b. Click in the upper left corner and select a region and project.
   c. Click in the upper left corner of the page and choose **Databases** > **GaussDB**.
   d. On the **Instances** page, check whether the target instance is available.

      Figure 8-2 Instance status

      If the instance is in an abnormal state, contact Huawei Cloud technical support.
2. Check monitoring metrics.
   a. Click in the upper left corner of the page, and choose **Management & Governance** > **Cloud Eye**.
   b. In the navigation pane, choose **Cloud Service Monitoring** > **GaussDB**.
   c. On the **Cloud Service Monitoring** page, click the target instance to go to the metric monitoring page.
      - On the **DB Instance** tab, view the value of **Instance Disk Usage** to check whether the disk usage is insufficient.
      - On the **Node** tab, view the value of **CPU Usage** to check whether the CPU usage remains high for a long time.
      - On the **Node** tab, view the value of **Memory Usage** to check whether the memory usage increases sharply.

      If any of the metrics are abnormal, contact Huawei Cloud technical support.

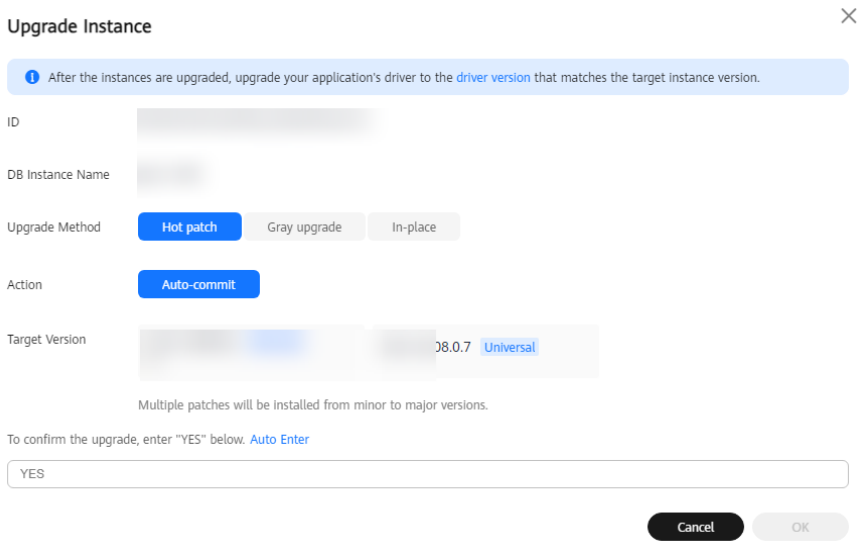
Step 2: Perform the Upgrade

[**Method 1: upgrading a single instance**]

1. Click in the upper left corner of the page and choose **Databases** > **GaussDB**.
2. On the **Instances** page, click **More** in the **Operation** column of the target instance and choose **Upgrade**.
3. In the **Upgrade Instance** dialog box, select **Hot patch** for **Upgrade Method**, enter **YES**, and click **OK**.

> **NOTE**
>
> All available patch versions are displayed in the **Target Version** area. If multiple patches are to be installed, they will be installed in ascending order of version numbers after the upgrade task is submitted.

Figure 8-3 Upgrading an instance

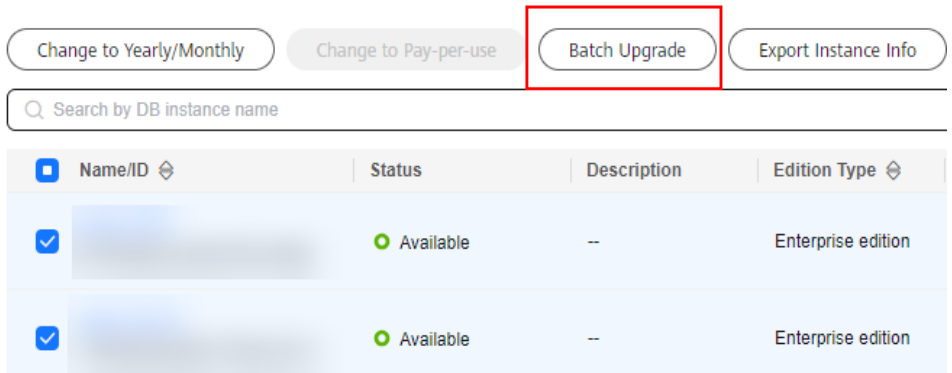


4. Check the upgrade result on the **Instances** page.
   - During the upgrade, the instance status is **Upgrading version**.
   - After the upgrade is complete, the instance status changes to **Available**.

[**Method 2: upgrading instances in batches**]

1. Click ☰ in the upper left corner of the page and choose **Databases** > **GaussDB**.
2. On the **Instances** page, select the target instances and click **Batch Upgrade**.

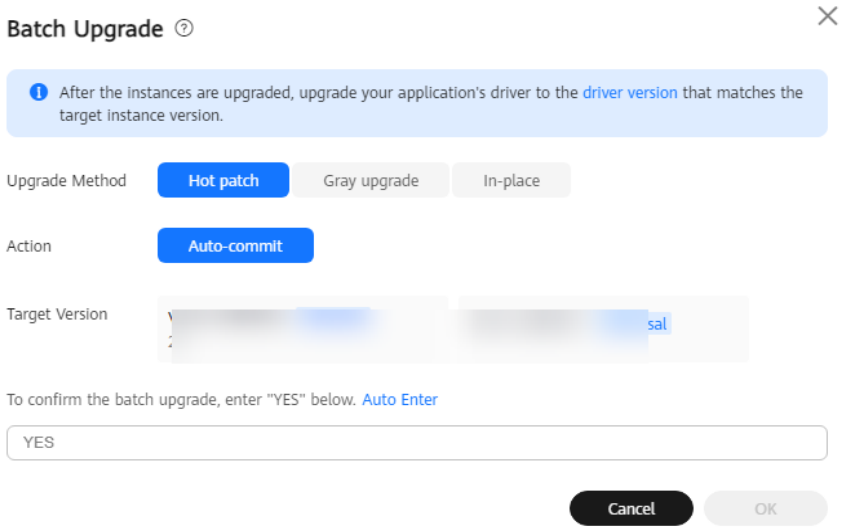
Figure 8-4 Batch upgrade



3. In the **Batch Upgrade** dialog box, select **Hot patch** for **Upgrade Method**.

4. Enter **YES** and click **OK**.

Figure 8-5 Auto-commit of a hot patch update



NOTE

All available patch versions are displayed in the **Target Version** area. If multiple patches are to be installed, they will be installed in ascending order of version numbers after the upgrade task is submitted.

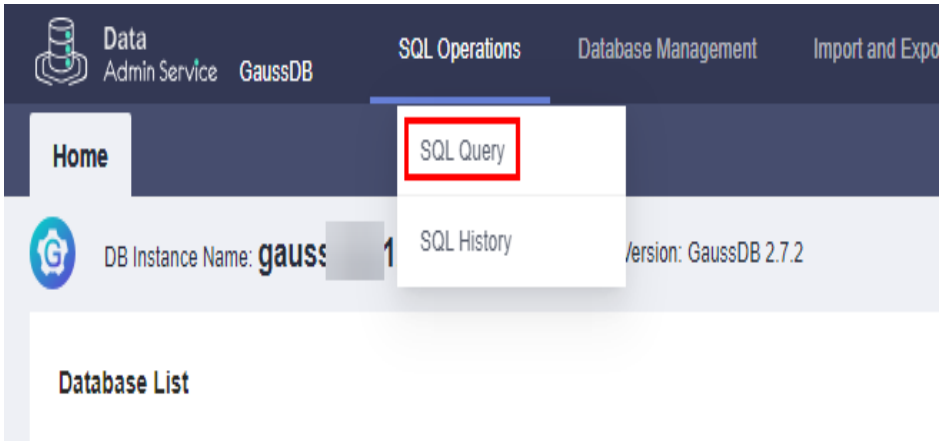
5. Check the upgrade result on the **Instances** page.
   - During the upgrade, the instance status is **Upgrading version**.
   - After the upgrade is complete, the instance status changes to **Available**.

Step 3: Verify the Upgrade

After the upgrade is complete, check the instance status, backup creation status, and instance connectivity, and whether you can add, delete, update, and query data in the instance.

1. On the **Instances** page, check whether **Status** of the target instance is **Available**.
2. On the **Instances** page, click the name of the target instance. On the **Basic Information** page that is displayed, check that the target versions are displayed in the **Upgraded Hot Patch Version** and **Upgraded Kernel Hot Patch Version** fields in the **Configuration** area.
3. Check that the instance is properly connected and you can add, delete, update, and query data in the instance.
   a. Log in to the database. For details, see **Connecting to an Instance Through DAS**.
   b. Go to the **SQL Query** page.

Figure 8-6 SQL query



c. Create a database.

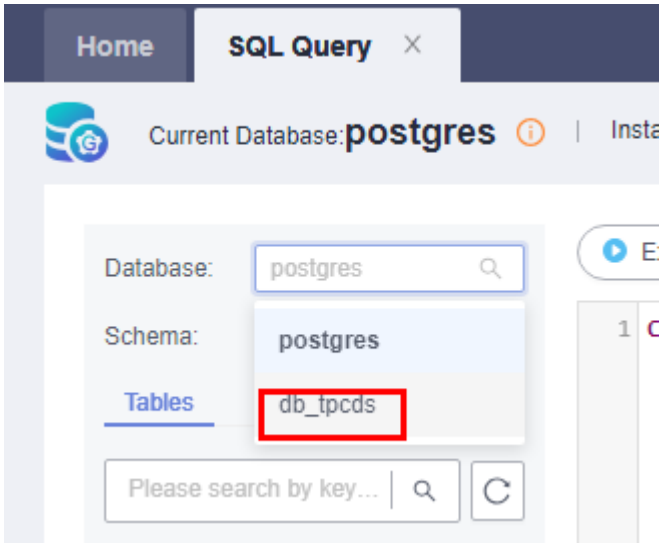
CREATE DATABASE *database_name*;

In this example, run the following command to create a database named **db_tpcds**:

CREATE DATABASE db_tpcds;

Switch to the newly created database in the upper left corner.

Figure 8-7 Switching to the new database



d. Create a table and add, delete, update, and query data in the table.

i. Create a schema.

CREATE SCHEMA *myschema*;

ii. Create a table named **mytable** that has only one column. The column name is **firstcol** and the column type is integer.

CREATE TABLE myschema.mytable *(firstcol int)*;

iii. Insert data into the table.

INSERT INTO myschema.mytable VALUES (100);

iv. View the data in the table.

SELECT * FROM myschema.mytable;

```
 | firstcol |
---+----------+
1 |   100    |
```

v. Update data in the table.

UPDATE myschema.mytable SET firstcol = 200;

vi. View the data in the table again.

SELECT * FROM myschema.mytable;

```
 | firstcol |
---+----------+
1 |   200    |
```

vii. Drop the table.

DROP TABLE myschema.mytable;

Rollback Operations

If a rollback is required after the upgrade, perform the following operations to roll back an instance to the source version.

NOTE

After the upgrade is rolled back, you can perform the upgrade again. If the problem persists, contact Huawei Cloud technical support, and Huawei Cloud engineers will help you upgrade the instance if necessary.

[Method 1: Rolling Back a Single Instance]

Step 1 In the **Upgrade Instance** dialog box, select **Hot patch** for **Upgrade Method**.

Step 2 Select **Rollback** for **Action**.

Step 3 Select the target version, enter **YES**, and click **OK**.

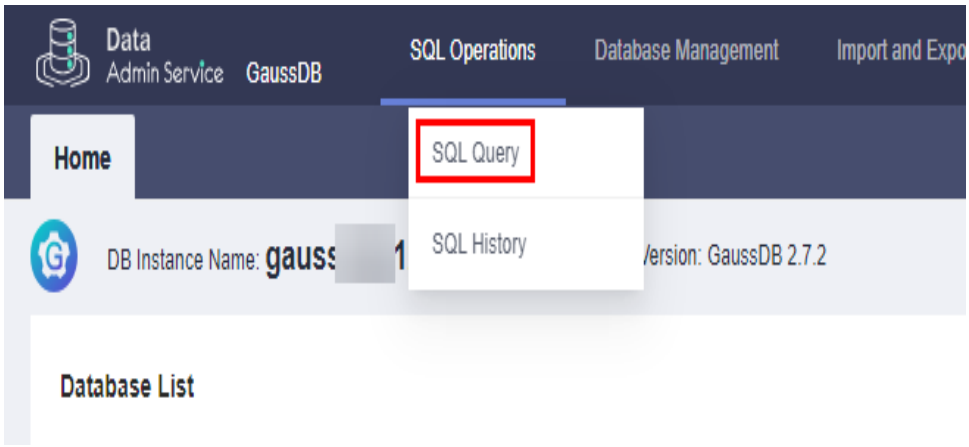
Step 4 On the **Instances** page, check the rollback status. After the rollback is complete, the instance status changes to **Available**.

Step 5 On the **Instances** page, click the name of the target instance. On the **Basic Information** page that is displayed, check that the target versions are not displayed in the **Upgraded Hot Patch Version** and **Upgraded Kernel Hot Patch Version** fields in the **Configuration** area.

Step 6 Check that the instance is properly connected and you can add, delete, update, and query data in the instance.

1. Log in to the database. For details, see **Connecting to an Instance Through DAS**.
2. Go to the **SQL Query** page.

Figure 8-8 SQL query



3. Create a database.

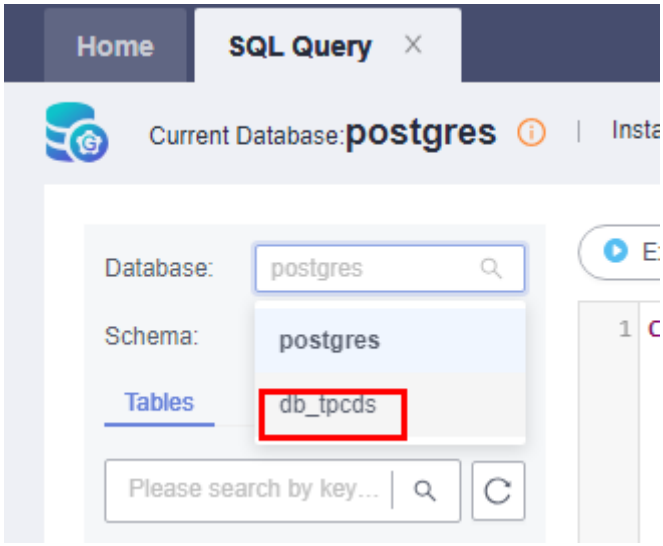
   CREATE DATABASE *database_name*;

   In this example, run the following command to create a database named **db_tpcds**:

   CREATE DATABASE db_tpcds;

   Switch to the newly created database in the upper left corner.

Figure 8-9 Switching to the new database



4. Create a table and add, delete, update, and query data in the table.

   a. Create a schema.

      CREATE SCHEMA *myschema*;

   b. Create a table named **mytable** that has only one column. The column name is **firstcol** and the column type is integer.

      CREATE TABLE myschema.mytable *(firstcol int)*;

   c. Insert data into the table.

      INSERT INTO myschema.mytable values (100);

   d. View the data in the table.

SELECT * FROM myschema.mytable;

```
 | firstcol |
---+----------+
1 |   100    |
```

e. Update data in the table.

UPDATE myschema.mytable SET firstcol = 200;

f. View the data in the table again.

SELECT * FROM myschema.mytable;

```
 | firstcol |
---+----------+
1 |   200    |
```

g. Drop the table.

DROP TABLE myschema.mytable;

----End

[Method 2: Rolling Back Instances in Batches]

Step 1 On the **Instances** page, select the target instances and click **Batch Upgrade**.

Step 2 In the **Batch Upgrade** dialog box, select **Hot patch** for **Upgrade Method** and **Rollback** for **Action**, select a target version, enter **YES**, and click **OK**.

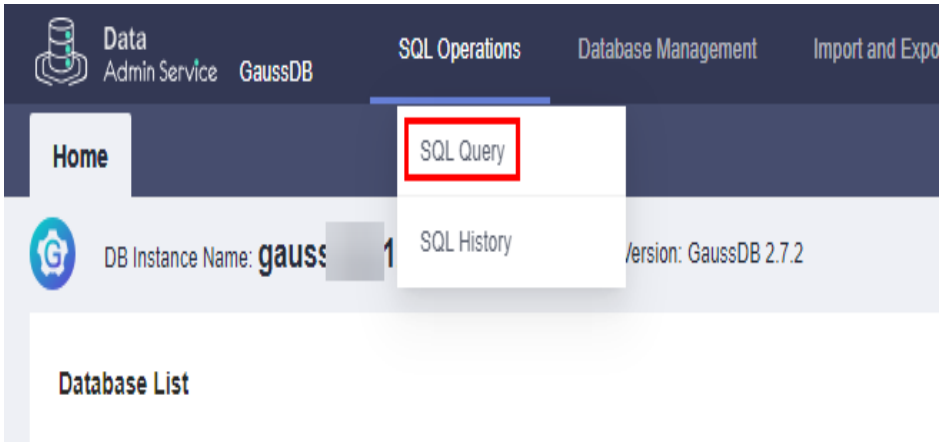
Step 3 On the **Instances** page, check the rollback status. After the rollback is complete, the instance status changes to **Available**.

Step 4 On the **Instances** page, click the name of the target instance. On the **Basic Information** page that is displayed, check that the target versions are not displayed in the **Upgraded Hot Patch Version** and **Upgraded Kernel Hot Patch Version** fields in the **Configuration** area.

Step 5 Check that the instance is properly connected and you can add, delete, update, and query data in the instance.

1. Log in to the database. For details, see **Connecting to an Instance Through DAS**.
2. Go to the **SQL Query** page.

Figure 8-10 SQL query



3. Create a database.

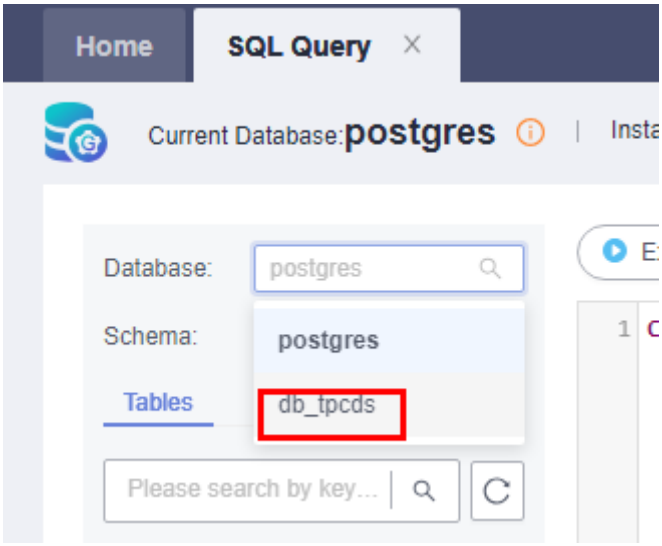
   CREATE DATABASE *database_name*;

   In this example, run the following command to create a database named **db_tpcds**:

   CREATE DATABASE db_tpcds;

   Switch to the newly created database in the upper left corner.

   Figure 8-11 Switching to the new database

4. Create a table and add, delete, update, and query data in the table.

   a. Create a schema.

      CREATE SCHEMA *myschema*;

   b. Create a table named **mytable** that has only one column. The column name is **firstcol** and the column type is integer.

      CREATE TABLE myschema.mytable *(firstcol int)*;

   c. Insert data into the table.

      INSERT INTO myschema.mytable VALUES (100);

   d. View the data in the table.

      SELECT * FROM myschema.mytable;

      ```
        | firstcol |
      ---+----------+
      1 |   100    |
      ```

   e. Update data in the table.

      UPDATE myschema.mytable SET firstcol = 200;

   f. View the data in the table again.

      SELECT * FROM myschema.mytable;

      ```
        | firstcol |
      ---+----------+
      1 |   200    |
      ```

   g. Drop the table.

      DROP TABLE myschema.mytable;

----End

8.3 In-place Upgrade

Scenarios

You can use in-place upgrade to upgrade your instance if a version upgrade is required for new functions or issue rectification. During an in-place upgrade, all nodes are upgraded at the same time, and all services are interrupted.

Precautions

- The DN disk usage cannot be greater than the configured disk usage threshold minus 10%.

  NOTE

  To check the current DN disk usage, go to the metric monitoring page on the management console.

  To obtain the disk usage threshold, contact technical support.
- Version upgrade is unavailable if instance nodes are in an abnormal state.
- During an upgrade, the following operations cannot be performed: scaling up storage, changing specifications, backing up data, resetting passwords, rebooting instances, and deleting instances.
- If this method is used for a major version upgrade, log archiving will be disabled before the upgrade, and you cannot use archive logs for Point-In-Time Recovery (PITR), which may result in data loss.

  NOTE

  Example major version upgrade: from V2.0-1.x to V2.0-2.x or from V2.0-2.x to V2.0-2.y
- If the upgrade fails, the system automatically rolls back the instance to the source version. You can contact Huawei Cloud customer service, and Huawei Cloud engineers will help you upgrade the instance if necessary.
- Services are interrupted for about 30 minutes during the in-place upgrade.
- After the upgrade is complete, an automated backup task will be triggered and log archiving will be enabled. However, for a single-replica instance upgraded to V2.0-3.0 or later from earlier versions, automated backup is disabled by default and will not be triggered. An automated backup task will also not be triggered in the case of minor version upgrades.

  NOTE

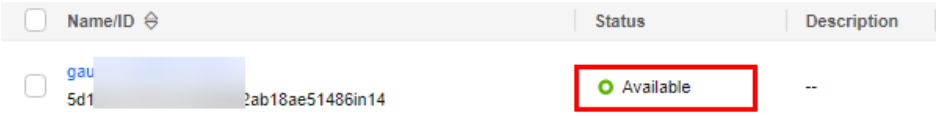
  Example minor version upgrade: from V2.0-1.a.x to V2.0-1.a.y or from V2.0-2.a.x to V2.0-2.a.y
- In-place upgrade does not require manual rollback.
- A maximum of 30 instances can be selected at a time for batch upgrade.

Step 1: Perform a Pre-upgrade Check

Before an upgrade, check the instance status and whether monitoring metrics such as the CPU usage, memory usage, and disk usage of the instance are normal.

1. Check instance status.
   a. **Log in to the management console**.
   b. Click in the upper left corner and select a region and project.
   c. Click in the upper left corner of the page and choose **Databases** > **GaussDB**.
   d. On the **Instances** page, check whether the target instance is available.

      Figure 8-12 Instance status

      If the instance is in an abnormal state, contact Huawei Cloud technical support.
2. Check monitoring metrics.
   a. Click in the upper left corner of the page, and choose **Management & Governance** > **Cloud Eye**.
   b. In the navigation pane, choose **Cloud Service Monitoring** > **GaussDB**.
   c. On the **Cloud Service Monitoring** page, click the target instance to go to the metric monitoring page.
      - On the **DB Instance** tab, view the value of **Instance Disk Usage** to check whether the disk usage is insufficient.
      - On the **Node** tab, view the value of **CPU Usage** to check whether the CPU usage remains high for a long time.
      - On the **Node** tab, view the value of **Memory Usage** to check whether the memory usage increases sharply.

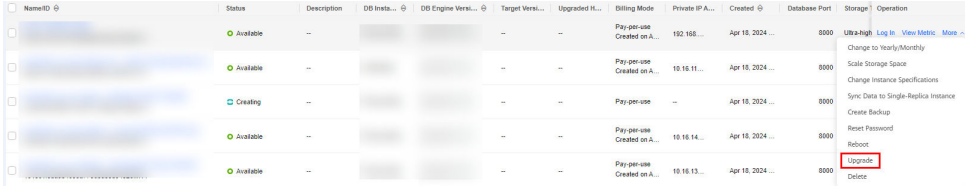
      If any of the metrics are abnormal, contact Huawei Cloud technical support.

Step 2: Perform the Upgrade

[Method 1: upgrading a single instance]

1. Click in the upper left corner of the page and choose **Databases** > **GaussDB**.
2. On the **Instances** page, click **More** in the **Operation** column of the target instance and choose **Upgrade**.

   Figure 8-13 Instances

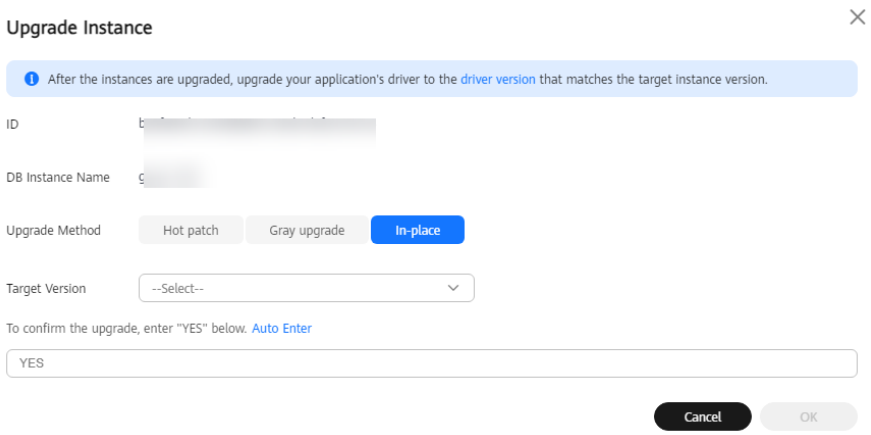
   

Alternatively, click the target instance name to go to the **Basic Information** page. In the **Configuration** area, click **Upgrade Instance** in the **DB Engine Version** field.

Figure 8-14 Basic information

3. In the **Upgrade Instance** dialog box, select **In-place** for **Upgrade Method**, select the target version, enter **YES**, and click **OK**.

Figure 8-15 Upgrading an instance

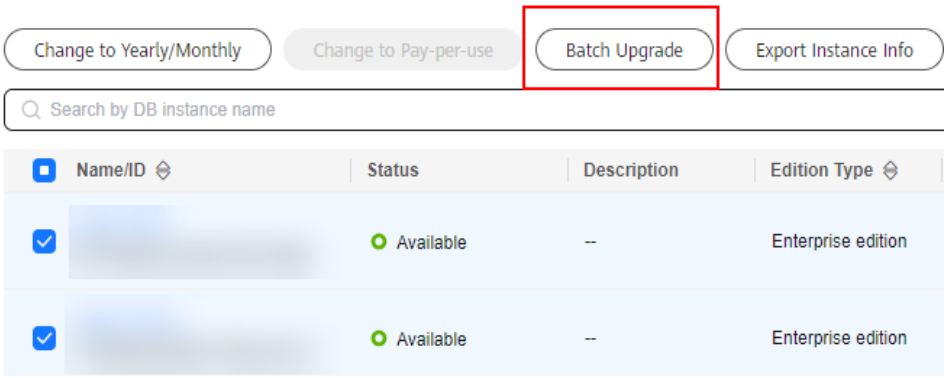


4. Check the upgrade result on the **Instances** page.
   - During the upgrade, the instance status is **Upgrading version**.
   - After the upgrade is complete, the instance status changes to **Available**.

[Method 2: upgrading instances in batches]

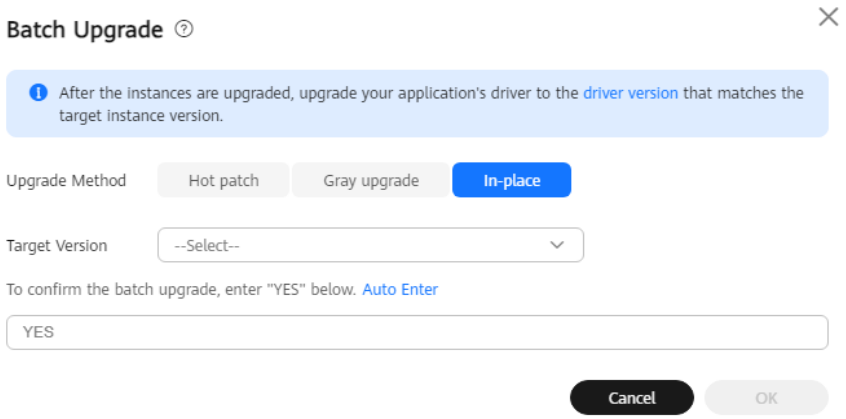
1. Click ☰ in the upper left corner of the page and choose **Databases** > **GaussDB**.
2. On the **Instances** page, select the target instances and click **Batch Upgrade**.

Figure 8-16 Batch upgrade



3. In the **Batch Upgrade** dialog box, select **In-place** for **Upgrade Method**, select the target version, enter **YES**, and click **OK**.

Figure 8-17 In-place upgrade



4. Check the upgrade result on the **Instances** page.
   - During the upgrade, the instance status is **Upgrading version**.
   - After the upgrade is complete, the instance status changes to **Available**.

Step 3: Verify the Upgrade

After the upgrade is complete, check the instance status, backup creation status, and instance connectivity, and whether you can add, delete, update, and query data in the instance.

1. On the **Instances** page, check whether **Status** of the target instance is **Available**.
2. On the **Instances** page, click the name of the target instance. On the **Basic Information** page that is displayed, check whether the value of **DB Engine Version** in the **Configuration** area is the target version.
3. Check that the automated backup triggered after the upgrade is successfully created.

a. On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

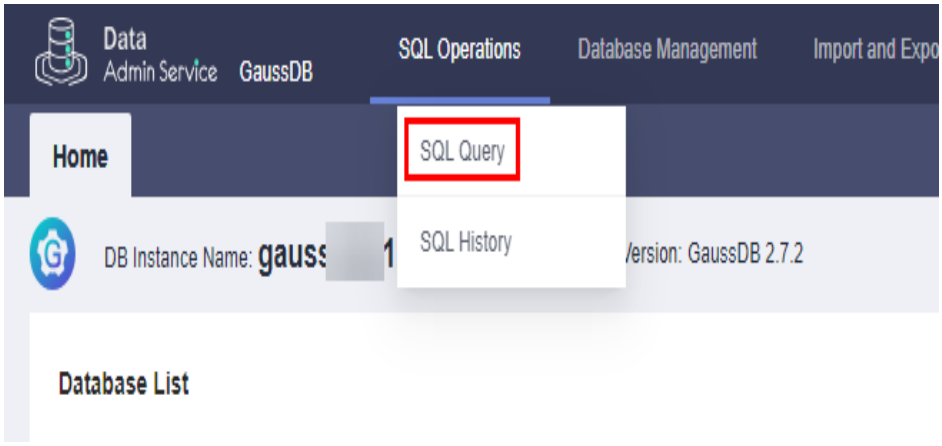
b. In the navigation pane, choose **Backups**. Check that a backup has been created and the backup status is **Completed**.

4. Check that the instance is properly connected and you can add, delete, update, and query data in the instance.

a. Log in to the database. For details, see **Connecting to an Instance Through DAS**.

b. Go to the **SQL Query** page.

Figure 8-18 SQL query



c. Create a database.

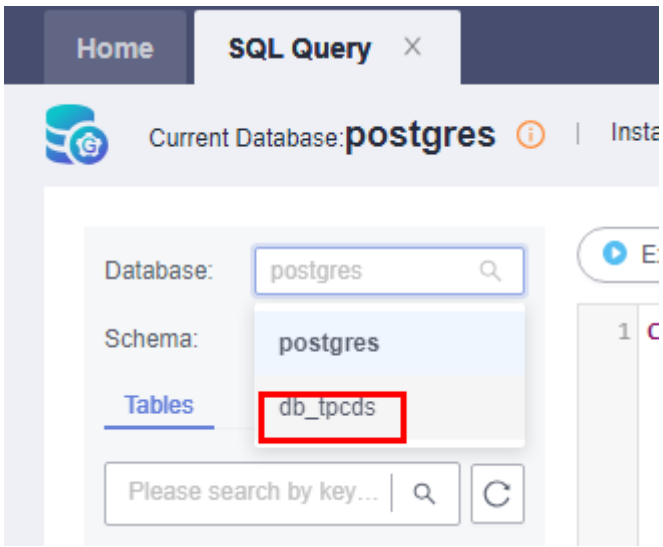
CREATE DATABASE *database_name*;

In this example, run the following command to create a database named **db_tpcds**:

CREATE DATABASE db_tpcds;

Switch to the newly created database in the upper left corner.

Figure 8-19 Switching to the new database



d. Create a table and add, delete, update, and query data in the table.

i. Create a schema.

CREATE SCHEMA *myschema*;

ii. Create a table named **mytable** that has only one column. The column name is **firstcol** and the column type is integer.

CREATE TABLE myschema.mytable *(firstcol int)*;

iii. Insert data into the table.

INSERT INTO myschema.mytable VALUES (100);

iv. View the data in the table.

SELECT * FROM myschema.mytable;

```
 | firstcol |
---+----------+
1 |   100    |
```

v. Update data in the table.

UPDATE myschema.mytable SET firstcol = 200;

vi. View the data in the table again.

SELECT * FROM myschema.mytable;

```
 | firstcol |
---+----------+
1 |   200    |
```

vii. Drop the table.

DROP TABLE myschema.mytable;

8.4 Gray Upgrade

Scenarios

You can use gray upgrade to upgrade your GaussDB instance if a version upgrade is required for new functions or issue rectification. You can either select auto-commit after the upgrade or perform a rolling upgrade.

- In the auto-commit mode, all standby DNs are upgraded first and then primary DNs and CNs in sequence. After the upgrade is complete, the upgrade is automatically committed.
- The rolling upgrade mode is also known as the upgrade observation mode. In this mode, the system enters the observation state after the upgrade is complete. During this period, you can observe the service status and either commit or roll back the upgrade based on service status.
  - Distributed instances are upgraded by shard. For details, see **Upgrading a Distributed Instance**.
  - Centralized instances are upgraded by AZ. For details, see **Upgrading a Centralized Instance**.

Operation Process

Step	Description
Step 1: Perform a Pre-upgrade Check	Before an upgrade, check the instance status and whether monitoring metrics such as the CPU usage, memory usage, and disk usage of the instance are normal.
Step 2: Perform the Upgrade	Select either auto-commit after the upgrade or perform a rolling upgrade. You can upgrade a single instance or multiple instances in batches as required.
Step 3: Verify the Upgrade	After the upgrade is complete, check the instance status, backup creation status, and instance connectivity, and whether you can add, delete, update, and query data in the instance.

Precautions

- The DN disk usage cannot be greater than the configured disk usage threshold minus 10%.

  NOTE

  To check the current DN disk usage, go to the metric monitoring page on the management console.

  To obtain the disk usage threshold, contact technical support.
- Version upgrade is unavailable if instance nodes are in an abnormal state.
- The rolling upgrade mode supports manual rollback, but the auto-commit mode does not support manual rollback.
- During an upgrade or rollback, the following operations cannot be performed: scaling up storage, changing specifications, backing up data, resetting passwords, rebooting instances, and deleting instances.
- You are advised to perform an upgrade during off-peak hours because there are more idle CPU, disk, and memory resources.
- If upgrade auto-commit is used for a major version upgrade, log archiving will be disabled before the upgrade, and you cannot use archive logs for PITR, which may result in data loss.
- If rolling upgrade is used for a major version upgrade, full backup cannot be triggered during the upgrade, and differential backup may fail. Manual full backups cannot be created until the upgrade operations in all AZs are complete during the rolling upgrade and observation period. Archive logs are still generated before the upgrade is committed, and you can use archive logs for PITR to prevent data loss. In the upgrade commit phase, log archiving is disabled.
- If the upgrade fails, the system automatically rolls back the instance to the source version. You can contact Huawei Cloud technical support, and Huawei Cloud engineers will help you upgrade the instance if necessary.
- Services are interrupted for about 10 seconds during the upgrade of primary DNs and during the upgrade of CNs.

- After the upgrade is complete, an automated backup task will be triggered and log archiving will be enabled. However, an automated backup task will not be triggered in the case of minor version upgrades.

NOTICE

Log archiving is available only for instances of versions later than V2.0-2.2.

Example minor version upgrade: from V2.0-1.a.x to V2.0-1.a.y or from V2.0-2.a.x to V2.0-2.a.y

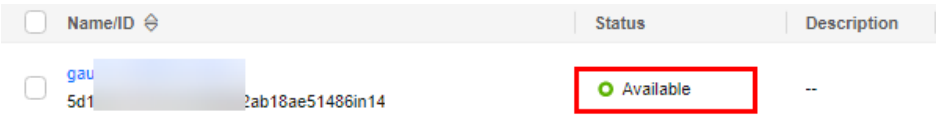
Example major version upgrade: from V2.0-1.x to V2.0-2.x or from V2.0-2.x to V2.0-2.y

Step 1: Perform a Pre-upgrade Check

Before an upgrade, check the instance status and whether monitoring metrics such as the CPU usage, memory usage, and disk usage of the instance are normal.

1. Check instance status.
   a. **Log in to the management console**.
   b. Click  in the upper left corner and select a region and project.
   c. Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.
   d. On the **Instances** page, check whether the target instance is available.

      Figure 8-20 Instance status

      If the instance is in an abnormal state, contact Huawei Cloud technical support.
2. Check monitoring metrics.
   a. Click  in the upper left corner of the page, and choose **Management & Governance** > **Cloud Eye**.
   b. In the navigation pane, choose **Cloud Service Monitoring** > **GaussDB**.
   c. On the **Cloud Service Monitoring** page, click the target instance to go to the metric monitoring page.
      - On the **DB Instance** tab, view the value of **Instance Disk Usage** to check whether the disk usage is insufficient.
      - On the **Node** tab, view the value of **CPU Usage** to check whether the CPU usage remains high for a long time.

- On the **Node** tab, view the value of **Memory Usage** to check whether the memory usage increases sharply.

If any of the metrics are abnormal, contact Huawei Cloud technical support.

Step 2: Perform the Upgrade

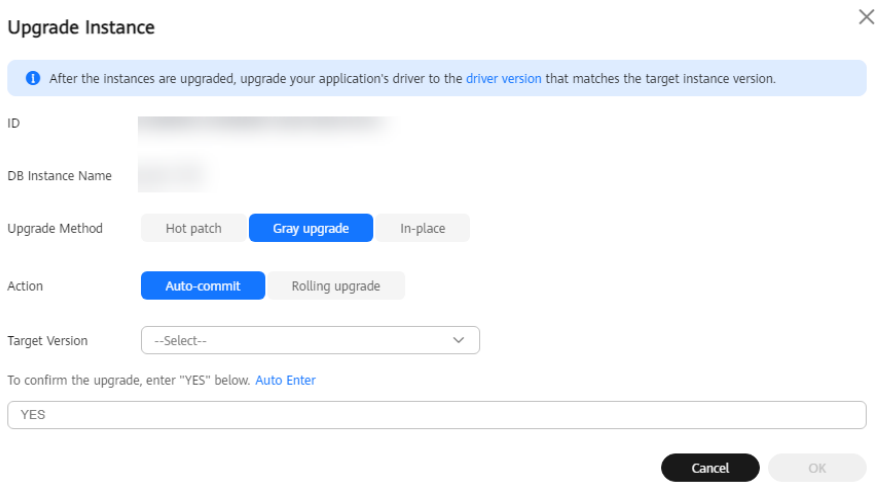
You can select auto-commit after the upgrade or perform a rolling upgrade for gray upgrade as required.

Upgrade Auto-commit

[Method 1: upgrading a single instance]

1. Click ☰ in the upper left corner of the page and choose **Databases** > **GaussDB**.
2. On the **Instances** page, click **More** in the **Operation** column of the target instance and choose **Upgrade**.
3. In the **Upgrade Instance** dialog box, select **Gray upgrade** for **Upgrade Method**.
4. Select **Auto-commit** for **Action**.

Figure 8-21 Upgrading an instance



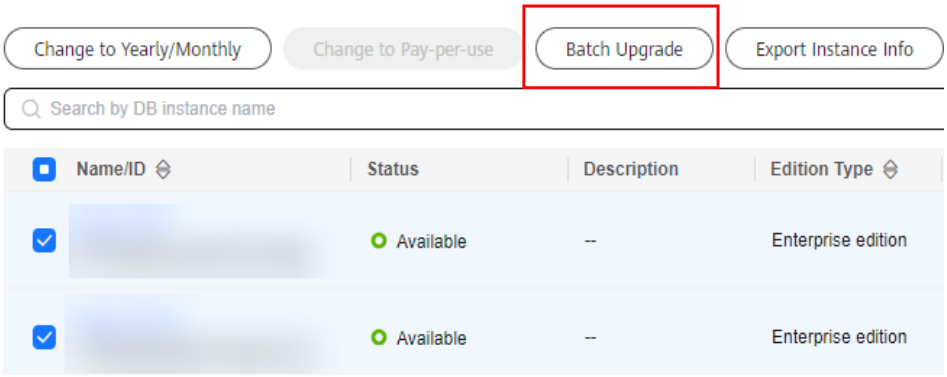
5. Select the target version, enter **YES**, and click **OK**.
6. Check the upgrade result on the **Instances** page.
   - During the upgrade, the instance status is **Upgrading version**.
   - After the upgrade is complete, the instance status changes to **Available**.

[Method 2: upgrading instances in batches]

1. Click ☰ in the upper left corner of the page and choose **Databases** > **GaussDB**.

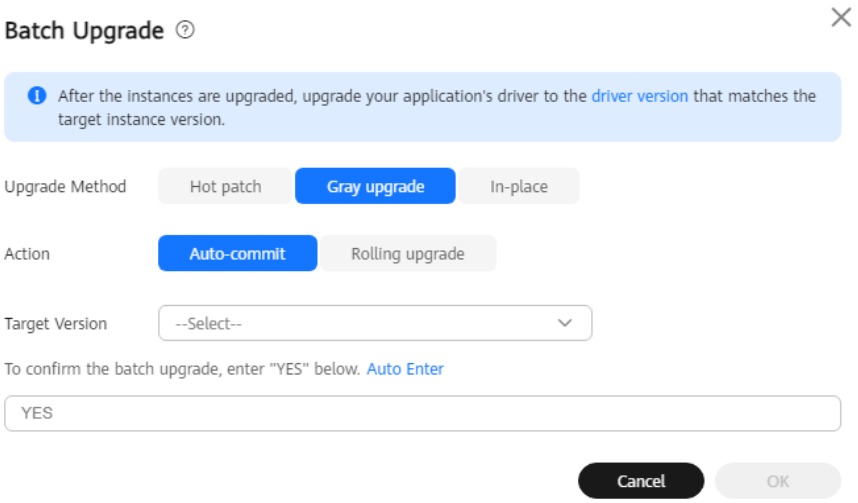
2. On the **Instances** page, select the target instances and click **Batch Upgrade**.

Figure 8-22 Batch upgrade



3. In the **Batch Upgrade** dialog box, select **Gray upgrade** for **Upgrade Method**.
4. Select **Auto-commit** for **Action**.
5. Select the target version, enter **YES**, and click **OK**.

Figure 8-23 Auto-commit of a gray upgrade



6. Check the upgrade result on the **Instances** page.
   - During the upgrade, the instance status is **Upgrading version**.
   - After the upgrade is complete, the instance status changes to **Available**.

Rolling Upgrade

[Method 1: upgrading a single instance]

- Upgrading a distributed instance

  a. Click ☰ in the upper left corner of the page and choose **Databases** > **GaussDB**.

b. On the **Instances** page, click **More** in the **Operation** column of the target instance and choose **Upgrade**.

Alternatively, click the target instance name to go to the **Basic Information** page. In the **Configuration** area, click **Upgrade Instance** in the **DB Engine Version** field.

Figure 8-24 Basic information

c. In the **Upgrade Instance** dialog box, select **Gray upgrade** for **Upgrade Method**.

d. Select **Rolling upgrade** for **Action**.

e. Set **Shards to Upgrade**, select a target version, enter **YES**, and click **OK**.

f. Check the upgrade result on the **Instances** page.

i. During the upgrade, the instance status is **Upgrading version**.

ii. After the upgrade is complete, the instance status changes to **Observing version upgrade**.

g. Check that all shards are upgraded and services are running properly before committing the upgrade.

In the **Upgrade Instance** dialog box, select **Commit** for **Action**, select a target version, enter **YES**, and click **OK**.

NOTICE

If you choose to upgrade shards one by one, repeat **b** to **f** until all shards are upgraded, and then commit the upgrade.

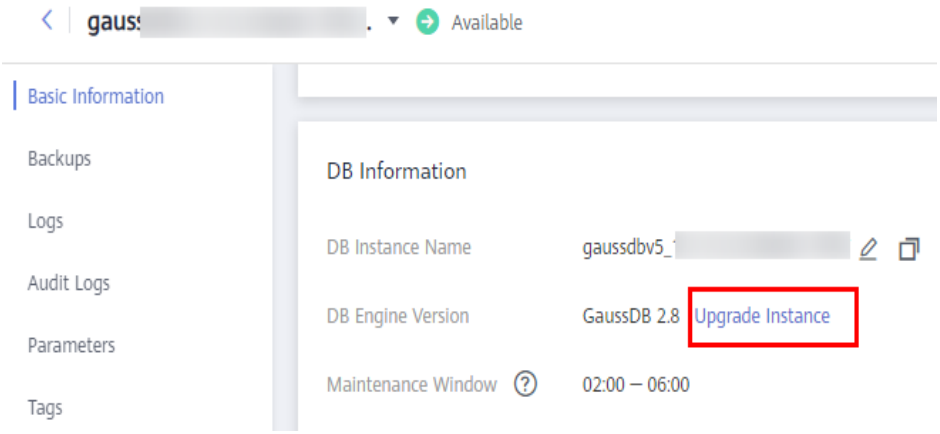
- Upgrading a centralized instance

a. Click ☰ in the upper left corner of the page and choose **Databases** > **GaussDB**.

b. On the **Instances** page, click **More** in the **Operation** column of the target instance and choose **Upgrade**.

Alternatively, click the target instance name to go to the **Basic Information** page. In the **Configuration** area, click **Upgrade Instance** in the **DB Engine Version** field.

Figure 8-25 Basic information



c. In the **Upgrade Instance** dialog box, select **Gray upgrade** for **Upgrade Method**.

d. Select **Rolling upgrade** for **Action**.

e. Set **AZs to Upgrade**, select a target version, enter **YES**, and click **OK**.

NOTE

You can upgrade a single AZ or multiple AZs at a time as needed.

f. Check the upgrade result on the **Instances** page.

i. During the upgrade, the instance status is **Upgrading version**.

ii. After the upgrade is complete, the instance status changes to **Observing version upgrade**.

g. Check that all AZs are upgraded and services are running properly before committing the upgrade.

In the **Upgrade Instance** dialog box, select **Commit** for **Action**, select a target version, enter **YES**, and click **OK**.

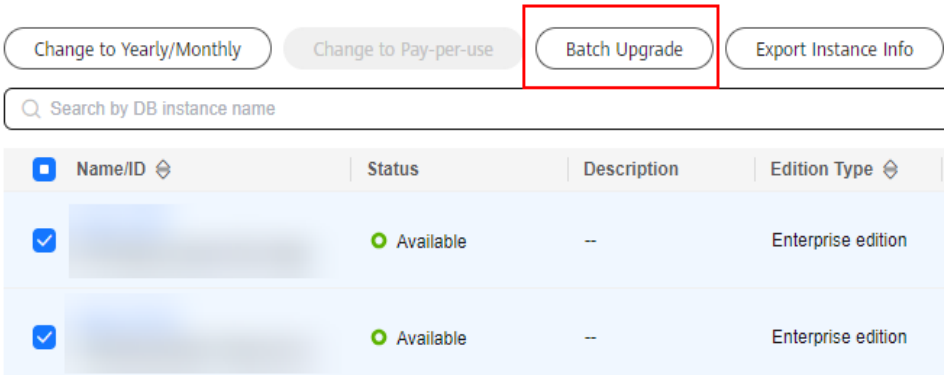
NOTICE

If you choose to upgrade AZs one by one, repeat **b** to **f** until all AZs are upgraded, and then commit the upgrade.

[Method 2: upgrading instances in batches]

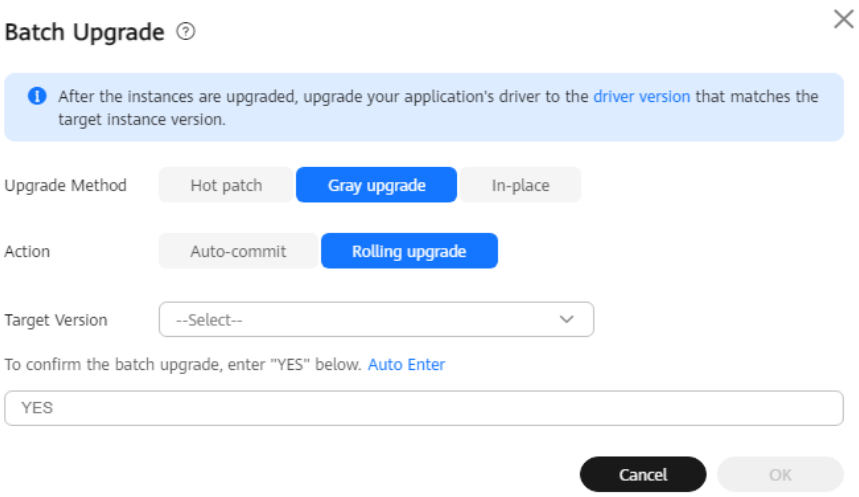
1. Click ☰ in the upper left corner of the page and choose **Databases** > **GaussDB**.
2. On the **Instances** page, select the target instances and click **Batch Upgrade**.

Figure 8-26 Batch upgrade



3. In the **Batch Upgrade** dialog box, select **Gray upgrade** for **Upgrade Method**.
4. Select **Rolling upgrade** for **Action**.
5. Select the target version, enter **YES**, and click **OK**.

Figure 8-27 Rolling upgrade of a gray upgrade



NOTE

During a batch upgrade in rolling upgrade mode, all AZs or shards of the selected instances are upgraded by default.

6. Check the upgrade result on the **Instances** page.
   - During the upgrade, the instance status is **Upgrading version**.
   - After the upgrade is complete, the instance status changes to **Observing version upgrade**.
7. Check that all shards or AZs are upgraded and services are running properly before committing the upgrade.

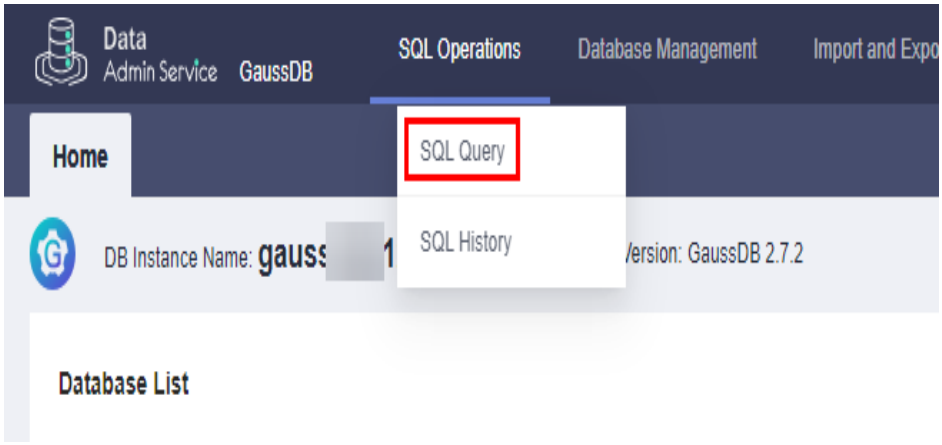
   In the **Batch Upgrade** dialog box, select **Commit** for **Action**, select a target version, enter **YES**, and click **OK**.

Step 3: Verify the Upgrade

After the upgrade is complete, check the instance status, backup creation status, and instance connectivity, and whether you can add, delete, update, and query data in the instance.

1. On the **Instances** page, check whether **Status** of the target instance is **Available**.
2. On the **Instances** page, click the name of the target instance. On the **Basic Information** page that is displayed, check whether the value of **DB Engine Version** in the **Configuration** area is the target version.
3. Check that the automated backup triggered after the upgrade is successfully created.
   a. On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.
   b. In the navigation pane, choose **Backups**. Check that a backup has been created and the backup status is **Completed**.
4. Check that the instance is properly connected and you can add, delete, update, and query data in the instance.
   a. Log in to the database. For details, see **Connecting to an Instance Through DAS**.
   b. Go to the **SQL Query** page.

   Figure 8-28 SQL query

   c. Create a database.

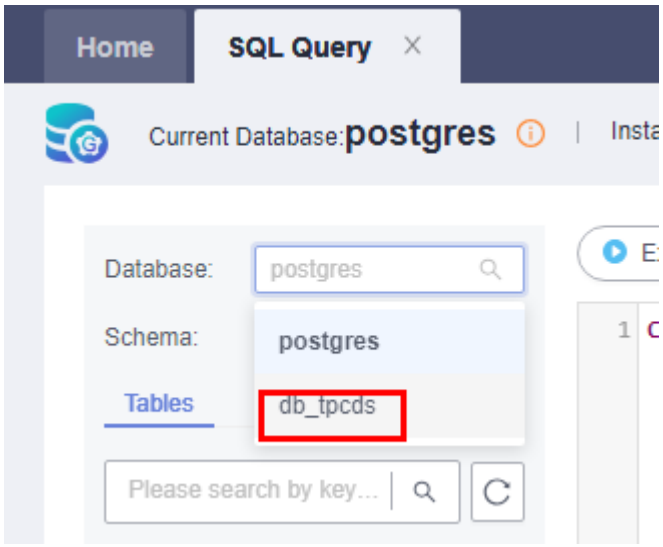
   CREATE DATABASE *database_name*;

   In this example, run the following command to create a database named **db_tpcds**:

   CREATE DATABASE db_tpcds;

   Switch to the newly created database in the upper left corner.

Figure 8-29 Switching to the new database



d. Create a table and add, delete, update, and query data in the table.

i. Create a schema.

CREATE SCHEMA *myschema*;

ii. Create a table named **mytable** that has only one column. The column name is **firstcol** and the column type is integer.

CREATE TABLE myschema.mytable *(firstcol int)*;

iii. Insert data into the table.

INSERT INTO myschema.mytable VALUES (100);

iv. View the data in the table.

SELECT * FROM myschema.mytable;

```
 | firstcol |
---+----------+
1 |   100    |
```

v. Update data in the table.

UPDATE myschema.mytable SET firstcol = 200;

vi. View the data in the table again.

SELECT * FROM myschema.mytable;

```
 | firstcol |
---+----------+
1 |   200    |
```

vii. Drop the table.

DROP TABLE myschema.mytable;

Rollback Operations

During upgrade observation, if a rollback is required due to service reasons or the upgrade using the rolling upgrade mode fails, you can manually roll back the upgrade by performing the steps in this section.

NOTE

- If the rollback is successful, you can perform the upgrade again.
- If the rollback fails, you can perform the rollback again.

If the problem persists, contact Huawei Cloud technical support, and Huawei Cloud engineers will help you upgrade the instance if necessary.

[Method 1: Rolling Back a Single Instance]

Step 1 In the **Upgrade Instance** dialog box, select **Rollback** for **Action**, select a target version, enter **YES**, and click **OK**.

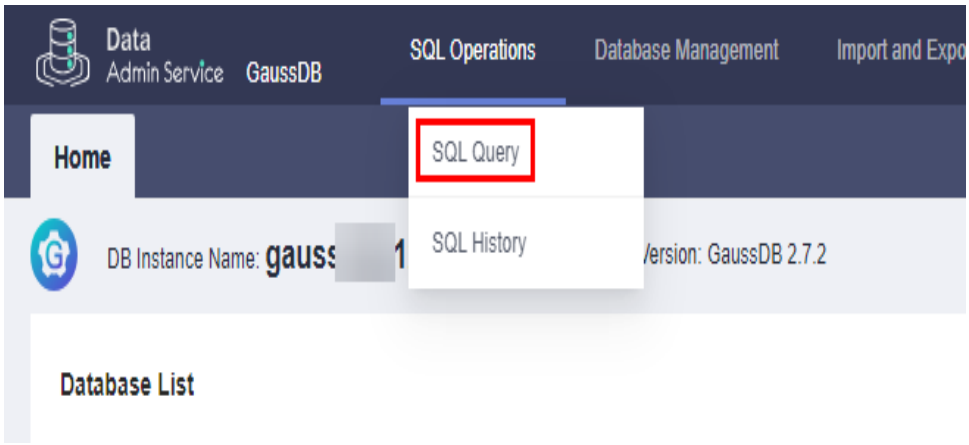
Step 2 On the **Instances** page, check the rollback status. After the rollback is complete, the instance status changes to **Available**.

Step 3 On the **Instances** page, click the name of the target instance. On the **Basic Information** page that is displayed, check that the value of **DB Engine Version** in the **Configuration** area is the source version, that is, the version before upgrade.

Step 4 Check that the instance is properly connected and you can add, delete, update, and query data in the instance.

1. Log in to the database. For details, see **Connecting to an Instance Through DAS**.
2. Go to the **SQL Query** page.

Figure 8-30 SQL query



3. Create a database.

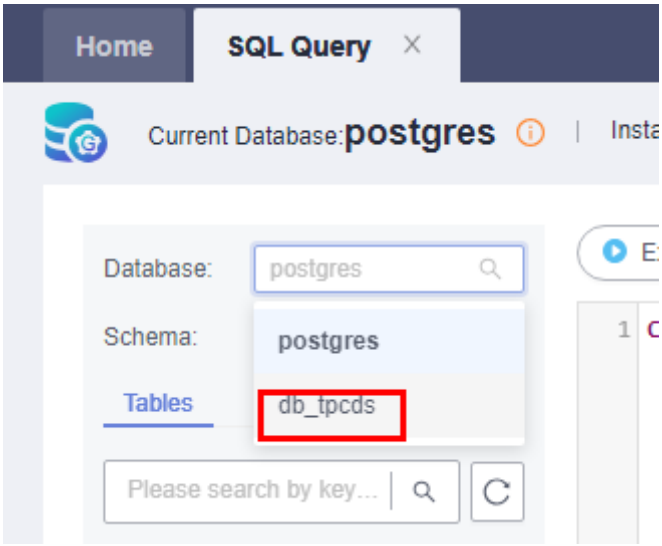
   CREATE DATABASE *database_name*;

   In this example, run the following command to create a database named **db_tpcds**:

   CREATE DATABASE db_tpcds;

   Switch to the newly created database in the upper left corner.

Figure 8-31 Switching to the new database



4. Create a table and add, delete, update, and query data in the table.
   a. Create a schema.
      CREATE SCHEMA *myschema*;
   b. Create a table named **mytable** that has only one column. The column name is **firstcol** and the column type is integer.
      CREATE TABLE myschema.mytable *(firstcol int)*;
   c. Insert data into the table.
      INSERT INTO myschema.mytable values (100);
   d. View data in the table.
      SELECT * FROM myschema.mytable;

```
 | firstcol |
---+----------+
1 |   100    |
```

   e. Update data in the table.
      UPDATE myschema.mytable SET firstcol = 200;
   f. View the data in the table again.
      SELECT * FROM myschema.mytable;

```
 | firstcol |
---+----------+
1 |   200    |
```

   g. Drop the table.
      DROP TABLE myschema.mytable;

----End

[Method 2: Rolling Back Instances in Batches]

Step 1 On the **Instances** page, select the target instances and click **Batch Upgrade**.

Step 2 In the **Batch Upgrade** dialog box, select **Rollback** for **Action**, select a target version, enter **YES**, and click **OK**.

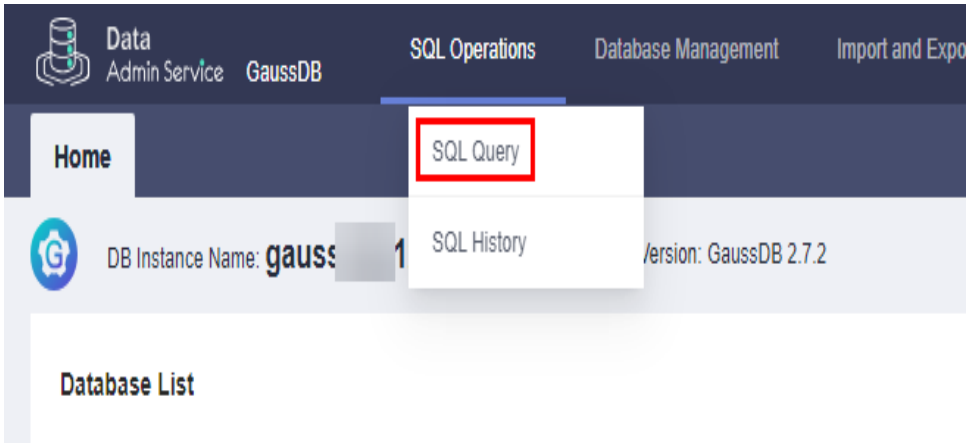
Step 3 On the **Instances** page, check the rollback status. After the rollback is complete, the instance status changes to **Available**.

Step 4 On the **Instances** page, click the name of the target instance. On the **Basic Information** page that is displayed, check that the value of **DB Engine Version** in the **Configuration** area is the source version, that is, the version before upgrade.

Step 5 Check that the instance is properly connected and you can add, delete, update, and query data in the instance.

1. Log in to the database. For details, see **Connecting to an Instance Through DAS**.
2. Go to the **SQL Query** page.

Figure 8-32 SQL query



3. Create a database.

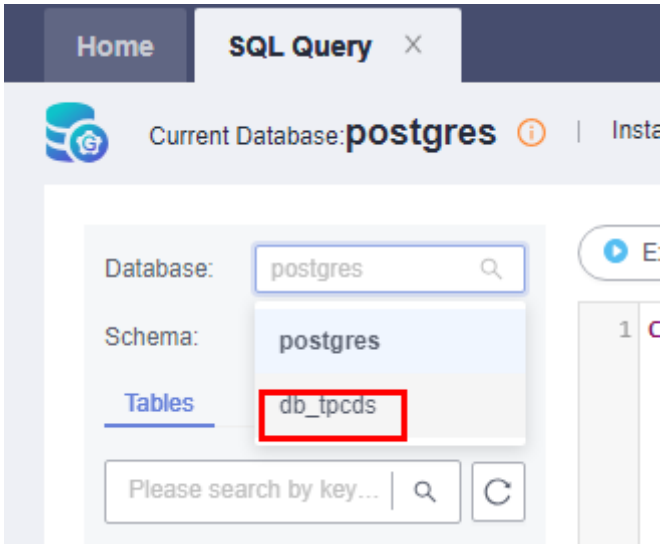
   CREATE DATABASE *database_name*;

   In this example, run the following command to create a database named **db_tpcds**:

   CREATE DATABASE db_tpcds;

   Switch to the newly created database in the upper left corner.

Figure 8-33 Switching to the new database



4. Create a table and add, delete, update, and query data in the table.
   a. Create a schema.

      CREATE SCHEMA *myschema*;

   b. Create a table named **mytable** that has only one column. The column name is **firstcol** and the column type is integer.

      CREATE TABLE myschema.mytable *(firstcol int)*;

   c. Insert data into the table.

      INSERT INTO myschema.mytable values (100);

   d. View data in the table.

      SELECT * FROM myschema.mytable;

      ```
        | firstcol |
      ---+----------+
      1 |   100    |
      ```

   e. Update data in the table.

      UPDATE myschema.mytable SET firstcol = 200;

   f. View the data in the table again.

      SELECT * FROM myschema.mytable;

      ```
        | firstcol |
      ---+----------+
      1 |   200    |
      ```

   g. Drop the table.

      DROP TABLE myschema.mytable;

----End

9 Plug-in Management

9.1 Installing a Plug-in

Scenarios

You can install kernel plug-ins for your GaussDB instances to enhance kernel functions. Only the PostGIS plug-in provided by Yukon is supported.

Prerequisites

The plug-in package has been uploaded to the OBS of the end tenant. For details about how to upload a plug-in package, see **Uploading an Object**.

Precautions

- During the installation, the instance will be rebooted, which will temporarily interrupt database services.
- Contact the third-party vendor to obtain the download URL, SHA-256 hash, and license information of the plug-in package.
- The plug-in installation requests can be submitted repeatedly. If a plug-in is installed for the first time, the license must be configured.
- After the **enable_default_ustore_table** parameter is set to **off**, the storage mode of new tables changes to Astore, but the storage mode of existing tables remains unchanged.

Constraints

- This function is available only for instances of version V2.0-8.100.0 or later.
- If the instance or node status is abnormal, the plug-in cannot be installed.
- The plug-in cannot be uninstalled after being installed.
- During plug-in installation, operations such as node repair and replacement, capacity expansion, and hot patch installation are not supported.
- Extensions can be enabled or disabled for a maximum of 200 databases in an instance and for a maximum of 10 databases at a time.

- Before using the PostGIS plug-in, check the value of the **behavior_compat_options** system. If the value contains **allow_procedure_compile_check**, **proc_implicit_for_loop_variable**, or **proc_outparam_override**, the plug-in may fail to be used. You can modify the parameter on the console. For details, see **Modifying GaussDB Instance Parameters**.
- Before using the PostGIS plug-in, check whether the values of **enable_default_ustore_table** and **forbid_public_funcname_same_with_sysfunc** are **off**. If their values are **on**, the plug-in cannot be used. You can change the parameter value to **off** on the console. For details, see **Modifying GaussDB Instance Parameters**.
- Currently, plug-ins cannot be installed for instances running Huawei Cloud EulerOS.
- Plug-ins cannot be installed for distributed instances.
- The PostGIS plug-in cannot be installed on primary and standby instances involved in a DR relationship.

Procedure

Step 1 **Log in to the management console**.

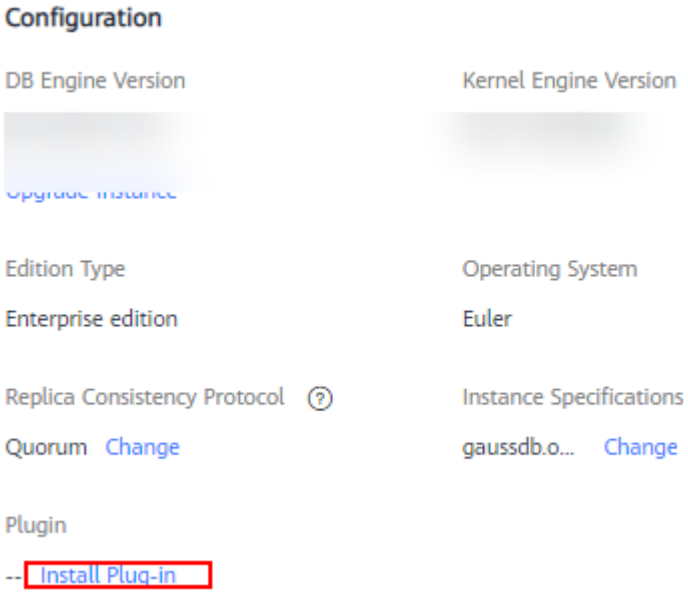
Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the **Configuration** area, click **Install Plug-in**.

Figure 9-1 Installing a plug-in



Step 6 Select the plug-in name, enter the correct license, download URL, and SHA-256 hash, and click **OK**.

Figure 9-2 Installing a plug-in

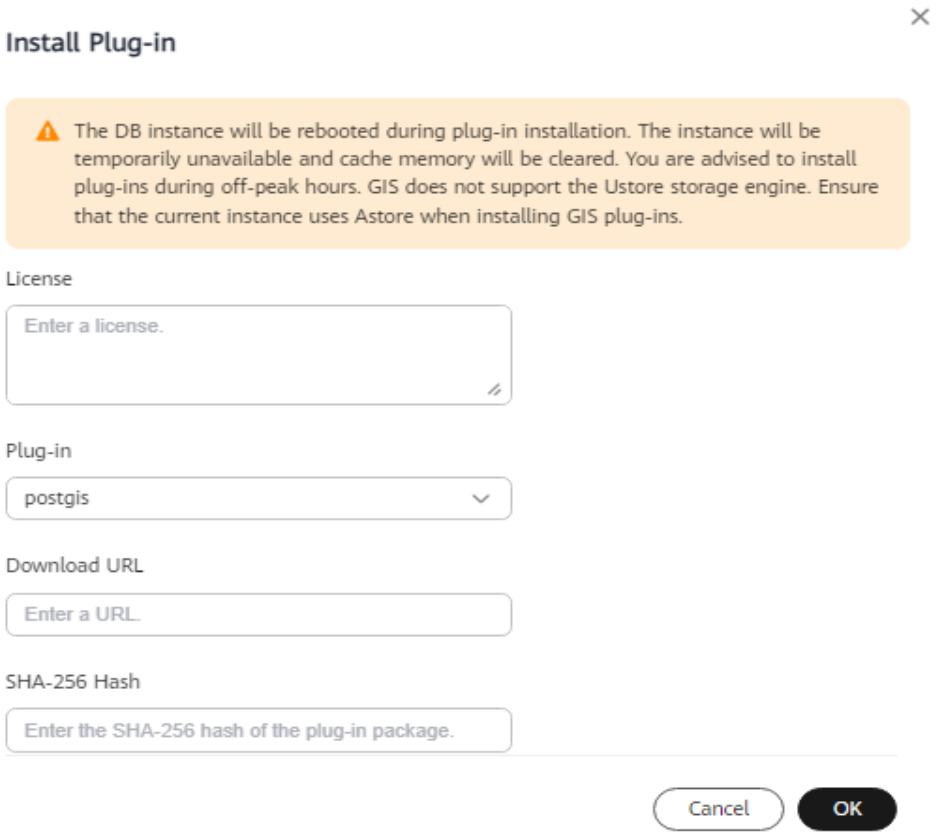


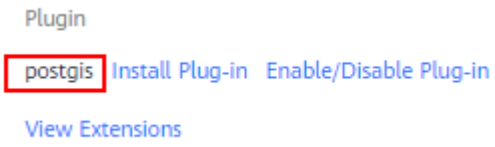
Table 9-1 Parameter description

Parameter	Description
License	Provide the ESN for the third-party vendor to obtain the license file. The ESN is the ID of the instance where the plug-in is to be installed.
Plug-in	Select the name of the plug-in to be installed. The default plug-in name is **postgis**.
Download URL	Enter the shared object URL of the plug-in package provided in OBS. For details, see **Accessing an Object Using Its URL**.
SHA-256 Hash	Enter the SHA-256 hash provided by the third-party vendor.

Step 7 Check the installation result.

During plug-in installation, the instance status is **Installing plug-in**. After the plug-in is installed, the instance status becomes **Available**. After the plug-in is installed, if the plug-in name is displayed in the **Plugin** field in the **Configuration** area, the plug-in is successfully installed.

Figure 9-3 Checking the installation result



----End

9.2 Enabling or Disabling a Plug-in

You can enable or disable plug-in extensions for a GaussDB instance. The PostGIS plug-in only supports the following extensions: **postgis**, **postgis_sfcgal**, **postgis_raster**, **yukon_geomodel**, **yukon_geogridcoder**, and **postgis_topology**.

Precautions

- The plug-in extension enabling or disabling requests can be submitted repeatedly.
- A single extension can be enabled or disabled for multiple user databases.
- When enabling plug-in extensions, enable **postgis** first and then the other extensions.

Constraints

- If the instance or node status is abnormal, the plug-in cannot be enabled or disabled.
- Before using the PostGIS plug-in, check whether the values of **enable_default_ustore_table** and **forbid_public_funcname_same_with_sysfunc** are **off**. If their values are **on**, the plug-in cannot be used. You can change the parameter value to **off** on the console. For details, see **Modifying GaussDB Instance Parameters**.

Procedure

Step 1 **Log in to the management console**.

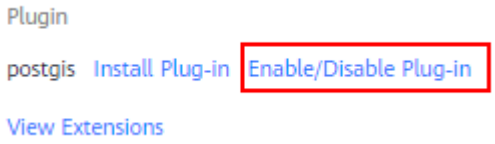
Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the **Configuration** area, click **Enable/Disable Plug-in**.

Figure 9-4 Enabling/Disabling a plug-in



Step 6 Set **User Database**, **Extension**, and **Enable/Disable Plug-in**, and click **OK**.

Table 9-2 Parameter description

Parameter	Description
User Database	Select one or more user-created service databases.
Extension	Six types of expansions are supported. The available extensions displayed vary from instance to instance. • postgis • postgis_sfcgal • postgis_raster • yukon_geomodel • yukon_geogridcoder • postgis_topology
Enable/Disable Plug-in	Select **Enable** or **Disable**.

----End

9.3 Viewing Extensions

After a plug-in is installed, you can check whether a plug-in extension is enabled or disabled for a specified user database.

Procedure

Step 1 Log in to the management console.

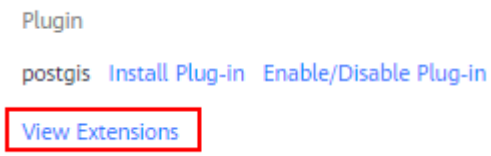
Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

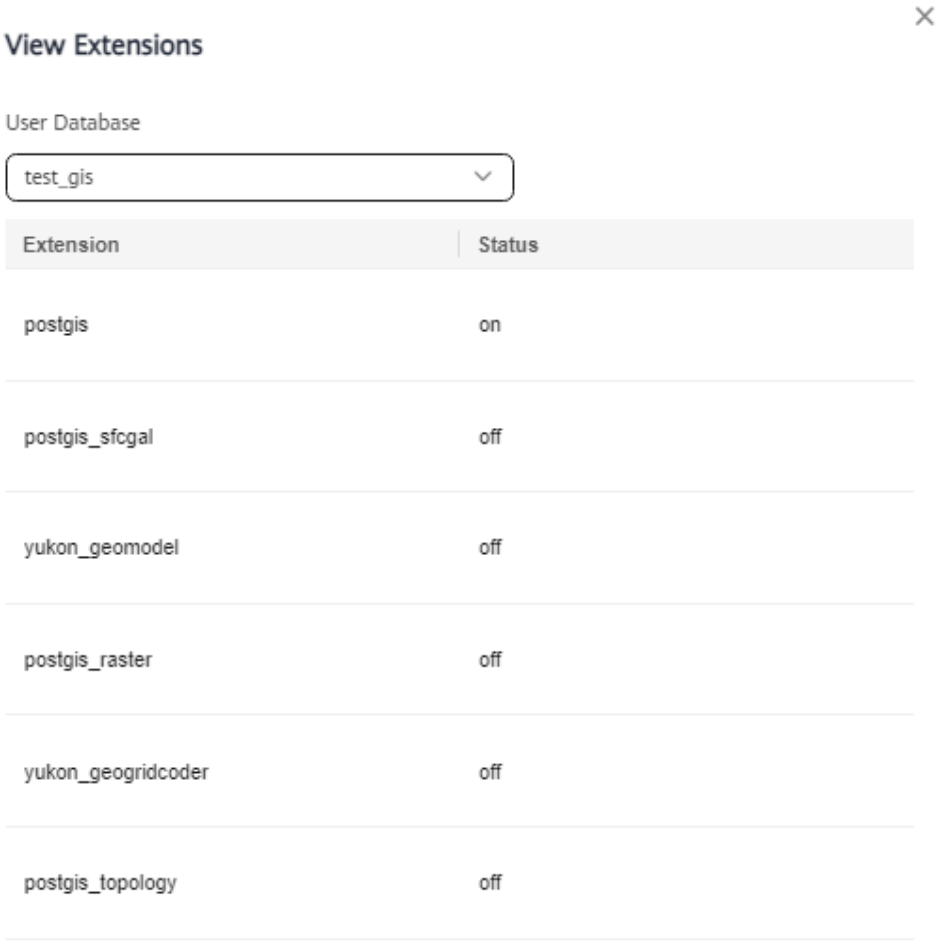
Step 5 In the **Configuration** area, click **View Extensions**.

Figure 9-5 Viewing extensions



Step 6 In the dialog box that is displayed, check whether an extension is enabled or disabled.

Figure 9-6 Viewing extensions



----End

10 Data Backup

10.1 Working with Backups

You can back up your GaussDB instances to ensure data reliability. Currently, backups are stored in an unencrypted form.

Backups are stored in OBS buckets.

In standard environments, 2 TB of data can be fully backed up and restored within 8 hours.

Precautions

Xlogs are not reclaimed during backup.

Functions

Although GaussDB supports high availability, if a database or table is maliciously or mistakenly deleted, data on the standby nodes is also deleted. In this case, you can only restore the deleted data from backups.

Full Backup

A full backup involves all data of a database at the backup point in time. The time required for full backup is long (in direct proportion to the total data volume of the database). You can use a full backup to restore data of a complete database. A full backup backs up all data even if the data has not changed since the last backup.

Differential Backup

A differential backup involves only incremental data modified after a specified time point. It takes less time than a full backup in direct proportion to how much data has changed (The total data volume is irrelevant). However, a differential backup cannot be used to restore all of the data of a database. By default, the system automatically backs up updated data every 30 minutes since the last automated backup. The backup period can be changed from 15 minutes to 1,440 minutes.

Table-Level Full Backup

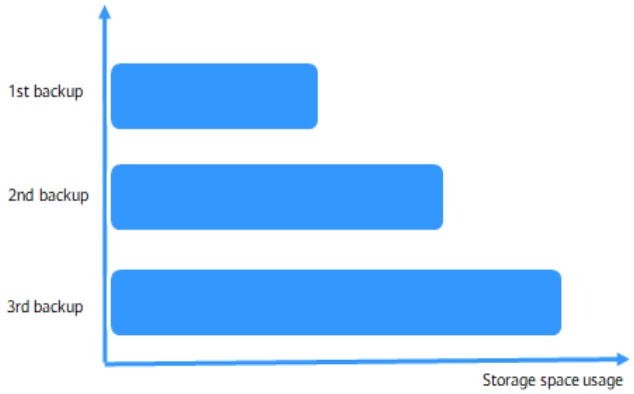
A table-level full backup involves all the data of a specified table at a specific point in time. The time required for table backup is long (in direct proportion to the total data volume of the table). You can use a full backup to restore all the data of a complete table. A table full backup backs up all data even if the data has not changed since the last backup.

Table-Level Differential Backup

A table differential backup involves only incremental data modified after a specified time point. It takes a short period of time (in direct proportion to the incremental data volume and irrelevant to the total data volume). However, a differential cannot restore all data of a table. By default, the system automatically backs up the updated data every 30 minutes since the last automated backup. The backup period can be changed from 15 minutes to 72 hours.

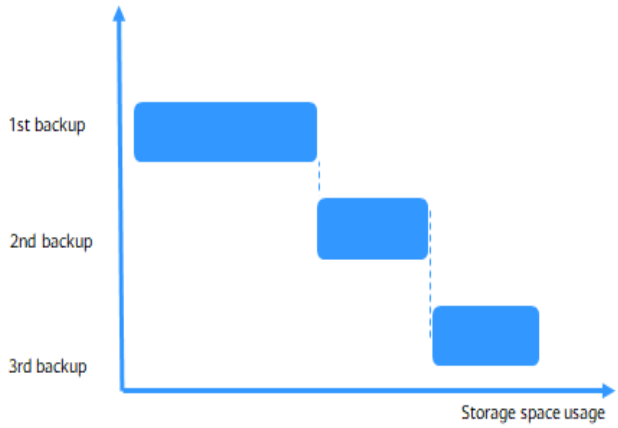
Backup Principles

Figure 10-1 Full backup



Full backup: After the first full backup, all data is backed up in the second and third backups regardless of whether the data is changed.

Figure 10-2 Differential backup



Differential backup: After the first full backup, the second backup backs up only the changed data, and the third backup backs up only the data changed after the second backup.

Automated Backup

Automated backups are created during the backup time window of your GaussDB instances. The system saves automated backups based on a retention period you specify. An automated backup is triggered after CNs or shards are added.

Table-Level Automated Backup

Automated backups of specified tables are created during the backup time window of your GaussDB instances. GaussDB saves automated table backups based on a retention period you specify. If necessary, you can restore the saved backups. If you have set a table-level backup policy, the system automatically backs up the table after CNs or shards are added.

Manual Backup

Manual backups are user-initiated full backups of instances. They are retained until you delete them manually.

Table-Level Manual Backup

Table-level manual backups are user-initiated full backups of instance tables. They are retained until you delete them manually.

10.2 Backup Execution

10.2.1 Configuring an Automated Backup Policy for GaussDB Instances

Scenarios

When you create a GaussDB instance, an instance-level automated backup policy is enabled by default. After your instance is created, you can modify the automated backup policy as needed. GaussDB backs up data based on the automated backup policy you specified.

If a database is faulty or data is damaged, you can restore it from backups to ensure data reliability. Backups are saved as packages in OBS buckets to ensure data confidentiality and durability. Since backing up data affects the database read and write performance, you are advised to perform automated backups during off-peak hours.

The automated backup policy is enabled by default as follows:

- Retention period: 7 days by default.
- Time window: An hour within 24 hours, such as 01:00-02:00 or 12:00-13:00. The backup time is in UTC format. If the DST or standard time is switched, the backup time segment changes with the time zone.

- Backup cycle: Monday to Sunday by default.
- Differential backup policy: Backup files are saved every 30 minutes by default.
- Backup flow control: The default value is **75 MB/s**.
- Prefetch pages: The default value is **64**.
- Standby node backup: This option is enabled by default.

NOTE

To ensure that data can be restored to a point in time, the latest full backup that exceeds the backup retention period will not be deleted immediately. For example, if **Backup Cycle** is set to **All** and **Retention Period** to one day and backup 1 is generated on November 1, this backup will not be deleted on November 2 when backup 2 is generated, but will be deleted on November 3 when backup 3 is generated.

Constraints

The instance-level automated backup policy cannot be configured for GaussDB single-replica instances of versions earlier than V2.0-3.0.

Billing

Backups are saved as packages in OBS buckets. For the billing details, see **How Is GaussDB Backup Data Charged?**

Modifying an Automated Backup Policy

Step 1 **Log in to the management console**.

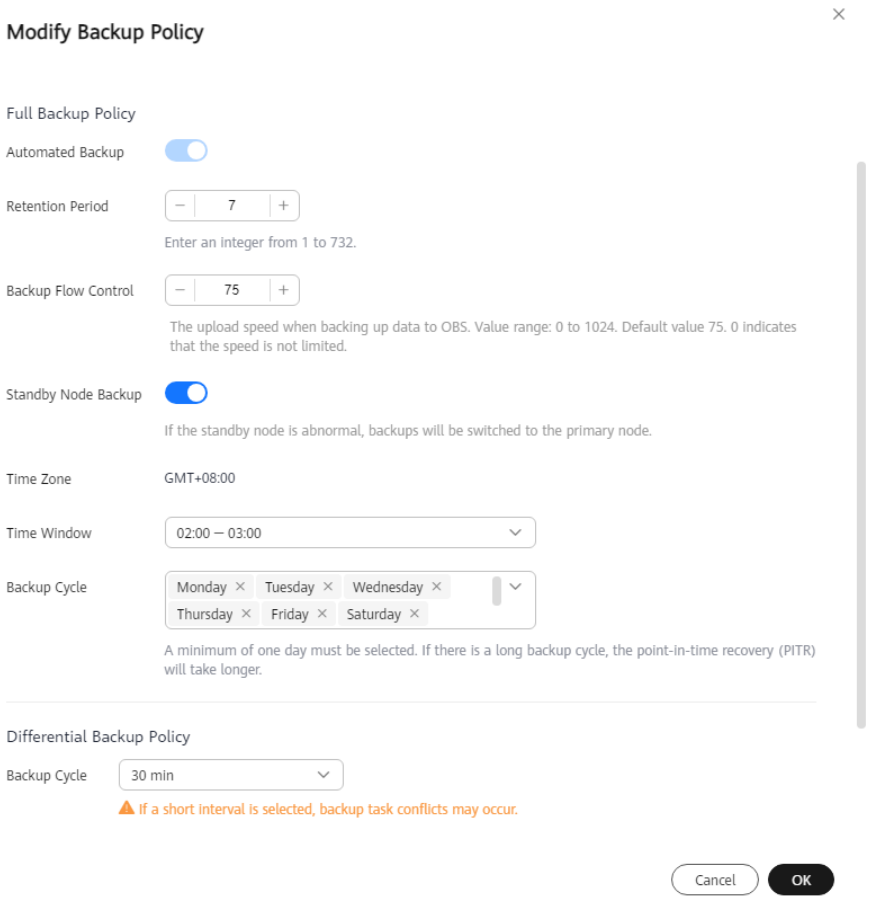
Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the navigation pane on the left, choose **Backups**. On the displayed page, click **Modify Backup Policy**. You can view the configured backup policy. To modify the backup policy, adjust the parameter values as needed.

Figure 10-3 Modifying the backup policy



Step 6 Configure parameters.

- Full backup policy:
  - **Retention Period**: Specify **Retention Period**, which indicates the number of days that your automated backups can be retained. Increasing the retention period will improve data reliability. The default value is **7**. However, even if the retention period has expired, the most recent backup will be retained.
    - Extending the retention period improves data reliability. You can extend the retention period as needed.
    - If you shorten the retention period, the new backup policy takes effect for existing backups. Any automated backups (including full and incremental backups) that have expired will be automatically deleted. Manual backups will not be automatically deleted but you can delete them manually.

    Policy for automatically deleting automated full backups:

    To ensure data integrity, even after the retention period expires, the most recent backup will be retained.

    If **Backup Cycle** was set to **Monday** and **Tuesday** and the **Retention Period** was set to **2**:

  - The full backup generated on Monday will be automatically deleted on Thursday. The reasons are as follows:

    The backup generated on Monday expires on Wednesday, but it was the last backup, so it will be retained until a new backup expires. The next backup will be generated on Tuesday and will expire on Thursday. So the full backup generated on Monday will not be automatically deleted until Thursday.

  - The full backup generated on Tuesday will be automatically deleted on the following Wednesday. The reasons are as follows:

    The backup generated on Tuesday will expire on Thursday, but as it is the last backup, it will be retained until a new backup expires. The next backup will be generated on the following Monday and will expire on the following Wednesday, so the full backup generated on Tuesday will not be automatically deleted until the following Wednesday.

- **Backup Flow Control**: Specify the rate at which data is uploaded from the data disk of the instance to the backup storage device (such as OBS). The default rate is 75 MB/s. The value **0** indicates that the upload rate is not limited. However, the actual upload rate is still restricted by factors such as the network, instance specifications, and disk I/O.
- **Standby Node Backup**: If this policy is enabled, full and differential backups of the instance are performed on the host where the standby DN resides.
- **Time Window**: An hour within 24 hours, such as 01:00-02:00 or 12:00-13:00. The backup time is in UTC format. If the DST or standard time is switched, the backup time segment changes with the time zone.
- **Backup Cycle**: Select at least one day from Monday to Sunday as required. By default, all options are selected.

  NOTE

  The backup retention period is from 1 to 732 days. To extend the retention period, contact technical support. Automated backups can be retained for up to 2,562 days.

  A time window is one hour. A total of 24 time windows are available. You are advised to select an off-peak time window for full backups. By default, each day of the week is selected for **Backup Cycle**. You can change the backup cycle. At least one day must be selected.

  A full backup is immediately triggered after a DB instance is created. Then, a full backup or differential backup is performed based on the time window and backup cycle you specified. We recommend that you set the automated backup time window to an off-peak hour.

- Differential backup policy:
  - **Backup Cycle**: Select the backup cycle for performing a differential backup. The default value is 30 minutes.
  - **Prefetch Pages**: Set the number of prefetch pages from the modified pages in the disk table file during a differential backup. The default value is **64**. When modified pages are adjacent (for example, with a bulk data load), you can set this parameter to a large value. When modified pages are scattered (for example, random update), you can set this parameter to a small value. If this parameter is set to a large value, the occupied I/O

increases. In this case, other services are affected and the database performance deteriorates.

Step 7 Click **OK**.

Step 8 Check the result.

After the task is submitted, click **Modify Backup Policy** to check whether the modification is successful.

----End

10.2.2 Configuring an Automated Backup Policy for GaussDB Tables

Scenarios

When you create a GaussDB instance, the table-level automated backup policy is disabled by default. After your instance is created, you can modify the automated backup policy for tables as needed. GaussDB backs up data of specified tables based on the automated backup policy you specified.

Backups are saved as packages in OBS buckets to ensure data confidentiality and durability. Since backing up data affects the database read and write performance, you are advised to perform automated backups for tables during off-peak hours.

Precautions

- After an instance-level restoration is complete, the databases or tables configured in the table-level automated backup policy may no longer exist. In this case, you need to configure the table-level automated backup policy again.
- To ensure that data can be restored to a point in time, the latest full backup that exceeds the backup retention period will not be deleted immediately. For example, if **Backup Cycle** is set to **All** and **Retention Period** to one day and backup 1 is generated on November 1, this table backup will not be deleted on November 2 when backup 2 is generated, but will be deleted on November 3 when backup 3 is generated.
- Tables in the **postgres**, **template0**, **template1**, **templatem**, **templatea**, and **templatepdb** system databases cannot be backed up using the table-level backup method.
- Tables in system schemas (for example, **public**) cannot be backed up using the table-level backup method.
- Tables in M-compatible databases cannot be backed up using the table-level backup method.
- A maximum of 100 databases or tables can be restored at the same time. If there are more than 100 databases or tables, you are advised to use instance-level backup and restoration.
- Table-level restoration does not support segment-page tables, column-store tables, tables containing user-defined types, synonym tables, temporary tables (including global tables), encrypted tables, TDE-enabled tables, unlogged tables, compressed tables, tables of private users, and tables of ledger databases.

- Table-level backup is unavailable to single-node instances.
- If the name of a table, schema, or database to be backed up is changed, you need to reconfigure the table-level backup policy.

Modifying an Automated Backup Policy for Tables

Step 1 **Log in to the management console**.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the navigation pane on the left, choose **Backups**. On the displayed page, choose **Full Backups** > **Table Backup**. Click **Modify Backup Policy**.

Step 6 Configure parameters.

- Full backup policy:

  Specify **Retention Period**, which indicates the number of days that your automated backups can be retained. Increasing the retention period will improve data reliability. The default value is **7**.

  Specify **Backup Cycle**. At least one day must be selected.

  NOTE

  If you shorten the retention period, the new backup policy takes effect for all backup files. The backup files that have expired will be deleted.

  The backup retention period is from 1 to 732 days. To extend the retention period, contact customer service. Automated backups can be retained for up to 2,562 days.

  The time window is one hour. You are advised to select an off-peak time window for full backups. By default, each day of the week is selected for **Backup Cycle**. You can change the backup cycle. At least one day must be selected.

  After a DB instance is created, a full backup for tables is not triggered. After a table backup policy is configured, full backups and differential backups for tables are performed based on the backup time window and backup cycle specified in the policy. We recommend that you set the full backup time window to off-peak hours.

  If a new table needs to be backed up, the new table will be backed up at the next automatic full backup or differential backup.

  If a table in the table-level backup policy is deleted, modify the table-level backup policy immediately to delete the deleted table. Otherwise, the table-level backup will always fail.

- Differential backup policy

  You need to select a backup cycle for differential table backups. 30 minutes is used by default.

  To disable the table-level differential backup policy, submit a request by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.

Step 7 Click **Modify Backup Policy**.

----End

10.2.3 Creating a Manual Backup for GaussDB Instances

Scenarios

GaussDB allows you to create instance-level manual backups for running instances. You can restore data using manual backups to ensure data reliability.

Precautions

- Manual backups are user-initiated full backups of instances. They are retained until you delete them manually.
- You can back up data of instances that are in the **Available** state.
- A user can perform only one instance-level backup operation for a DB instance at a time.
- Instance-level manual backups cannot be created for GaussDB single-replica instances of versions earlier than V2.0-3.0.

Billing

Backups are saved as packages in OBS buckets. For the billing details, see **How Is GaussDB Backup Data Charged?**

After a DB instance is deleted, the free backup space of the DB instance is automatically canceled. Manual backups are billed based on the space required. For details, see **Product Pricing Details**.

Method 1

Step 1 **Log in to the management console**.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, locate the instance and choose **More** > **Create Backup** in the **Operation** column.

Step 5 In the displayed dialog box, enter a backup name and enter the description as needed. Then, click **OK**. If you want to cancel the backup creation task, click **Cancel**.

To check whether the backup has been successfully created, click  in the upper right corner of the page. If the instance status is **Available**, the backup has been created. You can manage the backup by following the instructions provided in **Step 6**.

- The backup name must consist of 4 to 64 characters and start with a letter. It can contain only uppercase letters, lowercase letters, digits, hyphens (-), and underscores (_).
- The description can contain up to 256 characters, but cannot contain carriage returns and the following special characters: >!<"&'=

- During the creation process, the instance status is **Backing up**. The time required for creating a manual backup depends on the data volume.

Step 6 View and manage the created backup on the **Backups** page.

Alternatively, click the instance name. On the **Backups** page, you can view and manage the manual backups.

----End

Method 2

Step 1 **Log in to the management console**.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the navigation pane on the left, choose **Backups**. On the displayed page, click **Create Backup**.

Step 6 In the displayed dialog box, enter a backup name and description and click **OK**.

- The backup name must consist of 4 to 64 characters and start with a letter. It can contain only uppercase letters, lowercase letters, digits, hyphens (-), and underscores (_).
- The description can contain up to 256 characters, but cannot contain carriage returns and the following special characters: >!<"&'=
- During the creation process, the manual backup status is **Creating**. The time required for creating a manual backup depends on the data volume.

Step 7 View and manage the created backup on the current page.

Alternatively, go back to the instance list page, and click **Backups** to view and manage the backup.

----End

10.2.4 Creating a Manual Backup for GaussDB Tables

Scenarios

GaussDB allows you to create table-level manual backups for available instances. You can use these backups to restore data for ensuring data reliability.

NOTE

When a DB instance is deleted, the automated backups for tables are also deleted, but manual backups for tables are not deleted.

Precautions

- You can back up data of instances that are in the **Available** state.
- A user can perform only one table-level backup operation for a DB instance at a time.
- Keep your account balance above zero so that table-level backups can be created.
- Tables in the **postgres**, **template0**, **template1**, **templatea**, and **templatepdb** system databases cannot be backed up using the table-level backup method.
- Tables in M-compatible databases cannot be backed up using the table-level backup method.
- Tables in system schemas (for example, **public**) cannot be backed up using the table-level backup method.

NOTICE

By default, **public** is a system schema. However, you can run commands to change the owner of **public**. When the owner is changed to a non-system user, you can back up tables in the **public** schema.

Procedure

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the navigation pane on the left, choose **Backups**. On the displayed page, click **Table Backup** under **Full Backups** and click **Create Backup**.

Step 6 Enter the backup name and description, select data to be backed up, and click **Create Backup**.

- The backup name must consist of 4 to 64 characters and start with a letter. It can contain only uppercase letters, lowercase letters, digits, hyphens (-), and underscores (_).
- The description can contain up to 256 characters, but cannot contain carriage returns and the following special characters: >!<"&'=
- During the creation process, the status is **Creating**. The time required for creating a manual backup depends on the data volume.

Step 7 View and manage the created backup on the current page.

Alternatively, go back to the instance list page, and click **Backups** to view and manage the backup.

----End

10.3 Backup Management

10.3.1 Exporting Backup Information About GaussDB Instances

Scenarios

You can export backup information of instances to a CSV file for further analysis. The exported information includes the backup ID, backup name, instance name, instance ID, DB engine, backup type, backup start time, backup end time, backup status, backup size, and backup description.

Precautions

Backup information cannot be exported for GaussDB single-replica instances of versions earlier than V2.0-3.0.

Procedure

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 In the navigation pane on the left, choose **Backups**. Select the backups you want to export and click Export to export the backup information.

NOTE

Only the backup information displayed on the current page can be exported. The backup information displayed on other pages cannot be exported.

Alternatively, click the instance name. On the **Backups** page, select the backups you want to export and click **Export** above the backup list to export the backup information.

The exported backup information is in a CSV file which facilitates your analysis.

Step 5 View the exported backup information.

----End

10.3.2 Stopping a Backup for a GaussDB Instance

Scenarios

You can stop instance-level backup tasks as needed for your GaussDB instance, including automated full backups, manual full backups, and differential backups.

Precautions

Stopping a backup for an instance will stop all its ongoing full and differential backup tasks.

- Backups can be stopped only for instances whose DB engine version is V2.0-2.8 or later. You can view the backup stopping task information of an instance on the **Task Center** page.
- Backup tasks may fail to be stopped in the following scenarios:
  - The backup task is about to complete, that is, the backup is complete, but the status has not been updated. Wait until the backup task status changes to complete.
  - The backup task has just started executing and the backup process has not been started. In this case, try again later to stop the backup.
- You are advised not to stop the first automated backup after an instance is changed or restored. Forcibly stopping that backup may cause incremental and differential backups between the current time and the next automated full backup to fail, and point-in-time restoration may be unavailable. Stop that backup only when absolutely necessary.

Procedure

Step 1 Log in to the management console.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the navigation pane, choose **Backups**. On the displayed page, click **Stop Backup**.

Step 6 Click **OK**.

Step 7 Choose **Task Center** in the navigation pane on the left. On the displayed page, view the task details.

----End

10.3.3 Deleting a Manual Backup of a GaussDB Instance

Scenarios

You can delete manual backups for GaussDB instances to release storage space.

NOTICE

- Deleted manual backups cannot be recovered. Exercise caution when performing this operation.
- Automated backups cannot be manually deleted.
- Backups that are being restored cannot be deleted.
- To delete a backup, you must log in to the account that the backup belongs to.
- Manual backups cannot be deleted for GaussDB single-replica instances of versions earlier than V2.0-3.0.

Procedure

Step 1 Log in to the management console.

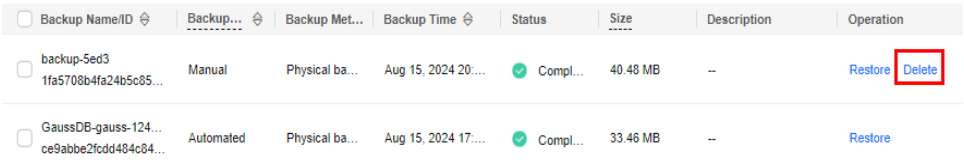
Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 In the navigation pane on the left, choose **Backups**. On the displayed page, locate the manual backup you want to delete and click **Delete** in the **Operation** column.

Alternatively, on the **Instances** page, click the instance name to go to the **Basic Information** page. In the navigation pane on the left, choose **Backups**. On the displayed page, locate the manual backup you want to delete and click **Delete** in the **Operation** column.

Figure 10-4 Deleting a manual backup



Step 5 Click **OK**.

After the backup is deleted, it will not be displayed on the **Backups** page.

Step 6 If you have enabled operation protection, click **Send Code** in the displayed **Identity Verification** dialog box and enter the obtained verification code. Then, click **OK**.

Two-factor authentication improves the security of your account. For details about how to view and enable high-risk operation protection, see ***Identity and Access Management User Guide***.

----End

11 Data Restoration

11.1 GaussDB Restoration Methods for Data Misoperations

You can choose an appropriate data restoration method based on the site requirements.

Table 11-1 Restoration methods for misoperations

Scenario	Recovery Method	Restoration Scope	Restore To	Operation Guide
An instance is deleted by mistake.	Recycle bin: Locate the deleted instance in the recycle bin and rebuild the instance to restore it.	All databases and tables	Current instance	**Rebuilding a GaussDB Instance**
	Instance backup: If a **manual backup** has been created before the instance is deleted, restore the instance on the **Backups** page.	All databases and tables	• A new instance • An existing instance • Current instance	• **Restoring a Backup File to a GaussDB Instance** • **Restoring a GaussDB Instance to a Specific Point in Time**

Scenario	Recovery Method	Restoration Scope	Restore To	Operation Guide
A table is deleted by mistake.	Restore the tables that are deleted by mistake by referring to the methods of restoring an instance.	All databases and tables	• A new instance • An existing instance • Current instance	• **Restoring a Backup File to a GaussDB Instance** • **Restoring a GaussDB Instance to a Specific Point in Time**
	Use the database and table restoration method to restore the table.	• All databases and tables • Certain databases and tables		• **Restoring a GaussDB Database or Table Using a Backup File** • **Restoring a GaussDB Database or Table to a Specific Point in Time**
A database is deleted by mistake.	Restore the databases that are deleted by mistake by referring to the methods of restoring an instance.	All databases and tables	• A new instance • An existing instance • Current instance	• **Restoring a Backup File to a GaussDB Instance** • **Restoring a GaussDB Instance to a Specific Point in Time**

Scenario	Recovery Method	Restoration Scope	Restore To	Operation Guide
	Use the database and table restoration method to restore the database.	• All databases and tables • Certain databases and tables		• **Restoring a GaussDB Database or Table Using a Backup File** • **Restoring a GaussDB Database or Table to a Specific Point in Time**
An entire table is overwritten, or the columns, rows, or data in a table is deleted or modified by mistake.	Restore the data that is deleted by mistake by referring to the methods of restoring an instance.	All databases and tables	• A new instance • An existing instance • Current instance	• **Restoring a Backup File to a GaussDB Instance** • **Restoring a GaussDB Instance to a Specific Point in Time**
	Use the database and table restoration method to restore table data.	• All databases and tables • Certain databases and tables		• **Restoring a GaussDB Database or Table Using a Backup File** • **Restoring a GaussDB Database or Table to a Specific Point in Time**

11.2 Restoring a Backup File to a GaussDB Instance

Scenarios

You can use an instance-level automated or manual backup to restore data to the point in time when the backup was created. The restoration is at the DB instance level.

Data can be restored to a new DB instance, an existing DB instance, or the original DB instance.

Constraints

- Restoration will fail if the instance is in the **Abnormal**, or **Storage full** state.
- GaussDB currently only supports restoration between DB instances running the same major version. For example, backup data can only be restored from version V2.0-1.4.*x* to version V2.0-1.4.*y*.
- Data cannot be restored to an existing instance for which a DR relationship has been established.

Procedure

Step 1 **Log in to the management console**.

Step 2 Click  in the upper left corner and select a region and project.

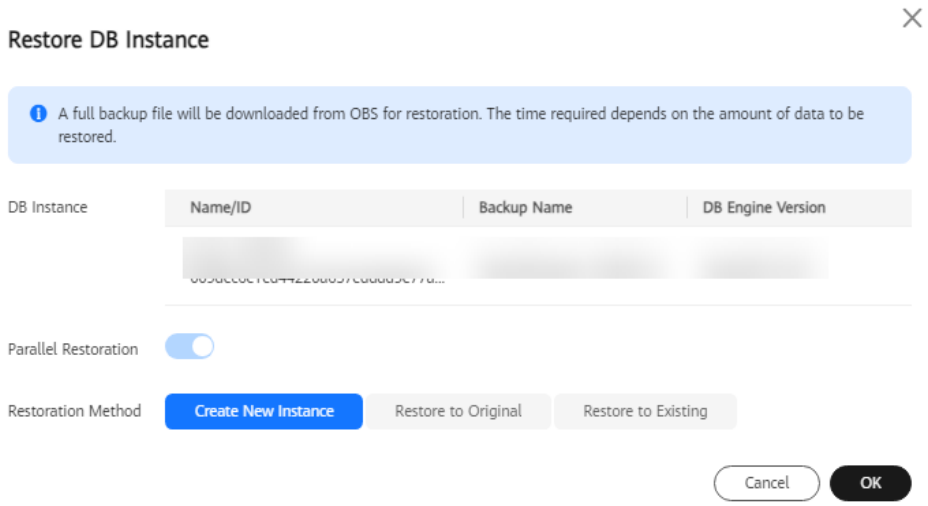
Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 In the navigation pane, choose **Backups**. On the **Backups** page, locate the backup to be restored and click **Restore** in the **Operation** column.

Alternatively, click the name of the target instance on the **Instances** page. In the navigation pane, choose **Backups**. On the **Full Backups** page, click the **Instance Backup** tab, and click **Restore** in the **Operation** column of the backup to be used for restoration.

Step 5 Click **OK**.

Figure 11-1 Restoring data from a backup



NOTE

- If parallel restoration is enabled, all replicas in shards download backup data from the OBS server at the same time during the restoration. Compared with serial restoration by default, parallel restoration requires *N* times as much bandwidth, where *N* is the number of replicas of each shard. If there is not enough available bandwidth, the restoration will slow down. If there are more than 5 shards in the instance to be restored, you are advised to consult the O&M personnel about the bandwidth available for the OBS server and if enabling parallel restoration is appropriate.
- Centralized instances support only parallel restoration.
- Parallel restoration cannot be enabled if the DB engine version is earlier than V2.0-1.4.
- In addition to full backups and incremental backups, the system also backs up incremental log files to ensure data consistency. It takes some time to back up and upload incremental log files (The time depends on the network and OBS traffic control). Note that the backup completion time does not represent the data consistency point that can be specified when this backup set is used to restore data. (Generally, the data consistency point is within several minutes before the backup completion time.) If you have strict requirements on data consistency after restoration, restore data to a specified point in time.

- Restoring data to a new DB instance:
  - The original and new DB instances must have the same major version. For example, backup data can only be restored from version V2.0-1.4.*x* to version V2.0-1.4.*y*.
  - The storage space of the new instance is the same as that of the original DB instance by default and the new instance must be at least as large as the original DB instance. The storage space for a single shard starts from 40 GB and can be increased at a step of 4 GB.
  - The administrator password needs to be reset.
  - By default, the instance specifications of the new instance are the same as those of the original instance. To change the instance specifications, ensure that the instance specifications of the new instance are at least those of the original instance.

  - If there are M-compatible databases in the instance to be restored, the **Tables Names in M-compatible Databases** parameter of the new instance must be set to the same value as that of the original instance. The default value is **Case-sensitive**.

  Configure the basic information about the new instance, click **Next**, and then click **Submit**.

- Restoring data to the original DB instance
  - The instance version and node configuration must be the same as those of the original DB instance.
  - Restoring to the original DB instance will overwrite all data on it and cause the DB instance to be unavailable during the restoration.
  - You are advised to manually back up data before the restoration.
  - If you use a backup created before advanced compression is enabled to restore data to the current instance, you must enable this feature for the instance again.
- Restoring data to an existing DB instance
  - Restoring to an existing DB instance will overwrite all data on it and cause the DB instance to be unavailable during the restoration.
  - Manually backing up data of the selected DB instance before the restoration.
  - If there are M-compatible databases in the instance to be restored, the **Tables Names in M-compatible Databases** parameter of the new instance must be set to the same value as that of the original instance. The default value is **Case-sensitive**.

Step 6 View the restoration results.

- Restoring data to a new DB instance

  A new instance is created using the backup data. When the instance status changes from **Creating** to **Available**, the restoration is complete.

  The new instance is independent from the original one.

- Restoring data to the original DB instance

  On the **Instances** page, the status of the DB instance changes from **Restoring** to **Available**. After the restoration is complete, an instance-level full backup will be automatically triggered.

  After the restoration is complete, check whether the restored data is consistent with the time point to which the data is restored.

- Restoring data to an existing DB instance

  On the **Instances** page, the status of the DB instance changes from **Restoring** to **Available**. After the restoration is complete, an instance-level full backup will be automatically triggered.

----End

11.3 Restoring a GaussDB Database or Table Using a Backup File

Scenarios

You can use an instance-level automated or manual backup to restore data of specified databases or tables to the point in time when the backup was created. This operation restores data in specified databases or tables.

You can use a table-level automated or manual backup to restore data in tables to the point in time when the backup was created. This operation restores data only in specified tables.

Data can be restored to a new DB instance, an existing DB instance, or the original DB instance.

Constraints

- Keep your account balance above zero so that backups can be restored to a new instance.
- Only DB instances of version V2.0-3.200 or later can be restored.
- Restoration will fail if the instance is in the **Abnormal**, or **Storage full** state.
- Backup data cannot be restored across major versions. For example, backup data can only be restored from version V2.0-3.200.*x* to version V2.0-3.200.*y*.
- A maximum of 100 databases or tables can be restored at the same time. If there are more than 100 databases or tables, you are advised to use instance-level restoration.
- The table names in a given database and schema as well as the database names must be different before and after the restoration.
- Table-level restoration does not support column-store tables, user-defined tables, synonym tables, temporary tables (including global tables), unlogged tables, tables of private users, and tables of ledger databases.
- Currently, only heap data, index data, and auto-increment column data related to tables can be restored. Other data related to tables, such as foreign key information, triggers, and SQL throttling rules, cannot be restored.
- After table-level restoration, row-level access control and dynamic masking information will be lost.
- System databases (**postgres**, **template0**, **template1**, **templatem**, **templatea**, and **templatepdb**) and their tables cannot be selected for database- and table-level restoration.
- The system schema (for example, **public**) and its tables cannot be selected for database- and table-level restoration.
- The M-compatible database and its tables cannot be selected for database- and table-level restoration.
- Single-replica instances do not support database- and table-level backup and restoration.
- Snapshot-based backups cannot be used to restore databases or tables.

- Data cannot be restored to an existing instance for which a DR relationship has been established.
- To enable the function of restoring databases and tables from backup files to existing instances, submit an application by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.

Procedure

Step 1 **Log in to the management console**.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 In the navigation pane, choose **Backups**. On the **Backups** page, locate the backup to be restored and click **Restore** in the **Operation** column.

- If the selected backup is an instance-level backup, you can restore data to specified databases or tables.
- If the selected backup is a table-level backup, you can only restore data to specified tables.

Alternatively, on the **Instances** page, click the instance name to go to the **Basic Information** page. In the navigation pane on the left, choose **Backups**. Click **Instance Backup** or **Table Backup** under **Full Backups**. On the displayed page, select the backup to be restored and click **Restore** in the **Operation** column.

Step 5 Click **OK**.

When you attempt to restore data to tables, a maximum of 200 tables can be displayed for **Backup Tables** by default. If the required table is not displayed, click the **+** icon in the upper right corner to add a custom table. (You can also use this method to add required tables if the instance is faulty or deleted.)

NOTE

In addition to table-level full backups and incremental backups, the system backs up incremental log files to ensure data consistency. It takes some time to back up and upload incremental log files (The time depends on the network and OBS traffic control). Note that the backup completion time does not represent the data consistency point that can be specified when this backup set is used to restore data. (Generally, the data consistency point is within several minutes before the backup completion time.) If you have strict requirements on data consistency after restoration, restore data to a specified point in time.

- Restoring data to a new DB instance:
  - The original and new DB instances must have the same major version. For example, backup data can only be restored from version V2.0-3.200.*x* to version V2.0-3.200.*y*.
  - The storage space of the new instance is the same as that of the original DB instance by default and the new instance must be at least as large as the original DB instance.
  - The administrator password needs to be reset.
  - By default, the specifications of the new instance are the same as those of the original instance. You can configure higher specifications for the

new instance. For instances of version V2.0-8.200 or later, an instance backup can be restored to an instance with lower specifications than the original instance. (Minimum specifications for the destination instance: 8 vCPUs | 32 GB)

- Parameters of the original instance will not be automatically restored to the new instance. To use the original parameter settings, select the required parameter template for **Parameter Template** when creating an instance for restoring data to a new instance. After the new instance is created, click the instance name and manually change the parameter values on the **Parameters** page.

Configure the basic information about the new instance and click **Apply Now**.

- Restoring data to the original DB instance
  - The instance version and node configuration must be the same as those of the original DB instance.
  - Databases and tables will be created on the target DB instance. During the restoration, the source database can be used properly.
  - You are advised to manually back up data before the restoration.
  - After a database (for example, **db1**) is restored to the original instance, you need to wait for the DB instance to automatically perform a full backup or differential backup before restoring the data of **db1** using other restoration processes. The time to wait depends on the backup policy. During point-in-time recovery, if you select a time point that is later than when the current database-level restoration is complete but earlier than when the next backup is performed, **db1** cannot be restored.
- Restoring data to an existing DB instance
  - Databases and tables will be created on the target instance. During the restoration, the databases can be used properly.
  - Manually backing up data of the selected DB instance before the restoration.

Step 6 View the restoration results.

- Restoring data to a new DB instance

  A new instance is created. The backup data is stored in the instance database or table. The instance status changes from **Creating** to **Available**.

  The new instance is independent from the original one.
- Restoring data to the original DB instance

  On the **Instances** page, the status of the DB instance changes from **Restoring** to **Available**.
- Restoring data to an existing instance

  On the **Instances** page, the status of the DB instance changes from **Restoring** to **Available**.

----End

11.4 Restoring a GaussDB Instance to a Specific Point in Time

Scenarios

You can use an instance-level automated backup to restore a GaussDB instance to a specified point in time.

You can restore backup data to the original GaussDB instance, an existing instance, or a new one.

Precautions

- Only instances of version V2.0-2.1 or later can be restored to any point in time. Single-replica instances are not supported.
- If nodes are being added, instance version is being upgraded, or data is being restored to an existing instance, the data cannot be restored a specific point in time.
- If a DB instance is faulty or a CN is removed, archive logs cannot be generated and the instance cannot be restored to a specific point in time.
- If you restore backup data to a new DB instance:
  - The DB engine and major version are the same as those of the original DB instance and cannot be changed.
  - The administrator password needs to be reset.
- If you restore backup data to the original DB instance, data on the original instance will be overwritten and the original DB instance will be unavailable during the restoration. Additionally, log archiving stops. After the restoration is complete, the **Confirm Data Integrity** button is displayed. Before clicking **Confirm Data Integrity**, you can restore data for multiple times. Once data integrity has been confirmed, any logs archived after the point in time data was restored from will be lost, but normal log archiving will be restored.
- When a DB instance is deleted, all archive logs are deleted by default and cannot be retained. After an instance is deleted, it cannot be rebuilt or restored to any point in time.

Procedure

Step 1 Log in to the management console.

Step 2 Click  in the upper left corner and select a region and project.

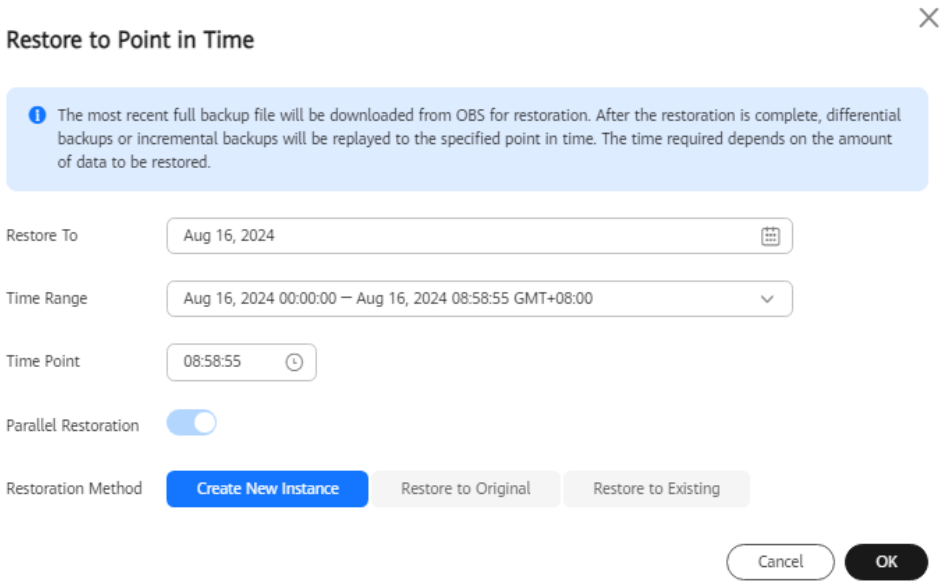
Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the navigation pane on the left, choose **Backups**. On the displayed page, click **Restore to Point in Time**.

Step 6 Click **OK**.

Figure 11-2 Restoring data to a specified point in time



NOTE

- If parallel restoration is enabled, all replicas in shards download backup data from the OBS server at the same time during the restoration. Compared with serial restoration by default, parallel restoration requires *N* times as much bandwidth, where *N* is the number of replicas of each shard. If there is not enough available bandwidth, the restoration will slow down. If there are more than 5 shards in the instance to be restored, you are advised to consult the O&M personnel about the bandwidth available for the OBS server and if enabling parallel restoration is appropriate.
- Centralized instances support only parallel restoration.
- Parallel restoration cannot be enabled if the DB engine version is earlier than V2.0-1.4.

- Restoring data to a new DB instance:
  - The original and new DB instances must have the same major version. For example, backup data can only be restored from version V2.0-1.4.*x* to version V2.0-1.4.*y*.
  - The storage space of the new instance is the same as that of the original DB instance by default and the new instance must be at least as large as the original DB instance.
  - The administrator password needs to be reset.
  - By default, the instance specifications of the new instance are the same as those of the original instance. To change the instance specifications, ensure that the instance specifications of the new instance are at least those of the original instance.
  - If there are M-compatible databases in the instance to be restored, the **Tables Names in M-compatible Databases** parameter of the new instance must be set to the same value as that of the original instance. The default value is **Case-sensitive**.

Configure the basic information about the new instance, click **Next**, and then click **Submit**.

- Restoring data to the original DB instance
  - The instance version and node configuration must be the same as those of the original DB instance.
  - Restoring to the original DB instance will overwrite all data on it and cause the DB instance to be unavailable during the restoration.
  - You are advised to manually back up data before the restoration.
  - If you use a backup created before advanced compression is enabled to restore data to the current instance, you must enable this feature for the instance again.
- Restoring data to an existing DB instance
  - Restoring to an existing DB instance will overwrite all data on it and cause the DB instance to be unavailable during the restoration.
  - Manually backing up data of the selected DB instance before the restoration.
  - If there are M-compatible databases in the instance to be restored, the **Tables Names in M-compatible Databases** parameter of the target instance must be set to the same value as that of the original instance. The default value is **Case-sensitive**.

Step 7 View the restoration results.

- Restoring data to a new DB instance

  A new instance is created using the backup data. When the instance status changes from **Creating** to **Available**, the restoration is complete.

  The new instance is independent from the original one.
- Restoring data to the original DB instance

  On the **Instances** page, the status of the DB instance changes from **Restoring** to **Available**. After the restoration is complete, a full backup will be automatically triggered.

  After the restoration is complete, check whether the restored data is consistent with the time point to which the data is restored.
- Restoring data to an existing DB instance

  On the **Instances** page, the status of the DB instance changes from **Restoring** to **Available**. After the restoration is complete, a full backup will be automatically triggered.

----End

11.5 Restoring a GaussDB Database or Table to a Specific Point in Time

Scenarios

You can use an instance-level automated backup to restore data in databases or tables to a specified point in time.

You can use a table-level automated backup to restore data in tables to a specified point in time.

You can restore backup data to the original GaussDB instance, an existing instance, or a new one.

Precautions

- To use this function, your DB instance cannot be a single-replica instance and its version must be V2.0-3.200 or later.
- If nodes are being added, instance version is being upgraded, or data is being restored to an existing instance, the data cannot be restored a specific point in time.
- If a DB instance is faulty or a CN is removed, archive logs cannot be generated and the instance cannot be restored to a specific point in time.
- If you restore backup data to a new DB instance:
  - The DB engine and major version are the same as those of the original DB instance and cannot be changed.
  - The administrator password needs to be reset.
- If you restore backup data to the original DB instance, a new database or table is created and the original database or table is still available. The archive logs are normal.
- When a DB instance is deleted, all archive logs are deleted by default and cannot be retained. After an instance is deleted, it cannot be rebuilt or restored to any point in time.
- The table names in a given database and schema as well as the database names must be different before and after the restoration. If they are same, you need to rename the tables and databases that the data is restored to.
- If **ALTER DATABASE SET TABLESPACE** and **ALTER TABLE SET TABLESPACE** are executed in the original instance, table-level data can be restored only after a differential backup or full backup is triggered in the original instance.
- A maximum of 100 databases or tables can be restored at the same time. If there are more than 100 databases or tables, you are advised to use instance-level restoration.
- Table-level restoration does not support column-store tables, user-defined tables, synonym tables, temporary tables (including global tables), unlogged tables, tables of private users, and tables of ledger databases.
- Currently, only heap data, index data, and auto-increment column data related to tables can be restored. Other data related to tables, such as foreign key information, triggers, and SQL throttling rules, cannot be restored.
- After table-level restoration, row-level access control and dynamic masking information will be lost.
- System databases (**postgres**, **template0**, **template1**, **templatem**, **templatea**, and **templatepdb**) and their tables cannot be selected for database- and table-level restoration.
- The system schema (for example, **public**) and its tables cannot be selected for database- and table-level restoration.
- The M-compatible database and its tables cannot be selected for database- and table-level restoration.

- Single-replica instances do not support database- and table-level backup and restoration.
- PITR based on snapshots does not support database- or table-level restoration.
- Data cannot be restored to an existing instance for which a DR relationship has been established.
- To apply for the permission needed for restoring data to existing instances, submit an application by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.

Procedure

Step 1 **Log in to the management console**.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the navigation pane, choose **Backups**. On the displayed page, click the **Instance Backup** or **Table Backup** tab.

- On the **Instance Backup** page, click **Restore to Point in Time** to restore data of specified databases or tables.
- On the **Table Backup** page, click **Restore to Point in Time** to restore data of specified tables.

Step 6 Click **OK**.

When **Restore Data From** is set to **Tables**, a maximum of 200 tables can be displayed for **Backup Tables** by default. If the required table is not displayed, click the **+** icon in the upper right corner to add a custom table. (You can also use this method to add required tables if the instance is faulty or deleted.)

- Restoring data to a new DB instance:
  - The original and new DB instances must have the same major version. For example, backup data can only be restored from version V2.0-3.200.*x* to version V2.0-3.200.*y*.
  - The storage space of the new instance is the same as that of the original DB instance by default and the new instance must be at least as large as the original DB instance.
  - The administrator password needs to be reset.
  - By default, the instance specifications of the new instance are the same as those of the original instance. To change the instance specifications, ensure that the instance specifications of the new instance are at least those of the original instance.
  - Parameters of the original instance will not be automatically restored to the new instance. To use the original parameter settings, select the required parameter template for **Parameter Template** when creating an instance for restoring data to a new instance. After the new instance is

created, click the instance name and manually change the parameter values on the **Parameters** page.

Configure the basic information about the new instance and click **Apply Now**.

- Restoring data to the original DB instance:
  - The instance version and node configuration must be the same as those of the original DB instance.
  - New databases or tables are created on the original DB instance. The databases to be restored are available during the restoration.
  - After a database (for example, **db1**) is restored to the original instance, you need to wait for the DB instance to automatically perform a full backup or differential backup before restoring the data of **db1** using other restoration processes. The time to wait depends on the backup policy. During point-in-time recovery, if you select a time point that is later than when the current database-level restoration is complete but earlier than when the next backup is performed, **db1** cannot be restored.
  - You are advised to manually back up data before the restoration.
- Restoring data to an existing DB instance:
  - Databases and tables will be created on the target instance. During the restoration, the databases can be used properly.
  - You are advised to manually back up data of the selected instance before the restoration.
  - To restore data to an existing instance, the backup media, instance type, deployment model, replica consistency protocol, transaction consistency, CPU architecture, resource specifications, and failover priority of the target instance must be the same as those of the source instance.
  - For instances of version V2.0-8.200.0 or later, an instance backup can be restored to an existing instance with a different compute resource type, deployment model (as long as it has the same number of DN shards and is not a single-node instance), and different specifications (with different vCPUs and memory).

    NOTICE

    You can restore an instance with high specifications to one with low specifications; the reverse is also supported. To restore an instance with high specifications to one with low specifications, ensure that the specifications of the destination instance are greater than or equal to 8 vCPUs | 32 GB.

  - If TDE is enabled for the source instance, table-level data cannot be restored to an existing instance.
- If you use an instance-level backup to restore data to databases, you can select **Time-specific Databases** or **Recent Databases** on the **Restore to Point in Time** page. If you use an instance-level backup to restore data to tables, you can select **Time-specific Tables** or **Recent Tables** on the **Restore to Point in Time** page.

  There are some precautions to take when restoring data to databases or tables. The following uses table restoration as an example:

  - Operations such as table creation may have been performed on the instance during the period from the time when the backup was created to the current time. Therefore, the tables listed when **Time-specific Tables** is selected may be different from those listed when **Recent Tables** is selected. If you attempt to restore a table in **Recent Tables** to a specified time point, the restoration may fail.
    - **Time-specific Tables** indicates the table information stored when the backup was created.
    - **Recent Tables** indicates the most recent table information of the current instance.
  - If some tables have been deleted from the instance, specific tables may not be available regardless of whether **Time-specific Tables** or **Recent Tables** is selected. If you are sure that a particular table is available when the backup was created, click the **+** icon in the upper right corner to add the table.

Step 7 View the restoration results.

- Restoring data to a new DB instance

  A new instance is created using the backup data. The instance status changes from **Creating** to **Available**.

  The new instance is independent from the original one.
- Restoring data to the original DB instance

  On the **Instances** page, the status of the DB instance changes from **Restoring** to **Available**.
- Restoring data to an existing DB instance

  On the **Instances** page, the status of the DB instance changes from **Restoring** to **Available**. After the restoration is complete, an instance-level full backup will be automatically triggered.

----End

12 Parameter Management

12.1 Configurable DB Instance Parameters

This section describes the GaussDB instance parameters that can be modified.

- **Configurable Parameters for Version V2.0-8.x**
- **Configurable Parameters for Version V2.0-3.x**
- **Configurable Parameters for Version V2.0-2.x**

Configurable Parameters for Version V2.0-8.*x*

The following table describes the parameters that can be modified.

Configurable Parameters for Independently Deployed Distributed Instances of Version V2.0-8.*x*

Table 12-1 Parameters for distributed instances that are independently deployed

Parameter	Description
dn:qrw_inlist2join_optmode	Specifies whether to enable inlist-to-join query rewriting.
dn:recovery_max_workers	Specifies the number of concurrent replayer threads.
cn:enable_security_policy	Controls whether unified auditing and dynamic data masking policies are applied.
cn:behavior_compat_options	Specifies database compatibility configuration items. After the value of **proc_outparam_override** is changed, the database must be connected again or the instance must be rebooted. Otherwise, stored procedures and functions cannot be correctly called.

Parameter	Description
dn:recyclebin_retention_time	Specifies how long files will be kept in the recycle bin, in seconds. Files in the recycle bin will be automatically deleted after this length of time.
dn:track_stmt_session_slot	Specifies the maximum number of full or slow SQL statements that can be cached in a session. If the number of full or slow SQL statements exceeds this value, new statements will not be traced until the flush thread flushes the cached statements to the disk to reserve free space. The default value is recommended.
timezone	Specifies the time zone for displaying and interpreting time stamps.
cn:auto_increment_offset	Specifies the initial value of an auto-increment column. The auto-increment value is calculated by the following formula: **auto_increment_offset** + *N* × **auto_increment_increment**. *N* is a positive integer. If the value of this parameter is greater than that of **auto_increment_increment**, there will be an error when the values in the auto-increment column automatically increase.
dn:enable_xid_abort_check	Specifies whether to check the status of transaction ID rollback when a transaction is committed.
cn:audit_internal_event	Specifies whether to audit the connections and operations of internal tools cm_agent, gs_clean, and WDRXdb, and whether to audit the logins and logouts from CNs on DNs.
cn:codegen_compile_thread_num	Specifies the number of Codegen compilation threads.
dn:static_thread_pool_num	Specifies the number of threads used to create a static thread pool (static pool). This parameter takes effect only on CNs of a distributed instance after **enable_thread_pool** is enabled.
cn:auto_increment_increment	Specifies the auto-increment step of an auto-increment column. The auto-increment value is calculated by the following formula: **auto_increment_offset** + *N* × **auto_increment_increment**. *N* is a positive integer. If the value of this parameter is smaller than that of **auto_increment_offset**, there will be an error when the values in the auto-increment column automatically increase.
cn:sql_mode	Specifies the SQL behavior control configuration item in M-compatible mode.

Parameter	Description
wal_level	Specifies the level of information to be written to the WAL. This is a required value and cannot be commented out. Determines how much information is written to the WAL. When this parameter is set to **logical**, logical logs are extracted and primary key information is recorded in Xlogs.
dn:enable_pbe_optimization	Specifies whether the optimizer optimizes the query plan for statements executed in Parse Bind Execute (PBE) mode.
cn:llvm_max_memory	Specifies the upper limit of the memory used by IRs (including cached and in-use IRs) generated during compilation in Codegen execution mode. The memory used by Codegen is not applied for by preoccupation. It is a part of **max_dynamic_memory** and is restricted by the **llvm_max_memory** parameter. Unit: KB
cn:hll_default_log2explicit	Specifies the threshold for switching from the **explicit** mode to the **sparse** mode.
dn:fix_func_selection	Specifies whether to optimize the function matching policy. The value **catlist** indicates the catlist sequence is optimized. (The non-B-compatible mode has been optimized. In non-B-compatible mode, system functions are always preferentially selected and executed. The policy in B-compatible mode is the same as that in versions earlier than 505.1.0. An error message indicating that the function is not unique may be displayed, or a system function may be selected for execution.)
dn:default_limit_rows	Specifies the estimated number of rows to return by default for generating a generic plan, that is, the default value for the LIMIT clause. If this parameter is set to a negative number, the value is converted to a percentage, for example, **-5** is equivalent to 5%, indicating that 5% of the total rows will be returned.
global_syscache_threshold	Specifies the maximum memory usage of the global system cache. Recommended value range: 2,048 KB to 16,384 KB on average per database. If this parameter is set to a large value, the cache link may be too long and performance will deteriorate. If this parameter is set to a small value, the parameter is not applied, and after the memory usage can exceed the threshold, performance deteriorates. Unit: KB

Parameter	Description
dn:sql_mode	Specifies the SQL behavior control configuration item in M-compatible mode.
dn:verify_log_buffers	Specifies the size or pages of verifyLog buffers in memory mode. The unit is 8 KB. For example, if the value of this parameter is 4, the requested memory is 4 x 8 KB = 32 KB. This parameter is valid only when **page_version_check** is set to **persistence**. If **page_version_check** is set to another value, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to **persistence**.
dn:resilience_ctrlstmt_control_iopslimit	Specifies the maximum IOPS that can be used by slow SQL statements after normal SQL statements are marked as slow SQL statements. This parameter is only valid for SELECT statements that are executed by non-sysadmin/monitoradmin users. **0(None)**: The IOPS is not limited. **10(LOW)**: The limit level for IOPS is **LOW**. **20(MEDIUM)**: The limit level for IOPS is **MEDIUM**. **50(HIGH)**: The limit level for IOPS is **HIGH**.
cn:convert_illegal_char_mode	Specifies the placeholders of invalid characters that can be displayed on the client when **enable_convert_illegal_char** is enabled. Value range: 95 characters whose decimal codes range from 32 to 126 in the ASCII coding table.
cn:page_version_check	Specifies the type of page version verification. **off** indicates that page version verification is disabled. **memory** indicates that page version verification in pure memory mode is enabled. The page version information will be lost after a restart. **persistence** indicates that persistent page version verification is enabled. The page version information will not be lost after a restart.
cn:audit_thread_num	Specifies the number of audit threads. Value range: 1 to 48.
recovery_time_target	Specifies whether the standby DB instance completes log writing and replay in streaming DR mode. If this parameter is set to a small value, the performance of the primary node is affected. If it is set to a large value, the log flow is not effectively controlled. The value **0** indicates that log flow control is disabled. Unit: second
dn:enable_convert_illegal_char	Specifies whether the database supports characters not included the character sets.

Parameter	Description
dn:system_view_version	Determines the version of the system view. All versions are backward compatible. For example, when **system_view_version** is set to **3**, all features of version 2 and version 1 are also supported. For details, see the product documentation.
dn:audit_dml_state_select	Determines whether to audit the SELECT operation.
dn:codegen_compile_thread_num	Specifies the number of Codegen compilation threads.
dn:resilience_ctrlstmt_detect_timelimit	Specifies the execution time of a normal SQL statement that will be marked as a slow SQL statement. The value **0** indicates that slow SQL statements are not identified. A value greater than 0 indicates that a normal SQL statement whose execution time exceeds the value of this parameter is marked as a slow SQL statement. This parameter is only valid for SELECT statements that are executed by non-sysadmin/monitoradmin users. Unit: millisecond
cn:a_format_date_timestamp	Specifies whether to enable **current_date**, **current_timestamp**, and **localtimestamp** to return the system time, instead of the transaction start time, when a transaction starts.
cn:recovery_max_workers	Specifies the number of concurrent replayer threads.
dn:max_compile_functions	Specifies the maximum number of function compilation results stored in the server. Excessive functions and compilation results of stored procedures may occupy large memory space. Setting this parameter to an appropriate value can reduce the memory usage and improve system performance. Before modifying this parameter, determine your application scenario and fully verify the change in a test environment. For details, see the reference document.
cn:audit_function_exec	Specifies whether to record the audit information during the execution of the stored procedures, anonymous blocks, or user-defined functions (excluding system functions). The value **0** means to disable the function, and **1** means to enable it.
dn:auto_explain_log_min_duration	Specifies the minimum duration of execution plans that are automatically printed. Only execution plans whose duration is greater than the value of **auto_explain_log_min_duration** will be printed. Unit: millisecond

Parameter	Description
cn:num_internal_lock_partitions	Specifies the number of internal lightweight lock partitions. Changing the value of this parameter affects performance and memory usage. Before modifying this parameter, determine your application scenario and fully verify the change in a test environment. For details, see the reference document.
dn:recovery_parse_workers	Specifies the number of ParseRedoRecord threads in the ultimate RTO feature. This parameter must be used together with **recovery_redo_workers**. If both **recovery_parse_workers** and **recovery_redo_workers** are greater than **1**, ultimate RTO is enabled. If you do not want to enable ultimate RTO, retain the default value **1** for **recovery_parse_workers**. When enabling ultimate RTO, ensure that **replication_type** is set to **1**. If both ultimate RTO and parallel replay are enabled, only ultimate RTO takes effect, and parallel replay is ineffective. Ultimate RTO does not support column-store tables. Therefore, disable ultimate RTO in a system where column-store tables are used or are to be used. Ultimate RTO also does not support flow control. Flow control is determined by the **recovery_time_target** parameter.
dn:enable_default_ustore_table	Specifies whether to enable the Ustore storage engine by default. If this parameter is set to **on**, all created tables are Ustore tables.
cn:enable_auto_explain	Specifies whether to automatically print execution plans. This parameter is used to locate slow stored procedures or slow queries.
dn:page_version_check	Specifies the type of page version verification. **off** indicates that page version verification is disabled. **memory** indicates that page version verification in pure memory mode is enabled. The page version information will be lost after a restart. **persistence** indicates that persistent page version verification is enabled. The page version information will not be lost after a restart.
cn:enable_pbe_optimization	Specifies whether the optimizer optimizes the query plan for statements executed in Parse Bind Execute (PBE) mode.
cn:auto_explain_log_min_duration	Specifies the minimum duration of execution plans that are automatically printed. Only execution plans whose duration is greater than the value of **auto_explain_log_min_duration** will be printed. Unit: millisecond

Parameter	Description
dn:num_internal_lock_partitions	Specifies the number of internal lightweight lock partitions. Changing the value of this parameter affects performance and memory usage. Before modifying this parameter, determine your application scenario and fully verify the change in a test environment. For details, see the reference document.
cn:page_version_max_num	Specifies the maximum number of page versions that can be cached in memory. This parameter is only valid when **page_version_check** is not set to off. If **page_version_check** is set to **off**, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to a value other than **off**. For details about the value range, see the product documentation.
dn:enable_auto_explain	Specifies whether to automatically print execution plans. This parameter is used to locate slow stored procedures or slow queries.
dn:audit_function_exec	Specifies whether to record the audit information during the execution of the stored procedures, anonymous blocks, or user-defined functions (excluding system functions). The value **0** means to disable the function, and **1** means to enable it.
cn:random_page_cost	Specifies the estimated cost for the optimizer to fetch an out-of-sequence disk page.

Parameter	Description
dn:auto_increment_cache	Specifies the number of reserved auto-increment cache values when auto-increment is triggered by batch insertion or import of auto-increment columns. When auto-increment values are reserved, the auto-increment counter value is updated to the maximum auto-increment cache value. Before the cache values are used up, the auto-increment counter value remains unchanged, and the triggered auto-increment uses the cache values. If this parameter is set to **0**, the auto-increment cache values are automatically reserved. When auto-increment is triggered for the first time, if the number of rows to be inserted into the auto-increment column is known, the number is the reserved value. If the number of rows is unknown, 2^n values are reserved each time. For example, one value is reserved in the first auto-increment, two values are reserved in the second auto-increment, four values are reserved in the third auto-increment, and eight values are reserved for in fourth auto-increment. However, if the number of reserved values exceeds 65,535, 65,535 values are reserved. If this parameter is not set to **0**, the number of reserved cache values is the value of this parameter. When auto-increment is triggered for the first time, if the number of rows to be inserted into the auto-increment column is known, the number is the reserved value. If the number of rows is unknown, the value of **auto_increment_cache** is the number of auto-increment values reserved each time. The reserved cache values are valid only in the statement. If the reserved auto-increment cache values are used up and subsequent INSERT statements trigger auto-increment based on the auto-increment counter, the values in the auto-increment column in the table are discontinuous. This parameter does not affect the auto-increment column in the local temporary table.
dn:enable_codegen	Specifies whether code optimization can be enabled. Currently, code optimization uses the LLVM optimization.
dn:instr_unique_sql_combination_options	Specifies the configuration items of combining unique SQL statements of the same type. If this feature is enabled, the IDs of unique SQL statements of the same type are normalized, and the generated unique SQL strings are normalized.

Parameter	Description
dn:behavior_compat_options	Specifies database compatibility configuration items. After the value of **proc_outparam_override** is changed, the database must be connected again or the instance must be rebooted. Otherwise, stored procedures and functions cannot be correctly called.
dn:enable_early_free	Specifies whether the operator memory can be released in advance.
dn:page_missing_dirty_check	Specifies whether to enable the verification for pages not marked as dirty. The verification checks whether the modified pages are not marked as dirty. This parameter is only valid when **page_version_check** is not set to **off**. If **page_version_check** is set to **off**, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to a value other than **off**.
dn:enable_security_policy	Controls whether unified auditing and dynamic data masking policies are applied.
cn:undo_retention_time	Specifies how long undo logs are kept. This parameter is only used for flashback query. Note: 1. The undo space of the local disk increases. 2. In subsequent incremental backups, the size of the backup set increases, because extra undo content is retained. Unit: second
dn:wdr_snapshot_full_backup_interval	Specifies the interval at which a full WDR snapshot is created. The interval specified by this parameter is about a number instead of time. For example, if the parameter is set to **12**, a full snapshot and then 11 incremental snapshots are generated for each group. If the parameter is set to **1**, all snapshots generated are full snapshots.
cms:datastorage_threshold_value_check	Specifies the disk usage threshold to put a database node into read-only mode. If the disk usage of a data directory exceeds this threshold, the database node is automatically changed to read-only. Unit: percentage (%)
dn:audit_thread_num	Specifies the number of audit threads. Value range: 1 to 48.
dn:wdr_snapshot_space_threshold	Specifies the threshold for controlling the space used by snapshots. When the space used by snapshots reaches 80% of the value of this parameter, the control logic of the database is enabled to stabilize the space usage. Unit: KB

Parameter	Description
cn:audit_dml_state	Determines whether to audit the INSERT, UPDATE, and DELETE operations on a specific table. **0**: These operations are not audited. **1**: These operations are audited.
dn:hll_duplicate_check	Specifies whether duplicate check is enabled by default.
cn:gs_perf_interval	Specifies the automatic perf data collection interval. The value **0** indicates that the collection is stopped. If the value is greater than 0 and less than 5, the value **5** is used. Unit: minute
cn:qrw_inlist2join_optmode	Specifies whether to enable inlist-to-join query rewriting.
dn:gs_perf_interval	Specifies the automatic perf data collection interval. The value **0** indicates that the collection is stopped. If the value is greater than 0 and less than 5, the value **5** is used. Unit: minute
dn:extra_float_digits	Adjusts the number of digits displayed for floating-point values, including float4, float8, and geometric data types. The parameter value is added to the standard number of digits (FLT_DIG or DBL_DIG as appropriate). This parameter can also be set to a negative value to suppress unwanted digits.
cn:disable_keyword_options	Specifies the configuration items for database compatibility. Multiple items are separated by commas (,). An identifier with this parameter set will not be used as a keyword.
dn:audit_dml_state	Determines whether to audit the INSERT, UPDATE, and DELETE operations on a specific table. **0**: These operations are not audited. **1**: These operations are audited.
cn:max_standby_archive_delay	Specifies the wait period before queries on a standby node are canceled when the queries conflict with WAL processing and archiving in hot standby mode. **-1** indicates that the standby node waits until the conflicting queries are complete. Unit: millisecond
track_stmt_stat_level	Controls the level of statement execution tracking.

Parameter	Description
dn:max_standby_archive_delay	Specifies the wait period before queries on a standby node are canceled when the queries conflict with WAL processing and archiving in hot standby mode. **-1** indicates that the standby node waits until the conflicting queries are complete. Unit: millisecond
cn:b_format_dev_version	Specifies the compatibility configuration item of database platform minor versions.
dn:copy_special_character_version	Specifies whether to report an error when there are invalid characters during data import and export using COPY FROM.
cn:page_version_recycler_thread_num	Specifies the number of threads for recycling and verifying page version information. This parameter is only valid when **page_version_check** is not set to **off**. If **page_version_check** is set to **off**, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to a value other than **off**.
enable_wdr_snapshot	Specifies whether to enable WDR snapshots. If this parameter is enabled, the kernel periodically takes performance snapshots.
dn:effective_cache_size	Specifies the size of the disk buffer available to the DN optimizer in a single query. Unit: 8 KB
cn:wdr_snapshot_full_backup_interval	Specifies the interval at which a full WDR snapshot is created. The interval specified by this parameter is about a number instead of time. For example, if the parameter is set to **12**, a full snapshot and then 11 incremental snapshots are generated for each group. If the parameter is set to **1**, all snapshots generated are full snapshots.
cn:extra_float_digits	Adjusts the number of digits displayed for floating-point values, including float4, float8, and geometric data types. The parameter value is added to the standard number of digits (FLT_DIG or DBL_DIG as appropriate). This parameter can also be set to a negative value to suppress unwanted digits.
cn:system_view_version	Determines the version of the system view. All versions are backward compatible. For example, when **system_view_version** is set to **3**, all features of version 2 and version 1 are also supported. For details, see the product documentation.

Parameter	Description
cn:recyclebin_retention_time	Specifies how long files will be kept in the recycle bin. Files in the recycle bin will be automatically deleted after this length of time. Unit: second
cn:password_encryption_type	Specifies how user passwords are encrypted. **0**: Passwords are encrypted using MD5. **1**: Passwords are encrypted using SHA-256 and MD5. **2**: Passwords are encrypted using SHA-256. **3**: Passwords are encrypted using SM3. MD5 is not recommended because it is a weak encryption algorithm.
cn:check_disconnect_query	Specifies whether to enable the function of terminating statement execution on the server after the client is disconnected due to timeout.
password_effect_time	Specifies the validity period of an account password. Unit: day
cn:verify_log_buffers	Specifies the size or pages of verifyLog buffers in memory mode. The unit is 8 KB. For example, if the value of this parameter is 4, the requested memory is 4 x 8 KB = 32 KB. This parameter is valid only when **page_version_check** is set to **persistence**. If **page_version_check** is set to another value, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to **persistence**.
dn:random_page_cost	Specifies the estimated cost for the optimizer to fetch an out-of-sequence disk page.
dn:resilience_ctrlslot_available_maxpercent	Specifies the maximum percentage of threads in the thread pool that can be occupied by slow SQL statements. This parameter is only valid for SELECT statements that are executed by non-sysadmin/ monitoradmin users.
cn:default_limit_rows	Specifies the estimated number of rows to return by default for generating a generic plan, that is, the default value for the LIMIT clause. If this parameter is set to a negative number, the value is converted to a percentage, for example, **-5** is equivalent to 5%, indicating that 5% of the total rows will be returned.
cn:enable_rls_match_index	Specifies whether indexes of a base table can be scanned based on target predicate conditions in row-level security scenarios. Target scenario: Row level security (RLS) policies are set and enabled in the base table, and the query predicate contains the **unleakproof** system function or **like** operator.

Parameter	Description
dn:enable_analyze_check	Specifies whether to check if statistics were collected about tables whose **reltuples** and **relpages** are displayed as **0** in pg_class during plan generation.
cn:resilience_ctrlstmt_control_iopslimit	Specifies the maximum IOPS that can be used by slow SQL statements after normal SQL statements are marked as slow SQL statements. This parameter is only valid for SELECT statements that are executed by non-sysadmin/monitoradmin users. **0(None)**: The IOPS is not limited. **10(LOW)**: The limit level for IOPS is **LOW**. **20(MEDIUM)**: The limit level for IOPS is **MEDIUM**. **50(HIGH)**: The limit level for IOPS is **HIGH**.
dn:auto_increment_increment	Specifies the auto-increment step of an auto-increment column. The auto-increment value is calculated by the following formula: **auto_increment_offset** + *N* × **auto_increment_increment**. *N* is a positive integer. If the value of this parameter is smaller than that of **auto_increment_offset**, there will be an error when the values in the auto-increment column automatically increase.
cn:copy_special_character_version	Specifies whether to report an error when there are invalid characters during data import and export using COPY FROM.
cn:local_syscache_threshold	Specifies the size of system catalog cache in a session. Unit: KB
dn:gs_format_behavior_compat_options	Specifies the configuration items of GaussDB internal system functions.
cn:hll_duplicate_check	Specifies whether duplicate check is enabled by default.
cn:max_standby_streaming_delay	Specifies the wait period before queries on the standby node are canceled when the queries conflict with WAL data receiving through streaming replication in hot standby mode. **-1** indicates that the standby node waits until the conflicting queries are complete. Unit: millisecond
dn:b_format_dev_version	Specifies the compatibility configuration item of database platform minor versions.
dn:check_disconnect_query	Specifies whether to enable the function of terminating statement execution on the server after the client is disconnected due to timeout.

Parameter	Description
cn:gs_format_behavior_compat_options	Specifies the configuration items of GaussDB internal system functions.
cn:audit_xid_info	Determines whether to record the transaction IDs of SQL statements in detail_info. **0**: The transaction IDs are not recorded. **1**: The transaction IDs are recorded.
wdr_snapshot_retention_days	Specifies how many days database monitoring snapshots are saved for.
cn:static_thread_pool_num	Specifies the number of threads used to create a static thread pool (static pool). This parameter takes effect only on CNs of a distributed instance after **enable_thread_pool** is enabled.
dn:max_standby_streaming_delay	Specifies the wait period before queries on the standby node are canceled when the queries conflict with WAL data receiving through streaming replication in hot standby mode. **-1** indicates that the standby node waits until the conflicting queries are complete. Unit: millisecond
audit_system_object	Specifies whether to audit the CREATE, DROP, and ALTER operations on GaussDB database objects. GaussDB database objects include databases, users, schemas, and tables. You can change the value of this parameter to audit only the operations on required database objects. In the scenario where the leader node is forcibly selected, you are advised to set **audit_system_object** to the maximum value and audit all DDL objects. For details about the value range, see the product documentation.
cn:enable_default_ustore_table	Specifies whether to enable the Ustore storage engine by default. If this parameter is set to **on**, all created tables are Ustore tables.
cn:resilience_ctrlslot_available_maxpercent	Specifies the maximum percentage of threads in the thread pool that can be occupied by slow SQL statements. This parameter is only valid for SELECT statements that are executed by non-sysadmin/monitoradmin users.

Parameter	Description
dn:page_version_partitions	Specifies the number of hash table partitions in cached page version information in the memory. This parameter directly affects the hash query efficiency and hash conflict probability, and is valid only when **page_version_check** is not set to **off**. If **page_version_check** is set to **off**, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to a value other than **off**. For details about the value range, see the product documentation.
dn:page_version_recycler_thread_num	Specifies the number of threads for recycling and verifying page version information. This parameter is only valid when **page_version_check** is not set to **off**. If **page_version_check** is set to **off**, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to a value other than **off**.
dn:password_encryption_type	Specifies how user passwords are encrypted. **0**: Passwords are encrypted using MD5. **1**: Passwords are encrypted using SHA-256 and MD5. **2**: Passwords are encrypted using SHA-256. **3**: Passwords are encrypted using SM3. MD5 is not recommended because it is a weak encryption algorithm.
dn:dcf_thread_effective_time	Specifies the effective time of the DCF flushing thread. This parameter is used to determine whether the disk I/O hangs. If the DCF cannot access I/O resources within the period specified by this parameter, the DCF considers that the thread I/O hangs and a primary/standby switchover is triggered. If this parameter is set to **0**, I/O hang detection is disabled. Unit: second
dn:a_format_date_timestamp	Specifies whether to enable **current_date**, **current_timestamp**, and **localtimestamp** to return the system time, instead of the transaction start time, when a transaction starts.
dn:hll_default_log2sparse	Specifies the default threshold for switching from Sparse mode to Full mode.
cn:enable_xid_abort_check	Specifies whether to check the status of transaction ID rollback when a transaction is committed.
session_timeout	Specifies how long to wait before a server connection is disconnected due to inactivity. The value **0** indicates there is no time limit. Unit: second

Parameter	Description
cn:hll_default_log2sparse	Specifies the default threshold for switching from Sparse mode to Full mode.
dn:disable_keyword_options	Specifies the configuration items for database compatibility. Multiple items are separated by commas (,). An identifier with this parameter set will not be used as a keyword.
cn:fix_func_selection	Specifies whether to optimize the function matching policy. The value **catlist** indicates the catlist sequence is optimized. (The non-B-compatible mode has been optimized. In non-B-compatible mode, system functions are always preferentially selected and executed. The policy in B-compatible mode is the same as that in versions earlier than 505.1.0. An error message indicating that the function is not unique may be displayed, or a system function may be selected for execution.)
cn:support_binary_copy_version	Specifies whether to verify the binary file encoding information when data is imported using COPY FROM in BINARY mode. If forward compatibility is required, leave this parameter empty. Otherwise, retain the default value.
autoanalyze	Specifies whether to automatically collect statistics on tables that have no statistics when a plan is generated.
password_lock_time	Specifies the maximum number of incorrect password attempts before an account is locked. The account will be automatically unlocked after the time specified in **password_lock_time** elapses. Only the **sysadmin** user can set this parameter.
wdr_snapshot_interval	Specifies the interval at which the background thread Snapshot automatically takes snapshots of the database monitoring data. Unit: minute
update_lockwait_timeout	Specifies the maximum duration that a lock waits for concurrent updates on a row to complete when the concurrent update feature is enabled. If the lock wait time exceeds this value, the system will report an error. Unit: millisecond
dn:undo_retention_time	Specifies how long undo logs are kept. This parameter is only used for flashback query. Note: 1. The undo space of the local disk increases. 2. In subsequent incremental backups, the size of the backup set increases, because extra undo content is retained. Unit: second

Parameter	Description
cn:recovery_parse_workers	Specifies the number of ParseRedoRecord threads in the ultimate RTO feature. This parameter must be used together with **recovery_redo_workers**. If both **recovery_parse_workers** and **recovery_redo_workers** are greater than **1**, ultimate RTO is enabled. If you do not want to enable ultimate RTO, retain the default value **1** for **recovery_parse_workers**. When enabling ultimate RTO, ensure that **replication_type** is set to **1**. If both ultimate RTO and parallel replay are enabled, only ultimate RTO takes effect, and parallel replay is ineffective. Ultimate RTO does not support column-store tables. Therefore, disable ultimate RTO in a system where column-store tables are used or are to be used. Ultimate RTO also does not support flow control. Flow control is determined by the **recovery_time_target** parameter.
cn:undo_space_limit_size	Specifies the threshold for forcibly recycling undo space. When the undo space usage reaches 80% of the threshold, forcible recycling starts. It is recommended that the value of this parameter be greater than or equal to the value of **undo_limit_size_per_transaction**. Unit: 8 KB

Parameter	Description
cn:auto_increment_cache	Specifies the number of reserved auto-increment cache values when auto-increment is triggered by batch insertion or import of auto-increment columns. When auto-increment values are reserved, the auto-increment counter value is updated to the maximum auto-increment cache value. Before the cache values are used up, the auto-increment counter value remains unchanged, and the triggered auto-increment uses the cache values. If this parameter is set to **0**, the auto-increment cache values are automatically reserved. When auto-increment is triggered for the first time, if the number of rows to be inserted into the auto-increment column is known, the number is the reserved value. If the number of rows is unknown, 2^n values are reserved each time. For example, one value is reserved in the first auto-increment, two values are reserved in the second auto-increment, four values are reserved in the third auto-increment, and eight values are reserved for in fourth auto-increment. However, if the number of reserved values exceeds 65,535, 65,535 values are reserved. If this parameter is not set to **0**, the number of reserved cache values is the value of this parameter. When auto-increment is triggered for the first time, if the number of rows to be inserted into the auto-increment column is known, the number is the reserved value. If the number of rows is unknown, the value of **auto_increment_cache** is the number of auto-increment values reserved each time. The reserved cache values are valid only in the statement. If the reserved auto-increment cache values are used up and subsequent INSERT statements trigger auto-increment based on the auto-increment counter, the values in the auto-increment column in the table are discontinuous. This parameter does not affect the auto-increment column in the local temporary table.
dn:enable_hotkeys_collection	Specifies whether to collect statistics on accessed key values in databases.
cn:enable_early_free	Specifies whether the operator memory can be released in advance.

Parameter	Description
cn:max_concurrent_autonomous_transactions	Specifies the maximum number of autonomous transaction connections, that is, the maximum number of concurrent autonomous transactions executed at the same time. If this parameter is set to **0**, autonomous transactions cannot be executed. The theoretical maximum value is **10000**. Set this parameter based on workload requirements and hardware configurations. It is recommended that this parameter be set to a value less than or equal to 1/10 of **max_connections**.
cn:audit_set_parameter	Determines whether to audit the SET operation. **0**: The SET operation is not audited. **1**: The SET operation is audited.
cn:enable_enhance_toast_table	Specifies whether to use the enhanced TOAST out-of-line storage table. The value **on** indicates that the enhanced TOAST out-of-line storage table is used. The value **off** indicates that the TOAST out-of-line storage table is used.
cn:archive_interval	Specifies the archiving interval. Log files are forcibly archived when the period specified by this parameter has elapsed. A large value of this parameter affects the RPO of PITR. The default value is recommended. Unit: second
dn:undo_limit_size_per_transaction	Specifies the maximum undo space for a single transaction. If the undo space of a transaction exceeds this parameter value, the transaction is rolled back due to an error. It is recommended that the value of this parameter be smaller than the value of **undo_space_limit_size**. If this parameter value is larger, the value of **undo_space_limit_size** will be used as the maximum undo space for a single transaction. If this undo space is greater than 1 TB, system performance and stability may be affected. Unit: 8 KB
dn:convert_illegal_char_mode	Specifies the placeholders of invalid characters that can be displayed on the client when **enable_convert_illegal_char** is enabled. Value range: 95 characters whose decimal codes range from 32 to 126 in the ASCII coding table.
cn:enable_analyze_check	Specifies whether to check if statistics were collected about tables whose **reltuples** and **relpages** are displayed as **0** in pg_class during plan generation.

Parameter	Description
dn:tde_index_default_encrypt	When **tde_index_default_encrypt** is set to **on** and an index is created based on an encrypted table, the database automatically converts the index to an encrypted index.
failed_login_attempts	Specifies the maximum number of incorrect password attempts before an account is locked. The account will be automatically unlocked after the time specified in **password_lock_time** elapses. Only the **sysadmin** user can set this parameter.
dn:enable_bbox_dump	Specifies whether to enable the black box function so that core files can be generated even when the core mechanism is not configured in the system.
dn:enable_recyclebin	Enables or disables the recycle bin in real time.
cn:wdr_snapshot_space_threshold	Specifies the threshold for controlling the space used by snapshots. When the space used by snapshots reaches 80% of the value of this parameter, the control logic of the database is enabled to stabilize the space usage. Unit: KB
enable_global_syscache	Determines whether to enable global system cache.
dn:audit_set_parameter	Determines whether to audit the SET operation. **0**: The SET operation is not audited. **1**: The SET operation is audited.
cn:max_compile_functions	Specifies the maximum number of function compilation results stored in the server. Excessive functions and compilation results of stored procedures may occupy large memory space. Setting this parameter to an appropriate value can reduce the memory usage and improve system performance. Before modifying this parameter, determine your application scenario and fully verify the change in a test environment. For details, see the reference document.
dn:auto_increment_offset	Specifies the initial value of an auto-increment column. The auto-increment value is calculated by the following formula: **auto_increment_offset** + *N* × **auto_increment_increment**. *N* is a positive integer. If the value of this parameter is greater than that of **auto_increment_increment**, there will be an error when the values in the auto-increment column automatically increase.

Parameter	Description
cn:track_stmt_session_slot	Specifies the maximum number of full or slow SQL statements that can be cached in a session. If the number of full or slow SQL statements exceeds this value, new statements will not be traced until the flush thread flushes the cached statements to the disk to reserve free space. The default value is recommended.
cn:page_version_partitions	Specifies the number of hash table partitions in cached page version information in the memory. This parameter directly affects the hash query efficiency and hash conflict probability, and is valid only when **page_version_check** is not set to **off**. If **page_version_check** is set to **off**, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to a value other than **off**. For details about the value range, see the product documentation.
autoanalyze_timeout	Specifies the autoanalyze timeout period. If the duration of autoanalyze on a table exceeds the value of **autoanalyze_timeout**, the autoanalyze operation is automatically canceled. The value **0** indicates that there is no timeout limit. Unit: second
dn:index_txntype	Sets the index page format to PCR or RCR. This parameter is left unconfigured during system initialization. By default, the created indexes are compatible with the index type (RCR) of earlier versions. Once this parameter is specified, it cannot be left unconfigured again.
cn:enable_recyclebin	Enables or disables the recycle bin in real time.
log_min_duration_statement	Specifies the threshold for logging the duration of a completed statement. If a statement runs for a period greater than or equal to the specified value, its duration will be logged. The value **-1** disables logging statement durations. If this parameter is set to a small value, the load throughput may be affected. Unit: millisecond
max_replication_slots	Specifies the number of log replication slots on the primary node.
cn:enable_bbox_dump	Specifies whether to enable the black box function so that core files can be generated even when the core mechanism is not configured in the system.
datestyle	Specifies the display format for date and time.

Parameter	Description
dn:hll_default_log2m	Specifies the number of buckets for HLL data.
enable_slot_log	Specifies whether to enable primary/standby synchronization for logical replication slots. Currently, only archive slots and backup slots are involved. Set this parameter to **on** only in cloud scenarios where logical replication is enabled. In other scenarios, set this parameter to **off**.
cn:enable_convert_illegal_char	Specifies whether the database supports characters not included the character sets.
cn:enable_workload_rule	Specifies whether to enable SQL throttling.
cn:index_txntype	Sets the index page format to PCR or RCR. This parameter is left unconfigured during system initialization. By default, the created indexes are compatible with the index type (RCR) of earlier versions. Once this parameter is specified, it cannot be left unconfigured again.
dn:enable_enhance_toast_table	Specifies whether to use the enhanced TOAST out-of-line storage table. The value **on** indicates that the enhanced TOAST out-of-line storage table is used. The value **off** indicates that the TOAST out-of-line storage table is used.
dn:support_binary_copy_version	Specifies whether to verify the binary file encoding information when data is imported using COPY FROM in BINARY mode. If forward compatibility is required, leave this parameter empty. Otherwise, retain the default value.
enable_stream_operator	Specifies the query optimizer's use of streams. If **enable_stream_operator** is disabled, a large number of logs indicating that the plans cannot be pushed down are recorded. If you do not need these logs, you are advised to disable both **enable_unshipping_log** and **enable_stream_operator**. The default value is recommended.
cn:enable_dynamic_samplesize	Specifies whether to dynamically adjust the number of sampled rows. For a large table with more than one million rows, the number of sampled rows is dynamically adjusted during statistics collection to improve statistics accuracy.
cn:gs_perf_retention_days	Specifies how many days the flame graph files in HTML format are retained. Unit: day

Parameter	Description
dn:audit_xid_info	Determines whether to record the transaction IDs of SQL statements in detail_info. **0**: The transaction IDs are not recorded. **1**: The transaction IDs are recorded.
dn:cost_model_version	Specifies the version of the optimizer cost model. It is a protective parameter. It prevents new optimizer cost models from being applied, so you can keep the current model consistent with the plan of an existing version. If the value of this parameter is changed, many SQL plans may be changed. Exercise caution when modifying this parameter.
dn:enable_rls_match_index	Specifies whether indexes of a base table can be scanned based on target predicate conditions in row-level security scenarios. Target scenario: Row level security (RLS) policies are set and enabled in the base table, and the query predicate contains the **unleakproof** system function or **like** operator.
dn:audit_internal_event	Specifies whether to audit the connections and operations of internal tools cm_agent, gs_clean, and WDRXdb, and whether to audit the logins and logouts from CNs on DNs.
cn:effective_cache_size	Specifies the size of the disk buffer available to the CN optimizer in a single query. Unit: 8 KB
enable_seqscan	Specifies whether to enable the optimizer's use of sequential scan plan types. It is impossible to completely suppress sequential scans, but setting this parameter to **off** allows the optimizer to choose other methods if available.
dn:hll_default_log2explicit	Specifies the threshold for switching from the **explicit** mode to the **sparse** mode.
cn:instr_unique_sql_combination_options	Specifies the configuration items of combining unique SQL statements of the same type.
dn:undo_space_limit_size	Specifies the threshold for forcibly recycling undo space. When the undo space usage reaches 80% of the threshold, forcible recycling starts. It is recommended that the value of this parameter be greater than or equal to the value of **undo_limit_size_per_transaction**. Unit: 8 KB
dn:enable_dynamic_samplesize	Specifies whether to dynamically adjust the number of sampled rows. For a large table with more than one million rows, the number of sampled rows is dynamically adjusted during statistics collection to improve statistics accuracy.

Parameter	Description
dn:llvm_max_memory	Specifies the upper limit of the memory used by IRs (including cached and in-use IRs) generated during compilation in Codegen execution mode. The memory used by Codegen is not applied for by preoccupation. It is a part of **max_dynamic_memory** and is restricted by the **llvm_max_memory** parameter. Unit: KB
dn:local_syscache_threshold	Specifies the size of system catalog cache in a session. Unit: KB
cn:tde_index_default_encrypt	When **tde_index_default_encrypt** is set to **on** and an index is created based on an encrypted table, the database automatically converts the index to an encrypted index.
dn:enable_workload_rule	Specifies whether to enable SQL throttling.
dn:archive_interval	Specifies the archiving interval. Log files are forcibly archived when the period specified by this parameter has elapsed. A large value of this parameter affects the RPO of PITR. The default value is recommended. Unit: second
cn:enable_hotkeys_collection	Specifies whether to collect statistics on accessed key values in databases.
cn:enable_codegen	Specifies whether code optimization can be enabled. Currently, code optimization uses the LLVM optimization.
max_wal_senders	Specifies the maximum number of WAL sender threads that can be created. Processes created for the following purposes occupy WAL sender threads: Standby DNs connect to primary DNs to obtain physical logs, and logical replication tools connect to primary DNs to obtain logical logs. If this parameter is set to a value smaller than 20, scale-out may fail. The value of this parameter must be smaller than that of **max_connections**.
cn:undo_limit_size_per_transaction	Specifies the maximum undo space for a single transaction. If the undo space of a transaction exceeds this parameter value, the transaction is rolled back due to an error. It is recommended that the value of this parameter be smaller than the value of **undo_space_limit_size**. If this parameter value is larger, the value of **undo_space_limit_size** will be used as the maximum undo space for a single transaction. If this undo space is greater than 1 TB, system performance and stability may be affected. Unit: 8 KB

Parameter	Description
dn:max_concurrent_autonomous_transactions	Specifies the maximum number of autonomous transaction connections, that is, the maximum number of concurrent autonomous transactions executed at the same time. If this parameter is set to **0**, autonomous transactions cannot be executed. The theoretical maximum value is **10000**. Set this parameter based on workload requirements and hardware configurations. It is recommended that this parameter be set to a value less than or equal to 1/10 of **max_connections**.
cn:page_missing_dirty_check	Specifies whether to enable the verification for pages not marked as dirty. The verification checks whether the modified pages are not marked as dirty. This parameter is only valid when **page_version_check** is not set to **off**. If **page_version_check** is set to **off**, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to a value other than **off**.
cn:cost_model_version	Specifies the version of the optimizer cost model. It is a protective parameter. It prevents new optimizer cost models from being applied, so you can keep the current model consistent with the plan of an existing version. If the value of this parameter is changed, many SQL plans may be changed. Exercise caution when modifying this parameter.
cn:audit_dml_state_select	Determines whether to audit the SELECT operation.
cn:hll_default_log2m	Specifies the number of buckets for HLL data.
dn:use_workload_manager	Specifies whether to enable resource management. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:use_workload_manager	Specifies whether to enable resource management. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_save_datachanged_timestamp	Specifies whether to record the time when INSERT, UPDATE, DELETE, or EXCHANGE/TRUNCATE/DROP PARTITION is performed on table data. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_save_datachanged_timestamp	Specifies whether to record the time when INSERT, UPDATE, DELETE, or EXCHANGE/TRUNCATE/DROP PARTITION is performed on table data. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
dn:max_process_memory	Specifies the maximum physical memory allowed for a DN. Unit: KB This parameter is supported in GaussDB V2.0-8.201 or later.
cn:max_process_memory	Specifies the maximum physical memory allowed for a CN. Unit: KB This parameter is supported in GaussDB V2.0-8.201 or later.
dn:standby_shared_buffers_fraction	Specifies the **shared_buffers** proportion used on the server where a standby instance is deployed. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:standby_shared_buffers_fraction	Specifies the **shared_buffers** proportion used on the server where a standby instance is deployed. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:pagewriter_sleep	Specifies the interval (in milliseconds) for the page writer thread to flush dirty pages to disks after the incremental checkpoint is enabled. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:pagewriter_sleep	Specifies the interval (in milliseconds) for the page writer thread to flush dirty pages to disks after the incremental checkpoint is enabled. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:xloginsert_locks	Specifies the number of locks on concurrent write-ahead logging. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:xloginsert_locks	Specifies the number of locks on concurrent write-ahead logging. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_light_proxy	Specifies whether the optimizer optimizes the execution of simple queries on CNs. This parameter does not take effect if the character set of the application side does not match that of the kernel side. You are advised to set the character set to UTF-8 when creating a database. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_light_proxy	Specifies whether the optimizer optimizes the execution of simple queries on CNs. This parameter does not take effect if the character set of the application side does not match that of the kernel side. You are advised to set the character set to UTF-8 when creating a database. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
dn:synchronous_commit	Specifies the synchronization mode of the current transaction. Generally, logs generated by a transaction are synchronized in the following sequence: The primary node writes the logs to the local memory. The primary node writes logs in the local memory to the local file system. The primary node flushes the logs in the local file system to disks. The primary node sends the logs to the standby node. The standby node receives the logs and saves them to its local memory. The standby node writes the logs in the local memory to the local file system. The standby node flushes the logs in the local file system to disks. The standby node replays the logs to complete the incremental update of data files. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:synchronous_commit	Specifies the synchronization mode of the current transaction. Generally, logs generated by a transaction are synchronized in the following sequence: The primary node writes the logs to the local memory. The primary node writes logs in the local memory to the local file system. The primary node flushes the logs in the local file system to disks. The primary node sends the logs to the standby node. The standby node receives the logs and saves them to its local memory. The standby node writes the logs in the local memory to the local file system. The standby node flushes the logs in the local file system to disks. The standby node replays the logs to complete the incremental update of data files. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_incremental_checkpoint	Specifies whether to enable incremental checkpoint. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_incremental_checkpoint	Specifies whether to enable incremental checkpoint. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:bypass_workload_manager	Enables or disables independent I/O control. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:bypass_workload_manager	Enables or disables independent I/O control. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
enable_resource_track	Specifies whether to enable real-time resource monitoring. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_instr_rt_percentile	Specifies whether to enable the function of calculating the response time of 80% and 95% of the SQL statements in the system. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_instr_rt_percentile	Specifies whether to enable the function of calculating the response time of 80% and 95% of the SQL statements in the system. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_instr_cpu_timer	Specifies whether to capture the CPU time consumed during SQL statement execution. In the x86-based centralized deployment scenario where 32 vCPUs and 256 GB of memory are configured, the performance tested by BenchmarkSQL 5.0 fluctuates by about 3.5% by enabling or disabling this parameter. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_instr_cpu_timer	Specifies whether to capture the CPU time consumed during SQL statement execution. In the x86-based centralized deployment scenario where 32 vCPUs and 256 GB of memory are configured, the performance tested by BenchmarkSQL 5.0 fluctuates by about 3.5% by enabling or disabling this parameter. This parameter is supported in GaussDB V2.0-8.201 or later.
track_counts	Determines whether to enable collection of statistics on database activities. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_user_metric_persistent	Specifies whether to dump the historical monitoring data of user resources. If this parameter is set to **on**, data in the **PG_TOTAL_USER_RESOURCE_INFO** view is periodically sampled and saved to the system catalog **GS_WLM_USER_RESOURCE_HISTORY**. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_user_metric_persistent	Specifies whether to dump the historical monitoring data of user resources. If this parameter is set to **on**, data in the **PG_TOTAL_USER_RESOURCE_INFO** view is periodically sampled and saved to the system catalog **GS_WLM_USER_RESOURCE_HISTORY**. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
dn:log_hostname	By default, connection log messages only show the IP address of the connecting host. The host name can be recorded when this parameter is set to **on**. It may take some time to parse the host name. Therefore, the database performance may be affected. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:log_hostname	By default, connection log messages only show the IP address of the connecting host. The host name can be recorded when this parameter is set to **on**. It may take some time to parse the host name. Therefore, the database performance may be affected. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_instr_track_wait	Specifies whether to enable real-time collection of wait event information. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_instr_track_wait	Specifies whether to enable real-time collection of wait event information. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_instance_metric_persistent	Specifies whether to dump instance resource monitoring data. When this parameter is set to **on**, the instance monitoring data is saved to the system catalog **GS_WLM_INSTANCE_HISTORY**. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_instance_metric_persistent	Specifies whether to dump instance resource monitoring data. When this parameter is set to **on**, the instance monitoring data is saved to the system catalog **GS_WLM_INSTANCE_HISTORY**. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_logical_io_statistics	Specifies whether to collect logical I/O statistics during resource monitoring. If this function is enabled, the **read_kbytes**, **write_kbytes**, **read_counts**, **write_counts**, **read_speed**, and **write_speed** columns in the PG_TOTAL_USER_RESOURCE_INFO view will collect statistics on the byte count, number of times, and speed of logical read and write. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
cn:enable_logical_io_statistics	Specifies whether to collect logical I/O statistics during resource monitoring. If this function is enabled, the **read_kbytes**, **write_kbytes**, **read_counts**, **write_counts**, **read_speed**, and **write_speed** columns in the PG_TOTAL_USER_RESOURCE_INFO view will collect statistics on the byte count, number of times, and speed of logical read and write. Columns related to logical read and write in the system catalogs GS_WLM_USER_RESOURCE_HISTORY and GS_WLM_INSTANCE_HISTORY will collect statistics on the logical read and write of related users and instances. This parameter is supported in GaussDB V2.0-8.201 or later.
enable_sort	Specifies whether the query optimizer uses sort methods. It is impossible to suppress explicit sorts entirely, but setting this variable to **off** encourages the optimizer to choose other methods if available. This parameter is supported in GaussDB V2.0-8.201 or later.
enable_material	Specifies whether the query optimizer uses materialization. It is impossible to suppress materialization entirely, but setting this variable to **off** prevents the optimizer from inserting materialized nodes. This parameter is supported in GaussDB V2.0-8.201 or later.
gtm:gtm_max_trans	Specifies the maximum number of connections accepted by the GTM. You are advised not to change the value. This parameter is supported in GaussDB V2.0-8.201 or later.
gtm:csn_sync_interval	Specifies the interval for synchronizing CSNs between the primary and standby GTMs, in seconds. This parameter is supported in GaussDB V2.0-8.201 or later.
gtm:restore_duration	Specifies the reply interval of XIDs or CSNs on the GTM, that is, the number of XIDs or CSNs for reply. This parameter is supported in GaussDB V2.0-8.201 or later.
cms:coordinator_heartbeat_timeout	Specifies the heartbeat timeout (in seconds) that triggers the automatic removal of faulty CNs. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
dn:client_min_messages	Specifies which level of messages will be sent to the client. A higher level covers the messages of all the lower levels. The lower the level is, the fewer messages are sent. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:client_min_messages	Specifies which level of messages will be sent to the client. A higher level covers the messages of all the lower levels. The lower the level is, the fewer messages are sent. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:pagewriter_thread_num	Specifies the number of threads for background page flushing after the incremental checkpoint is enabled. Dirty pages are flushed in sequence to disks, updating recovery points. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:pagewriter_thread_num	Specifies the number of threads for background page flushing after the incremental checkpoint is enabled. Dirty pages are flushed in sequence to disks, updating recovery points. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:bgwriter_delay	Specifies the interval at which the background writer writes dirty shared buffers, in milliseconds. Each time, the background writer initiates write operations for some dirty buffers. In full checkpoint mode, the **bgwriter_lru_maxpages** parameter is used to control the amount of data to be written each time, and the process is restarted after the period of hibernation specified by **bgwriter_delay** (in milliseconds). In incremental checkpoint mode, the number of target idle buffer pages is calculated based on the value of **candidate_buf_percent_target**. If the number of idle buffer pages is insufficient, a batch of pages are flushed to disks at the interval specified by **bgwriter_delay** (in milliseconds). The number of flushed pages is calculated based on the target difference percentage. The maximum number of flushed pages is limited by **max_io_capacity**. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
cn:bgwriter_delay	Specifies the interval at which the background writer writes dirty shared buffers, in milliseconds. Each time, the background writer initiates write operations for some dirty buffers. In full checkpoint mode, the **bgwriter_lru_maxpages** parameter is used to control the amount of data to be written each time, and the process is restarted after the period of hibernation specified by **bgwriter_delay** (in milliseconds). In incremental checkpoint mode, the number of target idle buffer pages is calculated based on the value of **candidate_buf_percent_target**. If the number of idle buffer pages is insufficient, a batch of pages are flushed to disks at the interval specified by **bgwriter_delay** (in milliseconds). The number of flushed pages is calculated based on the target difference percentage. The maximum number of flushed pages is limited by **max_io_capacity**. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:max_files_per_process	Specifies the maximum number of files that can be opened simultaneously by each server process. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:candidate_buf_percent_target	Specifies the proportion (%) of available buffers in **shared_buffer** in the candidate buffer chain when the incremental checkpoint is enabled. If the percentage is less than the value of this parameter, the **bgwriter** thread starts to flush dirty pages based on the **max_io_capacity** parameter. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:candidate_buf_percent_target	Specifies the proportion (%) of available buffers in **shared_buffer** in the candidate buffer chain when the incremental checkpoint is enabled. If the percentage is less than the value of this parameter, the **bgwriter** thread starts to flush dirty pages based on the **max_io_capacity** parameter. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:checkpoint_segments	Specifies the minimum number of WAL segment files in the period specified by **checkpoint_timeout**. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
cn:checkpoint_segments	Specifies the minimum number of WAL segment files in the period specified by **checkpoint_timeout**. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:checkpoint_timeout	Specifies the maximum time between automatic WAL checkpoints, in seconds. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:checkpoint_timeout	Specifies the maximum time between automatic WAL checkpoints, in seconds. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:incremental_checkpoint_timeout	Specifies the maximum interval (in seconds) between automatic WAL checkpoints when the incremental checkpoint is enabled. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:incremental_checkpoint_timeout	Specifies the maximum interval (in seconds) between automatic WAL checkpoints when the incremental checkpoint is enabled. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:wal_writer_delay	Specifies the delay (in milliseconds) between activity rounds for the WAL writer. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:wal_writer_delay	Specifies the delay (in milliseconds) between activity rounds for the WAL writer. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:wal_buffers	Specifies the shared memory for storing WALs, that is, multiples of **XLOG_BLCKSZ**, in increments of 8 KB. For details about the value range, see **wal_buffers**. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:wal_buffers	Specifies the shared memory for storing WALs, that is, multiples of **XLOG_BLCKSZ**, in increments of 8 KB. For details about the value range, see **wal_buffers**. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:walwriter_cpu_bind	Specifies the CPU core bound to the WAL write thread. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:bulk_write_ring_size	Specifies the size of the ring buffer used by the operation when a large amount of data is written (for example, the copy operation). This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
cn:bulk_write_ring_size	Specifies the size of a ring buffer used for parallel data import. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:recovery_redo_workers	Specifies the number of **PageRedoWorker** threads corresponding to each **ParseRedoRecord** thread when ultimate RTO is enabled. **recovery_redo_workers** must be used together with **recovery_parse_workers**. The value of **recovery_redo_workers** takes effect only when **recovery_parse_workers** is greater than 1. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:recovery_redo_workers	Specifies the number of **PageRedoWorker** threads corresponding to each **ParseRedoRecord** thread when ultimate RTO is enabled. **recovery_redo_workers** must be used together with **recovery_parse_workers**. The value of **recovery_redo_workers** takes effect only when **recovery_parse_workers** is greater than 1. This parameter is supported in GaussDB V2.0-8.201 or later.
cms:phony_dead_effective_time	Specifies the maximum number of times DN processes are detected as zombie. If the number of times a process is detected as zombie is greater than the specified value, the process is considered to be a zombie process and will be restarted. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_ai_watchdog	Enables or disables the AI watchdog function. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_ai_watchdog	Enables or disables the AI watchdog function. This parameter is supported in GaussDB V2.0-8.201 or later.
enable_unshipping_log	Specifies whether to log statements that are not pushed down. The logs help locate performance issues that may be caused by statements not pushed down. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:max_connections	Specifies the maximum number of concurrent connections. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:max_connections	Specifies the maximum number of concurrent connections. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
dn:resilience_memory_reject_percent	Specifies the dynamic memory usage for escape from memory overload. This parameter is only applied when **use_workload_manager** and **enable_memory_limit** are enabled. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:resilience_memory_reject_percent	Specifies the dynamic memory usage for escape from memory overload. This parameter is only applied when **use_workload_manager** and **enable_memory_limit** are enabled. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:resilience_threadpool_reject_cond	Specifies the proportion of accumulated sessions in the thread pool for escape from overload. This parameter is only applied when **enable_thread_pool** and **use_workload_manager** are enabled. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:resilience_threadpool_reject_cond	Specifies the proportion of accumulated sessions in the thread pool for escape from overload. This parameter is only applied when **enable_thread_pool** and **use_workload_manager** are enabled. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:vacuum_cost_delay	Specifies how long the process sleeps when **vacuum_cost_limit** has been exceeded, in milliseconds. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:vacuum_cost_delay	Specifies how long the process sleeps when **vacuum_cost_limit** has been exceeded, in milliseconds. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:b_format_version	Specifies the SQL behavior control configuration item in B-compatible mode. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:b_format_version	Specifies the forward compatibility in MySQL-compatible mode. This is a configuration item for controlling database platform compatibility behavior. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_control_group	Specifies whether to enable the Cgroups. This parameter must be applied on both CNs and DNs. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
cn:enable_control_group	Specifies whether to enable the Cgroups. This parameter must be applied on both CNs and DNs. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:track_stmt_parameter	After **track_stmt_parameter** is enabled, the executed statements recorded in **statement_history** are not normalized. The complete SQL statement information can be displayed to help the database administrator locate faults. For a simple query, the complete statement information is displayed. For a PBE statement, the complete statement information and information about each variable value are displayed. The format is "query string; parameters: $1=value1,$2=value2, ...". This parameter is used to display full SQL information for users and is not controlled by the **track_activity_query_size** parameter. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:track_stmt_parameter	After **track_stmt_parameter** is enabled, the executed statements recorded in **statement_history** are not normalized. The complete SQL statement information can be displayed to help the database administrator locate faults. For a simple query, the complete statement information is displayed. For a PBE statement, the complete statement information and information about each variable value are displayed. The format is "query string; parameters: $1=value1,$2=value2, ...". This parameter is used to display full SQL information for users and is not controlled by the **track_activity_query_size** parameter. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:track_activity_query_size	Specifies byte counts of the current running commands used to trace each active session. If the number of bytes in a command is greater than the value of this parameter, the command will be truncated. This parameter affects the memory usage of multiple system views. Adjusting the value of this parameter has a great impact on the memory usage. You can estimate the memory usage based on the value calculated from **instr_unique_sql_count** times **track_activity_query_size**. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
cn:track_activity_query_size	Specifies byte counts of the current running commands used to trace each active session. If the number of bytes in a command is greater than the value of this parameter, the command will be truncated. This parameter is supported in GaussDB V2.0-8.201 or later.

Configurable Parameters for Distributed Instances of Version V2.0-8.*x* in Combined Deployment Mode

Table 12-2 Parameters for distributed instances (combined deployment)

Parameter	Description
recyclebin_retention_time	Specifies how long files will be kept in the recycle bin. Files in the recycle bin will be automatically deleted after this length of time. Unit: second
autoanalyze_timeout	Specifies the autoanalyze timeout period. If the duration of autoanalyze on a table exceeds the value of **autoanalyze_timeout**, the autoanalyze operation is automatically canceled. The value **0** indicates that there is no timeout limit. Unit: second
cn:page_version_partitions	Specifies the number of hash table partitions in cached page version information in the memory. This parameter directly affects the hash query efficiency and hash conflict probability, and is valid only when **page_version_check** is not set to **off**. If **page_version_check** is set to **off**, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to a value other than **off**. For details about the value range, see the product documentation.
dn:index_txntype	Sets the index page format to PCR or RCR. This parameter is left unconfigured during system initialization. By default, the created indexes are compatible with the index type (RCR) of earlier versions. Once this parameter is specified, it cannot be left unconfigured again.

Parameter	Description
log_min_duration_statement	Specifies the threshold for logging the duration of a completed statement. If a statement runs for a period greater than or equal to the specified value, its duration will be logged. The value **-1** disables logging statement durations. If this parameter is set to a small value, the load throughput may be affected. Unit: millisecond
datestyle	Specifies the display format for date and time.
max_replication_slots	Specifies the number of log replication slots on the primary node.
timezone	Specifies the time zone for displaying and interpreting time stamps.
cn:auto_increment_offset	Specifies the initial value of an auto-increment column. The auto-increment value is calculated by the following formula: **auto_increment_offset** + *N* × **auto_increment_increment**. *N* is a positive integer. If the value of this parameter is greater than that of **auto_increment_increment**, there will be an error when the values in the auto-increment column automatically increase.
dn:enable_xid_abort_check	Specifies whether to check the status of transaction ID rollback when a transaction is committed.
cn:audit_internal_event	Specifies whether to audit the connections and operations of internal tools cm_agent, gs_clean, and WDRXdb, and whether to audit the logins and logouts from CNs on DNs.
enable_default_ustore_table	Specifies whether to enable the Ustore storage engine by default. If this parameter is set to **on**, all created tables are Ustore tables.
enable_slot_log	Specifies whether to enable primary/standby synchronization for logical replication slots. Currently, only archive slots and backup slots are involved. Set this parameter to **on** only in cloud scenarios where logical replication is enabled. In other scenarios, set this parameter to **off**.
cn:enable_convert_illegal_char	Specifies whether the database supports characters not included the character sets.
cn:codegen_compile_thread_num	Specifies the number of Codegen compilation threads.
cn:enable_workload_rule	Specifies whether to enable SQL throttling.

Parameter	Description
dn:enable_enhance_toast_table	Specifies whether to use the enhanced TOAST out-of-line storage table. The value **on** indicates that the enhanced TOAST out-of-line storage table is used. The value **off** indicates that the TOAST out-of-line storage table is used.
dn:static_thread_pool_num	Specifies the number of threads used to create a static thread pool (static pool). This parameter takes effect only on CNs of a distributed instance after **enable_thread_pool** is enabled.
enable_stream_operator	Specifies the query optimizer's use of streams. If **enable_stream_operator** is disabled, a large number of logs indicating that the plans cannot be pushed down are recorded. If you do not need these logs, you are advised to disable both **enable_unshipping_log** and **enable_stream_operator**. The default value is recommended.
cn:auto_increment_increment	Specifies the auto-increment step of an auto-increment column. The auto-increment value is calculated by the following formula: **auto_increment_offset** + *N* × **auto_increment_increment**. *N* is a positive integer. If the value of this parameter is smaller than that of **auto_increment_offset**, there will be an error when the values in the auto-increment column automatically increase.
dn:support_binary_copy_version	Specifies whether to verify the binary file encoding information when data is imported using COPY FROM in BINARY mode. If forward compatibility is required, leave this parameter empty. Otherwise, retain the default value.
cn:enable_dynamic_sample size	Specifies whether to dynamically adjust the number of sampled rows. For a large table with more than one million rows, the number of sampled rows is dynamically adjusted during statistics collection to improve statistics accuracy.
cn:gs_perf_retention_days	Specifies how many days the flame graph files in HTML format are retained. Unit: day
dn:cost_model_version	Specifies the version of the optimizer cost model. It is a protective parameter. It prevents new optimizer cost models from being applied, so you can keep the current model consistent with the plan of an existing version. If the value of this parameter is changed, many SQL plans may be changed. Exercise caution when modifying this parameter.

Parameter	Description
wal_level	Specifies the level of information to be written to the WAL. This is a required value and cannot be commented out. Determines how much information is written to the WAL. When this parameter is set to **logical**, logical logs are extracted and primary key information is recorded in Xlogs.
cn:sql_mode	Specifies the SQL behavior control configuration item in M-compatible mode.
dn:enable_rls_match_index	Specifies whether indexes of a base table can be scanned based on target predicate conditions in row-level security scenarios. Target scenario: Row level security (RLS) policies are set and enabled in the base table, and the query predicate contains the **unleakproof** system function or **like** operator.
dn:audit_internal_event	Specifies whether to audit the connections and operations of internal tools cm_agent, gs_clean, and WDRXdb, and whether to audit the logins and logouts from CNs on DNs.
cn:llvm_max_memory	Specifies the upper limit of the memory used by IRs (including cached and in-use IRs) generated during compilation in Codegen execution mode. The memory used by Codegen is not applied for by preoccupation. It is a part of **max_dynamic_memory** and is restricted by the **llvm_max_memory** parameter. Unit: KB
dn:fix_func_selection	Specifies whether to optimize the function matching policy. The value **catlist** indicates the catlist sequence is optimized. (The non-B-compatible mode has been optimized. In non-B-compatible mode, system functions are always preferentially selected and executed. The policy in B-compatible mode is the same as that in versions earlier than 505.1.0. An error message indicating that the function is not unique may be displayed, or a system function may be selected for execution.)
enable_seqscan	Specifies whether to enable the optimizer's use of sequential scan plan types. It is impossible to completely suppress sequential scans, but setting this parameter to **off** allows the optimizer to choose other methods if available.
comm_no_delay	Specifies whether to use the **no_delay** attribute of a communication library connection.
enable_recyclebin	Enables or disables the recycle bin in real time.

Parameter	Description
dn:sql_mode	Specifies the SQL behavior control configuration item in M-compatible mode.
cn:instr_unique_sql_combination_options	Specifies the configuration items of combining unique SQL statements of the same type.
dn:enable_dynamic_sample size	Specifies whether to dynamically adjust the number of sampled rows. For a large table with more than one million rows, the number of sampled rows is dynamically adjusted during statistics collection to improve statistics accuracy.
dn:verify_log_buffers	Specifies the size or pages of verifyLog buffers in memory mode. The unit is 8 KB. For example, if the value of this parameter is 4, the requested memory is 4 x 8 KB = 32 KB. This parameter is valid only when **page_version_check** is set to **persistence**. If **page_version_check** is set to another value, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to **persistence**.
cn:convert_illegal_char_mode	Specifies the placeholders of invalid characters that can be displayed on the client when **enable_convert_illegal_char** is enabled. Value range: 95 characters whose decimal codes range from 32 to 126 in the ASCII coding table.
dn:llvm_max_memory	Specifies the upper limit of the memory used by IRs (including cached and in-use IRs) generated during compilation in Codegen execution mode. The memory used by Codegen is not applied for by preoccupation. It is a part of **max_dynamic_memory** and is restricted by the **llvm_max_memory** parameter. Unit: KB
cn:page_version_check	Specifies the type of page version verification. **off** indicates that page version verification is disabled. **memory** indicates that page version verification in pure memory mode is enabled. The page version information will be lost after a restart. **persistence** indicates that persistent page version verification is enabled. The page version information will not be lost after a restart.
undo_space_limit_size	Specifies the threshold for forcibly recycling undo space. When the undo space usage reaches 80% of the threshold, forcible recycling starts. It is recommended that the value of this parameter be greater than or equal to the value of **undo_limit_size_per_transaction**. Unit: 8 KB

Parameter	Description
cn:tde_index_default_encrypt	When **tde_index_default_encrypt** is set to **on** and an index is created based on an encrypted table, the database automatically converts the index to an encrypted index.
recovery_time_target	Specifies the time for the standby node to write and replay logs. Unit: second
dn:enable_convert_illegal_char	Specifies whether the database supports characters not included the character sets.
dn:system_view_version	Determines the version of the system view. All versions are backward compatible. For example, when **system_view_version** is set to **3**, all features of version 2 and version 1 are also supported. For details, see the product documentation.
dn:enable_workload_rule	Specifies whether to enable SQL throttling.
dn:archive_interval	Specifies the archiving interval. Log files are forcibly archived when the period specified by this parameter has elapsed. A large value of this parameter affects the RPO of PITR. The default value is recommended. Unit: second
cn:enable_codegen	Specifies whether code optimization can be enabled. Currently, code optimization uses the LLVM optimization.
dn:codegen_compile_thread_num	Specifies the number of Codegen compilation threads.
max_wal_senders	Specifies the maximum number of WAL sender threads that can be created. Processes created for the following purposes occupy WAL sender threads: Standby DNs connect to primary DNs to obtain physical logs, and logical replication tools connect to primary DNs to obtain logical logs. If this parameter is set to a value smaller than 20, scale-out may fail. The value of this parameter must be smaller than that of **max_connections**.
dn:resilience_ctrlstmt_detect_timelimit	Specifies the execution time of a normal SQL statement that will be marked as a slow SQL statement. The value **0** indicates that slow SQL statements are not identified. A value greater than 0 indicates that a normal SQL statement whose execution time exceeds the value of this parameter is marked as a slow SQL statement. This parameter is only valid for SELECT statements that are executed by non-sysadmin/monitoradmin users. Unit: millisecond

Parameter	Description
cn:a_format_date_timestamp	Specifies whether to enable **current_date**, **current_timestamp**, and **localtimestamp** to return the system time, instead of the transaction start time, when a transaction starts.
cn:page_missing_dirty_check	Specifies whether to enable the verification for pages not marked as dirty. The verification checks whether the modified pages are not marked as dirty. This parameter is only valid when **page_version_check** is not set to **off**. If **page_version_check** is set to **off**, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to a value other than **off**.
cn:cost_model_version	Specifies the version of the optimizer cost model. It is a protective parameter. It prevents new optimizer cost models from being applied, so you can keep the current model consistent with the plan of an existing version. If the value of this parameter is changed, many SQL plans may be changed. Exercise caution when modifying this parameter.
dn:max_compile_functions	Specifies the maximum number of function compilation results stored in the server. Excessive functions and compilation results of stored procedures may occupy large memory space. Setting this parameter to an appropriate value can reduce the memory usage and improve system performance. Before modifying this parameter, determine your application scenario and fully verify the change in a test environment. For details, see the reference document.
dn:tde_index_default_encrypt	When **tde_index_default_encrypt** is set to **on** and an index is created based on an encrypted table, the database automatically converts the index to an encrypted index.

Parameter	Description
dn:recovery_parse_workers	Specifies the number of ParseRedoRecord threads in the ultimate RTO feature. This parameter must be used together with **recovery_redo_workers**. If both **recovery_parse_workers** and **recovery_redo_workers** are greater than **1**, ultimate RTO is enabled. If you do not want to enable ultimate RTO, retain the default value **1** for **recovery_parse_workers**. When enabling ultimate RTO, ensure that **replication_type** is set to **1**. If both ultimate RTO and parallel replay are enabled, only ultimate RTO takes effect, and parallel replay is ineffective. Ultimate RTO does not support column-store tables. Therefore, disable ultimate RTO in a system where column-store tables are used or are to be used. Ultimate RTO also does not support flow control. Flow control is determined by the **recovery_time_target** parameter.
failed_login_attempts	Specifies the maximum number of incorrect password attempts before an account is locked. The account will be automatically unlocked after the time specified in **password_lock_time** elapses. Only the **sysadmin** user can set this parameter.
undo_retention_time	Specifies how long undo logs are kept. This parameter is only used for flashback query. Note: 1. The undo space of the local disk increases. 2. In subsequent incremental backups, the size of the backup set increases, because extra undo content is retained. Unit: second
cn:wdr_snapshot_space_threshold	Specifies the threshold for controlling the space used by snapshots. When the space used by snapshots reaches 80% of the value of this parameter, the control logic of the database is enabled to stabilize the space usage. Unit: KB
dn:page_version_check	Specifies the type of page version verification. **off** indicates that page version verification is disabled. **memory** indicates that page version verification in pure memory mode is enabled. The page version information will be lost after a restart. **persistence** indicates that persistent page version verification is enabled. The page version information will not be lost after a restart.

Parameter	Description
cn:max_compile_functions	Specifies the maximum number of function compilation results stored in the server. Excessive functions and compilation results of stored procedures may occupy large memory space. Setting this parameter to an appropriate value can reduce the memory usage and improve system performance. Before modifying this parameter, determine your application scenario and fully verify the change in a test environment. For details, see the reference document.
cn:page_version_max_num	Specifies the maximum number of page versions that can be cached in memory. This parameter is only valid when **page_version_check** is not set to off. If **page_version_check** is set to **off**, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to a value other than **off**. For details about the value range, see the product documentation.
dn:auto_increment_offset	Specifies the initial value of an auto-increment column. The auto-increment value is calculated by the following formula: **auto_increment_offset** + *N* × **auto_increment_increment**. *N* is a positive integer. If the value of this parameter is greater than that of **auto_increment_increment**, there will be an error when the values in the auto-increment column automatically increase.

Parameter	Description
dn:auto_increment_cache	Specifies the number of reserved auto-increment cache values when auto-increment is triggered by batch insertion or import of auto-increment columns. When auto-increment values are reserved, the auto-increment counter value is updated to the maximum auto-increment cache value. Before the cache values are used up, the auto-increment counter value remains unchanged, and the triggered auto-increment uses the cache values. If this parameter is set to **0**, the auto-increment cache values are automatically reserved. When auto-increment is triggered for the first time, if the number of rows to be inserted into the auto-increment column is known, the number is the reserved value. If the number of rows is unknown, 2^n values are reserved each time. For example, one value is reserved in the first auto-increment, two values are reserved in the second auto-increment, four values are reserved in the third auto-increment, and eight values are reserved for in fourth auto-increment. However, if the number of reserved values exceeds 65,535, 65,535 values are reserved. If this parameter is not set to **0**, the number of reserved cache values is the value of this parameter. When auto-increment is triggered for the first time, if the number of rows to be inserted into the auto-increment column is known, the number is the reserved value. If the number of rows is unknown, the value of **auto_increment_cache** is the number of auto-increment values reserved each time. The reserved cache values are valid only in the statement. If the reserved auto-increment cache values are used up and subsequent INSERT statements trigger auto-increment based on the auto-increment counter, the values in the auto-increment column in the table are discontinuous. This parameter does not affect the auto-increment column in the local temporary table.
dn:enable_codegen	Specifies whether code optimization can be enabled. Currently, code optimization uses the LLVM optimization.
dn:instr_unique_sql_combination_options	Specifies the configuration items of combining unique SQL statements of the same type. If this feature is enabled, the IDs of unique SQL statements of the same type are normalized, and the generated unique SQL strings are normalized.
rewrite_rule	Sets query rewriting rules.

Parameter	Description
dn:page_missing_dirty_check	Specifies whether to enable the verification for pages not marked as dirty. The verification checks whether the modified pages are not marked as dirty. This parameter is only valid when **page_version_check** is not set to **off**. If **page_version_check** is set to **off**, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to a value other than **off**.
dn:wdr_snapshot_full_backup_interval	Specifies the interval at which a full WDR snapshot is created. The interval specified by this parameter is about a number instead of time. For example, if the parameter is set to **12**, a full snapshot and then 11 incremental snapshots are generated for each group. If the parameter is set to **1**, all snapshots generated are full snapshots.
dn:wdr_snapshot_space_threshold	Specifies the threshold for controlling the space used by snapshots. When the space used by snapshots reaches 80% of the value of this parameter, the control logic of the database is enabled to stabilize the space usage. Unit: KB
cn:gs_perf_interval	Specifies the automatic perf data collection interval. The value **0** indicates that the collection is stopped. If the value is greater than 0 and less than 5, the value **5** is used. Unit: minute
dn:gs_perf_interval	Specifies the automatic perf data collection interval. The value **0** indicates that the collection is stopped. If the value is greater than 0 and less than 5, the value **5** is used. Unit: minute
dn:extra_float_digits	Adjusts the number of digits displayed for floating-point values, including float4, float8, and geometric data types. The parameter value is added to the standard number of digits (FLT_DIG or DBL_DIG as appropriate). This parameter can also be set to a negative value to suppress unwanted digits.
cn:disable_keyword_options	Specifies the configuration items for database compatibility. Multiple items are separated by commas (,). An identifier with this parameter set will not be used as a keyword.
track_stmt_stat_level	Controls the level of statement execution tracking.
cn:b_format_dev_version	Specifies the compatibility configuration item of database platform minor versions.

Parameter	Description
dn:copy_special_character_version	Specifies whether to report an error when there are invalid characters during data import and export using COPY FROM.
cn:page_version_recycler_thread_num	Specifies the number of threads for recycling and verifying page version information. This parameter is only valid when **page_version_check** is not set to **off**. If **page_version_check** is set to **off**, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to a value other than **off**.
cn:wdr_snapshot_full_backup_interval	Specifies the interval at which a full WDR snapshot is created. The interval specified by this parameter is about a number instead of time. For example, if the parameter is set to **12**, a full snapshot and then 11 incremental snapshots are generated for each group. If the parameter is set to **1**, all snapshots generated are full snapshots.
cn:extra_float_digits	Adjusts the number of digits displayed for floating-point values, including float4, float8, and geometric data types. The parameter value is added to the standard number of digits (FLT_DIG or DBL_DIG as appropriate). This parameter can also be set to a negative value to suppress unwanted digits.
cn:system_view_version	Determines the version of the system view. All versions are backward compatible. For example, when **system_view_version** is set to **3**, all features of version 2 and version 1 are also supported. For details, see the product documentation.
support_batch_bind	Specifies whether to bind and execute PBE (Parse, Bind, Execute) statements in batches through interfaces such as JDBC, ODBC, and libpq.
cn:check_disconnect_query	Specifies whether to enable the function of terminating statement execution on the server after the client is disconnected due to timeout.
password_effect_time	Specifies the validity period of an account password. Unit: day

Parameter	Description
cn:verify_log_buffers	Specifies the size or pages of verifyLog buffers in memory mode. The unit is 8 KB. For example, if the value of this parameter is 4, the requested memory is 4 x 8 KB = 32 KB. This parameter is valid only when **page_version_check** is set to **persistence**. If **page_version_check** is set to another value, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to **persistence**.
cn:enable_slot_log	Specifies whether to enable primary/standby synchronization for logical replication slots. Currently, only archive slots and backup slots are involved. Set this parameter to **on** only in cloud scenarios where logical replication is enabled. In other scenarios, set this parameter to **off**.
undo_limit_size_per_transaction	Specifies the threshold for forcibly recycling undo space. When the undo space usage reaches 80% of the threshold, forcible recycling starts. It is recommended that the value of this parameter be greater than or equal to the value of **undo_limit_size_per_transaction**. Unit: 8 KB
dn:resilience_ctrlslot_available_maxpercent	Specifies the maximum percentage of threads in the thread pool that can be occupied by slow SQL statements. This parameter is only valid for SELECT statements that are executed by non-sysadmin/ monitoradmin users.
cn:enable_rls_match_index	Specifies whether indexes of a base table can be scanned based on target predicate conditions in row-level security scenarios. Target scenario: Row level security (RLS) policies are set and enabled in the base table, and the query predicate contains the **unleakproof** system function or **like** operator.
dn:auto_increment_increment	Specifies the auto-increment step of an auto-increment column. The auto-increment value is calculated by the following formula: **auto_increment_offset** + *N* × **auto_increment_increment**. *N* is a positive integer. If the value of this parameter is smaller than that of **auto_increment_offset**, there will be an error when the values in the auto-increment column automatically increase.
cn:copy_special_character_version	Specifies whether to report an error when there are invalid characters during data import and export using COPY FROM.

Parameter	Description
behavior_compat_options	Specifies database compatibility configuration items. After the value of **proc_outparam_override** is changed, the database must be connected again or the instance must be rebooted. Otherwise, stored procedures and functions cannot be correctly called.
dn:gs_format_behavior_compat_options	Specifies the configuration items of GaussDB internal system functions.
dn:b_format_dev_version	Specifies the compatibility configuration item of database platform minor versions.
dn:check_disconnect_query	Specifies whether to enable the function of terminating statement execution on the server after the client is disconnected due to timeout.
cn:gs_format_behavior_compat_options	Specifies the configuration items of GaussDB internal system functions.
wdr_snapshot_retention_days	Specifies how many days database monitoring snapshots are saved for.
cn:static_thread_pool_num	Specifies the number of threads used to create a static thread pool (static pool). This parameter takes effect only on CNs of a distributed instance after **enable_thread_pool** is enabled.
audit_system_object	Specifies whether to audit the CREATE, DROP, and ALTER operations on GaussDB database objects. GaussDB database objects include databases, users, schemas, and tables. You can change the value of this parameter to audit only the operations on required database objects. In the scenario where the leader node is forcibly selected, you are advised to set **audit_system_object** to the maximum value and audit all DDL objects. For details about the value range, see the product documentation.
cn:resilience_ctrlslot_available_maxpercent	Specifies the maximum percentage of threads in the thread pool that can be occupied by slow SQL statements. This parameter is only valid for SELECT statements that are executed by non-sysadmin/monitoradmin users.
dn:enable_slot_log	Specifies whether to enable primary/standby synchronization for logical replication slots. Currently, only archive slots and backup slots are involved. Set this parameter to **on** only in cloud scenarios where logical replication is enabled. In other scenarios, set this parameter to **off**.

Parameter	Description
dn:page_version_partitions	Specifies the number of hash table partitions in cached page version information in the memory. This parameter directly affects the hash query efficiency and hash conflict probability, and is valid only when **page_version_check** is not set to **off**. If **page_version_check** is set to **off**, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to a value other than **off**. For details about the value range, see the product documentation.
dn:page_version_recycler_thread_num	Specifies the number of threads for recycling and verifying page version information. This parameter is only valid when **page_version_check** is not set to **off**. If **page_version_check** is set to **off**, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to a value other than **off**.
dn:dcf_thread_effective_time	Specifies the effective time of the DCF flushing thread. This parameter is used to determine whether the disk I/O hangs. If the DCF cannot access I/O resources within the period specified by this parameter, the DCF considers that the thread I/O hangs and a primary/standby switchover is triggered. If this parameter is set to **0**, I/O hang detection is disabled. Unit: second
dn:a_format_date_timestamp	Specifies whether to enable **current_date**, **current_timestamp**, and **localtimestamp** to return the system time, instead of the transaction start time, when a transaction starts.
cn:enable_xid_abort_check	Specifies whether to check the status of transaction ID rollback when a transaction is committed.
session_timeout	Specifies how long to wait before a server connection is disconnected due to inactivity. The value **0** indicates there is no time limit. Unit: second
dn:disable_keyword_options	Specifies the configuration items for database compatibility. Multiple items are separated by commas (,). An identifier with this parameter set will not be used as a keyword.

Parameter	Description
cn:fix_func_selection	Specifies whether to optimize the function matching policy. The value **catlist** indicates the catlist sequence is optimized. (The non-B-compatible mode has been optimized. In non-B-compatible mode, system functions are always preferentially selected and executed. The policy in B-compatible mode is the same as that in versions earlier than 505.1.0. An error message indicating that the function is not unique may be displayed, or a system function may be selected for execution.)
autoanalyze	Specifies whether to automatically collect statistics on tables that have no statistics when a plan is generated.
password_lock_time	Specifies the maximum number of incorrect password attempts before an account is locked. The account will be automatically unlocked after the time specified in **password_lock_time** elapses. Only the **sysadmin** user can set this parameter.
cn:support_binary_copy_version	Specifies whether to verify the binary file encoding information when data is imported using COPY FROM in BINARY mode. If forward compatibility is required, leave this parameter empty. Otherwise, retain the default value.
update_lockwait_timeout	Specifies the maximum duration that a lock waits for concurrent updates on a row to complete when the concurrent update feature is enabled. If the lock wait time exceeds this value, the system will report an error. Unit: millisecond
cn:recovery_parse_workers	Specifies the number of ParseRedoRecord threads in the ultimate RTO feature. This parameter must be used together with **recovery_redo_workers**. If both **recovery_parse_workers** and **recovery_redo_workers** are greater than **1**, ultimate RTO is enabled. If you do not want to enable ultimate RTO, retain the default value **1** for **recovery_parse_workers**. When enabling ultimate RTO, ensure that **replication_type** is set to **1**. If both ultimate RTO and parallel replay are enabled, only ultimate RTO takes effect, and parallel replay is ineffective. Ultimate RTO does not support column-store tables. Therefore, disable ultimate RTO in a system where column-store tables are used or are to be used. Ultimate RTO also does not support flow control. Flow control is determined by the **recovery_time_target** parameter.

Parameter	Description
cn:auto_increment_cache	Specifies the number of reserved auto-increment cache values when auto-increment is triggered by batch insertion or import of auto-increment columns. When auto-increment values are reserved, the auto-increment counter value is updated to the maximum auto-increment cache value. Before the cache values are used up, the auto-increment counter value remains unchanged, and the triggered auto-increment uses the cache values. If this parameter is set to **0**, the auto-increment cache values are automatically reserved. When auto-increment is triggered for the first time, if the number of rows to be inserted into the auto-increment column is known, the number is the reserved value. If the number of rows is unknown, 2^n values are reserved each time. For example, one value is reserved in the first auto-increment, two values are reserved in the second auto-increment, four values are reserved in the third auto-increment, and eight values are reserved for in fourth auto-increment. However, if the number of reserved values exceeds 65,535, 65,535 values are reserved. If this parameter is not set to **0**, the number of reserved cache values is the value of this parameter. When auto-increment is triggered for the first time, if the number of rows to be inserted into the auto-increment column is known, the number is the reserved value. If the number of rows is unknown, the value of **auto_increment_cache** is the number of auto-increment values reserved each time. The reserved cache values are valid only in the statement. If the reserved auto-increment cache values are used up and subsequent INSERT statements trigger auto-increment based on the auto-increment counter, the values in the auto-increment column in the table are discontinuous. This parameter does not affect the auto-increment column in the local temporary table.
cn:enable_enhance_toast_table	Specifies whether to use the enhanced TOAST out-of-line storage table. The value **on** indicates that the enhanced TOAST out-of-line storage table is used. The value **off** indicates that the TOAST out-of-line storage table is used.
cn:archive_interval	Specifies the archiving interval. Log files are forcibly archived when the period specified by this parameter has elapsed. A large value of this parameter affects the RPO of PITR. The default value is recommended. Unit: second

Parameter	Description
dn:convert_illegal_char_mode	Specifies the placeholders of invalid characters that can be displayed on the client when **enable_convert_illegal_char** is enabled. Value range: 95 characters whose decimal codes range from 32 to 126 in the ASCII coding table.
dn:use_workload_manager	Specifies whether to enable resource management. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:use_workload_manager	Specifies whether to enable resource management. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_save_datachanged_timestamp	Specifies whether to record the time when INSERT, UPDATE, DELETE, or EXCHANGE/TRUNCATE/DROP PARTITION is performed on table data. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_save_datachanged_timestamp	Specifies whether to record the time when INSERT, UPDATE, DELETE, or EXCHANGE/TRUNCATE/DROP PARTITION is performed on table data. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:max_process_memory	Specifies the maximum physical memory allowed for a DN. Unit: KB This parameter is supported in GaussDB V2.0-8.201 or later.
cn:max_process_memory	Specifies the maximum physical memory allowed for a CN. Unit: KB This parameter is supported in GaussDB V2.0-8.201 or later.
dn:standby_shared_buffers_fraction	Specifies the **shared_buffers** proportion used on the server where a standby instance is deployed. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:standby_shared_buffers_fraction	Specifies the **shared_buffers** proportion used on the server where a standby instance is deployed. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:pagewriter_sleep	Specifies the interval (in milliseconds) for the page writer thread to flush dirty pages to disks after the incremental checkpoint is enabled. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:pagewriter_sleep	Specifies the interval (in milliseconds) for the page writer thread to flush dirty pages to disks after the incremental checkpoint is enabled. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
dn:xloginsert_locks	Specifies the number of locks on concurrent write-ahead logging. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:xloginsert_locks	Specifies the number of locks on concurrent write-ahead logging. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_light_proxy	Specifies whether the optimizer optimizes the execution of simple queries on CNs. This parameter does not take effect if the character set of the application side does not match that of the kernel side. You are advised to set the character set to UTF-8 when creating a database. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_light_proxy	Specifies whether the optimizer optimizes the execution of simple queries on CNs. This parameter does not take effect if the character set of the application side does not match that of the kernel side. You are advised to set the character set to UTF-8 when creating a database. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:synchronous_commit	Specifies the synchronization mode of the current transaction. Generally, logs generated by a transaction are synchronized in the following sequence: The primary node writes the logs to the local memory. The primary node writes logs in the local memory to the local file system. The primary node flushes the logs in the local file system to disks. The primary node sends the logs to the standby node. The standby node receives the logs and saves them to its local memory. The standby node writes the logs in the local memory to the local file system. The standby node flushes the logs in the local file system to disks. The standby node replays the logs to complete the incremental update of data files. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
cn:synchronous_commit	Specifies the synchronization mode of the current transaction. Generally, logs generated by a transaction are synchronized in the following sequence: The primary node writes the logs to the local memory. The primary node writes logs in the local memory to the local file system. The primary node flushes the logs in the local file system to disks. The primary node sends the logs to the standby node. The standby node receives the logs and saves them to its local memory. The standby node writes the logs in the local memory to the local file system. The standby node flushes the logs in the local file system to disks. The standby node replays the logs to complete the incremental update of data files. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_incremental_checkpoint	Specifies whether to enable incremental checkpoint. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_incremental_checkpoint	Specifies whether to enable incremental checkpoint. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:bypass_workload_manager	Enables or disables independent I/O control. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:bypass_workload_manager	Enables or disables independent I/O control. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_resource_track	Specifies whether to enable real-time resource monitoring. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_resource_track	Specifies whether to enable real-time resource monitoring. This parameter must be applied on both CNs and DNs. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_instr_rt_percentile	Specifies whether to enable the function of calculating the response time of 80% and 95% of the SQL statements in the system. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_instr_rt_percentile	Specifies whether to enable the function of calculating the response time of 80% and 95% of the SQL statements in the system. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
dn:enable_instr_cpu_timer	Specifies whether to capture the CPU time consumed during SQL statement execution. In the x86-based centralized deployment scenario where 32 vCPUs and 256 GB of memory are configured, the performance tested by BenchmarkSQL 5.0 fluctuates by about 3.5% by enabling or disabling this parameter. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_instr_cpu_timer	Specifies whether to capture the CPU time consumed during SQL statement execution. In the x86-based centralized deployment scenario where 32 vCPUs and 256 GB of memory are configured, the performance tested by BenchmarkSQL 5.0 fluctuates by about 3.5% by enabling or disabling this parameter. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:track_counts	Determines whether to enable collection of statistics on database activities. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:track_counts	Determines whether to enable collection of statistics on database activities. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_user_metric_persistent	Specifies whether to dump the historical monitoring data of user resources. If this parameter is set to **on**, data in the **PG_TOTAL_USER_RESOURCE_INFO** view is periodically sampled and saved to the system catalog **GS_WLM_USER_RESOURCE_HISTORY**. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_user_metric_persistent	Specifies whether to dump the historical monitoring data of user resources. If this parameter is set to **on**, data in the **PG_TOTAL_USER_RESOURCE_INFO** view is periodically sampled and saved to the system catalog **GS_WLM_USER_RESOURCE_HISTORY**. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:log_hostname	By default, connection log messages only show the IP address of the connecting host. The host name can be recorded when this parameter is set to **on**. It may take some time to parse the host name. Therefore, the database performance may be affected. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
cn:log_hostname	By default, connection log messages only show the IP address of the connecting host. The host name can be recorded when this parameter is set to **on**. It may take some time to parse the host name. Therefore, the database performance may be affected. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_instr_track_wait	Specifies whether to enable real-time collection of wait event information. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_instr_track_wait	Specifies whether to enable real-time collection of wait event information. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_instance_metric_persistent	Specifies whether to dump instance resource monitoring data. When this parameter is set to **on**, the instance monitoring data is saved to the system catalog **GS_WLM_INSTANCE_HISTORY**. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_instance_metric_persistent	Specifies whether to dump instance resource monitoring data. When this parameter is set to **on**, the instance monitoring data is saved to the system catalog **GS_WLM_INSTANCE_HISTORY**. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_logical_io_statistics	Specifies whether to collect logical I/O statistics during resource monitoring. If this function is enabled, the **read_kbytes**, **write_kbytes**, **read_counts**, **write_counts**, **read_speed**, and **write_speed** columns in the PG_TOTAL_USER_RESOURCE_INFO view will collect statistics on the byte count, number of times, and speed of logical read and write. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
cn:enable_logical_io_statistics	Specifies whether to collect logical I/O statistics during resource monitoring. If this function is enabled, the **read_kbytes**, **write_kbytes**, **read_counts**, **write_counts**, **read_speed**, and **write_speed** columns in the PG_TOTAL_USER_RESOURCE_INFO view will collect statistics on the byte count, number of times, and speed of logical read and write. Columns related to logical read and write in the system catalogs GS_WLM_USER_RESOURCE_HISTORY and GS_WLM_INSTANCE_HISTORY will collect statistics on the logical read and write of related users and instances. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_sort	Specifies whether the query optimizer uses sort methods. It is impossible to suppress explicit sorts entirely, but setting this variable to **off** encourages the optimizer to choose other methods if available. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_sort	Specifies whether the query optimizer uses sort methods. It is impossible to suppress explicit sorts entirely, but setting this variable to **off** encourages the optimizer to choose other methods if available. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_material	Specifies whether the query optimizer uses materialization. It is impossible to suppress materialization entirely, but setting this variable to **off** prevents the optimizer from inserting materialized nodes. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_material	Specifies whether the query optimizer uses materialization. It is impossible to suppress materialization entirely, but setting this variable to **off** prevents the optimizer from inserting materialized nodes. This parameter is supported in GaussDB V2.0-8.201 or later.
gtm:gtm_max_trans	Specifies the maximum number of connections accepted by the GTM. You are advised not to change the value. This parameter is supported in GaussDB V2.0-8.201 or later.
gtm:csn_sync_interval	Specifies the interval for synchronizing CSNs between the primary and standby GTMs, in seconds. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
gtm:restore_duration	Specifies the reply interval of XIDs or CSNs on the GTM, that is, the number of XIDs or CSNs for reply. This parameter is supported in GaussDB V2.0-8.201 or later.
cms:coordinator_heartbeat_timeout	Specifies the heartbeat timeout (in seconds) that triggers the automatic removal of faulty CNs. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:client_min_messages	Specifies which level of messages will be sent to the client. A higher level covers the messages of all the lower levels. The lower the level is, the fewer messages are sent. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:client_min_messages	Specifies which level of messages will be sent to the client. A higher level covers the messages of all the lower levels. The lower the level is, the fewer messages are sent. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:pagewriter_thread_num	Specifies the number of threads for background page flushing after the incremental checkpoint is enabled. Dirty pages are flushed in sequence to disks, updating recovery points. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:pagewriter_thread_num	Specifies the number of threads for background page flushing after the incremental checkpoint is enabled. Dirty pages are flushed in sequence to disks, updating recovery points. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
dn:bgwriter_delay	Specifies the interval at which the background writer writes dirty shared buffers, in milliseconds. Each time, the background writer initiates write operations for some dirty buffers. In full checkpoint mode, the **bgwriter_lru_maxpages** parameter is used to control the amount of data to be written each time, and the process is restarted after the period of hibernation specified by **bgwriter_delay** (in milliseconds). In incremental checkpoint mode, the number of target idle buffer pages is calculated based on the value of **candidate_buf_percent_target**. If the number of idle buffer pages is insufficient, a batch of pages are flushed to disks at the interval specified by **bgwriter_delay** (in milliseconds). The number of flushed pages is calculated based on the target difference percentage. The maximum number of flushed pages is limited by **max_io_capacity**. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:bgwriter_delay	Specifies the interval at which the background writer writes dirty shared buffers, in milliseconds. Each time, the background writer initiates write operations for some dirty buffers. In full checkpoint mode, the **bgwriter_lru_maxpages** parameter is used to control the amount of data to be written each time, and the process is restarted after the period of hibernation specified by **bgwriter_delay** (in milliseconds). In incremental checkpoint mode, the number of target idle buffer pages is calculated based on the value of **candidate_buf_percent_target**. If the number of idle buffer pages is insufficient, a batch of pages are flushed to disks at the interval specified by **bgwriter_delay** (in milliseconds). The number of flushed pages is calculated based on the target difference percentage. The maximum number of flushed pages is limited by **max_io_capacity**. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:max_files_per_process	Specifies the maximum number of files that can be opened simultaneously by each server process. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
dn:candidate_buf_percent_target	Specifies the proportion (%) of available buffers in **shared_buffer** in the candidate buffer chain when the incremental checkpoint is enabled. If the percentage is less than the value of this parameter, the **bgwriter** thread starts to flush dirty pages based on the **max_io_capacity** parameter. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:candidate_buf_percent_target	Specifies the proportion (%) of available buffers in **shared_buffer** in the candidate buffer chain when the incremental checkpoint is enabled. If the percentage is less than the value of this parameter, the **bgwriter** thread starts to flush dirty pages based on the **max_io_capacity** parameter. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:checkpoint_segments	Specifies the minimum number of WAL segment files in the period specified by **checkpoint_timeout**. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:checkpoint_segments	Specifies the minimum number of WAL segment files in the period specified by **checkpoint_timeout**. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:checkpoint_timeout	Specifies the maximum time between automatic WAL checkpoints, in seconds. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:checkpoint_timeout	Specifies the maximum time between automatic WAL checkpoints, in seconds. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:incremental_checkpoint_timeout	Specifies the maximum interval (in seconds) between automatic WAL checkpoints when the incremental checkpoint is enabled. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:incremental_checkpoint_timeout	Specifies the maximum interval (in seconds) between automatic WAL checkpoints when the incremental checkpoint is enabled. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:wal_writer_delay	Specifies the delay (in milliseconds) between activity rounds for the WAL writer. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:wal_writer_delay	Specifies the delay (in milliseconds) between activity rounds for the WAL writer. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
dn:wal_buffers	Specifies the shared memory for storing WALs, that is, multiples of **XLOG_BLCKSZ**, in increments of 8 KB. For details about the value range, see **wal_buffers**. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:wal_buffers	Specifies the shared memory for storing WALs, that is, multiples of **XLOG_BLCKSZ**, in increments of 8 KB. For details about the value range, see **wal_buffers**. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:walwriter_cpu_bind	Specifies the CPU core bound to the WAL write thread. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:bulk_write_ring_size	Specifies the size of the ring buffer used by the operation when a large amount of data is written (for example, the copy operation). This parameter is supported in GaussDB V2.0-8.201 or later.
cn:bulk_write_ring_size	Specifies the size of a ring buffer used for parallel data import. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:recovery_redo_workers	Specifies the number of **PageRedoWorker** threads corresponding to each **ParseRedoRecord** thread when ultimate RTO is enabled. **recovery_redo_workers** must be used together with **recovery_parse_workers**. The value of **recovery_redo_workers** takes effect only when **recovery_parse_workers** is greater than 1. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:recovery_redo_workers	Specifies the number of **PageRedoWorker** threads corresponding to each **ParseRedoRecord** thread when ultimate RTO is enabled. **recovery_redo_workers** must be used together with **recovery_parse_workers**. The value of **recovery_redo_workers** takes effect only when **recovery_parse_workers** is greater than 1. This parameter is supported in GaussDB V2.0-8.201 or later.
cms:phony_dead_effective_time	Specifies the maximum number of times DN processes are detected as zombie. If the number of times a process is detected as zombie is greater than the specified value, the process is considered to be a zombie process and will be restarted. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
dn:enable_ai_watchdog	Enables or disables the AI watchdog function. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_ai_watchdog	Enables or disables the AI watchdog function. This parameter is supported in GaussDB V2.0-8.201 or later.
enable_unshipping_log	Specifies whether to log statements that are not pushed down. The logs help locate performance issues that may be caused by statements not pushed down. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:max_connections	Specifies the maximum number of concurrent connections. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:max_connections	Specifies the maximum number of concurrent connections. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:resilience_memory_reject_percent	Specifies the dynamic memory usage for escape from memory overload. This parameter is only applied when **use_workload_manager** and **enable_memory_limit** are enabled. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:resilience_memory_reject_percent	Specifies the dynamic memory usage for escape from memory overload. This parameter is only applied when **use_workload_manager** and **enable_memory_limit** are enabled. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:resilience_threadpool_reject_cond	Specifies the proportion of accumulated sessions in the thread pool for escape from overload. This parameter is only applied when **enable_thread_pool** and **use_workload_manager** are enabled. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:resilience_threadpool_reject_cond	Specifies the proportion of accumulated sessions in the thread pool for escape from overload. This parameter is only applied when **enable_thread_pool** and **use_workload_manager** are enabled. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
dn:vacuum_cost_delay	Specifies how long the process sleeps when **vacuum_cost_limit** has been exceeded, in milliseconds. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:vacuum_cost_delay	Specifies how long the process sleeps when **vacuum_cost_limit** has been exceeded, in milliseconds. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:b_format_version	Specifies the SQL behavior control configuration item in B-compatible mode. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:b_format_version	Specifies the forward compatibility in MySQL-compatible mode. This is a configuration item for controlling database platform compatibility behavior. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_control_group	Specifies whether to enable the Cgroups. This parameter must be applied on both CNs and DNs. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:enable_control_group	Specifies whether to enable the Cgroups. This parameter must be applied on both CNs and DNs. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:track_stmt_parameter	After **track_stmt_parameter** is enabled, the executed statements recorded in **statement_history** are not normalized. The complete SQL statement information can be displayed to help the database administrator locate faults. For a simple query, the complete statement information is displayed. For a PBE statement, the complete statement information and information about each variable value are displayed. The format is "query string; parameters: $1=value1,$2=value2, ...". This parameter is used to display full SQL information for users and is not controlled by the **track_activity_query_size** parameter. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
cn:track_stmt_parameter	After **track_stmt_parameter** is enabled, the executed statements recorded in **statement_history** are not normalized. The complete SQL statement information can be displayed to help the database administrator locate faults. For a simple query, the complete statement information is displayed. For a PBE statement, the complete statement information and information about each variable value are displayed. The format is "query string; parameters: $1=value1,$2=value2, ...". This parameter is used to display full SQL information for users and is not controlled by the **track_activity_query_size** parameter. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:track_activity_query_size	Specifies byte counts of the current running commands used to trace each active session. If the number of bytes in a command is greater than the value of this parameter, the command will be truncated. This parameter affects the memory usage of multiple system views. Adjusting the value of this parameter has a great impact on the memory usage. You can estimate the memory usage based on the value calculated from **instr_unique_sql_count** times **track_activity_query_size**. This parameter is supported in GaussDB V2.0-8.201 or later.
cn:track_activity_query_size	Specifies byte counts of the current running commands used to trace each active session. If the number of bytes in a command is greater than the value of this parameter, the command will be truncated. This parameter is supported in GaussDB V2.0-8.201 or later.

Configurable Parameters for Centralized Instances of Version V2.0-8.*x*

Table 12-3 Parameters for centralized instances

Parameter	Description
dn:qrw_inlist2join_optmode	Specifies whether to enable inlist-to-join query rewriting.
dn:recovery_max_workers	Specifies the number of concurrent replayer threads.
dn:enable_auto_clean_unique_sql	Specifies whether to enable automatic cleaning of unique SQL statements in hash tables.

Parameter	Description
dn:gs_perf_retention_days	Specifies how many days the flame graph files in HTML format are retained. Unit: day
dn:page_version_max_num	Specifies the maximum number of page versions that can be cached in memory. This parameter is only valid when **page_version_check** is not set to off. If **page_version_check** is set to **off**, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to a value other than **off**. For details about the value range, see the product documentation.
log_autovacuum_min_duration	Specifies the interval which should elapse before autovacuum operations are logged. Autovacuum operations equal to or beyond the specified interval will be logged. If it is set to **0**, all autovacuum operations will be logged. If it is set to **-1**, no autovacuum operations will be logged.
log_min_duration_statement	Specifies the threshold for logging the duration of a completed statement. If a statement runs for a period greater than or equal to the specified value, its duration will be logged. The value **-1** disables logging statement durations. If this parameter is set to a small value, the load throughput may be affected. Unit: millisecond
dn:max_connections	Specifies the maximum number of concurrent connections to DNs. The value of this parameter must be greater than that of **max_wal_senders**.
datestyle	Specifies the display format for date and time.
max_replication_slots	Specifies the number of log replication slots on the primary node.
timezone	Specifies the time zone for displaying and interpreting time stamps.
dn:enable_xid_abort_check	Specifies whether to check the status of transaction ID rollback when a transaction is committed.
enable_slot_log	Specifies whether to enable primary/standby synchronization for logical replication slots. Currently, only archive slots and backup slots are involved. Set this parameter to **on** only in cloud scenarios where logical replication is enabled. In other scenarios, set this parameter to **off**.

Parameter	Description
dn:enable_enhance_toast_table	Specifies whether to use the enhanced TOAST out-of-line storage table. The value **on** indicates that the enhanced TOAST out-of-line storage table is used. The value **off** indicates that the TOAST out-of-line storage table is used.
dn:support_binary_copy_version	Specifies whether to verify the binary file encoding information when data is imported using COPY FROM in BINARY mode.
dn:recovery_time_target	Specifies the time for the standby node to write and replay logs. Unit: second
plat_compat_allow_public_key_retrieval	Specifies the database configuration item in M-compatible mode. This parameter specifies whether a client can request the RSA public key. **on**: The kernel allows the client to request the RSA public key for password transmission encryption. **off**: The client is not allowed to request the RSA public key.
dn:audit_xid_info	Determines whether to record the transaction IDs of SQL statements in detail_info. **0**: The transaction IDs are not recorded. **1**: The transaction IDs are recorded.
dn:cost_model_version	Specifies the version of the optimizer cost model. It is a protective parameter. It prevents new optimizer cost models from being applied, so you can keep the current model consistent with the plan of an existing version. If the value of this parameter is changed, many SQL plans may be changed. Exercise caution when modifying this parameter.
wal_level	Specifies the level of information to be written to the WAL. This is a required value and cannot be commented out. Determines how much information is written to the WAL. When this parameter is set to **logical**, logical logs are extracted and primary key information is recorded in Xlogs.
dn:enable_rls_match_index	Specifies whether indexes of a base table can be scanned based on target predicate conditions in row-level security scenarios. Target scenario: Row level security (RLS) policies are set and enabled in the base table, and the query predicate contains the **unleakproof** system function or **like** operator.
dn:enable_pbe_optimization	Specifies whether the optimizer optimizes the query plan for statements executed in Parse Bind Execute (PBE) mode.

Parameter	Description
dn:audit_internal_event	Specifies whether to audit the connections and operations of internal tools cm_agent, gs_clean, and WDRXdb, and whether to audit the logins and logouts from CNs on DNs.
dn:numa_distribute_mode	Specifies the distribution of some shared data and threads among NUMA nodes. This parameter is used to optimize the performance of large-scale Arm servers with multiple NUMA nodes. Generally, you do not need to set this parameter. This parameter cannot be set to **all** for instances deployed on x86-based servers.
dn:max_compile_packages	Specifies the maximum number of package compilation results stored in the server. Default values for different specifications are calculated by rounding down (**max_process_memory** x 2%)/4.4, in MB.
dn:fix_func_selection	Specifies whether to optimize the function matching policy. The value **catlist** indicates the catlist sequence is optimized. (The non-B-compatible mode has been optimized. In non-B-compatible mode, system functions are always preferentially selected and executed. The policy in B-compatible mode is the same as that in versions earlier than 505.1.0. An error message indicating that the function is not unique may be displayed, or a system function may be selected for execution.)
enable_seqscan	Specifies whether to enable the optimizer's use of sequential scan plan types. It is impossible to completely suppress sequential scans, but setting this parameter to **off** allows the optimizer to choose other methods if available.
dn:default_limit_rows	Specifies the estimated number of rows to return by default for generating a generic plan, that is, the default value for the LIMIT clause. If this parameter is set to a negative number, the value is converted to a percentage, for example, **-5** is equivalent to 5%, indicating that 5% of the total rows will be returned.
dn:sql_mode	Specifies the SQL behavior control configuration item in M-compatible mode.
dn:enable_dynamic_samplesize	Specifies whether to dynamically adjust the number of sampled rows. For a large table with more than one million rows, the number of sampled rows is dynamically adjusted during statistics collection to improve statistics accuracy.

Parameter	Description
dn:undo_space_limit_size	Specifies the threshold for forcibly recycling undo space. When the undo space usage reaches 80% of the threshold, forcible recycling starts. It is recommended that the value of this parameter be greater than or equal to the value of **undo_limit_size_per_transaction**. Unit: 8 KB
dn:umdk_enabled	Specifies whether UMDK is enabled for the primary and standby DNs of the current instance. If the UMDK protocol is used for communication between the primary and standby DNs, the related log keyword on DNs is **umdk**. If the TCP protocol is used for communication between the primary and standby DNs, logs are recorded.
dn:verify_log_buffers	Specifies the size or pages of verifyLog buffers in memory mode. The unit is 8 KB. For example, if the value of this parameter is 4, the requested memory is 4 x 8 KB = 32 KB. This parameter is valid only when **page_version_check** is set to **persistence**. If **page_version_check** is set to another value, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to **persistence**.
dn:llvm_max_memory	Specifies the upper limit of the memory used by IRs (including cached and in-use IRs) generated during compilation in Codegen execution mode. The memory used by Codegen is not applied for by preoccupation. It is a part of **max_dynamic_memory** and is restricted by the **llvm_max_memory** parameter. Unit: KB
dn:enable_control_group	Specifies whether to enable the Cgroups.
dn:local_syscache_threshold	Specifies the size of system catalog cache in a session. Unit: KB
dn:enable_convert_illegal_char	Specifies whether the database supports characters not included the character sets.
dn:system_view_version	Determines the version of the system view. All versions are backward compatible. For example, when **system_view_version** is set to **3**, all features of version 2 and version 1 are also supported. For details, see the product documentation.

Parameter	Description
track_activity_query_size	Specifies the maximum number of bytes to be logged for each SQL statement. If the number of bytes of a SQL statement exceeds the specified parameter value, the SQL statement will be truncated. Setting this parameter to a large value can lead to excessive memory usage, potentially causing the system to run out of memory. You are advised to set this parameter to a value no more than 4096.
dn:enable_workload_rule	Specifies whether to enable SQL throttling.
dn:archive_interval	Specifies the archiving interval. Log files are forcibly archived when the period specified by this parameter has elapsed. A large value of this parameter affects the RPO of PITR. The default value is recommended. Unit: second
div_precision_increment	Specifies the database configuration item in M-compatible mode. This is a session-level parameter, which is used to set the value of precision that the division result can improve. The final precision is the precision of the first operation parameter added by the value of this parameter.
lockwait_timeout	Specifies the maximum duration that a lock waits for concurrent updates on a row to complete when the concurrent update feature is enabled. If the lock wait time exceeds this value, the system will report an error. Unit: millisecond
dn:adaptive_cost_min_time	Specifies the execution duration threshold of SQL statements for cardinality feedback collection. Only the feedback of statements whose execution duration is greater than the value of this parameter is collected. Unit: millisecond
dn:audit_dml_state_select	Determines whether to audit the SELECT operation.
dn:codegen_compile_thread_num	Specifies the number of Codegen compilation threads.
dn:enable_vectordb	Specifies whether vector indexes can be created, inserted, updated, and queried.

Parameter	Description
max_wal_senders	Specifies the maximum number of WAL sender threads that can be created. Processes created for the following purposes occupy WAL sender threads: Standby DNs connect to primary DNs to obtain physical logs, and logical replication tools connect to primary DNs to obtain logical logs. If this parameter is set to a value smaller than 20, scale-out may fail. The value of this parameter must be smaller than that of **max_connections**.
dn:max_concurrent_autonomous_transactions	Specifies the maximum number of autonomous transaction connections, that is, the maximum number of concurrent autonomous transactions executed at the same time. If this parameter is set to **0**, autonomous transactions cannot be executed. The theoretical maximum value is **10000**. Set this parameter based on workload requirements and hardware configurations. It is recommended that this parameter be set to a value less than or equal to 1/10 of **max_connections**.
dn:enable_extension	Specifies whether database extensions can be created. This parameter can be used only in specific scenarios after evaluation. Generally, this parameter does not need to be adjusted.
dn:max_compile_functions	Specifies the maximum number of function compilation results stored in the server. Excessive functions and compilation results of stored procedures may occupy large memory space. Setting this parameter to an appropriate value can reduce the memory usage and improve system performance. Before modifying this parameter, determine your application scenario and fully verify the change in a test environment. For details, see the reference document.
dn:auto_explain_log_min_duration	Specifies the minimum duration of execution plans that are automatically printed. Only execution plans whose duration is greater than the value of **auto_explain_log_min_duration** will be printed. Unit: millisecond
dn:enable_default_ustore_table	Specifies whether to enable the Ustore storage engine by default. If this parameter is set to **on**, all created tables are Ustore tables.
dn:enable_vacuum_control	Specifies whether to move the database permanent thread **autoVacuumWorker** to the **Vacuum** Cgroup.

Parameter	Description
dn:recovery_parse_workers	Specifies the number of ParseRedoRecord threads in the ultimate RTO feature. This parameter must be used together with **recovery_redo_workers**. If both **recovery_parse_workers** and **recovery_redo_workers** are greater than **1**, ultimate RTO is enabled. If you do not want to enable ultimate RTO, retain the default value **1** for **recovery_parse_workers**. When enabling ultimate RTO, ensure that **replication_type** is set to **1**. If both ultimate RTO and parallel replay are enabled, only ultimate RTO takes effect, and parallel replay is ineffective. Ultimate RTO does not support column-store tables. Therefore, disable ultimate RTO in a system where column-store tables are used or are to be used. Ultimate RTO also does not support flow control. Flow control is determined by the **recovery_time_target** parameter.
dn:page_version_check	Specifies the type of page version verification. **off** indicates that page version verification is disabled. **memory** indicates that page version verification in pure memory mode is enabled. The page version information will be lost after a restart. **persistence** indicates that persistent page version verification is enabled. The page version information will not be lost after a restart.
dn:track_activity_query_size	Specifies the maximum number of bytes to be logged for each SQL statement. If the number of bytes of a SQL statement exceeds the specified parameter value, the SQL statement will be truncated. Setting this parameter to a large value can lead to excessive memory usage, potentially causing the system to run out of memory. You are advised to set this parameter to a value no more than 4096.
dn:gsivfflat_secondary_probes	Specifies the number of level-2 inverted indexes to be searched. If the value exceeds the total number of level-2 inverted indexes, the entire table is searched.
dn:num_internal_lock_partitions	Specifies the number of internal lightweight lock partitions. Changing the value of this parameter affects performance and memory usage. Before modifying this parameter, determine your application scenario and fully verify the change in a test environment. For details, see the reference document.

Parameter	Description
dn:enable_auto_explain	Specifies whether to automatically print execution plans. This parameter is used to locate slow stored procedures or slow queries.
dn:audit_function_exec	Specifies whether to record the audit information during the execution of the stored procedures, anonymous blocks, or user-defined functions (excluding system functions). The value **0** means to disable the function, and **1** means to enable it.
dn:auto_increment_cache	Specifies the number of reserved auto-increment cache values when auto-increment is triggered by batch insertion or import of auto-increment columns. When auto-increment values are reserved, the auto-increment counter value is updated to the maximum auto-increment cache value. Before the cache values are used up, the auto-increment counter value remains unchanged, and the triggered auto-increment uses the cache values. If this parameter is set to **0**, the auto-increment cache values are automatically reserved. When auto-increment is triggered for the first time, if the number of rows to be inserted into the auto-increment column is known, the number is the reserved value. If the number of rows is unknown, 2^n values are reserved each time. For example, one value is reserved in the first auto-increment, two values are reserved in the second auto-increment, four values are reserved in the third auto-increment, and eight values are reserved for in fourth auto-increment. However, if the number of reserved values exceeds 65,535, 65,535 values are reserved. If this parameter is not set to **0**, the number of reserved cache values is the value of this parameter. When auto-increment is triggered for the first time, if the number of rows to be inserted into the auto-increment column is known, the number is the reserved value. If the number of rows is unknown, the value of **auto_increment_cache** is the number of auto-increment values reserved each time. The reserved cache values are valid only in the statement. If the reserved auto-increment cache values are used up and subsequent INSERT statements trigger auto-increment based on the auto-increment counter, the values in the auto-increment column in the table are discontinuous. This parameter does not affect the auto-increment column in the local temporary table.

Parameter	Description
dn:enable_codegen	Specifies whether code optimization can be enabled. Currently, code optimization uses the LLVM optimization.
dn:instr_unique_sql_combination_options	Specifies the configuration items of combining unique SQL statements of the same type. If this feature is enabled, the IDs of unique SQL statements of the same type are normalized, and the generated unique SQL strings are normalized.
plan_cache_mode	Specifies the policy of generating and caching execution plans of prepared statements.
dn:group_concat_max_len	Specifies the maximum permitted result length in bytes for the GROUP_CONCAT() function.
dn:hadr_recovery_point_target	Specifies the time allowed for the standby instance to flush logs to disks in streaming DR. This ensures that the log difference between the primary and standby instances is controlled within the period specified by **hadr_recovery_point_target** during the switchover and the standby instance can be promoted to primary. If this parameter is set to a small value, the performance of the primary node is affected. If it is set to a large value, the log flow is not effectively controlled. The value **0** indicates that log flow control is disabled. Unit: second
dn:behavior_compat_options	Specifies database compatibility configuration items. After the value of **proc_outparam_override** is changed, the database must be connected again or the instance must be rebooted. Otherwise, stored procedures and functions cannot be correctly called.
dn:page_missing_dirty_check	Specifies whether to enable the verification for pages not marked as dirty. The verification checks whether the modified pages are not marked as dirty. This parameter is only valid when **page_version_check** is not set to **off**. If **page_version_check** is set to **off**, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to a value other than **off**.
dn:enable_security_policy	Controls whether unified auditing and dynamic data masking policies are applied.

Parameter	Description
dn:wdr_snapshot_full_backup_interval	Specifies the interval at which a full WDR snapshot is created. The interval specified by this parameter is about a number instead of time. For example, if the parameter is set to **12**, a full snapshot and then 11 incremental snapshots are generated for each group. If the parameter is set to **1**, all snapshots generated are full snapshots.
dn:vacuum_defer_cleanup_age	Specifies the number of transactions used by VACUUM.
cms:datastorage_threshold_value_check	Specifies the disk usage threshold to put a database node into read-only mode. If the disk usage of a data directory exceeds this threshold, the database node is automatically changed to read-only. Unit: percentage (%)
dn:audit_thread_num	Specifies the number of audit threads. Value range: 1 to 48.
dn:wdr_snapshot_space_threshold	Specifies the threshold for controlling the space used by snapshots. When the space used by snapshots reaches 80% of the value of this parameter, the control logic of the database is enabled to stabilize the space usage. Unit: KB
dn:m_format_behavior_compat_options	Specifies the configuration items for the M-compatible mode.
dn:gs_perf_interval	Specifies the automatic perf data collection interval. The value **0** indicates that the collection is stopped. If the value is greater than 0 and less than 5, the value **5** is used. Unit: minute
dn:extra_float_digits	Adjusts the number of digits displayed for floating-point values, including float4, float8, and geometric data types. The parameter value is added to the standard number of digits (FLT_DIG or DBL_DIG as appropriate). This parameter can also be set to a negative value to suppress unwanted digits.
dn:audit_dml_state	Determines whether to audit the INSERT, UPDATE, and DELETE operations on a specific table. **0**: These operations are not audited. **1**: These operations are audited.
max_allowed_packet	Specifies the database configuration item in M-compatible mode. This parameter indicates the upper limit of the return value of a string function. The value must be a multiple of 1,024. Unit: KB
track_stmt_stat_level	Controls the level of statement execution tracking.

Parameter	Description
dn:max_standby_archive_delay	Specifies the wait period before queries on a standby node are canceled when the queries conflict with WAL processing and archiving in hot standby mode. **-1** indicates that the standby node waits until the conflicting queries are complete. Unit: millisecond
dn:copy_special_character_version	Specifies whether to report an error when there are invalid characters during data import and export using COPY FROM.
log_temp_files	Specifies whether to log temporary file details when they are deleted. A positive value indicates that the deletion information of temporary files whose size is larger than the value specified by **log_temp_files** is recorded. **0** indicates that the deletion information of all temporary files is recorded. **-1** indicates that the deletion information of any temporary files is not recorded. Unit: KB
enable_wdr_snapshot	Specifies whether to enable database monitoring snapshots on the standby node.
dn:m_format_dev_version	Specifies the compatibility configuration item of database platform minor versions.
password_effect_time	Specifies the validity period of an account password. Unit: day
dn:random_page_cost	Specifies the estimated cost for the optimizer to fetch an out-of-sequence disk page.
dn:immediate_analyze_threshold	Specifies the threshold for automatically analyzing inserted data. When the amount of inserted data reaches the original data amount multiplied by the value of **immediate_analyze_threshold**, and the total number of rows exceeds 100, ANALYZE is automatically triggered.
dn:auto_increment_increment	Specifies the auto-increment step of an auto-increment column. The auto-increment value is calculated by the following formula: **auto_increment_offset** + *N* × **auto_increment_increment**. *N* is a positive integer. If the value of this parameter is smaller than that of **auto_increment_offset**, there will be an error when the values in the auto-increment column automatically increase.
dn:gs_format_behavior_compat_options	Specifies the configuration items of GaussDB internal system functions.

Parameter	Description
enable_nestloop	Controls whether the query optimizer uses the nested-loop join plan type to fully scan inner tables.
dn:b_format_dev_version	Specifies the compatibility configuration item of database platform minor versions.
dn:check_disconnect_query	Specifies whether to enable the function of terminating statement execution on the server after the client is disconnected due to timeout.
dn:wal_keep_segments	Specifies the minimum number of transaction log files stored in the **pg_xlog** directory. Standby nodes obtain the logs from the primary node to perform streaming replication.
wdr_snapshot_retention_days	Specifies how many days database monitoring snapshots are saved for.
dn:max_standby_streaming_delay	Specifies the wait period before queries on the standby node are canceled when the queries conflict with WAL data receiving through streaming replication in hot standby mode. **-1** indicates that the standby node waits until the conflicting queries are complete. Unit: millisecond
audit_system_object	Specifies whether to audit the CREATE, DROP, and ALTER operations on GaussDB database objects. GaussDB database objects include databases, users, schemas, and tables. You can change the value of this parameter to audit only the operations on required database objects. In the scenario where the leader node is forcibly selected, you are advised to set **audit_system_object** to the maximum value and audit all DDL objects. For details about the value range, see the product documentation.
dn:enable_copy_server_files	Specifies whether to enable the privilege to copy server files. If this parameter is set to **on**, users with the SYSADMIN privilege or users who inherit the built-in role privilege **gs_role_copy_files** are allowed to run the COPY FROM FILENAME or COPY TO FILENAME command. If it is set to **off**, only the initial user is allowed to run the COPY FROM FILENAME or COPY TO FILENAME statement.
checkpoint_segments	Specifies the minimum number of WAL segment files in the period specified by **checkpoint_timeout**.

Parameter	Description
dn:page_version_partitions	Specifies the number of hash table partitions in cached page version information in the memory. This parameter directly affects the hash query efficiency and hash conflict probability, and is valid only when **page_version_check** is not set to **off**. If **page_version_check** is set to **off**, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to a value other than **off**. For details about the value range, see the product documentation.
dn:password_encryption_type	Specifies how user passwords are encrypted. **0**: Passwords are encrypted using MD5. **1**: Passwords are encrypted using SHA-256 and MD5. **2**: Passwords are encrypted using SHA-256. **3**: Passwords are encrypted using SM3. MD5 is not recommended because it is a weak encryption algorithm.
dn:enable_plsql_opfusion	Specifies whether to enable SQLBypass for stored procedures.
dn:page_version_recycler_thread_num	Specifies the number of threads for recycling and verifying page version information. This parameter is only valid when **page_version_check** is not set to **off**. If **page_version_check** is set to **off**, the parameter value will still be sent to the kernel, but the relevant function does not take effect until after **page_version_check** is set to a value other than **off**.
dn:dcf_thread_effective_time	Specifies the effective time of the DCF flushing thread. This parameter is used to determine whether the disk I/O hangs. If the DCF cannot access I/O resources within the period specified by this parameter, the DCF considers that the thread I/O hangs and a primary/standby switchover is triggered. If this parameter is set to **0**, I/O hang detection is disabled. Unit: second

Parameter	Description
dn:audit_login_logout	Specifies whether to audit users' logins (including successful and failed logins) and logouts. This parameter can be configured for specific PDBs. If this parameter is not specified for a PDB, the PDB inherits the global setting of this parameter. **0**: Disable user login and logout auditing. **1**: Audit only successful logins. **2**: Audit only failed logins. **3**: Audit both successful and failed logins. **4**: Audit only user logouts. **5**: Audit only user logouts and successful logins. **6**: Audit only user logouts and failed logins. **7**: Audit successful and failed logins, as well as user logouts.
dn:a_format_date_timestamp	Specifies whether to enable **current_date**, **current_timestamp**, and **localtimestamp** to return the system time, instead of the transaction start time, when a transaction starts.
session_timeout	Specifies how long to wait before a server connection is disconnected due to inactivity. The value **0** indicates there is no time limit. Unit: second
temp_file_limit	Specifies the maximum amount of disk space that a session can use for temporary files. The value **–1** indicates there is no limit. Unit: KB
dn:disable_keyword_options	Specifies the configuration items for database compatibility. Multiple items are separated by commas (,). An identifier with this parameter set will not be used as a keyword.
dn:hadr_recovery_time_target	Specifies whether the standby DB instance completes log writing and replay in streaming DR mode. If this parameter is set to a small value, the performance of the primary node is affected. If it is set to a large value, the log flow is not effectively controlled. The value **0** indicates that log flow control is disabled. Unit: second
password_lock_time	Specifies the duration for a locked account to be automatically unlocked. Unit: day
wdr_snapshot_interval	Specifies the interval at which the background thread Snapshot automatically takes snapshots of the database monitoring data. Unit: minute
update_lockwait_timeout	Specifies the maximum duration that a lock waits for concurrent updates on a row to complete when the concurrent update feature is enabled. If the lock wait time exceeds this value, the system will report an error. Unit: millisecond

Parameter	Description
dn:wal_file_preinit_bounds	Specifies the maximum number of WAL segment files that can be pre-expanded by the WAL writer auxiliary thread per second during service running. The WAL segment file size is 16 MB. If this parameter is set to **0**, there is no restriction.
dn:enable_feedback_cardest	Specifies whether to enable the feedback-based optimizer cardinality and cost correction functions. This parameter is used by developers to diagnose model-related problems. If enable_adaptive_cost is set to **off** and this parameter is set to **on**, the operator information is still collected and the API of cardinality estimation feedback is still called. However, in this case, the thread for background automatic model maintenance will not be enabled. You can use the **gs_acm_analyze_workload_manual()** function to manually train models for diagnosing problems.
dn:enable_global_plsqlcache	Specifies whether to globally cache compilation products of packages, stored procedures, and functions, and cache execution products at the session level. Enabling this function can reduce the memory usage of database nodes in high concurrency scenarios.
dn:convert_illegal_char_mode	Specifies the placeholders of invalid characters that can be displayed on the client when **enable_convert_illegal_char** is enabled. Value range: 95 characters whose decimal codes range from 32 to 126 in the ASCII coding table.
dn:advance_xlog_file_num	Specifies the number of Xlog files that are periodically initialized in advance in the background. This parameter is used to prevent the Xlog file initialization from affecting the performance during transaction commit. However, such a fault may occur only when the system is overloaded. Therefore, you do not need to set this parameter.
enable_mergejoin	Controls whether the query optimizer uses the merge-join plan type.
dn:tde_index_default_encrypt	When **tde_index_default_encrypt** is set to **on** and an index is created based on an encrypted table, the database automatically converts the index to an encrypted index.

Parameter	Description
failed_login_attempts	Specifies the maximum number of incorrect password attempts before an account is locked. The account will be automatically unlocked after the time specified in **password_lock_time** elapses. Only the **sysadmin** user can set this parameter.
undo_retention_time	Specifies how long undo logs are kept. This parameter is only used for flashback query. Note: 1. The undo space of the local disk increases. 2. In subsequent incremental backups, the size of the backup set increases, because extra undo content is retained. Unit: second
enable_global_syscache	Determines whether to enable global system cache.
dn:max_io_capacity	Specifies the maximum I/O per second for the background writer process to flush pages in batches. Set this parameter based on the service scenario and the disk I/O capability. If the RTO is short or the data volume is many times that of the shared memory and the service access data volume is random, the value of this parameter cannot be too small. A small value of **max_io_capacity** reduces the number of pages flushed by the background writer. If a large number of pages are evicted due to service triggering, the services are affected. Unit: KB
dn:audit_set_parameter	Determines whether to audit the SET operation. **0**: The SET operation is not audited. **1**: The SET operation is audited.
index_txntype	Sets the index page format to PCR or RCR. This parameter is left unconfigured during system initialization. By default, the created indexes are compatible with the index type (RCR) of earlier versions. Once this parameter is specified, it cannot be left unconfigured again.
dn:auto_increment_offset	Specifies the initial value of an auto-increment column. The auto-increment value is calculated by the following formula: **auto_increment_offset** + *N* × **auto_increment_increment**. *N* is a positive integer. If the value of this parameter is greater than that of **auto_increment_increment**, there will be an error when the values in the auto-increment column automatically increase.

Parameter	Description
autovacuum_naptime	Specifies the minimum delay between autovacuum runs on any given database. If this parameter is set to a smaller value, the load is more stable but the I/O increases. If this parameter is set to a larger value, the load may fluctuate more but the I/O decreases. Unit: second
dn:diskann_probe_ncandidates	Specifies the size of the candidate set when the gsdiskann index is used to retrieve vectors.
autoanalyze_timeout	Specifies the autoanalyze timeout period. If the duration of autoanalyze on a table exceeds the value of **autoanalyze_timeout**, the autoanalyze operation is automatically canceled. The value **0** indicates that there is no timeout limit. Unit: second
dn:use_workload_manager	Specifies whether to enable resource management. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_save_datachanged_timestamp	Specifies whether to record the time when INSERT, UPDATE, DELETE, or EXCHANGE/TRUNCATE/DROP PARTITION is performed on table data. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:max_process_memory	Specifies the maximum physical memory allowed for a DN. Unit: KB This parameter is supported in GaussDB V2.0-8.201 or later.
dn:standby_shared_buffers_fraction	Specifies the **shared_buffers** proportion used on the server where a standby instance is deployed. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:pagewriter_sleep	Specifies the interval (in milliseconds) for the page writer thread to flush dirty pages to disks after the incremental checkpoint is enabled. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:xloginsert_locks	Specifies the number of locks on concurrent write-ahead logging. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_light_proxy	Specifies whether the optimizer optimizes the execution of simple queries on CNs. This parameter does not take effect if the character set of the application side does not match that of the kernel side. You are advised to set the character set to UTF-8 when creating a database. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
dn:synchronous_commit	Specifies the synchronization mode of the current transaction. Generally, logs generated by a transaction are synchronized in the following sequence: The primary node writes the logs to the local memory. The primary node writes logs in the local memory to the local file system. The primary node flushes the logs in the local file system to disks. The primary node sends the logs to the standby node. The standby node receives the logs and saves them to its local memory. The standby node writes the logs in the local memory to the local file system. The standby node flushes the logs in the local file system to disks. The standby node replays the logs to complete the incremental update of data files. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_incremental_checkpoint	Specifies whether to enable incremental checkpoint. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:bypass_workload_manager	Enables or disables independent I/O control. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_instr_rt_percentile	Specifies whether to enable the function of calculating the response time of 80% and 95% of the SQL statements in the system. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_instr_cpu_timer	Specifies whether to capture the CPU time consumed during SQL statement execution. In the x86-based centralized deployment scenario where 32 vCPUs and 256 GB of memory are configured, the performance tested by BenchmarkSQL 5.0 fluctuates by about 3.5% by enabling or disabling this parameter. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_user_metric_persistent	Specifies whether to dump the historical monitoring data of user resources. If this parameter is set to **on**, data in the **PG_TOTAL_USER_RESOURCE_INFO** view is periodically sampled and saved to the system catalog **GS_WLM_USER_RESOURCE_HISTORY**. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
dn:log_hostname	By default, connection log messages only show the IP address of the connecting host. The host name can be recorded when this parameter is set to **on**. It may take some time to parse the host name. Therefore, the database performance may be affected. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_instr_track_wait	Specifies whether to enable real-time collection of wait event information. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_instance_metric_persistent	Specifies whether to dump instance resource monitoring data. When this parameter is set to **on**, the instance monitoring data is saved to the system catalog **GS_WLM_INSTANCE_HISTORY**. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_logical_io_statistics	Specifies whether to collect logical I/O statistics during resource monitoring. If this function is enabled, the **read_kbytes**, **write_kbytes**, **read_counts**, **write_counts**, **read_speed**, and **write_speed** columns in the PG_TOTAL_USER_RESOURCE_INFO view will collect statistics on the byte count, number of times, and speed of logical read and write. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:client_min_messages	Specifies which level of messages will be sent to the client. A higher level covers the messages of all the lower levels. The lower the level is, the fewer messages are sent. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:pagewriter_thread_num	Specifies the number of threads for background page flushing after the incremental checkpoint is enabled. Dirty pages are flushed in sequence to disks, updating recovery points. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
dn:bgwriter_delay	Specifies the interval at which the background writer writes dirty shared buffers, in milliseconds. Each time, the background writer initiates write operations for some dirty buffers. In full checkpoint mode, the **bgwriter_lru_maxpages** parameter is used to control the amount of data to be written each time, and the process is restarted after the period of hibernation specified by **bgwriter_delay** (in milliseconds). In incremental checkpoint mode, the number of target idle buffer pages is calculated based on the value of **candidate_buf_percent_target**. If the number of idle buffer pages is insufficient, a batch of pages are flushed to disks at the interval specified by **bgwriter_delay** (in milliseconds). The number of flushed pages is calculated based on the target difference percentage. The maximum number of flushed pages is limited by **max_io_capacity**. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:candidate_buf_percent_target	Specifies the proportion (%) of available buffers in **shared_buffer** in the candidate buffer chain when the incremental checkpoint is enabled. If the percentage is less than the value of this parameter, the **bgwriter** thread starts to flush dirty pages based on the **max_io_capacity** parameter. This parameter is supported in GaussDB V2.0-8.201 or later.
checkpoint_segments	Specifies the minimum number of WAL segment files in the period specified by **checkpoint_timeout**. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:checkpoint_timeout	Specifies the maximum time between automatic WAL checkpoints, in seconds. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:incremental_checkpoint_timeout	Specifies the maximum interval (in seconds) between automatic WAL checkpoints when the incremental checkpoint is enabled. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:wal_writer_delay	Specifies the delay (in milliseconds) between activity rounds for the WAL writer. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
dn:wal_buffers	Specifies the shared memory for storing WALs, that is, multiples of **XLOG_BLCKSZ**, in increments of 8 KB. For details about the value range, see **wal_buffers**. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:walwriter_cpu_bind	Specifies the CPU core bound to the WAL write thread. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:bulk_write_ring_size	Specifies the size of the ring buffer used by the operation when a large amount of data is written (for example, the copy operation). This parameter is supported in GaussDB V2.0-8.201 or later.
dn:recovery_redo_workers	Specifies the number of **PageRedoWorker** threads corresponding to each **ParseRedoRecord** thread when ultimate RTO is enabled. **recovery_redo_workers** must be used together with **recovery_parse_workers**. The value of **recovery_redo_workers** takes effect only when **recovery_parse_workers** is greater than 1. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_ai_watchdog	Enables or disables the AI watchdog function. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:max_connections	Specifies the maximum number of concurrent connections. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:resilience_memory_reject_percent	Specifies the dynamic memory usage for escape from memory overload. This parameter is only applied when **use_workload_manager** and **enable_memory_limit** are enabled. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:resilience_threadpool_reject_cond	Specifies the proportion of accumulated sessions in the thread pool for escape from overload. This parameter is only applied when **enable_thread_pool** and **use_workload_manager** are enabled. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:vacuum_cost_delay	Specifies how long the process sleeps when **vacuum_cost_limit** has been exceeded, in milliseconds. This parameter is supported in GaussDB V2.0-8.201 or later.

Parameter	Description
dn:b_format_version	Specifies the SQL behavior control configuration item in B-compatible mode. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:enable_control_group	Specifies whether to enable the Cgroups. This parameter must be applied on both CNs and DNs. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:track_stmt_parameter	After **track_stmt_parameter** is enabled, the executed statements recorded in **statement_history** are not normalized. The complete SQL statement information can be displayed to help the database administrator locate faults. For a simple query, the complete statement information is displayed. For a PBE statement, the complete statement information and information about each variable value are displayed. The format is "query string; parameters: $1=value1,$2=value2, ...". This parameter is used to display full SQL information for users and is not controlled by the **track_activity_query_size** parameter. This parameter is supported in GaussDB V2.0-8.201 or later.
track_activity_query_size	Specifies byte counts of the current running commands used to trace each active session. If the number of bytes in a command is greater than the value of this parameter, the command will be truncated. This parameter affects the memory usage of multiple system views. Adjusting the value of this parameter has a great impact on the memory usage. You can estimate the memory usage based on the value calculated from **instr_unique_sql_count** times **track_activity_query_size**. This parameter is supported in GaussDB V2.0-8.201 or later.
dn:auto_explain_level	Specifies the log level for automatically printing execution plans.

Parameter	Description
dn:instr_unique_sql_count	Specifies the maximum number of unique SQL records to be collected. The value **0** indicates that the function of collecting unique SQL information is disabled. If the value is changed from a larger one to a smaller one, the original data in the system will be cleared and data will be re-collected. There is no impact if the value is changed from a smaller one to a larger one. When the number of unique SQL records generated in the system is greater than the value of this parameter, the extra unique SQL records are not collected.
cma:log_saved_days	Specifies the maximum number of days to store logs for.
cma:log_threshold_check_interval	Specifies the interval for compressing and clearing logs, in seconds.
dn:wdr_snapshot_query_timeout	Specifies the execution timeout, in seconds, for the SQL statements associated with database monitoring snapshot operations.
dn:enable_wdr_snapshot_standby	Specifies whether to enable database monitoring snapshots on the standby node.
dn:backtrace_min_messages	Prints the function's stack information to the server's log file if the information generated is at least the level specified by this parameter.
dn:log_statement	Specifies whether to record SQL statements. For clients using extended query protocols, logging occurs when an execute message and values (enclosed by a pair of single quotation marks) of bound parameters are received.
dn:asp_flush_rate	Determines how many samples are flushed from memory to disk. When the number of samples in memory reaches the value of **asp_sample_num**, a portion of them will be flushed to disk based on the ratio specified by this parameter. If this parameter is set to **10**, the data flushing ratio is 10:1 (10%).
dn:asp_sample_interval	Specifies the sampling interval, in seconds.
dn:audit_space_limit	Specifies the maximum storage space occupied by audit files, in KB.

Parameter	Description
dn:enable_defer_calculate_snapshot	Determines if snapshot calculations for **xmin** and **oldestxmin** are deferred. If this parameter is set to **on**, calculations are triggered every 1,000 transactions or once per second. This reduces overhead under heavy load but slows down the advancement of **oldestxmin**, potentially delaying garbage collection. If this parameter is set to **off**, **xmin** and **oldestxmin** update in real time, improving tuple cleanup but with higher processing overhead.
dn:enable_global_plancache	Specifies whether to share the execution plan cache for statements in PBE queries and stored procedures. Enabling it can help reduce the memory needed on database nodes during high-concurrency workloads. With **enable_global_plancache** enabled, **local_syscache_threshold** is automatically set to a minimum of 16 MB for GPC to take effect. If the value of **local_syscache_threshold** is configured below 16 MB, it defaults to 16 MB. If set above, the specified value is used.
dn:enable_prevent_job_task_startup	Determines whether to start the job thread.
dn:log_min_messages	Specifies which level of messages will be written into the server log. A higher level covers the messages of all the lower levels. The lower the level is, the fewer messages will be written into the log. A same value for **client_min_messages** and **log_min_messages** does not indicate the same level. For some log information, after this parameter is enabled, you also need to set **logging_module** to enable log printing for the corresponding module.
dn:max_recursive_times	Specifies the maximum number of WITH RECURSIVE iterations.
dn:vacuum_freeze_min_age	Specifies whether VACUUM replaces the **xmin** column of a record with **FrozenXID** when scanning a table (in the same transaction). Although you can set this parameter to any value, VACUUM will limit the effective value to 50% of **autovacuum_freeze_max_age** by default.

Parameter	Description
dn:vacuum_freeze_table_age	Specifies when VACUUM scans the whole table and freezes old tuples. VACUUM performs a full table scan if the difference between the current transaction ID and the value of **pg_class.relfrozenxid64** is greater than the specified time. Although you can set this parameter to any value, VACUUM will limit the effective value to 95% of **autovacuum_freeze_max_age** by default.
dn:allow_create_sysobject	Specifies whether objects such as functions, stored procedures, synonyms, aggregate functions, and operators can be created or modified in the system schema. The system schema refers to the schema provided by the database after initialization, excluding the public schema. The OID of the system schema is usually less than 16384.
dn:standby_max_query_time	Specifies the maximum allowed query time (in seconds) on the standby node when ultimate RTO is enabled. If a query exceeds this limit, it will be canceled. The exact cancellation timing depends on the **standby_recycle_interval** for the recycling thread and the snapshot timing, so the actual query duration on the standby node may exceed this limit.
dn:exrto_standby_read_opt	Specifies whether to enable read optimization on standby nodes with ultimate RTO. This parameter is enabled by default. The setting of this parameter is not synchronized between primary and standby nodes.
dn:dirty_page_percent_max	Specifies the percentage of dirty pages relative to the value of **shared_buffers** after the incremental checkpoint is enabled. When the value of this parameter is reached, the background page flush thread flushes dirty pages based on the value of **max_io_capacity**.
dn:audit_enabled	Determines whether to enable or disable the audit process. After the audit process is enabled, it can read the auditing information written by the background process from the pipe and save it into audit files.
dn:enable_bitmapscan	Specifies whether the optimizer uses bitmap scan.
dn:autovacuum	Specifies whether to enable the autovacuum process. Before enabling the process, make sure the **track_counts** parameter is set to **on**.

Parameter	Description
dn:autovacuum_analyze_scale_factor	Specifies a fraction of the table size added to the **autovacuum_analyze_threshold** parameter when deciding whether to trigger an ANALYZE operation.
dn:autovacuum_analyze_threshold	Specifies the threshold for triggering ANALYZE. When the number of deleted, inserted, or updated rows in a table exceeds the specified threshold, the ANALYZE operation is executed on this table.
dn:autovacuum_freeze_max_age	Specifies the maximum age (in transactions) that a table's **pg_class.relfrozenxid** column can attain before a VACUUM operation is performed.
dn:autovacuum_io_limits	Determines the maximum number of input/output operations that the autovacuum process can execute per second.
dn:autovacuum_vacuum_threshold	Specifies the threshold for triggering VACUUM. When the number of deleted or updated rows in a table exceeds the specified threshold, the VACUUM operation is executed on this table.
dn:max_stack_depth	Specifies the maximum safe depth of the GaussDB Kernel's execution stack, in KB. The safety margin is required because the stack depth is not checked in every routine in the server, but only in key potentially-recursive routines, such as expression evaluation. If the value of **ulimit -s** minus 640 KB is greater than or equal to 2 MB, the default value of this parameter is 2 MB. If the value of **ulimit -s** minus 640 KB is less than 2 MB, that value will be used as the default value of this parameter.
dn:default_transaction_read_only	Specifies whether each new transaction is in read-only state. If this parameter is set to **on**, DML operations and write transactions cannot be executed.
dn:autovacuum_vacuum_cost_delay	Sets the amount of time, in milliseconds, that the autovacuum process will sleep when the cost limit has been exceeded.
dn:autovacuum_vacuum_cost_limit	Sets the accumulated cost that will cause the autovacuum process to sleep.

Configurable Parameters for Version V2.0-3.*x*

The following table describes the parameters that can be modified.

Configurable Parameters for Distributed Instances of Version V2.0-3.*x*

Table 12-4 Parameters for distributed instances

Parameter	Description
audit_system_object	Determines whether to audit the CREATE, DROP, and ALTER operations on GaussDB Kernel database objects. GaussDB Kernel database objects include databases, users, schemas, and tables. You can change the parameter value to audit only the operations on required database objects. During a forcible primary/standby failover, set **audit_system_object** to the maximum value and audit all DDL objects. If the parameter value is incorrectly changed, DDL audit logs will be lost. Contact technical support to change it.
autoanalyze	Specifies whether to automatically collect statistics on tables that have no statistics when a plan is generated.
autoanalyze_timeout	Specifies the autoanalyze timeout period. If the duration of autoanalyze on a table exceeds the value of **autoanalyze_timeout**, the autoanalyze operation is automatically canceled. **0** indicates there is no timeout. Unit: second
cn:effective_cache_size	Specifies the size of the disk buffer available to the CN optimizer in a single query. Unit: 8 KB
cn:enable_hotkeys_collection	Specifies whether to collect statistics on accessed key values in databases.
cn:track_stmt_session_slot	Specifies the maximum number of full or slow SQL statements that can be cached in a CN session.
datestyle	Specifies the display format for date and time.
dn:effective_cache_size	Specifies the size of the disk buffer available to the DN optimizer in a single query. Unit: 8 KB
dn:enable_hotkeys_collection	Specifies whether to collect statistics on accessed key values in databases.
dn:track_stmt_session_slot	Specifies the maximum number of full or slow SQL statements that can be cached in a DN session.
enable_seqscan	Specifies whether to enable the optimizer's use of sequential scan plan types. It is impossible to completely suppress sequential scans, but setting this parameter to **off** allows the optimizer to choose other methods if available.

Parameter	Description
enable_slot_log	Specifies whether to enable primary/standby synchronization for logical replication slots.
enable_stream_operator	Specifies the query optimizer's use of streams. When this parameter is set to **off**, a large number of logs indicating that the stream plans cannot be pushed down are recorded.
failed_login_attempts	Specifies the maximum number of incorrect password attempts before an account is locked. The account will be automatically unlocked after the time specified in **password_lock_time** elapses. Only the **sysadmin** user can set this parameter.
log_min_duration_statement	Specifies the threshold for logging the duration of a completed statement. If a statement runs for a period greater than or equal to the specified value, its duration will be logged. The value **-1** disables logging statement durations. If this parameter is set to a small value, the load throughput may be affected. Unit: millisecond
max_replication_slots	Specifies the number of log replication slots on the primary node.
max_wal_senders	Specifies the maximum number of WAL sender threads that can be created. Processes created for the following purposes occupy WAL sender threads: Standby DNs connect to primary DNs to obtain physical logs, and logical replication tools connect to primary DNs to obtain logical logs.
password_effect_time	Specifies the validity period of the password, in days.
password_lock_time	Specifies the duration for a locked account to be automatically unlocked, in days.
recovery_time_target	Specifies the time for the standby node to write and replay logs, in seconds.
session_timeout	Specifies how long to wait before a server connection is disconnected due to inactivity. The value **0** indicates there is no time limit. Unit: second
timezone	Specifies the time zone for displaying and interpreting time stamps.
track_stmt_stat_level	Controls the level of statement execution tracking.

Parameter	Description
update_lockwait_timeout	Specifies the maximum duration that a lock waits for concurrent updates on a row to complete when the concurrent update feature is enabled. If the lock wait time exceeds this value, the system will report an error. Unit: millisecond
wal_level	Specifies the level of information to be written to the WAL. This is a required value and cannot be commented out. Determines how much information is written to the WAL. When this parameter is set to **logical**, logical logs are extracted and primary key information is recorded in Xlogs.
cn:audit_thread_num	Specifies the number of audit threads. Value range: 1 to 48.
dn:audit_thread_num	Specifies the number of audit threads. Value range: 1 to 48.
cn:qrw_inlist2join_optmode	Specifies whether to enable inlist-to-join query rewriting.
dn:qrw_inlist2join_optmode	Specifies whether to enable inlist-to-join query rewriting.
cn:audit_xid_info	Determines whether to record the transaction IDs of SQL statements in detail_info. **0**: The transaction IDs are not recorded. **1**: The transaction IDs are recorded.
dn:audit_xid_info	Determines whether to record the transaction IDs of SQL statements in detail_info. **0**: The transaction IDs are not recorded. **1**: The transaction IDs are recorded.
cn:default_limit_rows	Specifies the estimated number of rows to return by default for generating a generic plan, that is, the default value for the LIMIT clause. If this parameter is set to a negative number, the value is converted to a percentage, for example, **-5** is equivalent to 5%, indicating that 5% of the total rows will be returned.
dn:default_limit_rows	Specifies the estimated number of rows to return by default for generating a generic plan, that is, the default value for the LIMIT clause. If this parameter is set to a negative number, the value is converted to a percentage, for example, **-5** is equivalent to 5%, indicating that 5% of the total rows will be returned.

Parameter	Description
cn:audit_dml_state_select	Determines whether to audit the SELECT operation.
dn:audit_dml_state_select	Determines whether to audit the SELECT operation.
cn:audit_dml_state	Determines whether to audit the INSERT, UPDATE, and DELETE operations on a specific table. **0**: These operations are not audited. **1**: These operations are audited.
dn:audit_dml_state	Determines whether to audit the INSERT, UPDATE, and DELETE operations on a specific table. **0**: These operations are not audited. **1**: These operations are audited.
cn:random_page_cost	Specifies the estimated cost for the optimizer to fetch an out-of-sequence disk page.
dn:random_page_cost	Specifies the estimated cost for the optimizer to fetch an out-of-sequence disk page.
cn:enable_security_policy	Controls whether unified auditing and dynamic data masking policies are applied.
dn:enable_security_policy	Controls whether unified auditing and dynamic data masking policies are applied.
cn:audit_set_parameter	Determines whether to audit the SET operation. **0**: The SET operation is not audited. **1**: The SET operation is audited.
dn:audit_set_parameter	Determines whether to audit the SET operation. **0**: The SET operation is not audited. **1**: The SET operation is audited.
cn:enable_pbe_optimization	Specifies whether the optimizer optimizes the query plan for statements executed in Parse Bind Execute (PBE) mode.
dn:enable_pbe_optimization	Specifies whether the optimizer optimizes the query plan for statements executed in Parse Bind Execute (PBE) mode.
wdr_snapshot_interval	Specifies the interval (in minutes) at which the background thread Snapshot automatically performs snapshot operations on the database monitoring data.
cn:enable_auto_explain	Specifies whether to automatically print execution plans. This parameter is used to locate slow stored procedures or slow queries.

Parameter	Description
dn:enable_auto_explain	Specifies whether to automatically print execution plans. This parameter is used to locate slow stored procedures or slow queries.
enable_wdr_snapshot	Specifies whether to enable WDR snapshots. If this parameter is enabled, the kernel periodically takes performance snapshots.
cn:max_concurrent_autonomous_transactions	Specifies the maximum number of autonomous transaction connections, that is, the maximum number of concurrent autonomous transactions executed at the same time. If this parameter is set to **0**, autonomous transactions cannot be executed.
dn:max_concurrent_autonomous_transactions	Specifies the maximum number of autonomous transaction connections, that is, the maximum number of concurrent autonomous transactions executed at the same time. If this parameter is set to **0**, autonomous transactions cannot be executed.
cn:max_standby_archive_delay	Specifies the wait period (in milliseconds) before queries on a standby node are canceled when the queries conflict with WAL processing and archiving in hot standby mode.
dn:max_standby_archive_delay	Specifies the wait period (in milliseconds) before queries on a standby node are canceled when the queries conflict with WAL processing and archiving in hot standby mode.
cn:max_standby_streaming_delay	Specifies how long a standby node waits before canceling queries, in milliseconds.
dn:max_standby_streaming_delay	Specifies how long a standby node waits before canceling queries, in milliseconds.
cn:recovery_max_workers	Specifies the number of concurrent replayer threads.
dn:recovery_max_workers	Specifies the number of concurrent replayer threads.
cn:auto_explain_log_min_duration	Specifies the minimum duration of execution plans that are automatically printed. Only execution plans whose duration is greater than the value of **auto_explain_log_min_duration** will be printed. Unit: ms

Parameter	Description
dn:auto_explain_log_min_duration	Specifies the minimum duration of execution plans that are automatically printed. Only execution plans whose duration is greater than the value of **auto_explain_log_min_duration** will be printed. Unit: ms
cn:audit_function_exec	Specifies whether to record the audit information during the execution of the stored procedures, anonymous blocks, or user-defined functions (excluding system functions).
dn:audit_function_exec	Specifies whether to record the audit information during the execution of the stored procedures, anonymous blocks, or user-defined functions (excluding system functions).
cn:local_syscache_threshold	Specifies the size of system catalog cache in a session. Unit: KB
dn:local_syscache_threshold	Specifies the size of system catalog cache in a session. Unit: KB
cms:datastorage_threshold_value_check	Specifies the disk usage threshold to put a database node into read-only mode. If the disk usage of a data directory exceeds this threshold, the database node is automatically changed to read-only. Unit: percentage (%)
wdr_snapshot_retention_days	Specifies how many days database monitoring snapshots are saved for.
cn:enable_default_ustore_table	Specifies whether to enable the Ustore storage engine by default. If this parameter is set to **on**, all created tables are Ustore tables.
dn:enable_default_ustore_table	Specifies whether to enable the Ustore storage engine by default. If this parameter is set to **on**, all created tables are Ustore tables.
cn:undo_space_limit_size	Specifies the undo forcible reclamation threshold. If 80% of the specified parameter value is reached, forcible reclamation is triggered. The unit is 8 KB. It is recommended that the value be at least the value of **undo_limit_size_per_transaction**.
dn:undo_space_limit_size	Specifies the undo forcible reclamation threshold. If 80% of the specified parameter value is reached, forcible reclamation is triggered. The unit is 8 KB. It is recommended that the value be at least the value of **undo_limit_size_per_transaction**.

Parameter	Description
cn:undo_limit_size_per_transaction	Specifies the maximum undo space for a single transaction. The unit is 8 KB. If the undo space of a transaction exceeds this parameter value, the transaction is rolled back due to an error. It is recommended that this parameter value be smaller than the value of **undo_space_limit_size**. If this parameter value is larger, the value of **undo_space_limit_size** will be used as the maximum undo space for a single transaction.
dn:undo_limit_size_per_transaction	Specifies the maximum undo space for a single transaction. The unit is 8 KB. If the undo space of a transaction exceeds this parameter value, the transaction is rolled back due to an error. It is recommended that this parameter value be smaller than the value of **undo_space_limit_size**. If this parameter value is larger, the value of **undo_space_limit_size** will be used as the maximum undo space for a single transaction.
cn:enable_recyclebin	Enables or disables the recycle bin in real time.
dn:enable_recyclebin	Enables or disables the recycle bin in real time.
cn:recyclebin_retention_time	Specifies how long files will be kept in the recycle bin, in seconds. Files in the recycle bin will be automatically deleted after this length of time.
dn:recyclebin_retention_time	Specifies how long files will be kept in the recycle bin, in seconds. Files in the recycle bin will be automatically deleted after this length of time.
cn:undo_retention_time	Specifies how long undo logs are kept, in seconds. This parameter is only used for flashback query. Note: 1. The undo space of the local disk increases. 2. In subsequent incremental backups, the size of the backup set increases, because extra undo content is retained.
dn:undo_retention_time	Specifies how long undo logs are kept, in seconds. This parameter is only used for flashback query. Note: 1. The undo space of the local disk increases. 2. In subsequent incremental backups, the size of the backup set increases, because extra undo content is retained.

Parameter	Description
cn:cost_model_version	Specifies the version of the optimizer cost model. It is a protective parameter. It prevents new optimizer cost models from being applied, so you can keep the current model consistent with the plan of an existing version. If the value of this parameter is changed, many SQL plans may be changed. Exercise caution when modifying this parameter.
dn:cost_model_version	Specifies the version of the optimizer cost model. It is a protective parameter. It prevents new optimizer cost models from being applied, so you can keep the current model consistent with the plan of an existing version. If the value of this parameter is changed, many SQL plans may be changed. Exercise caution when modifying this parameter.
cn:enable_dynamic_sample size	Specifies whether to dynamically adjust the number of sampled rows. For a large table with more than one million rows, the number of sampled rows is dynamically adjusted during statistics collection to improve statistics accuracy.
dn:enable_dynamic_sample size	Specifies whether to dynamically adjust the number of sampled rows. For a large table with more than one million rows, the number of sampled rows is dynamically adjusted during statistics collection to improve statistics accuracy.
cn:resilience_ctrlslot_availa ble_maxpercent	Specifies the maximum percentage of threads in the thread pool that can be occupied by slow SQL statements. This parameter is only valid for SELECT statements that are executed by non-sysadmin/monitoradmin users.
dn:resilience_ctrlslot_availa ble_maxpercent	Specifies the maximum percentage of threads in the thread pool that can be occupied by slow SQL statements. This parameter is only valid for SELECT statements that are executed by non-sysadmin/monitoradmin users.
cn:resilience_ctrlstmt_contr ol_iopslimit	Specifies the maximum IOPS that can be used by slow SQL statements after normal SQL statements are marked as slow SQL statements. This parameter is only valid for SELECT statements that are executed by non-sysadmin/monitoradmin users. **0(None)**: The IOPS is not limited. **10(LOW)**: The limit level for IOPS is **LOW**. **20(MEDIUM)**: The limit level for IOPS is **MEDIUM**. **50(HIGH)**: The limit level for IOPS is **HIGH**.

Parameter	Description
dn:resilience_ctrlstmt_control_iopslimit	Specifies the maximum IOPS that can be used by slow SQL statements after normal SQL statements are marked as slow SQL statements. This parameter is only valid for SELECT statements that are executed by non-sysadmin/monitoradmin users. **0(None)**: The IOPS is not limited. **10(LOW)**: The limit level for IOPS is **LOW**. **20(MEDIUM)**: The limit level for IOPS is **MEDIUM**. **50(HIGH)**: The limit level for IOPS is **HIGH**.
dn:resilience_ctrlstmt_detect_timelimit	Specifies the execution time of a normal SQL statement that will be marked as a slow SQL statement. The value **0** indicates that slow SQL statements are not identified. A value greater than 0 indicates that a normal SQL statement whose execution time exceeds the value of this parameter is marked as a slow SQL statement. This parameter is only valid for SELECT statements that are executed by non-sysadmin/monitoradmin users. Unit: ms

Configurable Parameters for Centralized Instances of Version V2.0-3.*x*

Table 12-5 Parameters for centralized instances

Parameter	Description
audit_system_object	Determines whether to audit the CREATE, DROP, and ALTER operations on GaussDB Kernel database objects. GaussDB Kernel database objects include databases, users, schemas, and tables. You can change the parameter value to audit only the operations on required database objects. During a forcible primary/standby failover, set **audit_system_object** to the maximum value and audit all DDL objects. If the parameter value is incorrectly changed, DDL audit logs will be lost. Contact technical support to change it.
autoanalyze	Specifies whether to automatically collect statistics on tables that have no statistics when a plan is generated.
autoanalyze_timeout	Specifies the autoanalyze timeout period. If the duration of autoanalyze on a table exceeds the value of **autoanalyze_timeout**, the autoanalyze operation is automatically canceled. **0** indicates there is no timeout. Unit: second
datestyle	Specifies the display format for date and time.

Parameter	Description
dn:wal_keep_segments	Specifies the minimum number of transaction log files stored in the **pg_xlog** directory. Standby nodes obtain the logs from the primary node to perform streaming replication.
enable_seqscan	Specifies whether to enable the optimizer's use of sequential scan plan types. It is impossible to completely suppress sequential scans, but setting this parameter to **off** allows the optimizer to choose other methods if available.
enable_slot_log	Specifies whether to enable primary/standby synchronization for logical replication slots.
failed_login_attempts	Specifies the maximum number of incorrect password attempts before an account is locked. The account will be automatically unlocked after the time specified in **password_lock_time** elapses. Only the **sysadmin** user can set this parameter.
log_min_duration_statement	Specifies the threshold for logging the duration of a completed statement. If a statement runs for a period greater than or equal to the specified value, its duration will be logged. The value **-1** disables logging statement durations. If this parameter is set to a small value, the load throughput may be affected. Unit: millisecond
max_replication_slots	Specifies the number of log replication slots on the primary node.
max_wal_senders	Specifies the maximum number of WAL sender threads that can be created. Processes created for the following purposes occupy WAL sender threads: Standby DNs connect to primary DNs to obtain physical logs, and logical replication tools connect to primary DNs to obtain logical logs.
password_effect_time	Specifies the validity period of the password, in days.
password_lock_time	Specifies the duration for a locked account to be automatically unlocked, in days.
session_timeout	Specifies how long to wait before a server connection is disconnected due to inactivity. The value **0** indicates there is no time limit. Unit: second
timezone	Specifies the time zone for displaying and interpreting time stamps.

Parameter	Description
update_lockwait_timeout	Specifies the maximum duration that a lock waits for concurrent updates on a row to complete when the concurrent update feature is enabled. If the lock wait time exceeds this value, the system will report an error. Unit: millisecond
wal_level	Specifies the level of information to be written to the WAL. This is a required value and cannot be commented out. Determines how much information is written to the WAL. When this parameter is set to **logical**, logical logs are extracted and primary key information is recorded in Xlogs.
dn:audit_thread_num	Specifies the number of audit threads. Value range: **1** to **48**.
dn:qrw_inlist2join_optmode	Specifies whether to enable inlist-to-join query rewriting.
dn:audit_xid_info	Determines whether to record the transaction IDs of SQL statements in detail_info. **0**: The transaction IDs are not recorded. **1**: The transaction IDs are recorded.
dn:default_limit_rows	Specifies the estimated number of rows to return by default for generating a generic plan, that is, the default value for the LIMIT clause. If this parameter is set to a negative number, the value is converted to a percentage, for example, **-5** is equivalent to 5%, indicating that 5% of the total rows will be returned.
dn:audit_dml_state_select	Determines whether to audit the SELECT operation.
dn:audit_dml_state	Determines whether to audit the INSERT, UPDATE, and DELETE operations on a specific table. **0**: These operations are not audited. **1**: These operations are audited.
dn:random_page_cost	Specifies the estimated cost for the optimizer to fetch an out-of-sequence disk page.
dn:enable_security_policy	Controls whether unified auditing and dynamic data masking policies are applied.
dn:audit_set_parameter	Determines whether to audit the SET operation. **0**: The SET operation is not audited. **1**: The SET operation is audited.
dn:max_standby_streaming_delay	Specifies how long a standby node waits before canceling queries, in milliseconds.

Parameter	Description
dn:vacuum_defer_cleanup_age	Specifies the number of transactions used by VACUUM.
dn:enable_pbe_optimization	Specifies whether the optimizer optimizes the query plan for statements executed in Parse Bind Execute (PBE) mode.
wdr_snapshot_interval	Specifies the interval (in minutes) at which the background thread Snapshot automatically performs snapshot operations on the database monitoring data.
undo_retention_time	Specifies how long undo logs are kept, in seconds. This parameter is only used for flashback query. Note: 1. The undo space of the local disk increases. 2. In subsequent incremental backups, the size of the backup set increases, because extra undo content is retained.
track_stmt_stat_level	Controls the level of statement execution tracking.
dn:enable_auto_explain	Specifies whether to automatically print execution plans. This parameter is used to locate slow stored procedures or slow queries.
enable_wdr_snapshot	Specifies whether to enable WDR snapshots. If this parameter is enabled, the kernel periodically takes performance snapshots.
dn:max_concurrent_autonomous_transactions	Specifies the maximum number of autonomous transaction connections, that is, the maximum number of concurrent autonomous transactions executed at the same time. If this parameter is set to **0**, autonomous transactions cannot be executed.
dn:max_standby_archive_delay	Specifies the wait period (in milliseconds) before queries on a standby node are canceled when the queries conflict with WAL processing and archiving in hot standby mode.
dn:max_standby_streaming_delay	Specifies how long a standby node waits before canceling queries, in milliseconds.
dn:recovery_max_workers	Specifies the number of concurrent replayer threads.

Parameter	Description
dn:track_stmt_parameter	If this parameter is enabled, the executed statements recorded in **statement_history** are not normalized. The complete SQL statement information can be displayed to help the database administrator locate faults. For a simple query, the complete statement information is displayed. For a PBE statement, the complete statement information and information about each variable value are displayed. The format is "querystring; parameters:$1=value1,$2=value2,...". This parameter is used to display full SQL information for users and is not controlled by the **track_activity_query_size** parameter.
dn:auto_explain_log_min_duration	Specifies the minimum duration of execution plans that are automatically printed. Only execution plans whose duration is greater than the value of **auto_explain_log_min_duration** will be printed. Unit: ms
dn:recovery_time_target	Specifies the time for the standby node to write and replay logs. Unit: second
dn:audit_function_exec	Specifies whether to record the audit information during the execution of the stored procedures, anonymous blocks, or user-defined functions (excluding system functions).
dn:local_syscache_threshold	Specifies the size of system catalog cache in a session. Unit: KB
cms:datastorage_threshold_value_check	Specifies the disk usage threshold to put a database node into read-only mode. If the disk usage of a data directory exceeds this threshold, the database node is automatically changed to read-only. Unit: percentage (%)
wdr_snapshot_retention_days	Specifies how many days database monitoring snapshots are saved for.
dn:undo_space_limit_size	Specifies the undo forcible reclamation threshold. If 80% of the specified parameter value is reached, forcible reclamation is triggered. The unit is 8 KB. It is recommended that the value be at least the value of **undo_limit_size_per_transaction**.
dn:group_concat_max_len	Specifies the maximum permitted result length in bytes for the GROUP_CONCAT() function.
dn:enable_extension	Specifies whether database extensions can be created. The extension plug-in is a lab feature and is not recommended.

Parameter	Description
dn:cost_model_version	Specifies the version of the optimizer cost model. It is a protective parameter. It prevents new optimizer cost models from being applied, so you can keep the current model consistent with the plan of an existing version. If the value of this parameter is changed, many SQL plans may be changed. Exercise caution when modifying this parameter.
dn:immediate_analyze_threshold	Specifies the threshold for automatically analyzing inserted data. When the amount of inserted data reaches the original data amount multiplied by the value of **immediate_analyze_threshold**, and the total number of rows exceeds 100, ANALYZE is automatically triggered.
dn:enable_dynamic_sample size	Specifies whether to dynamically adjust the number of sampled rows. For a large table with more than one million rows, the number of sampled rows is dynamically adjusted during statistics collection to improve statistics accuracy.
dn:max_io_capacity	Specifies the maximum I/O per second for the background writer process to flush pages in batches. Set this parameter based on the service scenario and the disk I/O capability. If the RTO is short or the data volume is many times that of the shared memory and the service access data volume is random, the value of this parameter cannot be too small. A small value of **max_io_capacity** reduces the number of pages flushed by the background writer process. If a large number of pages are evicted due to service triggering, the services are affected. Unit: KB
dn:max_connections	Specifies the maximum number of concurrent connections to DNs.
log_autovacuum_min_duration	Specifies the interval which should elapse before autovacuum operations are logged. Autovacuum operations equal to or beyond the specified interval will be logged. If it is set to **0**, all autovacuum operations will be logged. If it is set to **-1**, no autovacuum operations will be logged.

Parameter	Description
dn:sql_beta_feature	Specifies which SQL engine optional beta features are enabled, including optimization of row count estimation and query equivalence estimation. These optional features can improve performance in certain situations, but they may also degrade performance if not thoroughly tested. In certain scenarios, you can set the query rewriting rules through this GUC parameter to improve query efficiency.
dn:ai_watchdog_max_cpu_usage	Specifies the expected upper limit of the database CPU usage. The value is normalized based on the multi-core situation.
dn:enable_control_group	Specifies whether to enable the Cgroups.
dn:explain_perf_mode	Specifies the display format of EXPLAIN.
dn:enable_ai_watchdog	Enables or disables the AI watchdog function.
dn:resilience_ctrlstmt_detect_timelimit	Specifies the execution time of a normal SQL statement that will be marked as a slow SQL statement. The value **0** indicates that slow SQL statements are not identified. A value greater than 0 indicates that a normal SQL statement whose execution time exceeds the value of this parameter is marked as a slow SQL statement. This parameter is only valid for SELECT statements that are executed by non-sysadmin/monitoradmin users.
dn:resilience_ctrlslot_available_maxpercent	Specifies the maximum percentage of threads in the thread pool that can be occupied by slow SQL statements. This parameter is only valid for SELECT statements that are executed by non-sysadmin/monitoradmin users.
dn:require_ssl	Controls whether the server requires SSL connections. This parameter is only valid when **ssl** is set to **on**.
dn:enable_gpi_auto_update	Determines whether global indexes are updated by default in partition DDL commands.
dn:autovacuum_max_workers	Specifies the maximum number of autovacuum worker threads that can run at the same time. The upper limit of this parameter is related to the values of **max_connections** and **job_queue_processes**.
dn:effective_cache_size	Specifies the size of the disk buffer available to the optimizer in a single query.

Parameter	Description
dn:maintenance_work_mem	Specifies the maximum memory (in bytes) to be used for maintenance.
dn:work_mem	Specifies the amount of memory required by internal sort operations and hash tables before they write data to temporary disk files.
dn:enable_cachedplan_mgr	Specifies whether to enable the adaptive plan selection function.
dn:recovery_parse_workers	Specifies the number of **ParseRedoRecord** threads in the ultimate RTO feature.
dn:recovery_redo_workers	Specifies the number of **PageRedoWorker** threads corresponding to each **ParseRedoRecord** thread when ultimate RTO is enabled.
dn:max_standby_base_page_size	Sets the maximum storage space of base page files on the standby node after ultimate RTO is enabled.
dn:pagewriter_sleep	Specifies the interval for the page writer thread to flush dirty pages to disks after the incremental checkpoint is enabled.
dn:enable_double_write	Specifies whether to enable double-write. If both **enable_incremental_checkpoint** and **enable_double_write** are enabled, double-write will be used for protection and **full_page_writes** is not used to prevent half-page write.
dn:max_process_memory	Specifies the maximum physical memory allowed for a DN.
dn:bgwriter_delay	Specifies the interval at which the background writer writes dirty shared buffers. Each time, the background writer initiates write operations for some dirty buffers. In full checkpoint mode, the **bgwriter_lru_maxpages** parameter is used to control the amount of data to be written each time, and the process is restarted after the period of hibernation specified by **bgwriter_delay** (in milliseconds). In incremental checkpoint mode, the number of target idle buffer pages is calculated based on the value of **candidate_buf_percent_target**. If the number of idle buffer pages is insufficient, a batch of pages are flushed to disks at the interval specified by **bgwriter_delay** (in milliseconds). The number of flushed pages is calculated based on the target difference percentage. The maximum number of flushed pages is limited by **max_io_capacity**.

Parameter	Description
dn:wal_receiver_buffer_size	Sets the size of the memory buffer that stores the Xlogs received by the standby and secondary nodes. Currently, primary/standby/secondary deployment is not supported by default.
dn:enable_incremental_checkpoint	Specifies whether to enable incremental checkpoint.

Configurable Parameters for Version V2.0-2.*x*

The following table describes the parameters that can be modified.

Configurable Parameters for Distributed Instances of Version V2.0-2.*x*

Table 12-6 Parameters for distributed instances

Parameter	Description
audit_system_object	Determines whether to audit the CREATE, DROP, and ALTER operations on GaussDB Kernel database objects. GaussDB Kernel database objects include databases, users, schemas, and tables. You can change the parameter value to audit only the operations on required database objects. During a forcible primary/standby failover, set **audit_system_object** to the maximum value and audit all DDL objects. If the parameter value is incorrectly changed, DDL audit logs will be lost. Contact technical support to change it.
autoanalyze	Specifies whether to automatically collect statistics on tables that have no statistics when a plan is generated.
autoanalyze_timeout	Specifies the autoanalyze timeout period. If the duration of autoanalyze on a table exceeds the value of **autoanalyze_timeout**, the autoanalyze operation is automatically canceled. **0** indicates there is no timeout. Unit: second
cn:effective_cache_size	Specifies the size of the disk buffer available to the CN optimizer in a single query. Unit: 8 KB
cn:enable_hotkeys_collection	Specifies whether to collect statistics on accessed key values in databases.
cn:track_stmt_session_slot	Specifies the maximum number of full or slow SQL statements that can be cached in a CN session.

Parameter	Description
datestyle	Specifies the display format for date and time.
dn:effective_cache_size	Specifies the size of the disk buffer available to the DN optimizer in a single query. Unit: 8 KB
dn:enable_hotkeys_collection	Specifies whether to collect statistics on accessed key values in databases.
dn:track_stmt_session_slot	Specifies the maximum number of full or slow SQL statements that can be cached in a DN session.
enable_seqscan	Specifies whether to enable the optimizer's use of sequential scan plan types. It is impossible to completely suppress sequential scans, but setting this parameter to **off** allows the optimizer to choose other methods if available.
enable_slot_log	Specifies whether to enable primary/standby synchronization for logical replication slots.
enable_stream_operator	Specifies the query optimizer's use of streams. When this parameter is set to **off**, a large number of logs indicating that the stream plans cannot be pushed down are recorded.
failed_login_attempts	Specifies the maximum number of incorrect password attempts before an account is locked. The account will be automatically unlocked after the time specified in **password_lock_time** elapses. Only the **sysadmin** user can set this parameter.
log_min_duration_statement	Specifies the threshold for logging the duration of a completed statement. If a statement runs for a period greater than or equal to the specified value, its duration will be logged. The value **-1** disables logging statement durations. If this parameter is set to a small value, the load throughput may be affected. Unit: millisecond
max_replication_slots	Specifies the number of log replication slots on the primary node.
max_wal_senders	Specifies the maximum number of WAL sender threads that can be created. Processes created for the following purposes occupy WAL sender threads: Standby DNs connect to primary DNs to obtain physical logs, and logical replication tools connect to primary DNs to obtain logical logs.
password_effect_time	Specifies the validity period of the password, in days.

Parameter	Description
password_lock_time	Specifies the duration for a locked account to be automatically unlocked, in days.
recovery_time_target	Specifies the time for the standby node to write and replay logs, in seconds.
session_timeout	Specifies how long to wait before a server connection is disconnected due to inactivity. The value **0** indicates there is no time limit. Unit: second
timezone	Specifies the time zone for displaying and interpreting time stamps.
track_stmt_stat_level	Controls the level of statement execution tracking.
update_lockwait_timeout	Specifies the maximum duration that a lock waits for concurrent updates on a row to complete when the concurrent update feature is enabled. If the lock wait time exceeds this value, the system will report an error. Unit: millisecond
wal_level	Specifies the level of information to be written to the WAL. This is a required value and cannot be commented out. Determines how much information is written to the WAL. When this parameter is set to **logical**, logical logs are extracted and primary key information is recorded in Xlogs.
cn:audit_thread_num	Specifies the number of audit threads. Value range: **1** to **48**.
dn:audit_thread_num	Specifies the number of audit threads. Value range: **1** to **48**.
cn:qrw_inlist2join_optmode	Specifies whether to enable inlist-to-join query rewriting.
dn:qrw_inlist2join_optmode	Specifies whether to enable inlist-to-join query rewriting.
cn:audit_xid_info	Determines whether to record the transaction IDs of SQL statements in detail_info. **0**: The transaction IDs are not recorded. **1**: The transaction IDs are recorded.
dn:audit_xid_info	Determines whether to record the transaction IDs of SQL statements in detail_info. **0**: The transaction IDs are not recorded. **1**: The transaction IDs are recorded.

Parameter	Description
cn:default_limit_rows	Specifies the estimated number of rows to return by default for generating a generic plan, that is, the default value for the LIMIT clause. If this parameter is set to a negative number, the value is converted to a percentage, for example, **-5** is equivalent to 5%, indicating that 5% of the total rows will be returned.
dn:default_limit_rows	Specifies the estimated number of rows to return by default for generating a generic plan, that is, the default value for the LIMIT clause. If this parameter is set to a negative number, the value is converted to a percentage, for example, **-5** is equivalent to 5%, indicating that 5% of the total rows will be returned.
cn:audit_dml_state_select	Determines whether to audit the SELECT operation.
dn:audit_dml_state_select	Determines whether to audit the SELECT operation.
cn:audit_dml_state	Determines whether to audit the INSERT, UPDATE, and DELETE operations on a specific table. **0**: These operations are not audited. **1**: These operations are audited.
dn:audit_dml_state	Determines whether to audit the INSERT, UPDATE, and DELETE operations on a specific table. **0**: These operations are not audited. **1**: These operations are audited.
cn:random_page_cost	Specifies the estimated cost for the optimizer to fetch an out-of-sequence disk page.
dn:random_page_cost	Specifies the estimated cost for the optimizer to fetch an out-of-sequence disk page.
cn:enable_security_policy	Controls whether unified auditing and dynamic data masking policies are applied.
dn:enable_security_policy	Controls whether unified auditing and dynamic data masking policies are applied.
cn:audit_set_parameter	Determines whether to audit the SET operation. **0**: The SET operation is not audited. **1**: The SET operation is audited.
dn:audit_set_parameter	Determines whether to audit the SET operation. **0**: The SET operation is not audited. **1**: The SET operation is audited.

Parameter	Description
cn:enable_pbe_optimization	Specifies whether the optimizer optimizes the query plan for statements executed in Parse Bind Execute (PBE) mode.
dn:enable_pbe_optimization	Specifies whether the optimizer optimizes the query plan for statements executed in Parse Bind Execute (PBE) mode.
wdr_snapshot_interval	Specifies the interval (in minutes) at which the background thread Snapshot automatically performs snapshot operations on the database monitoring data.
enable_wdr_snapshot	Specifies whether to enable WDR snapshots. If this parameter is enabled, the kernel periodically takes performance snapshots.
cn:max_standby_archive_delay	Specifies the wait period (in milliseconds) before queries on a standby node are canceled when the queries conflict with WAL processing and archiving in hot standby mode.
dn:max_standby_archive_delay	Specifies the wait period (in milliseconds) before queries on a standby node are canceled when the queries conflict with WAL processing and archiving in hot standby mode.
cn:max_standby_streaming_delay	Specifies how long a standby node waits before canceling queries, in milliseconds.
dn:max_standby_streaming_delay	Specifies how long a standby node waits before canceling queries, in milliseconds.
cn:recovery_max_workers	Specifies the number of concurrent replayer threads.
dn:recovery_max_workers	Specifies the number of concurrent replayer threads.
cn:local_syscache_threshold	Specifies the size of system catalog cache in a session. Unit: KB
dn:local_syscache_threshold	Specifies the size of system catalog cache in a session. Unit: KB
cms:datastorage_threshold_value_check	Specifies the disk usage threshold to put a database node into read-only mode. If the disk usage of a data directory exceeds this threshold, the database node is automatically changed to read-only. Unit: percentage (%)
wdr_snapshot_retention_days	Specifies how many days database monitoring snapshots are saved for.

Configurable Parameters for Centralized Instances of Version V2.0-2.*x*

Table 12-7 Parameters for centralized instances

Parameter	Description
audit_system_object	Determines whether to audit the CREATE, DROP, and ALTER operations on GaussDB Kernel database objects. GaussDB Kernel database objects include databases, users, schemas, and tables. You can change the parameter value to audit only the operations on required database objects. During a forcible primary/standby failover, set **audit_system_object** to the maximum value and audit all DDL objects. If the parameter value is incorrectly changed, DDL audit logs will be lost. Contact technical support to change it.
autoanalyze	Specifies whether to automatically collect statistics on tables that have no statistics when a plan is generated.
autoanalyze_timeout	Specifies the autoanalyze timeout period. If the duration of autoanalyze on a table exceeds the value of **autoanalyze_timeout**, the autoanalyze operation is automatically canceled. **0** indicates there is no timeout. Unit: second
datestyle	Specifies the display format for date and time.
dn:wal_keep_segments	Specifies the minimum number of transaction log files stored in the **pg_xlog** directory. Standby nodes obtain the logs from the primary node to perform streaming replication.
enable_seqscan	Specifies whether to enable the optimizer's use of sequential scan plan types. It is impossible to completely suppress sequential scans, but setting this parameter to **off** allows the optimizer to choose other methods if available.
enable_slot_log	Specifies whether to enable primary/standby synchronization for logical replication slots.
failed_login_attempts	Specifies the maximum number of incorrect password attempts before an account is locked. The account will be automatically unlocked after the time specified in **password_lock_time** elapses. Only the **sysadmin** user can set this parameter.

Parameter	Description
log_min_duration_statement	Specifies the threshold for logging the duration of a completed statement. If a statement runs for a period greater than or equal to the specified value, its duration will be logged. The value **-1** disables logging statement durations. If this parameter is set to a small value, the load throughput may be affected. Unit: millisecond
max_replication_slots	Specifies the number of log replication slots on the primary node.
max_wal_senders	Specifies the maximum number of WAL sender threads that can be created. Processes created for the following purposes occupy WAL sender threads: Standby DNs connect to primary DNs to obtain physical logs, and logical replication tools connect to primary DNs to obtain logical logs.
password_effect_time	Specifies the validity period of the password, in days.
password_lock_time	Specifies the duration for a locked account to be automatically unlocked, in days.
session_timeout	Specifies how long to wait before a server connection is disconnected due to inactivity. The value **0** indicates there is no time limit. Unit: second
timezone	Specifies the time zone for displaying and interpreting time stamps.
update_lockwait_timeout	Specifies the maximum duration that a lock waits for concurrent updates on a row to complete when the concurrent update feature is enabled. If the lock wait time exceeds this value, the system will report an error. Unit: millisecond
wal_level	Specifies the level of information to be written to the WAL. This is a required value and cannot be commented out. Determines how much information is written to the WAL. When this parameter is set to **logical**, logical logs are extracted and primary key information is recorded in Xlogs.
dn:audit_thread_num	Specifies the number of audit threads. Value range: **1** to **48**.
dn:qrw_inlist2join_optmode	Specifies whether to enable inlist-to-join query rewriting.

Parameter	Description
dn:audit_xid_info	Determines whether to record the transaction IDs of SQL statements in detail_info. **0**: The transaction IDs are not recorded. **1**: The transaction IDs are recorded.
dn:default_limit_rows	Specifies the estimated number of rows to return by default for generating a generic plan, that is, the default value for the LIMIT clause. If this parameter is set to a negative number, the value is converted to a percentage, for example, **-5** is equivalent to 5%, indicating that 5% of the total rows will be returned.
dn:audit_dml_state_select	Determines whether to audit the SELECT operation.
dn:audit_dml_state	Determines whether to audit the INSERT, UPDATE, and DELETE operations on a specific table. **0**: These operations are not audited. **1**: These operations are audited.
dn:random_page_cost	Specifies the estimated cost for the optimizer to fetch an out-of-sequence disk page.
dn:enable_security_policy	Controls whether unified auditing and dynamic data masking policies are applied.
dn:audit_set_parameter	Determines whether to audit the SET operation. **0**: The SET operation is not audited. **1**: The SET operation is audited.
dn:max_standby_streaming_delay	Specifies how long a standby node waits before canceling queries, in milliseconds.
dn:vacuum_defer_cleanup_age	Specifies the number of transactions used by VACUUM.
dn:enable_pbe_optimization	Specifies whether the optimizer optimizes the query plan for statements executed in Parse Bind Execute (PBE) mode.
wdr_snapshot_interval	Specifies the interval (in minutes) at which the background thread Snapshot automatically performs snapshot operations on the database monitoring data.
undo_retention_time	Specifies how long undo logs are kept, in seconds. This parameter is only used for flashback query. Note: 1. The undo space of the local disk increases. 2. In subsequent incremental backups, the size of the backup set increases, because extra undo content is retained.

Parameter	Description
track_stmt_stat_level	Controls the level of statement execution tracking.
enable_wdr_snapshot	Specifies whether to enable WDR snapshots. If this parameter is enabled, the kernel periodically takes performance snapshots.
dn:max_standby_archive_delay	Specifies the wait period (in milliseconds) before queries on a standby node are canceled when the queries conflict with WAL processing and archiving in hot standby mode.
dn:max_standby_streaming_delay	Specifies how long a standby node waits before canceling queries, in milliseconds.
dn:recovery_max_workers	Specifies the number of concurrent replayer threads.
dn:recovery_time_target	Specifies the time for the standby node to write and replay logs. Unit: second
dn:local_syscache_threshold	Specifies the size of system catalog cache in a session. Unit: KB
cms:datastorage_threshold_value_check	Specifies the disk usage threshold to put a database node into read-only mode. If the disk usage of a data directory exceeds this threshold, the database node is automatically changed to read-only. Unit: percentage (%)
wdr_snapshot_retention_days	Specifies how many days database monitoring snapshots are saved for.
log_autovacuum_min_duration	Specifies the interval which should elapse before autovacuum operations are logged. Autovacuum operations equal to or beyond the specified interval will be logged. If it is set to **0**, all autovacuum operations will be logged. If it is set to **-1**, no autovacuum operations will be logged.
dn:max_connections	Specifies the maximum number of concurrent connections to DNs.

12.2 Modifying GaussDB Instance Parameters

You can modify parameters of a GaussDB instance to bring out the best possible performance of the instance. You can also check the parameter values of an instance.

GaussDB provides the following types of parameters:

- Public parameters: GaussDB uses a set of default running parameters after it is installed. You can modify the parameters to better fit your application scenarios and data volume.
- Parameters for data redistribution: These parameters are used to control the data redistribution policy during database scale-out.

Precautions

- Parameters for data redistribution can be modified only for distributed instances of version V2.0-2.6 or later.
- Parameters of read replicas can be modified only for centralized (1 primary + 2 standby) instances of version V2.0-2.7.1 or later.
- Before modifying parameters, make sure you understand their meanings and fully verify the changes in a test environment to avoid instance or service exceptions caused by inappropriate parameter settings.

Modifying Common Parameters of the Current Instance

Step 1 Log in to the management console.

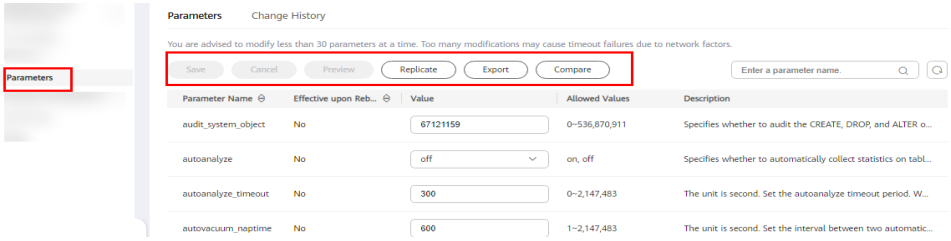
Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the navigation pane on the left, choose **Parameters**.

Figure 12-1 Parameters



- You can modify and query the parameters applied to the instance on this page. After modifying parameters, you can preview the changes or cancel the modification.

  After confirming that all changes are correct, click **Save**.

  NOTE

  The modification of some parameters takes effect only after the instance is rebooted. After you modify a parameter value, view the value in the **Effective upon Reboot** column.

  - If the value is **Yes** and the instance status on the **Instances** page is **Parameter change. Pending reboot**, you must reboot the instance for the modifications to take effect.
  - If the value is **No**, the modifications take effect immediately for the instance.

- You can click **Replicate** to save the parameters of the instance as a parameter template. You can view the parameter template under the **Custom Templates** tab of the **Parameter Templates** page. For details, see **Managing Parameter Templates for GaussDB Instances**.
- You can click **Export** to download the parameters of the instance to your local PC.
- You can click **Compare** to compare the parameter template applied to the current instance with an existing parameter template.

----End

Modifying Common Parameters of Multiple Instances at a Time

Step 1 **Log in to the management console**.

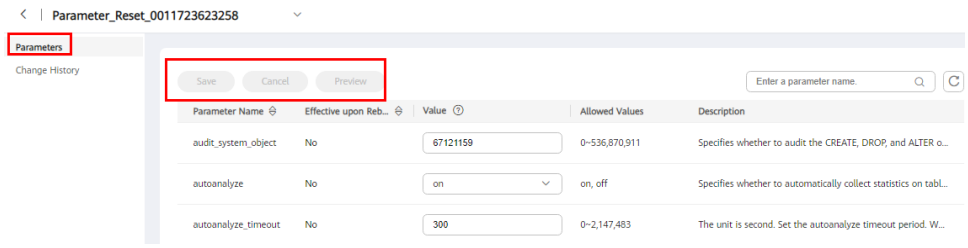
Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 In the navigation pane on the left, choose **Parameter Templates**. Click the **Custom Templates** tab, and click the name of the target parameter template.

Step 5 Modify parameters as needed.

Figure 12-2 Modifying parameters



- Click **Save**. In the displayed dialog box, click **Yes** to save the modification.
- To cancel your changes, click **Cancel**.
- To preview your changes, click **Preview**.

Step 6 After the parameters are modified, click **Change History** to view what changes have been made.

NOTICE

The changes take effect only after you apply the parameter template to instances. For details, see **Applying a Parameter Template**.

----End

Modifying Data Redistribution Parameters of the Current Instance

Step 1 **Log in to the management console**.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the navigation pane on the left, choose **Parameters**. On the displayed page, click **Change Parameters for Scale-out** or **Change Parameters for Redistribution**.

Figure 12-3 Parameters

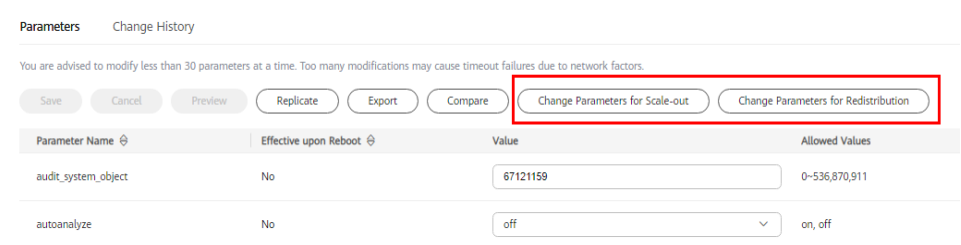


Figure 12-4 Changing parameters for scale-out

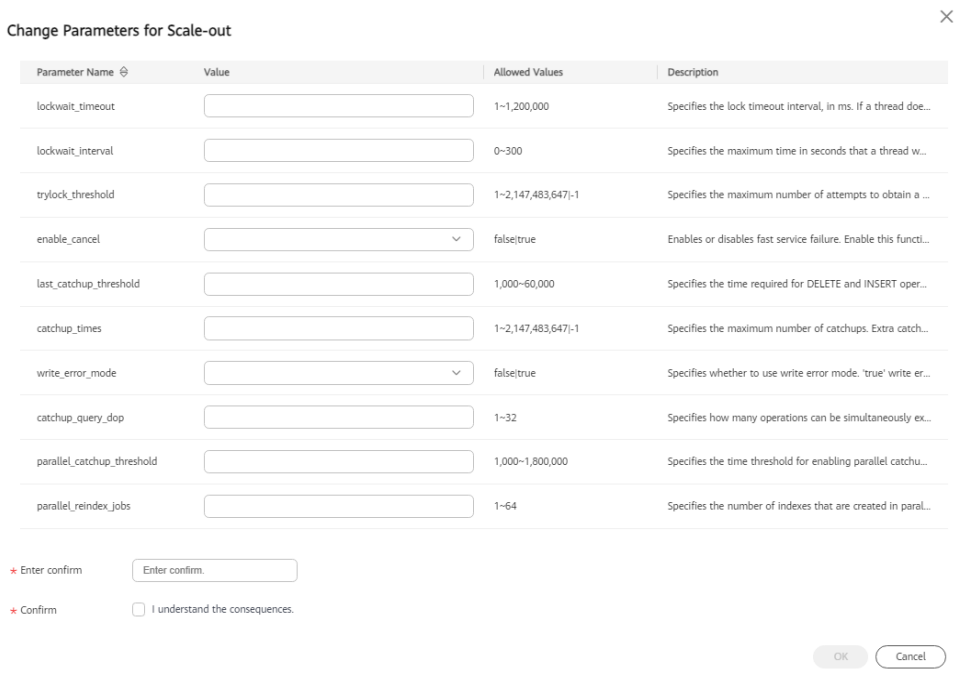
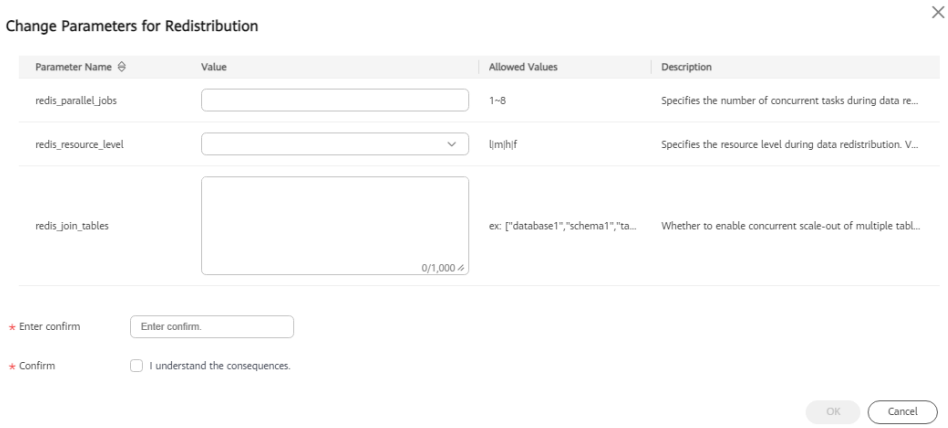


Figure 12-5 Changing parameters for redistribution



Step 6 Enter required parameter values, enter **confirm** in the text box, select the confirmation check box, and click **OK**.

----End

12.3 Viewing Parameter Change History of a GaussDB Instance

Scenarios

You can view the change history of DB instance parameters or custom parameter templates.

NOTE

- In a newly replicated or created parameter template, the change history is blank.
- The change history of the last seven days is displayed.
- The parameter change history of read replicas is available only for centralized (1 primary + 2 standby) instances of version V2.0-2.7.1 or later.

Viewing Change History of DB Instance Parameters

Step 1 Log in to the management console.

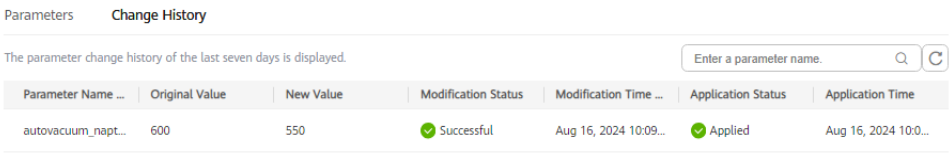
Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the navigation pane on the left, choose **Parameters**.

Figure 12-6 Viewing the parameter change history



Step 6 On the displayed page, click **Change History**.

You can view the parameter name, original parameter value, new parameter value, modification status, modification time, application status, and application time.

You can apply the parameter template to instances as required by referring to **Applying a Parameter Template**.

----End

Viewing Change History of a Parameter Template

Step 1 **Log in to the management console**.

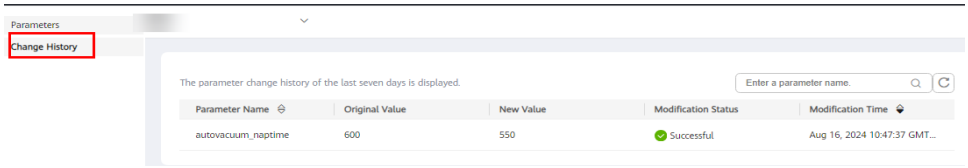
Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 Choose **Parameter Templates** in the navigation pane on the left. On the **Custom Templates** page, click the parameter template name.

Step 5 On the displayed page, choose **Change History** in the navigation pane on the left.

Figure 12-7 Viewing the change history of a parameter template



You can view the parameter name, original parameter value, new parameter value, modification status, and modification time.

----End

12.4 Exporting Parameters of a GaussDB Instance

Scenarios

You can export the parameter template information (parameter names, values, and descriptions) of a DB instance to a CSV file for analysis.

Precautions

The parameters of read replicas can be exported only for centralized (1 primary + 2 standby) instances of version V2.0-2.7.1 or later.

Procedure

Step 1 Log in to the management console.

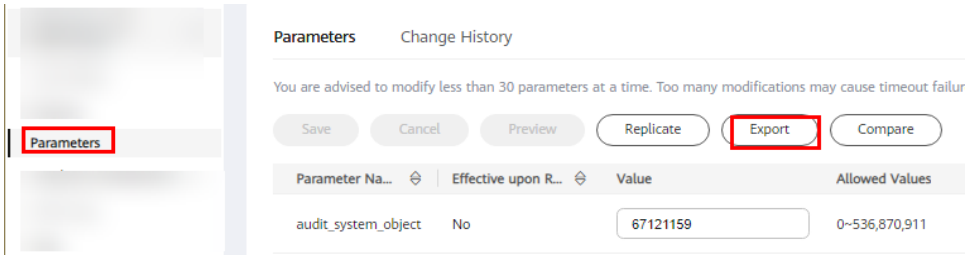
Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the navigation pane on the left, choose **Parameters**. On the displayed page, click **Export** above the parameter list.

Figure 12-8 Exporting parameters



Exporting to a file: You can export the parameter template information (parameter names, parameter values, and descriptions) of an instance to a CSV file for analysis.

Step 6 In the displayed dialog box, enter the file name and click **OK**.

NOTE

The file name must start with a letter and consist of 4 to 81 characters. It can contain only letters, numbers, hyphens (-), and underscores (_).

----End

12.5 Creating a Parameter Template for GaussDB Instances

You can use parameters in a parameter template to manage DB engine configurations. A parameter template can be applied to one or more instances.

If you create a DB instance without specifying a custom parameter template, a default parameter template is used. This default template contains DB engine defaults and system defaults based on the engine, compute specifications, and allocated storage of the instance. Default parameter templates cannot be

modified, but you can create your own parameter template to change parameter settings.

NOTICE

Not all DB engine parameters can be changed in a custom parameter template.

If you want to use your custom parameter template, create a parameter template and select it when you create a DB instance or apply it to an existing DB instance by following the instructions provided in **Applying a Parameter Template**.

When you have already created a parameter template and want to include most of the custom parameters and values from that template in a new parameter template, you can replicate that parameter template following the instructions provided in **Replicating a Parameter Template**.

The following are the key points you should know when using parameters in a parameter template:

- In the **Parameters** page, when you change a parameter value in a parameter template and save the change, the change applies only to current instance and does not affect the other instances.
- Some modifications take effect only after you manually reboot the DB instance.
- Improperly setting parameters in a parameter template may have unintended adverse effects, including degraded performance and system instability. Exercise caution when changing database parameters and you need to back up data before changing parameters in a parameter template. Do not perform the boundary testing in the parameter template, or the instance will be abnormal. Before applying parameter template changes to a production DB instance, you should try out these changes on a test DB instance.

NOTE

GaussDB parameter template quotas are not shared by DDS.

A maximum of 100 GaussDB database parameter templates can be created for each project. All GaussDB engines share the parameter template quota.

Procedure

Step 1 **Log in to the management console**.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 In the navigation pane on the left, choose **Parameter Templates**.

Step 5 On the **Parameter Templates** page, click **Create Parameter Template**.

Step 6 In the displayed dialog box, configure required information and click **OK**.

- Select a DB engine for the parameter template.

- The template name is case-sensitive and can contain 1 to 64 characters. Only uppercase letters, lowercase letters, digits, hyphens (-), underscores (_), and periods (.) are allowed.
- The template description can contain a maximum of 256 characters and cannot include carriage return characters and the following special characters: >!<"&'=

----End

12.6 Managing Parameter Templates for GaussDB Instances

You can perform the following operations to manage GaussDB parameter templates:

- **Applying a parameter template**: Modifications to parameters in a parameter template take effect for instances only after you apply this parameter template to target instances. A parameter template can be applied only to instances of the same version.
- **Replicating a parameter template**: You can replicate a custom parameter template that you have created. When you have already created a parameter template and want to include most of the custom parameters and values from that template in a new parameter template, you can replicate that parameter template. You can also replicate the parameter template used by an existing instance to generate a new one for future use.
- **Comparing instance parameters with a parameter template**: You can compare instance parameters with a parameter template to see the differences of parameter settings.
- **Comparing parameter templates**: You can compare two default GaussDB parameter templates to see the differences between them. You can also compare two custom parameter templates.
- **Resetting a parameter template**: You can reset all parameters in a parameter template you have created to their default settings.
- **Modifying the description of a parameter template**: You can edit the description of a parameter template you have created.
- **Deleting a parameter template**: You can create up to 100 parameter templates and delete parameter templates that are no longer used.

Applying a Parameter Template

Step 1 Log in to the management console.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Parameter Templates** page, perform the following operations based on the type of the parameter template to be applied:

- If you intend to apply a default parameter template to instances, click the **Default Templates** tab and click **Apply** in the **Operation** column of the target parameter template.
- If you intend to apply a custom parameter template to instances, click **Custom Templates** and choose **More** > **Apply** in the **Operation** column of the target parameter template.

A parameter template can be applied to one or more instances.

Step 5 In the displayed dialog box, select one or more instances to which the parameter template will be applied and click **OK**.

Step 6 After the parameter template is applied, check its application records.

- If you intend to check the application records of a default parameter template, click the **Default Templates** tab and click **View Application Record** in the **Operation** column of the target parameter template.
- If you intend to check the application records of a custom parameter template, click **Custom Templates** and choose **More** > **View Application Record** in the **Operation** column of the target parameter template.

If the application status of an instance is **Applying**, it will not be displayed in the instance list when you apply a parameter template again. If you want to apply the parameter template to the same instance again, ensure that the application status is **Successful**.

NOTE

After the parameter template is successfully applied, if you modify parameters in the parameter template and the instance status is **Parameter change. Pending reboot**, you must reboot the instance for the modifications to take effect. If no parameter that requires an instance reboot is modified, the instance status remains unchanged.

----End

Replicating a Parameter Template

Step 1 **Log in to the management console**.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Parameter Templates** page, click the **Custom Templates** tab. Locate the parameter template to be replicated and click **Replicate** in the **Operation** column.

Alternatively, on the **Instances** page, click the instance name to go to the **Basic Information** page. In the navigation pane, choose **Parameters**. On the displayed page, click **Replicate** to generate a new parameter template for future use.

Step 5 In the displayed dialog box, configure required details and click **OK**.

- The template name is case-sensitive and can contain 1 to 64 characters. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed.
- The template description can contain up to 256 characters, but cannot contain carriage returns and the following special characters: >!<"&'=

After the parameter template is replicated, a new template is generated in the list on the **Custom Templates** tab of the **Parameter Templates** page.

NOTE

- The new parameter template may not be displayed immediately. You are advised to wait for at least 5 minutes before using the new template.
- Default parameter templates cannot be replicated, but you can create parameter templates based on them.

----End

Comparing Instance Parameters with a Parameter Template

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the instance name to go to the **Basic Information** page.

Step 5 In the navigation pane on the left, choose **Parameters**.

Step 6 On the displayed page, click **Compare** to compare the parameters of the current instance.

Step 7 In the displayed dialog box, select a parameter template that you want to compare with parameters of the current instance and click **OK**.

- If their settings are different, the parameter names and values of both parameter templates are displayed.
- If their settings are the same, no data is displayed.

NOTICE

Parameters of read replicas can be compared only for centralized (1 primary + 2 standby) instances of version V2.0-2.7.1 or later.

----End

Comparing Parameter Templates

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Parameter Templates** page, click **Default Templates** or **Custom Templates**. Locate a parameter template and click **Compare** in the **Operation** column.

Step 5 In the displayed dialog box, select a parameter template that uses the same DB engine as the target template and click **OK**.

- If their settings are different, the parameter names and values of both parameter templates are displayed.
- If their settings are the same, no data is displayed.

----End

Resetting a Parameter Template

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Parameter Templates** page, click the **Custom Templates** tab. Locate the parameter template and choose **More** > **Reset** in the **Operation** column

Step 5 Click **Yes** to reset all parameters to their default values.

NOTE

After you reset a parameter template, you need to view the status of the instance to which the parameter template applies in the instance list. If its status is **Parameter change. Pending reboot**, you must reboot the instance for the reset to take effect.

----End

Modifying the Description of a Parameter Template

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Parameter Templates** page, click the **Custom Templates** tab. Locate the parameter template and click in the **Description** column.

Step 5 Enter a new description. You can click to submit or to cancel the modification.

- After you submit the modification, you can view the new description in the **Description** column.
- The description can contain up to 256 characters, but cannot contain the following special characters: >!<"&'=

NOTE

You cannot modify the description of any default parameter template.

----End

Deleting a Parameter Template

NOTICE

- Deleted parameter templates cannot be recovered. Exercise caution when performing this operation.
- Default parameter templates cannot be deleted.

Step 1 Log in to the management console.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Parameter Templates** page, click the **Custom Templates** tab. Locate the parameter template to be deleted and choose **More** > **Delete** in the **Operation** column.

Step 5 Click **Yes** to delete it.

----End

13 Monitoring and Alarming

13.1 Supported Metrics of GaussDB

Description

This section describes metrics reported by GaussDB as well as their namespaces and dimensions.

Namespace

SYS.GAUSSDBV5

Metric Collection Constraints

- Standby DNs of distributed instances: Metric data can be collected only when the instance version is V2.0-3.100.0 or later, and the transaction consistency must be eventual consistency.
- Standby DNs of centralized instances: Metric data can be collected only when the instance version is V2.0-2.0.10 or later

Supported Metrics

The following table lists the performance metrics of GaussDB.

Table 13-1 Monitoring metrics supported by GaussDB

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds001_cpu_util	CPU Usage	CPU usage of the monitored object	Current node	%	Node	60s

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds002_mem_util	Memory Usage	Memory usage of the monitored object	Current node	%	Node	60s
rds003_bytes_in	Data Write Volume	Average number of bytes sent by the VM of the monitored object in a measurement period	Current node	Byte/s	Node	60s
rds004_bytes_out	Outgoing Data Volume	Average number of bytes received by the VM of the monitored object in a measurement period	Current node	Byte/s	Node	60s
rds014_iops	Disk IOPS	Real-time value of data disk reads and writes per second of the monitored node	Current node	Count/s	Node	60s
rds016_disk_write_throughput	Disk Write Throughput	Real-time write throughput per second of the data disk on the monitored node	Current node	Byte/s	Node	60s

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds017_disk_read_throughput	Disk Read Throughput	Real-time read throughput per second of the data disk on the monitored node	Current node	Byte/s	Node	60s
rds020_avg_disk_ms_per_write	Time Required for per Data Disk Write	Average time required for a data disk write on the monitored node in a measurement period	Current node	ms	Node	60s
rds021_avg_disk_ms_per_read	Time Required for per Data Disk Read	Average time required for a data disk read on the monitored node in a measurement period	Current node	ms	Node	60s
io_bandwidth_usage	Disk I/O Bandwidth Usage	Percentage of current disk I/O bandwidth	Current node	%	Node	60s
iops_usage	IOPS Usage	Percentage of used IOPS in the total IOPS	Current node	%	Node	60s
rds005_instance_disk_used_size	Used Instance Disk Size	Real-time used data disk size of the monitored instance	Instance	GB	Instance	60s

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds006_instance_disk_total_size	Total Instance Disk Size	Real-time total data disk size of the monitored instance	Instance	GB	Instance	60s
rds007_instance_disk_usage	Instance Disk Usage	Real-time data disk usage of the monitored instance	Instance	%	Instance	60s
rds035_buffer_hit_ratio	Buffer Hit Rate	Buffer hit rate of the database	Instance	%	Instance	60s
rds036_deadlocks	Deadlocks	Incremental number of database transaction deadlocks in a measurement period	Instance	Count	Instance	60s
rds048_P80	Response Time of 80% SQL Statements	Real-time response time of 80% of database SQL statements	Instance	us	Instance	60s
rds049_P95	Response Time of 95% SQL Statements	Real-time response time of 95% of database SQL statements	Instance	us	Instance	60s
rds008_disk_used_size	Used Disk Size	Real-time used data disk size of the monitored node	Current node	GB	Component	60s

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds009_disk_total_size	Total Disk Size	Real-time total data disk size of the monitored node	Current node	GB	Component	60s
rds010_disk_usage	Disk Usage	Real-time data disk usage of the monitored node	Current node	%	Component	60s
rds024_current_sleep_time	Primary Node Flow Control Duration	Real-time primary node flow control duration on the monitored node	Distributed: standby DN Centralized: standby DN	us	Component	60s
rds025_current_rto	Standby Node RTO	Real-time Recovery Time Objective (RTO) of the primary/standby replication of the monitored node	Distributed: standby DN Centralized: standby DN	s	Component	60s
rds026_login_counter	User Logins per Second	Average number of logins per second in a measurement period	Distributed: all CNs Centralized: primary DN	Count/s	Component	60s

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds027_logout_counter	User Logouts per Second	Average number of logouts per second in a measurement period	Distributed: all CNs Centralized: primary DN	Count/s	Component	60s
rds028_standby_delay	Standby Node Redo Progress	Real-time redo progress of the standby node in a shard. It indicates the difference of the redo progress between the primary and standby nodes.	Distributed: standby DN Centralized: primary DN	Byte	Component	60s
rds030_wait_ratio	Lock-Waiting Session Rate	Real-time rate of lock waiting sessions to active sessions	Distributed: all CNs + primary DN Centralized: all DNs	%	Component	60s
rds031_active_ratio	Active Session Rate	Real-time rate of active sessions to all sessions	Distributed: all CNs + primary DN Centralized: all DNs	%	Component	60s

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds034_inuse_counter	CN Connections	Real-time number of in-use connections in the CN connection pool	Distributed: all CNs Centralized: N/A	Count	Component	60s
rds037_commit_counter	User Committed Transactions per Second	Average number of transactions committed by users per second in a measurement period	Distributed: all CNs Centralized: primary DN	Count/s	Component	60s
rds038_rollback_counter	User Rollback Transactions per Second	Average number of transactions rolled back by users per second in a measurement period	Distributed: all CNs Centralized: primary DN	Count/s	Component	60s
rds039_bg_commit_counter	Background Committed Transactions per Second	Average number of transactions committed by the background per second in a measurement period	Distributed: all CNs Centralized: primary DN	Count/s	Component	60s
rds040_bg_rollback_counter	Background Rollback Transactions per Second	Average number of transactions rolled back by the background per second in a measurement period	Distributed: all CNs Centralized: primary DN	Count/s	Component	60s

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds041_resp_avg	Average Response Time of User Transactions	Average response time of user transactions	Distributed: all CNs Centralized: primary DN	us	Component	60s
rds042_rollback_ratio	User Transaction Rollback Rate	Average rate of user rollback transactions to all user committed and rolled back transactions in a measurement period	Distributed: all CNs Centralized: primary DN	%	Component	60s
rds043_bg_rollback_ratio	Background Transaction Rollback Rate	Average rate of background rollback transactions to all user committed and rolled back transactions in a measurement period	Distributed: all CNs Centralized: primary DN	%	Component	60s
rds044_ddl_count	Data Definition Language/s	Average number of DDL statements in user load at the query layer in a measurement period	Distributed: all CNs + all DNs Centralized: all DNs	Count/s	Component	60s

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds045_dml_count	Data Manipulation Language/s	Average number of DML statements in user load at the query layer in a measurement period	Distributed: all CNs + all DNs Centralized: all DNs	Count/s	Component	60s
rds046_dcl_count	Data Control Language/s	Average number of DCL statements in user load at the query layer in a measurement period	Distributed: all CNs + all DNs Centralized: all DNs	Count/s	Component	60s
rds047_ddl_dcl_ratio	DDL and DCL Rate	Average rate of DDL and DCL statements to DDL, DCL, and DML statements in user load at the query layer in a measurement period	Distributed: all CNs + all DNs Centralized: all DNs	%	Component	60s
rds050_ckpt_delay	Data Volume to Be Flushed to Disks	Real-time amount of data to be flushed to disks during synchronization	Distributed: all CNs + primary DN Centralized: primary DN	Byte	Component	60s

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds051_phyrds	Physical Reads per Second	Average number of physical reads per second in a measurement period	Distributed: all CNs + primary DN Centralized: all DNs	Count/s	Component	60s
rds052_phywrts	Physical Writes per Second	Average number of physical writes per second in a measurement period	Distributed: all CNs + primary DN Centralized: all DNs	Count/s	Component	60s
rds053_online_session	Online Sessions	Real-time number of online sessions	Distributed: all CNs + all DNs Centralized: all DNs	Count	Component	60s
rds054_active_session	Active Sessions	Real-time number of active sessions	Distributed: all CNs + primary DN Centralized: primary DN	Count	Component	60s

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds055_online_ratio	Online Session Rate	Real-time percentage of online sessions on a CN (of a distributed instance) or a primary DN (of a centralized instance)	Distributed: all CNs + primary DN Centralized: all DNs	%	Component	60s
rds060_long_running_transaction_exectime	Maximum Execution Duration of Database Transactions	Maximum execution duration of database transactions	Distributed: all CNs + primary DN Centralized: all DNs	s	Component	60s
rds066_replication_slot_wal_log_size	WAL Log Size in the Replication Slot	Real-time size of WAL logs reserved in the replication slot of a primary DN	Distributed: primary DN Centralized: all DNs	Byte	Component	60s
rds067_xlog_lsn	Xlog Rate	Real-time rate of Xlogs on CNs or primary DNs	Distributed: all CNs + primary DN Centralized: primary DN	Byte/s	Component	60s
rds068_swap_used_ratio	Swap Memory Usage	Real-time swap memory usage of the OS	Current node	%	Node	60s

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds069_swap_total_size	Total Swap Memory	Real-time total swap memory size of the OS	Current node	MB	Node	60s
rds070_thread_pool	Thread Pool Usage	Real-time thread pool usage on a CN and DN	Distributed: all CNs + primary DN Centralized: all DNs	%	Component	60s
rds071_locks_session	Sessions Waiting for Locks	Number of sessions waiting for locks on a CN or primary DN. This metric is updated in real time	Distributed: all CNs + primary DN Centralized: all DNs	Count	Component	60s
rds072_streaming_dr_xlog_gap	Shard Log Gap of DR Cluster	Log difference between shards in the DR cluster and shards in the production cluster when streaming DR is enabled	Distributed: all CNs + primary DN Centralized: primary DN	Byte	Component	60s
rds073_streaming_dr_xlog_to_be_replayed	Size of Shard Logs to Be Replayed in DR Cluster	Size of the logs to be replayed of each shard in the DR cluster when streaming DR is enabled	Distributed: all CNs + primary DN Centralized: primary DN	Byte	Component	60s

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds074_streaming_dr_xlog_flushing_rate	Flushing Rate of Shard Logs in DR Cluster	Rate at which logs of each shard in the DR cluster are flushed to disk when streaming DR is enabled	Distributed: all CNs + primary DN Centralized: primary DN	Byte/s	Component	60s
rds075_streaming_dr_xlog_replay_rate	Replay Rate of Shard Logs in DR Cluster	Rate at which logs of each shard in the DR cluster are replayed when streaming DR is enabled	Distributed: all CNs + primary DN Centralized: primary DN	Byte/s	Component	60s
rds076_streaming_dr_rpo	Shard RPO	Real-time RPO of each shard when streaming DR is enabled	Distributed: all CNs + primary DN Centralized: primary DN	s	Component	60s
rds077_streaming_dr_rto	Shard RTO	Real-time RTO of each shard when streaming DR is enabled	Distributed: all CNs + primary DN Centralized: primary DN	s	Component	60s

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds078_inactive_replication_slot	Inactive Replication Slots	Number of physical and logical replication slots that are inactive	Distributed: all CNs + primary DN Centralized: all DNs	Count	Component	60s
rds079_standy_not_replayed_log	Size of Read Replica Logs Not Replayed	Difference between the number of replayed read replica logs and the number of received read replica logs	Distributed: standby DN Centralized: standby DN	Byte	Component	60s
rds080_xlog_num	Xlogs	Real-time number of Xlogs in the data directory on a CN or DN	Distributed: all CNs + all DNs Centralized: all DNs	Count	Component	60s
rds081_xlog_size	Xlog Size	Real-time size of Xlogs in the data directory on a CN or DN	Distributed: all CNs + all DNs Centralized: all DNs	MB	Component	60s
rds064_dynamic_used_memory	Used Dynamic Memory	Real-time used dynamic memory of the monitored object	Distributed: all CNs + all DNs Centralized: all DNs	MB	Component	60s

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds065_dynamic_used_memory_usage	Dynamic Memory Usage	Real-time dynamic memory usage of the monitored object	Distributed: all CNs + all DNs Centralized: all DNs	%	Component	60s
rds061_idle_in_transaction_num	Idle Transactions	Real-time reporting of how many idle transactions there are for the monitored object	Distributed: all CNs + all DNs Centralized: all DNs	Count	Component	60s
rds062_slowquery_sys	Slow SQL Statements in the System Database	Real-time number of slow SQL statements in the system database on the primary DN or CN in a measurement period	Distributed: all CNs Centralized: primary DN	Count	Component	60s
rds063_slowquery_user	Slow SQL Statements in the User Database	Real-time number of slow SQL statements in the user database on the primary DN or CN in a measurement period	Distributed: all CNs Centralized: primary DN	Count	Component	60s

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds082_gaussv5_wait_session	Waiting Sessions	Real-time number of waiting sessions	Distributed: all CNs + standby DN Centralized: all DNs	Count	Component	60s
rds083_cn_temp_dir_size	CN Temporary Directory Size	Real-time size of the temporary directories under the data directory on a CN	Distributed: all CNs	MB	Component	60s
rds084_sys_database_size	System Database Size	Real-time **postgres** database size on the monitored instance	Current node	Byte	Node	60s
rds085_user_database_size	User Database Total Size	Real-time user database size on the monitored instance	Current node	Byte	Node	60s
rds086_select_distribution	SELECT Distribution	Real-time percentage of SELECT statements	Distributed: all CNs + all DNs Centralized: all DNs	%	Component	60s

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds087_update_distribution	UPDATE Distribution	Real-time percentage of UPDATE statements	Distributed: all CNs + all DNs Centralized: all DNs	%	Component	60s
rds088_insert_distribution	INSERT Distribution	Real-time percentage of INSERT statements	Distributed: all CNs + all DNs Centralized: all DNs	%	Component	60s
rds089_delete_distribution	DELETE Distribution	Real-time percentage of DELETE statements	Distributed: all CNs + all DNs Centralized: all DNs	%	Component	60s
rds091_gaussv5_qps	Read Requests	Average number of read requests per second of a tenant in a specified period	Distributed: all CNs Centralized: all DNs	Count	Component	60s
rds092_gaussv5_tps_rt_insert	INSERT Request Response Time	Average response time for INSERT requests of a tenant in a specified period	Distributed: all CNs Centralized: all DNs	ms	Component	60s

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds093_gaussv5_tps_rt_update	UPDATE Request Response Time	Average response time for UPDATE requests of a tenant in a specified period	Distributed: all CNs Centralized: all DNs	ms	Component	60s
rds094_gaussv5_tps_rt_delete	DELETE Request Response Time	Average response time for DELETE requests of a tenant in a specified period	Distributed: all CNs Centralized: all DNs	ms	Component	60s
rds095_gaussv5_qps_rt	Read Request Response Time	Average response time for read requests of a tenant in a specified period	Distributed: all CNs Centralized: all DNs	ms	Component	60s
retrans_rate	Retransmission Ratio	Real-time retransmission ratio of TCP packets	Current node	%	Node	60s
rds096_process_used_memory	Process Used Memory	Real-time used memory by a CN or DN	Distributed: all CNs + all DNs Centralized: all DNs	MB	Component	60s
rds097_2pc_transaction_prepare	Oldest Two-Phase Commit Transaction Duration	Maximum duration of uncommitted transactions using two-phase commit	Centralized: primary DN	s	Component	60s

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds098_dn_instance_status	DN Status	Real-time status of a DN. **1**: a normal primary DN; **2**: a normal standby DN; **3**: a normal main standby DN; **4**: a normal cascaded standby DN; **10**: standby DN catching up with primary DN using Xlog files; **20**: a properly connected standby DN with abnormal replication status; **21**: a disconnected DN	Centralized: all DNs	N/A	Component	60s
rds099_replication_slot_dir_size	Replication Slot Directory Size	Real-time size of the replication slot directory	Centralized: all DNs	KB	Component	300s

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds100_standby_diff_redo_and_receive	Difference Between Redo and Receipt Positions on Standby Node	The difference (in bytes) between the redo position and data receipt position on the standby node. This metric is used to determine whether data inconsistency is caused by slow redo rate on the standby node or because the primary node has not sent redo data.	Distributed: standby DN Centralized: standby DN	Byte	Component	60s
rds101_online_distinct_client_addr_count	Online Clients	Number of online clients on each CN	Distributed: all CNs	Count	Component	60s
rds102_working_distinct_client_addr_count	Active Clients	Number of active client connections on each CN	Distributed: all CNs	Count	Component	60s

Metric ID	Metric	Description	Display Object	Unit	Monitored Object	Monitoring Period (Raw Data)
rds103_shard_min_rto	Shard RTO	Shortest possible RTO among multiple standby DNs within the current shard.	Centralized: primary DN Distributed: primary DN	s	Component	60s
rds104_invalid_usr_pwd_login_denied_count	Login Attempts with Incorrect Usernames or Passwords	The number of failed login attempts due to incorrect usernames or passwords in kernel logs. The value is the difference between two collected values (incremental value).	Centralized: all DNs Distributed: all CNs	Count	Component	300s

Dimensions

Table 13-2 Dimensions

Key	Value
gaussdbv5_instance_id	GaussDB instance
gaussdbv5_node_id	GaussDB node
gaussdbv5_component_id	GaussDB component

13.2 Querying GaussDB Monitoring Metrics

Scenarios

Cloud Eye monitors operating statuses of DB instances. You can view the DB instance monitoring metrics on the management console. For details, see **Viewing Metrics of DB Instances**.

Monitored data takes some time for transmission and display. The DB instance status displayed on the Cloud Eye console is the status of the last 5 to 10 minutes. If your DB instance is newly created, wait for 5 to 10 minutes and then view the monitoring data.

You can also view database metrics on the **Monitoring Dashboards** page of the GaussDB console. On the **Monitoring Dashboards** page, you can select all nodes or components of an instance at the same time to check their metrics. In this way, you can easily compare metric differences between components or nodes. In contrast, CES allows you to check metrics of only one node or component at a time.

Prerequisites

- A DB instance is running properly.

  Monitoring metrics of the DB instances that are faulty or have been deleted cannot be displayed on the Cloud Eye console. You can view their monitoring metrics after they are rebooted or restored to be normal.

  NOTE

  If a DB instance has been faulty for 24 hours, Cloud Eye considers that it does not exist and deletes it from the monitoring object list. You need to manually clear the alarm rules created for the DB instance.

- The DB instance keeps running properly for about 10 minutes.

  For a newly created DB instance, you need to wait for a while before viewing the monitoring metrics.

Viewing Metrics of DB Instances

Step 1 **Log in to the management console**.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Under **Management & Governance** of the service list, click **Cloud Eye**.

Step 4 In the navigation pane on the left, choose **Cloud Service Monitoring** > **GaussDB**.

Step 5 Click the target instance name to view its monitoring information.

Cloud Eye can monitor performance metrics in the last 1 hour, last 3 hours, last 12 hours, last 24 hours, or last 7 days.

You can also click in the upper left corner of the page and choose **Databases** > **GaussDB**. On the **Instances** page, click **View Metric** in the **Operation** column

of the row containing the target instance to go to the Cloud Eye console. Alternatively, click the name of the target DB instance on the **Instances** page. On the displayed page, click **View Metric** in the upper right corner to go to the Cloud Eye console.

----End

13.3 Checking GaussDB Monitoring Dashboards

Scenarios

On the GaussDB console, you can see real-time performance metrics for your instances and historical performance metrics on different components of the specified instance.

Precautions

- The instance and nodes must be specified.
- You can select a maximum of nine nodes at a time.
- You can select a maximum of nine components at a time.

Procedure

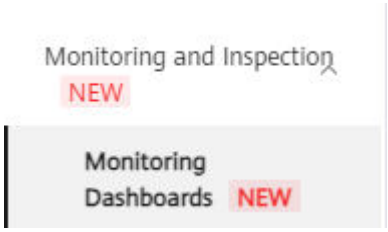
Step 1 **Log in to the management console**.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 Choose **Monitoring and Inspection** > **Monitoring Dashboards**.

Figure 13-1 Monitoring Dashboards



Step 5 On the **Monitoring Dashboards** page, select the specified instance, nodes, and components.

Figure 13-2 Selecting an instance

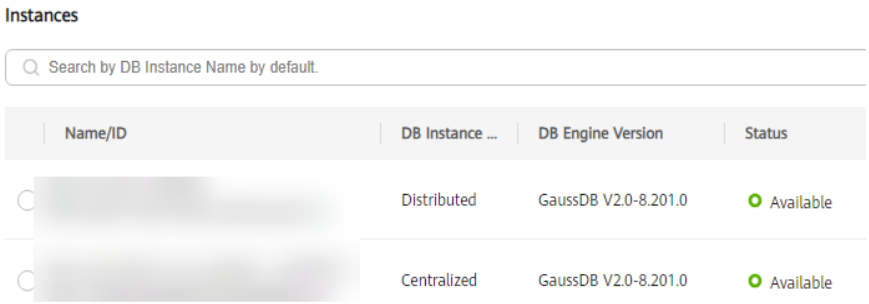


Figure 13-3 Selecting nodes

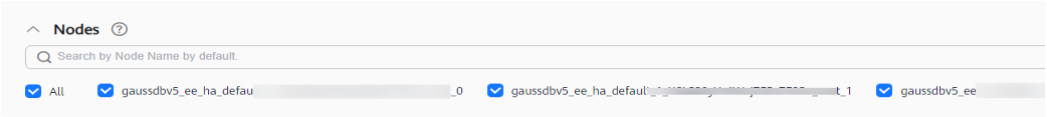
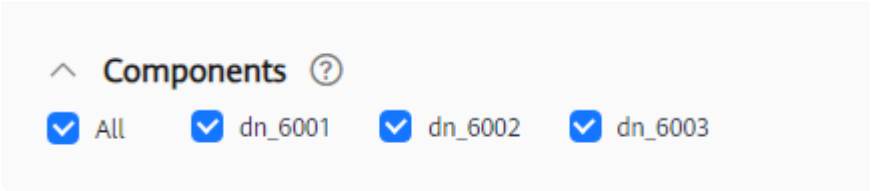


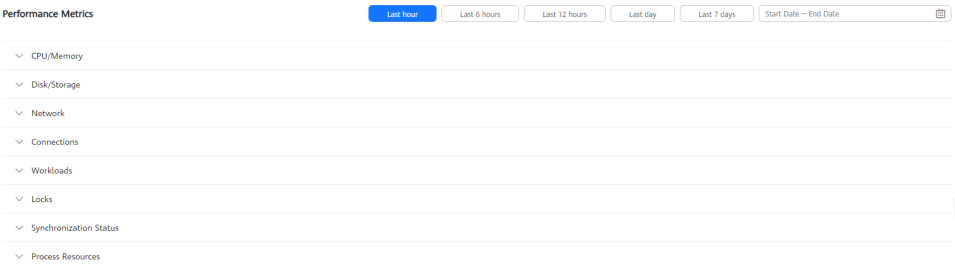
Figure 13-4 Selecting components



Step 6 Click **View Metrics**.

Step 7 Select a time segment and click to view corresponding metric data.

Figure 13-5 Viewing metric data



----End

13.4 Creating Alarm Rules for a GaussDB Instance

Scenarios

You can set alarm rules to customize the monitored objects and notification policies and to stay aware of the operating status.

The alarm rules include alarm rule names, resource type, dimensions, monitored objects, metrics, alarm thresholds, monitoring period, and whether to send notifications.

Procedure

Step 1 Log in to the management console.

Step 2 Under **Management & Governance** of the service list, click **Cloud Eye**.

Step 3 In the navigation pane on the left, choose **Cloud Service Monitoring** > **GaussDB**.

Step 4 Select the instance for which you want to create an alarm rule and click **Create Alarm Rule** in the **Operation** column.

Step 5 On the displayed page, set parameters as required.

- Select **Use existing template** (recommended) for **Method**. The default alarm template contains an alarm policy for the instance data disk usage.
- Specify **Name** and **Description**.
- Click to enable alarm notification. The validity period is 24 hours by default. If the topics you required are not displayed in the drop-down list, click **Create an SMN topic**. Then, select **Generated alarm** and **Cleared alarm** for **Trigger Condition**.

  NOTE

  Cloud Eye sends notifications only within the notification window specified in the alarm rule.

Step 6 Click **Create**. The alarm rule is created.

For details about how to create alarm rules, see **Creating an Alarm Rule**.

----End

13.5 Event Monitoring

13.5.1 Supported Events of GaussDB

Event monitoring provides event data reporting, query, and alarm reporting. You can create alarm rules for both system and custom events. When specific events occur, Cloud Eye generates alarms for you.

Events are key operations on database resources that are stored and monitored by Cloud Eye. You can view events to see operations performed by specific users on specific resources, for example, changing instance specifications.

Event monitoring provides an API for reporting custom events, which helps you collect and report abnormal events or important change events generated by services to Cloud Eye.

Event monitoring is enabled by default. You can view monitoring details about system events and custom events. For details about system events, see **Table 13-3**.

Table 13-3 Events supported by GaussDB

Source	Name	ID	Severity	Description	Handling Suggestion	Impact
GaussDB	Process status alarm	ProcessStatusAlarm	Major	Key processes exit, including CMS/CMA, ETCD, GTM, CN, and DN processes.	Wait until the process is automatically recovered or a primary/standby failover is automatically performed. Check whether services are recovered. If not, submit a service ticket by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.	If processes on primary nodes are faulty, services are interrupted and then rolled back. If processes on standby nodes are faulty, services are not affected.

Source	Name	ID	Severity	Description	Handling Suggestion	Impact
	Component status alarm	Component Status Alarm	Major	Key components do not respond, including CMA, ETCD, GTM, CN, and DN components.	Wait until the process is automatically recovered or a primary/standby failover is automatically performed. Check whether services are recovered. If not, submit a service ticket by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.	If processes on primary nodes do not respond, neither do the services. If processes on standby nodes are faulty, services are not affected.

Source	Name	ID	Severity	Description	Handling Suggestion	Impact
	Cluster status alarm	ClusterStatusAlarm	Major	The cluster is abnormal, including the following faults: The cluster is read-only. The majority of ETCD members are faulty. The cluster resources are unevenly distributed.	In the upper right corner of the management console, submit a service ticket by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.	If the cluster status is read-only, only read requests are processed. If the majority of ETCD members are faulty, the cluster is unavailable. If resources are unevenly distributed, the instance performance and reliability deteriorate.
	Hardware resource alarm	HardwareResourceAlarm	Major	A major hardware fault occurs in the instance, such as disk damage or GTM network fault.	In the upper right corner of the management console, submit a service ticket by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.	Some or all services are affected.

Source	Name	ID	Severity	Description	Handling Suggestion	Impact
	Status transition alarm	StateTransitionAlarm	Major	The following events occur in the instance: DN build attempt, DN build failure, forcible DN promotion, primary/standby DN switchover/failover, or primary/standby GTM switchover/failover.	Wait until the fault is automatically rectified and check whether services are recovered. If not, submit a service ticket by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.	Some services are interrupted.
	Other abnormal alarm	OtherAbnormalAlarm	Major	Disk usage threshold alarm	Monitor service changes and scale up storage space as needed.	If the used storage space exceeds the threshold, storage space cannot be scaled up.

Source	Name	ID	Severity	Description	Handling Suggestion	Impact
	Instance running status abnormal	TaurusInstanceRunningStatusAbnormal	Major	This event is a key alarm event and is reported when an instance is faulty due to a disaster or a server failure.	In the upper right corner of the management console, submit a service ticket by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.	The database service may be unavailable.
	Instance running status recovery	TaurusInstanceRunningStatusRecovered	Major	If a disaster occurs, GaussDB provides an HA tool to automatically or manually rectify the fault. After the fault is rectified, this event is reported.	No further action is required.	None

Source	Name	ID	Severity	Description	Handling Suggestion	Impact
	Faulty DB node	TaurusNodeRunningStatusAbnormal	Major	This event is a key alarm event and is reported when a database node is faulty due to a disaster or a server failure.	Check whether the database service is available and submit a service ticket by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.	The database service may be unavailable.
	DB node recovered	TaurusNodeRunningStatusRecovered	Major	If a disaster occurs, GaussDB provides an HA tool to automatically or manually rectify the fault. After the fault is rectified, this event is reported.	No further action is required.	None

Source	Name	ID	Severity	Description	Handling Suggestion	Impact
	Instance creation failure	GaussDBV5CreateInstanceFailed	Major	Instances fail to be created because the quota is insufficient or underlying resources are exhausted.	Release the instances that are no longer used and try to provision new instances again, or submit a service ticket by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console to adjust the quota.	Instances fail to be created.
	Node adding failure	GaussDBV5ExpandClusterFailed	Major	The underlying resources are insufficient.	Submit a service ticket by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console to coordinate resources, delete the nodes that failed to be added, and add nodes again.	None

Source	Name	ID	Severity	Description	Handling Suggestion	Impact
	Storage scale-up failure	GaussDBV5EnlargeVolumeFailed	Major	The underlying resources are insufficient.	Submit a service ticket by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console to coordinate resources and scale up storage again.	Services may be interrupted.
	Reboot failure	GaussDBV5RestartInstanceFailed	Major	The network is abnormal.	Retry the reboot operation or submit a service ticket by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.	The database service may be unavailable.

Source	Name	ID	Severity	Description	Handling Suggestion	Impact
	Full backup failure	GaussDBV5FullBackupFailed	Major	The backup files fail to be exported or uploaded.	Submit a service ticket by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.	Data cannot be backed up.
	Differential backup failure	GaussDBV5DifferentialBackupFailed	Major	The backup files fail to be exported or uploaded.	Submit a service ticket by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.	Data cannot be backed up.
	Backup deletion failure	GaussDBV5DeleteBackupFailed	Major	Backup files fail to be cleared.	Submit a service ticket by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.	There may be residual OBS files.

Source	Name	ID	Severity	Description	Handling Suggestion	Impact
	EIP binding failure	GaussDBV5BindEIPFailed	Major	The EIP has been used or EIP resources are insufficient.	Submit a service ticket by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.	The instance cannot be accessed from the public network.
	EIP unbinding failure	GaussDBV5UnbindEIPFailed	Major	The network or the EIP service is faulty.	Retry the unbinding operation or submit a service ticket by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.	There may be residual IP resources.
	Parameter template application failure	GaussDBV5ApplyParamFailed	Major	Changing a parameter group times out.	Change the parameter group again.	None
	Parameter modification failure	GaussDBV5UpdateInstanceParamGroupFailed	Major	Changing a parameter group times out.	Change the parameter group again.	None

Source	Name	ID	Severity	Description	Handling Suggestion	Impact
	Backup and restoration failure	GaussDBV5RestoreFromBcakupFailed	Major	The underlying resources are insufficient or backup files fail to be downloaded.	In the upper right corner of the management console, submit a service ticket by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.	The database service may be unavailable during the restoration failure.

13.5.2 Checking GaussDB Event Monitoring Data

Scenarios

Event monitoring provides event data reporting, query, and alarm reporting. When there are specified events, you will receive alarm notifications from Cloud Eye.

Event monitoring is enabled by default. You can view monitoring details about system events and custom events.

This section describes how to view the event monitoring data.

Procedure

Step 1 **Log in to the management console**.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page. Under **Management & Governance**, click **Cloud Eye**.

You can also click in the upper left corner of the page and choose **Databases** > **GaussDB**. On the **Instances** page, click **View Metric** in the **Operation** column of the row containing the target instance to go to the Cloud Eye console.

Alternatively, click the name of the target instance on the **Instances** page. On the displayed page, click **View Metric** in the upper right corner to go to the Cloud Eye console. Then, click  to return to the main page of Cloud Eye.

Step 4 In the navigation pane on the left, choose **Event Monitoring**.

Step 5 On the displayed page, check all system events of the last 24 hours that are displayed by default.

You can also click **1h**, **3h**, **12h**, **1d**, **7d**, or **30d** to view the events generated in different periods.

Step 6 Expand an event, and click **View Event** in the **Operation** column to view details about a specific event.

----End

13.5.3 Creating an Alarm Rule to Monitor a GaussDB Event

Scenarios

You can create alarm rules for event monitoring.

Procedure

Step 1 Log in to the management console.

Step 2 Click  in the upper left corner of the page, and choose **Management & Governance** > **Cloud Eye**.

Step 3 In the navigation pane, choose **Event Monitoring**. On the **Event Monitoring** page, click **Create Alarm Rule**.

Step 4 On the **Create Alarm Rule** page, configure required parameters.

Table 13-4 Parameters for creating an alarm rule

Parameter	Description
Name	Name of the alarm rule. The system generates a random name, and you can change it if needed.
Description	Supplementary information about the alarm rule. This parameter is optional.
Alarm Type	Alarm type corresponding to the alarm rule.
Event Type	Event type of the metric corresponding to the alarm rule.
Event Source	Service the event is generated for. Select **GaussDB**.
Monitoring Scope	Monitoring scope for event monitoring.

Parameter	Description
Method	Method you use to create the alarm rule. You can select **Configure manually**.
Event Name	Instantaneous operations users performed on system resources, such as login and logout.
Triggering mode	Select **Immediate trigger** or **Accumulative trigger** based on the operation severity.
Alarm Policy	Policy that triggers an alarm. For example, an alarm is triggered if the event occurred for three consecutive periods of 5 minutes. **NOTE** This parameter is mandatory when the triggering mode is set to **Accumulative trigger**.
Alarm Severity	Alarm severity, which can be **Critical**, **Major**, **Minor**, or **Informational**.
Operation	You can click **Delete** to delete an alarm policy.

Toggle on next to the **Alarm Notification** field to enable alarm notification. The notification window is 24 hours by default. If the topics you need are not displayed in the **Notification Object** drop-down list, click **Create an SMN topic** first. Then, select **Generated alarm** and **Cleared alarm** for **Trigger Condition**.

NOTE

Cloud Eye sends notifications only within the notification window specified in the alarm rule.

Table 13-5 Parameters for setting alarm notifications

Parameter	Description
Alarm Notification	Specifies whether to notify users when alarms are triggered. Notifications can be sent by email or text message, or through HTTP/HTTPS request to servers.
Notification Recipient	There are two options: **Notification group** and **Topic subscription**.
Notification Group	Notification group the alarm notification is to be sent to.

Parameter	Description
Notification Object	Object that receives alarm notifications. You can select the account contact or a topic. • Account contact is the mobile phone number and email address provided for registration. • A topic is used to publish messages and subscribe to notifications. If the required topic is unavailable, create one first and add subscriptions to it. For details, see **Creating a Topic** and **Adding Subscriptions**.
Notification Window	Cloud Eye sends notifications only within the validity period specified in the alarm rule. If **Notification Window** is set to **08:00-20:00**, Cloud Eye sends notifications only within 08:00-20:00.
Trigger Condition	Condition for triggering an alarm notification. You can select **Generated alarm** (when an alarm is generated), **Cleared alarm** (when an alarm is cleared), or both.

Step 5 Click **Create**. The alarm rule is created.

For details about alarm rule parameters, see **Creating an Alarm Rule** in *Cloud Eye User Guide*.

----End

14 Logs and Auditing

14.1 Downloading Error Logs and Slow Query Logs of a GaussDB Instance

GaussDB allows you to download slow query logs and error logs. Slow query logs help you locate slow SQL statement execution problems. Error logs help you locate instance problems.

Precautions

- CNs and DNs of the instance are normal.
- The IaaS network is normal.

Slow Query Logs

Step 1 Log in to the management console.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

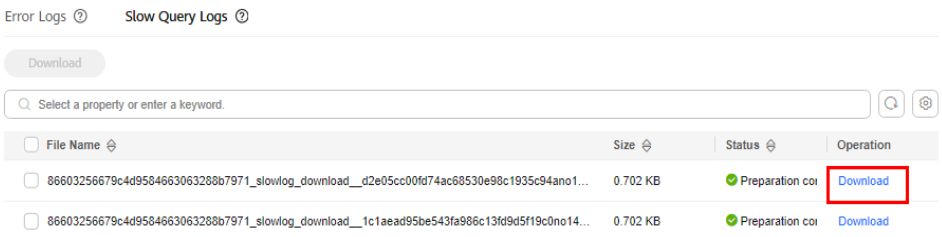
Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the navigation pane, choose **Log Analysis**.

Step 6 The system checks whether there has been a slow query log task in the last 5 minutes and, if there is not, generates one. Click the **Slow Query Logs** tab. Click **Download** in the **Operation** column of the record whose status is **Preparation completed**.

After the log is downloaded, you can analyze the log on the local PC.

Figure 14-1 Downloading a slow query log



NOTE

Logs generated in the last 12 hours are collected for the analysis of slow query logs.

Table 14-1 describes the fields contained in slow query logs.

Table 14-1 Fields in slow query logs

Field	Type	Description
message_version	String	Log format version. The value is fixed at **V1**.
db_name	name	Database name.
schema_name	name	Schema name.
origin_node	integer	Node name.
user_name	name	Username.
application_name	text	Name of the application that sends a request.
client_addr	text	IP address of the client that sends a request.
client_port	integer	Port number of the client that sends a request.
unique_query_id	bigint	ID of the normalized SQL statement.
debug_query_id	bigint	ID of the unique SQL statement. Some statements are not unique. For example, the value of **debug_query_id** in the Parse packet, DCL statements, and TCL statements is **0**.
query	text	Normalized SQL (available only on CNs). When **track_stmt_parameter** is enabled, complete SQL statements are displayed.
start_time	timestamp with time zone	Time when a statement starts.

Field	Type	Description
finish_time	timestamp with time zone	Time when a statement ends.
slow_sql_threshold	bigint	Standard for slow SQL statement execution.
transaction_id	bigint	Transaction ID.
thread_id	bigint	ID of an execution thread.
session_id	bigint	Session ID of a user.
n_soft_parse	bigint	Number of soft parses. The value of **n_soft_parse** plus the value of **n_hard_parse** may be greater than the value of **n_calls** because the number of subqueries is not counted in the value of **n_calls**.
n_hard_parse	bigint	Number of hard parses. The value of **n_soft_parse** plus the value of **n_hard_parse** may be greater than the value of **n_calls** because the number of subqueries is not counted in the value of **n_calls**.
query_plan	text	Statement execution plan.
n_returned_rows	bigint	Number of rows in the result set returned by the SELECT statement.
n_tuples_fetched	bigint	Number of rows randomly scanned.
n_tuples_returned	bigint	Number of rows sequentially scanned.
n_tuples_inserted	bigint	Number of rows inserted.
n_tuples_updated	bigint	Number of rows updated.
n_tuples_deleted	bigint	Number of rows deleted.
n_blocks_fetched	bigint	Number of buffer block access times.
n_blocks_hit	bigint	Number of buffer block hits.
db_time	bigint	Valid DB time, which is accumulated if multiple threads are involved (unit: microsecond).
cpu_time	bigint	CPU time (unit: microsecond).

Field	Type	Description
execution_time	bigint	Execution time in the executor (unit: microsecond).
parse_time	bigint	SQL parsing time (unit: microsecond).
plan_time	bigint	SQL plan generation time (unit: microsecond).
rewrite_time	bigint	SQL rewriting time (unit: microsecond).
pl_execution_time	bigint	Execution time of PL/pgSQL (unit: microsecond).
pl_compilation_time	bigint	Compilation time of PL/pgSQL (unit: microsecond).
data_io_time	bigint	I/O time (unit: microsecond).
net_send_info	text	Network status of messages sent through a physical connection, including the time (in microseconds), number of calls, and throughput (in bytes). In a distributed database, CNs communicate with each other, CNs communicate with the client, and CNs communicate with DNs through physical connections. This column can be used to analyze the network overhead of SQL statements in a distributed system. Example: **{"time":***xxx***, "n_calls":***xxx***, "size":***xxx***}**.
net_recv_info	text	Network status of messages received through a physical connection, including the time (in microseconds), number of calls, and throughput (in bytes). In a distributed database, CNs communicate with each other, CNs communicate with the client, and CNs communicate with DNs through physical connections. This column can be used to analyze the network overhead of SQL statements in a distributed system. Example: **{"time":***xxx***, "n_calls":***xxx***, "size":***xxx***}**.

Field	Type	Description
net_stream_send_info	text	Network status of messages sent through a logical connection, including the time (in microseconds), number of calls, and throughput (in bytes). In a distributed database, DNs of different shards communicate with each other through logical connections. This column can be used to analyze the network overhead of SQL statements in a distributed system. Example: **{"time":***xxx***, "n_calls":***xxx***, "size":***xxx***}**.
net_stream_recv_info	text	Network status of messages received through a logical connection, including the time (in microseconds), number of calls, and throughput (in bytes). In a distributed database, DNs of different shards communicate with each other through logical connections. This column can be used to analyze the network overhead of SQL statements in a distributed system. Example: **{"time":***xxx***, "n_calls":***xxx***, "size":***xxx***}**.
lock_count	bigint	Number of locks.
lock_time	bigint	Time required for locking.
lock_wait_count	bigint	Number of lock waits.
lock_wait_time	bigint	Time required for lock waiting.
lock_max_count	bigint	Maximum number of locks.
lwlock_count	bigint	Number of lightweight locks (reserved).
lwlock_wait_count	bigint	Number of lightweight lock waits.
lwlock_time	bigint	Time required for lightweight locking (reserved).

Field	Type	Description
details	bytea	List of wait events and statement lock events. When the value of the record level is greater than or equal to **L0**, the list of waiting events starts to be recorded. It displays statistics about wait events on the current node. For details about key events, see **Table: Waiting State List**, **Table: List of Wait Events Corresponding to Lightweight Locks**, **Table: List of I/O Wait Events**, and **Table: List of Wait Events Corresponding to Transaction Locks**. You can also view the list of all events in the system in the **wait_event_info** view. For details about the impact of each transaction lock on services, see **LOCK**. When the value of the record level is **L2**, the list of statement lock events is recorded. The list records events in chronological order. The number of records is affected by the value of the **track_stmt_details_size** parameter. This field is in binary format and needs to be read by using the parsing function **pg_catalog.statement_detail_decode**. For details, see **Table: statement_detail_decode Parameter Description**. Events include: • Start locking. • Complete locking. • Start lock waiting. • Complete lock waiting. • Start unlocking. • Complete unlocking. • Start lightweight lock waiting. • Complete lightweight lock waiting.
is_slow_sql	boolean	Specifies whether the SQL statement is a slow SQL statement. • **t** (true): yes. • **f** (false): no.
lwlock_wait_time	bigint	Time required for lightweight lock waiting.

----End

Error Logs

Step 1 Log in to the management console.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the navigation pane, choose **Log Analysis**.

Step 6 On the displayed page, click the **Error Logs** page, enable **Error Log Collection**, and click **Download** in the **Operation** column of the log whose status is **Preparation completed** to download the error log file. After the log is downloaded, you can analyze the log on the local PC.

Error logs are stored in the **gs_log** directory and are named in the format **gaussdb-***creation_time***.log**. The default format of each row of logs in the log file is as follows: *Date*+*Time*+*Node name*+*Username*+*Database name*+*IP address*+*Session ID*+*Transaction ID*+*Application name*+*Log level*+*Log content*

Table 14-2 Parameters of error logs

Field	Description
Date	Date when a log is generated. The format is yyyy-mm-dd.
Time	Time when a log is generated. The format is hh:mm:ss:ms.
Node name	The node to which an error is reported.
Username	Username of the database user who triggers log generation.
Database name	Name of the database that triggers log generation.
IP address	IP address of the client that triggers log generation.
Thread ID	Thread ID.
Session ID	ID of the session that triggers log generation.
Transaction ID	Transaction ID (**0** indicates that no transaction ID is assigned).
Thread name	Thread name.

Field	Description
Query ID	ID of a query initiated by a user, which is recorded in the background.
Module name	Module name.
Log level	Log level, such as **FATAL**, **ERROR**, or **LOG**. Different log levels indicate different severities.
Log content	Log content.

----End

14.2 Downloading Switchover/Failover Logs of a GaussDB Instance

You can download switchover/failover logs of a GaussDB instance. If switchover/failover log collection is enabled for an instance whose **Failover Priority** is **Availability**, GaussDB can collect Xlogs that cannot be replayed on the standby node in time when a switchover or failover occurs and convert the Xlogs into a SQL file. You can download the SQL file and run SQL statements to replay the data in the SQL file as required.

Precautions

- CNs and DNs of the instance are normal.
- The IaaS network is normal.
- Switchover/Failover logs are available only for distributed instances whose **Failover Priority** is **Availability**.

Switchover/Failover Logs

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the navigation pane on the left, choose **Log Analysis**.

Step 6 On the displayed page, click the **Switchover/Failover Logs** page, enable **Switchover/Failover Log Collection**, and click **Download** in the **Operation** column of the record whose status is **Preparation completed** to download the switchover/failover log file.

----End

14.3 Querying Audit Logs of GaussDB Instances on CTS

With CTS, you can record operations associated with GaussDB for future query, audit, and backtracking.

GaussDB Operations That Can Be Recorded by CTS

Table 14-3 Operations supported by CTS

Operation	Resource Type	Trace Name
Creating a DB instance or restoring data to a new DB instance	instance	createInstance
Deleting a DB instance	instance	deleteInstance
Changing instance specifications	instance	resizeFlavor
Upgrading the instance version	instance	upgradeVersion
Resetting a password	instance	resetPassword
Rebooting a DB instance	instance	instanceRestart
Binding an EIP	instance	setOrResetPublicIP
Unbinding an EIP	instance	setOrResetPublicIP
Modifying resource tags	instance	modifyTag
Deleting resource tags	instance	deleteTag
Adding resource tags	instance	createTag
Renaming a DB instance	instance	renameInstance
Adding nodes	instance	instanceAction
Deleting task records	workflowTask	deleteTaskRecord
Reducing the number of replicas	instance	reduceReplica
Deleting coordinator nodes	instance	reduceCoordinator-Node

Operation	Resource Type	Trace Name
Modifying the recycling policy	backup	setRecyclePolicy
Creating a manual backup	backup	createManualSnapshot
Deleting a manual backup	backup	deleteManualSnapshot
Modifying the backup policy	backup	setBackupPolicy
Restoring a DB instance	backup	restoreInstance
Restoring data of an instance using a backup	instance	restoreInstance
Changing the retention period of automated backups	instance	setBackupPolicy
Creating a parameter group	parameterGroup	createParameterGroup
Applying a parameter group	parameterGroup	applyParameterGroup
Replicating a parameter group	parameterGroup	copyParameterGroup
Deleting a parameter group	parameterGroup	deleteParameterGroup
Resetting a parameter group	parameterGroup	resetParameterGroup
Updating a parameter group	parameterGroup	updateParameterGroup
Changing the port	instance	modifyPort
Creating slow query log download tasks	instance	createSlowLogDownload
Enabling or disabling switchover/failover logs	instance	switchErrorLog
Scaling up storage for shards	instance	resizeVolume
Modifying storage autoscaling policies	instance	autoEnlargeVolume

Operation	Resource Type	Trace Name
Deleting shards	instance	reduceShard
Changing standby data nodes to log nodes	instance	switchReplica
Performing a primary/standby switchover	instance	switchShard
Changing the disk type	instance	changeVolumeType
Starting an instance or node	instance	startInstance
Stopping an instance or node	instance	stopInstance
Changing a single-replica instance to an instance with primary and standby nodes	instance	changeDeployment-Solution
Enabling advanced features	instance	updateFeatures
Updating the policy of agency permissions	feature	updateAgency

Querying Audit Logs

You can query GaussDB traces (audit logs) on the CTS console. For details, see **Querying Real-Time Traces**.

14.4 Interconnecting with LTS and Querying Database Audit Logs

Scenarios

Log Tank Service (LTS) collects, analyzes, and stores logs. If you enable **Upload Audit Logs to LTS**, GaussDB audit logs will be uploaded to LTS and you can search for logs, monitor logs, download logs, and view real-time logs.

- **Enabling Upload Audit Logs to LTS**: Enable log upload configurations for a single instance.
- **Disabling Upload Audit Logs to LTS**: Disable log upload configurations for a single instance.

Precautions

- To apply for the permission needed for enabling **Upload Audit Logs to LTS**, submit an application by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console. The default permission only allows you to upload audit logs for centralized instances. To also enable audit log upload for distributed instances, clarify your requirement in the application when creating the service ticket.
- The DB engine version must be V2.0-2.1.0 or later.
- Audit logs record all requests sent to your DB instance and are stored in LTS.
- Toggling on or off this function will not be applied immediately. There is a delay of about 10 minutes.
- For details about how to enable or disable the audit log function, configure **audit_enabled**.
- For details about the parameters for controlling audit logs, see **Audit Items**.
- You will be billed for this function. For details, see **LTS Pricing Details**.
- After this function is enabled, audit policies you configured are reported to LTS by default.

Enabling Upload Audit Logs to LTS

Step 1 **Log in to the management console**.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 In the navigation pane on the left, choose **Instances**.

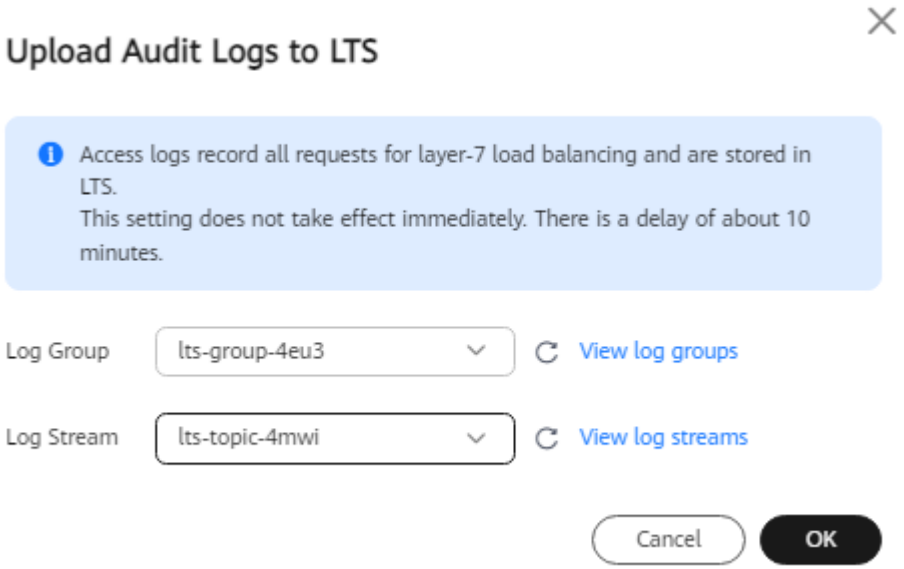
Step 5 Click the instance name to go to the **Basic Information** page.

Step 6 In the navigation pane on the left, click **Audit Logs**.

Step 7 Click  next to **Upload Audit Logs to LTS**.

Step 8 In the displayed dialog box, configure **Log Group** and **Log Stream**.

Figure 14-2 Enabling Upload Audit Logs to LTS



NOTE

If you enable this function for the first time, click **View Log Groups** to log in to the LTS console and configure log groups and log streams. For details, see **Managing Log Groups** and **Managing Log Streams**.

Step 9 Click **OK**.

After this function is enabled, audit logs will not be uploaded immediately to LTS. There is a delay of about 10 minutes. For details, see **Viewing Real-Time Logs**.

----End

Disabling Upload Audit Logs to LTS

Step 1 **Log in to the management console**.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 In the navigation pane on the left, choose **Instances**.

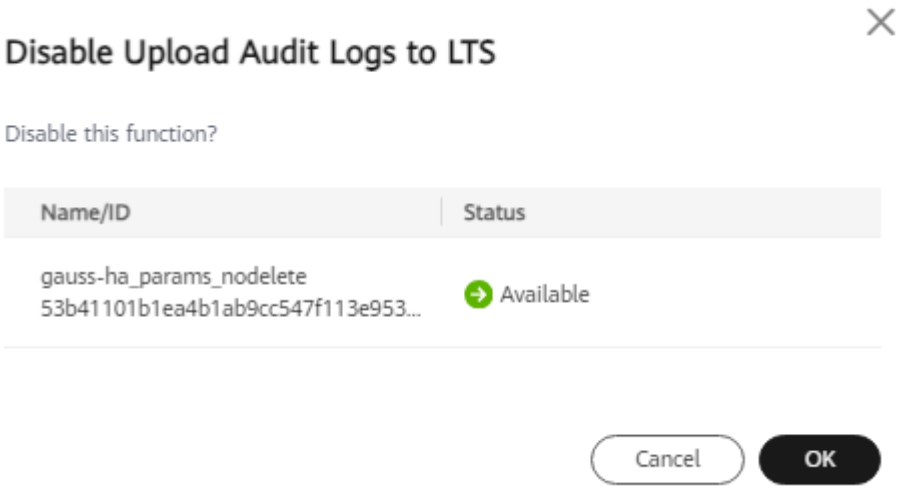
Step 5 Click the instance name to go to the **Basic Information** page.

Step 6 In the navigation pane on the left, click **Audit Logs**.

Step 7 Click next to **Upload Audit Logs to LTS**.

Step 8 In the displayed dialog box, confirm the information.

Figure 14-3 Disabling Upload Audit Logs to LTS



Step 9 In the displayed dialog box, click **OK**.

----End

15 Quota Adjustment

15.1 Adjusting Cloud Service Resource Quotas of GaussDB

What Is a Quota?

A quota is a limit on the quantity or capacity of a certain type of service resources available to you. Examples of GaussDB quotas include the maximum number of GaussDB instances that you can create. Quotas are put in place to prevent excessive resource usage.

If the existing resource quotas cannot meet your service requirements, you can request higher quotas.

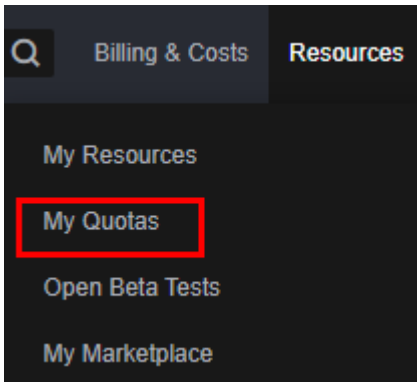
How Do I View My Quotas?

Step 1 **Log in to the management console**.

Step 2 Click in the upper left corner and select the desired region and project.

Step 3 In the upper right corner of the page, choose **Resources** > **My Quotas**.

Figure 15-1 My Quotas



Step 4 On the **Quotas** page, view the used and total quotas of each type of resources.

----End

How Do I Apply for a Higher Quota?

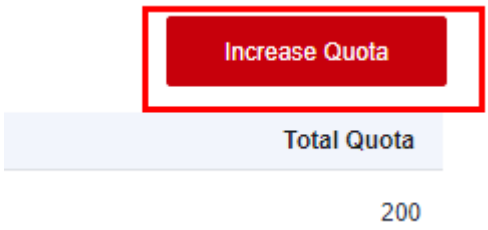
Step 1 **Log in to the management console**.

Step 2 Click in the upper left corner and select the desired region and project.

Step 3 In the upper right corner of the page, choose **Resources** > **My Quotas**.

Step 4 In the upper right corner of the page, click **Increase Quota**.

Figure 15-2 Increase Quota



Step 5 On the **Create Service Ticket** page, configure parameters as required.

In the **Problem Description** area, fill in the content and reason for quota adjustment.

Step 6 Read the agreements and confirm that you agree to them, and then click **Submit**.

----End

15.2 Adjusting GaussDB Resource Quotas of an Enterprise Project

The GaussDB management console on Huawei Cloud provides quota management for enterprise projects of tenants.

Quota management is available only for enterprise accounts configured in the whitelist. To apply for the permissions required, submit an application by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the management console.

Managing Quotas

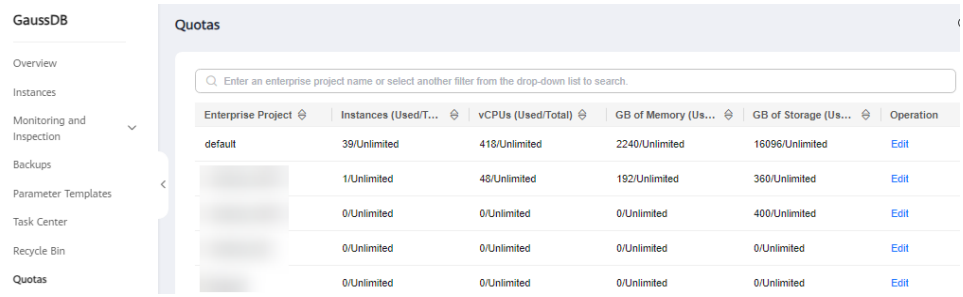
Step 1 **Log in to the management console**.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 In the navigation pane on the left, click **Quotas**.

Figure 15-3 Quotas



On this page, you can view the usage of instances, vCPUs, memory, and storage under each project.

Step 5 Locate the enterprise project to be managed, and click **Edit** in the **Operation** column.

Figure 15-4 Modifying quotas

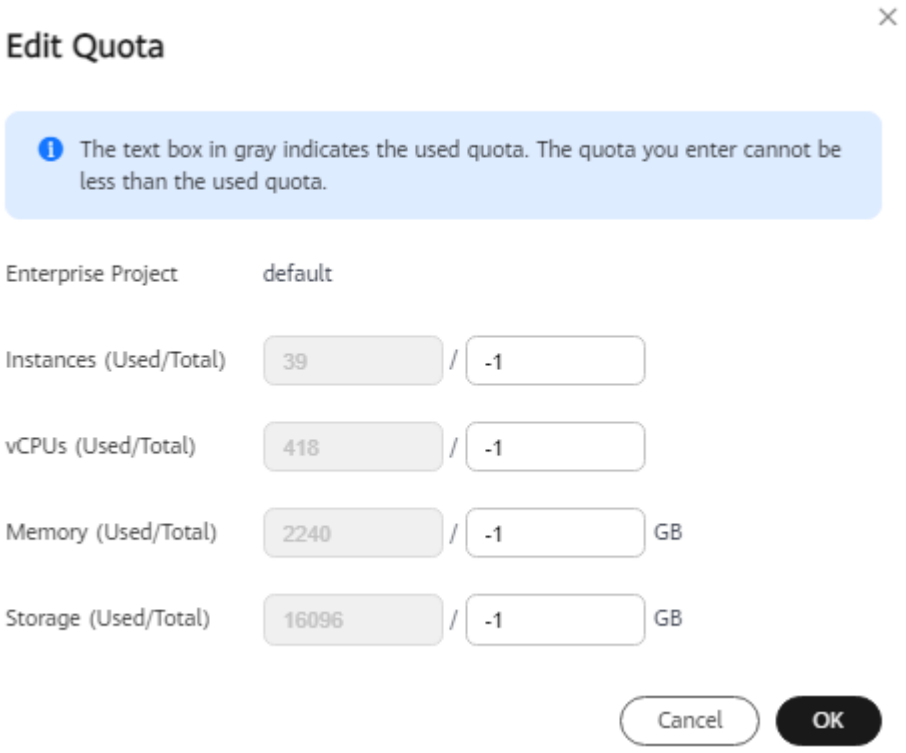


Table 15-1 Parameter description

Category	Description
Instances	• The first number indicates the number of existing instances in the enterprise project. • The second number indicates the maximum number of instances that can be created in the enterprise project. The minimum value must be greater than or equal to the number of existing instances. The maximum value is **100000**. If this parameter is set to **-1**, the number is not limited.
vCPUs	• The first number indicates the number of vCPUs used by existing instances in the enterprise project. • The second number indicates the maximum number of vCPUs that can be used by instances in the enterprise project. The minimum value must be greater than or equal to the number of vCPUs used by existing instances. The maximum value is **2147483646**. If this parameter is set to **-1**, the number is not limited.
Memory (GB)	• The first number indicates the size of memory used by existing instances in the enterprise project. • The second number indicates the maximum size of memory that can be used by instances in the enterprise project. The minimum value must be greater than or equal to the size of memory used by existing instances. The maximum value is **2147483646**. If this parameter is set to **-1**, the number is not limited.
Storage (GB)	• The first number indicates the storage space used by existing instances in the enterprise project. • The second number indicates the maximum storage space that can be used by instances in the enterprise project. The minimum value must be greater than or equal to the storage space used by existing instances. The maximum value is **2147483646**. If this parameter is set to **-1**, the number is not limited.

NOTE

When you access the **Quotas** page for the first time, **Settings** is displayed.

Step 6 In the displayed dialog box, enter a new quota. Click **OK**.

----End

16 SQL Diagnosis

16.1 Configuring SQL Throttling Rules

GaussDB supports SQL throttling. When there are abnormal SQL statements (for example, poor indexing) or the number of concurrent SQL requests increases, you can use this function to control the number of access requests or concurrency of specific SQL statements to ensure service availability.

Prerequisites

You have logged in using an account assigned the following actions: **gaussdb:instance:listFlowlimit**, **gaussdb:instance:flowlimitAddOrUpdate**, **gaussdb:instance:createFlowLimit**, **gaussdb:instance:updateFlowLimit**, **gaussdb:instance:deleteFlowLimit**, **gaussdb:instance:getFlowLimitInfo**, **gaussdb:instance:getFlowLimitTemplate**, **gaussdb:instance:listFlowLimitInfo**, **gaussdb:instance:syncFlowLimit**, and **gaussdb:instance:flowlimitDelete**

Constraints

- The DB engine version of GaussDB instances must be V2.0-8.0.1 or later, and the kernel version must be 505.0.0 or later.
- Auto throttling is unavailable to distributed instances.
- SQL ID-based throttling is unavailable to distributed instances.
- SQL throttling cannot be performed for system-level users (such as **root**).
- Currently, SQL throttling tasks cannot be deleted in batches.
- Each instance supports a maximum of 1,000 throttling tasks.
- For a keyword-based throttling task, keywords are separated by commas (,), and the sequence of keywords is not distinguished.
- If you used CLI commands to add, delete, or modify a SQL throttling task as a system user (for example, user **root**), the actual throttling task status will be different from that displayed on the web UI.
- When a throttling rule is being applied, CPU usage increases. To ensure system stability, do not create too many rules for a given instance.
- SQL statements that have been executed before a throttling rule is added are not counted.

- In DR scenarios, the throttling rules applied to the primary instance are automatically synchronized to the DR instance. After the DR instance is promoted to primary, the throttling rules cannot be added, deleted, modified, or queried. Throttling tasks can be managed again only after the original primary instance becomes primary.
- If data is lost when you restore a backup to a new DB instance or switch the original primary instance to primary, handle the data loss issue by following the instructions in *Troubleshooting*.
- You can still create, modify, and query a throttling rule if the instance status is abnormal or when instance specifications are being changed, but the operation may fail. If a failure or exception occurs, you can delete the throttling rule. Before deleting a throttling rule, ensure that the database is running properly.
- Throttling is not allowed for SQL statements involved in setting up database connections, for example, **select setting from pg_settings where name= \'delimiter_name\'**, because new connections cannot be set up if such statements are blocked.
- When an instance node is being created or rebooted (for example, for adding replicas or shards), the node may fail to be connected. As a result, the SQL throttling task details cannot be queried.
- During a gray upgrade or when there are two primary instances after the original DR instance is promoted to primary, an error message, indicating another operation is in progress, will be displayed if you attempt to create, modify, or delete the throttling rule applied to the original DR instance, and the throttling task sticks in the creating, modifying, or deleting state. In this case, click **Data Synchronization** to rectify the fault.
- You may receive a kernel error message like **ERROR: The workload rule takes effect and this request will be cancelled. rule_id: x, rule_name: "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"**. The rule name in such a message indicates the throttling task ID on this page and the task ID in audit logs.
- If a SQL statement matches multiple throttling task rules at the same time, the rules are applied based on the following priorities:
  a. The smaller value of **Max. Concurrency** or **Max. Active Sessions**, the higher the priority.
  b. If the values of **Max. Concurrency** or **Max. Active Sessions** in different tasks are the same, the priority is as follows: auto throttling > SQL ID-based throttling > keyword-based throttling
  c. If the values of **Max. Concurrency** or **Max. Active Sessions** and the throttling type are the same in different tasks, the task created later is preferentially used to block requests.
- Do not create tables with the same name. If a table with the same name exists in different databases or schemas, the throttling rule may not take effect when you create a SQL ID-based throttling task related to the table.

Creating a Throttling Task

Step 1 **Log in to the management console**.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click ☰ in the upper left corner of the page and choose **Database** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the navigation pane, choose **Diagnostics and Optimization** > **SQL Diagnosis**. Then, click the **SQL Throttling** tab.

Step 6 On the **SQL Throttling Tasks** or **Auto Throttling Tasks** tab, click **Create Task**. In the right pane, set the throttling parameters described in **Table 16-1**.

- Creating a SQL throttling task

Figure 16-1 Creating a SQL throttling task

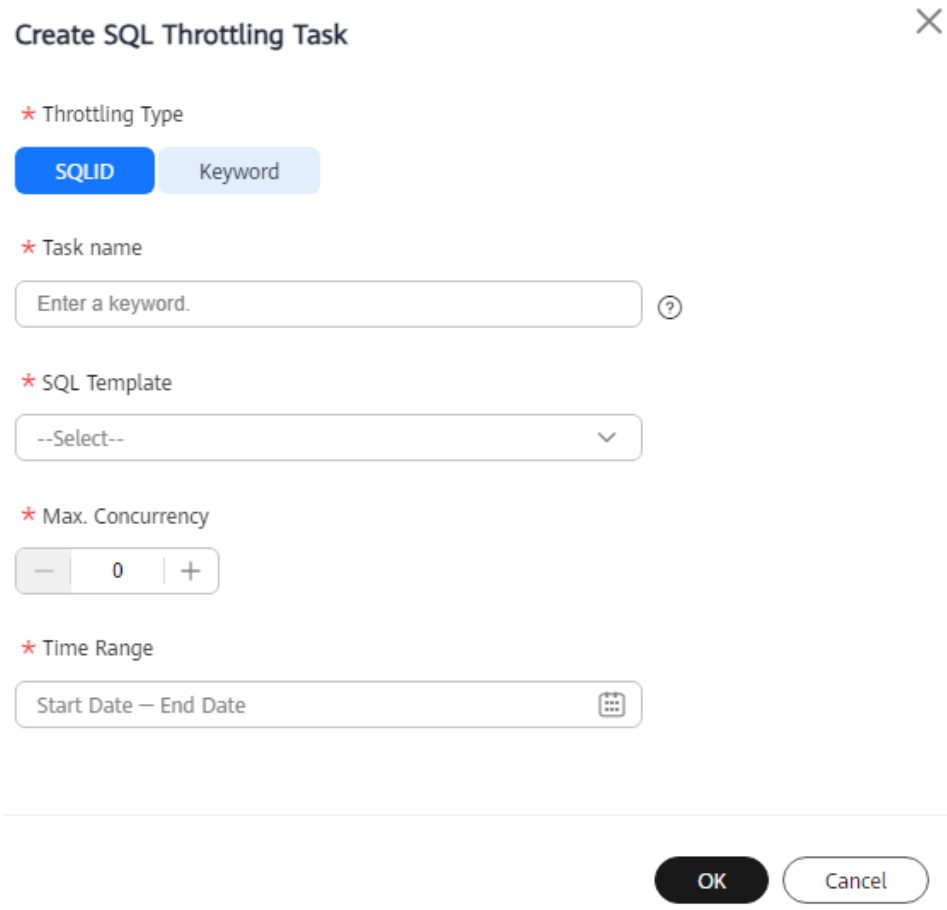


Table 16-1 Parameters for creating a SQL throttling task

Throttling Type	Parameter	Description
SQL ID	Task Name	Name of the SQL throttling task.

Throttling Type	Parameter	Description
	SQL Template	The system records the SQL statement execution records of each node and converts the records into a template. You can select the template for performing SQL throttling on each node. **NOTE** When an instance is rebooted, the SQL template of the instance node will be cleared.
	Max. Concurrency	Maximum number of active connections for executing the throttled SQL statement on the specified node at the same time.
	Time Range	Time period when the throttling takes effect.
Keyword **NOTE** – 2 to 100 keywords can be specified for a single task. Each keyword can contain 2 to 64 characters. – Keywords are case-insensitive. – The fewer the keywords, the larger the range of SQL statements to be throttled.	Task Name	Name of the SQL throttling task.
	SQL Statement Type	SQL statement type, which can be SELECT, INSERT, UPDATE, DELETE, or MERGE.
	Keyword	Keywords used for throttling. The throttling task throttles the statements that match the specified keywords and SQL statement type.
	Target Databases	Databases where the throttling takes effect.
	Max. Concurrency	Maximum number of active connections for executing the throttled SQL statement on the specified node at the same time.
	Time Range	Time period when the throttling takes effect.

- Creating an auto throttling task

Figure 16-2 Create Auto Throttling Task

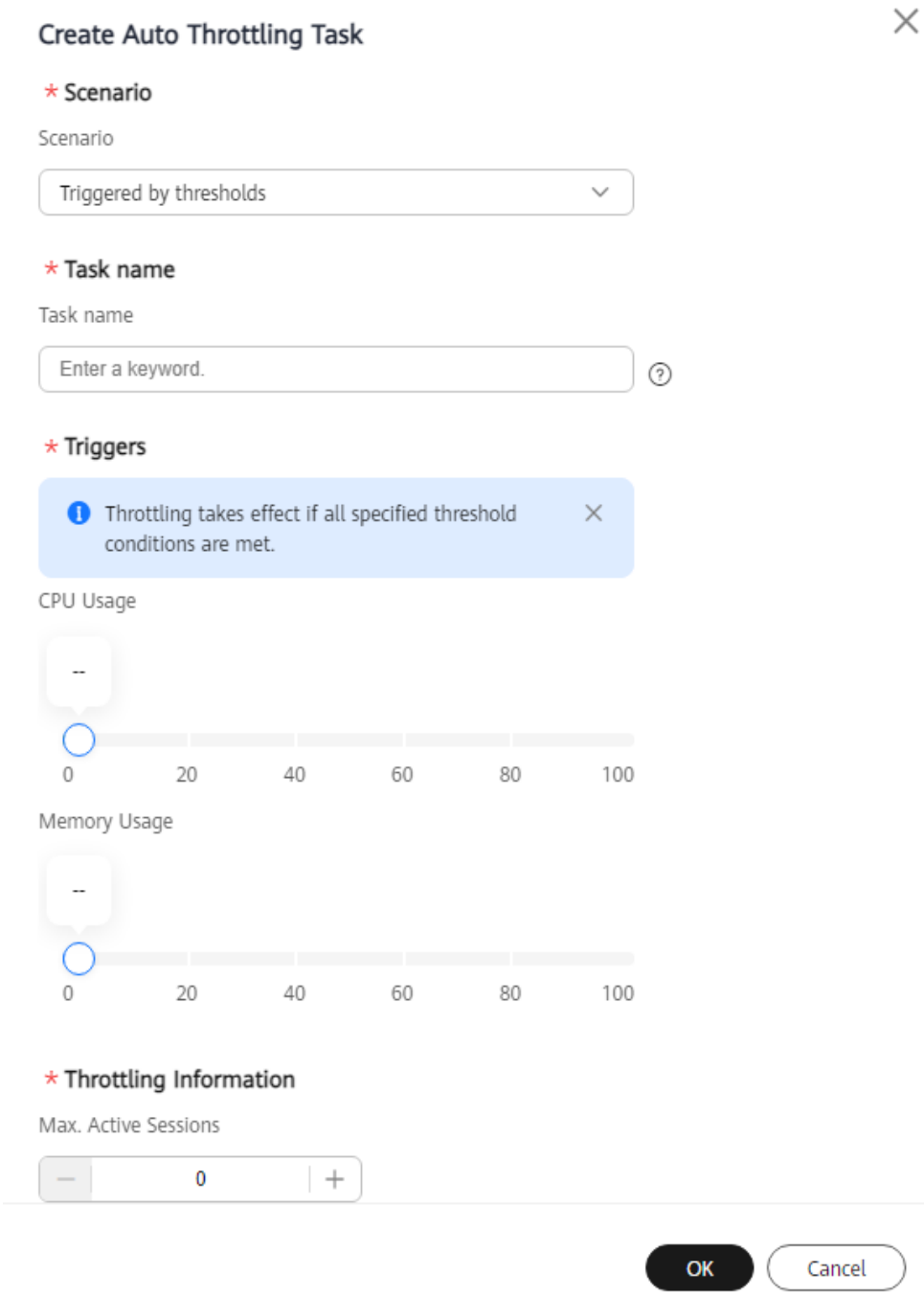


Table 16-2 Parameters for creating an auto throttling task

Parameter	Description
Scenario	Condition for triggering throttling. Currently, only **Triggered by thresholds** is available.
Task Name	Name of the throttling task.

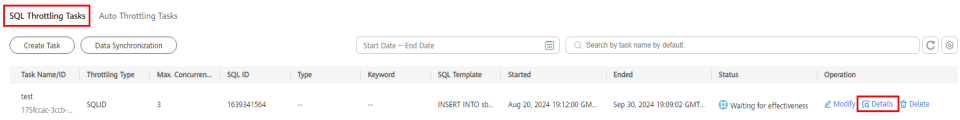
Parameter	Description
CPU Usage	CPU usage threshold for triggering throttling on the specified node. Throttling takes effect if both the CPU and memory usage threshold conditions are met. The CPU usage and memory usage thresholds cannot both be set to **0**. If you only need one of them for throttling, set the other threshold to **0**.
Memory Usage	Memory usage threshold for triggering throttling on the specified node. Throttling takes effect if both the CPU and memory usage threshold conditions are met. The CPU usage and memory usage thresholds cannot both be set to **0**. If you only need one of them for throttling, set the other threshold to **0**.
Max. Active Sessions	Maximum number of active connections for executing SQL statements on the specified node at the same time.

Step 7 Click **OK**.

Step 8 After a throttling task is created, view throttling details or modify the task.

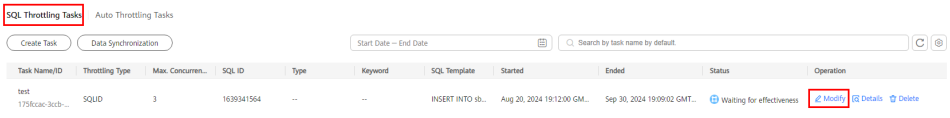
- To view task details, click **Details** in the **Operation** column of the task.

  Figure 16-3 Viewing SQL throttling task details

- To modify a throttling task, click **Modify** in the **Operation** column of the task.

  Figure 16-4 Modifying a throttling task

- To delete a throttling task, click **Delete** in the **Operation** column of the task.

NOTE

After the throttling rule of a task is modified, the instance reboots, or a primary/standby switchover is performed, the number of blocked connections of the task will be cleared.

----End

Synchronizing Data

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

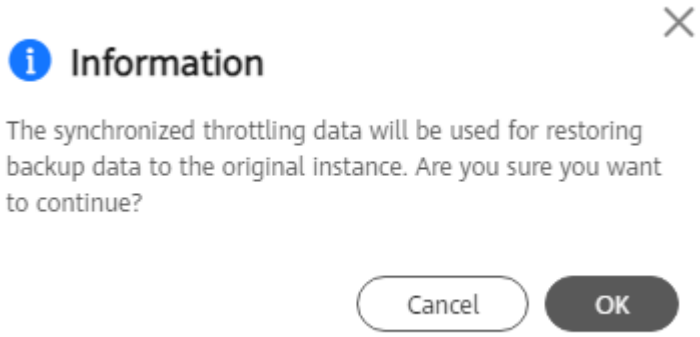
Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the navigation pane, choose **Diagnostics and Optimization** > **SQL Diagnosis**. Then, click the **SQL Throttling** tab.

Step 6 Click **Data Synchronization**.

Step 7 In the displayed dialog box, click **OK** to synchronize data after the original DB instance is backed up.

Figure 16-5 Data synchronization



----End

17 Disaster Recovery Management

17.1 Constraints

GaussDB provides the cross-region disaster recovery (DR) capability to improve your data security. If a disaster occurs in the primary data center, the DR data center takes over services quickly. The cross-region DR is suitable for data centers with a distance greater than 200 km.

The differences between cross-cloud DR and regular on-cloud DR are as follows:

- Cross-cloud DR also supports cross-region DR, and you can create a remote DR relationship on the web UI.

Constraints

Constraints on DR relationship establishment:

- Currently, a primary instance can only establish a DR relationship with one DR instance. If you want to re-establish a new DR relationship, you must stop the DR relationship between the primary and original DR instances and then select an available instance to be the DR instance.
- To establish a DR relationship for distributed instances, the number of shards of the primary instance must be the same as that of the DR instance.
- The instances for which a DR relationship is established must use the same OS. DR relationships cannot be established between instances running different OSs, for example, they run on EulerOS and HCE separately.

Constraints during DR relationship establishment:

- The DR instance is only used to synchronize data of the primary instance. It cannot read or write data.
- The DR instance does not support full backup or differential backup.
- Storage of the primary and DR instances cannot be scaled up.
- For the primary instance, if the instance is forcibly started when a majority of AZs are faulty, data cannot be synchronized to the DR instance. In this case, you need to disconnect the DR relationship.

- If a minority of nodes for the DR instance are faulty, the DR instance still works properly.
- If a majority of nodes for the DR instance are faulty, the DR instance cannot be promoted to primary.
- When you restore the primary instance to a new instance, the DR user account cannot be automatically deleted. You need to delete the account manually.
- The DR instance in a DR relationship does not support adding replicas. After the DR instance is promoted to primary, replicas in the instance can be added.
- The resource types (virtualization types) of the primary and DR instances must be the same.
- It is recommended that the specifications of the DR instance be the same as those of the primary instance. Smaller specifications of the DR instance will slow down Xlog replay on the DR instance.
- Before setting up cross-cloud DR, you need to delegate the DR operation permissions to a system user and connect the two instances properly.
  - No action is required if the two instances are in the same VPC.
  - If the two instances are deployed in different VPCs, create a VPC peering connection on the **Virtual Private Cloud** > **VPC Peering Connections** page of the VPC Endpoint console and configure routes between subnets.
  - In cross-cloud scenarios, create a Direct Connect connection.
  - Ensure that the security group allows IPv4 inbound traffic from port 12016.
- Compatibility constraints:
  - In the cross-cloud DR scenario, the peer DR instance name is not displayed in the DR relationship list.
- If the pre-verification for a cross-cloud DR operation fails, you can deliver the same operation again on the page after the fault is rectified.

17.2 Creating a DR Task

This section describes how to create a disaster recovery (DR) task.

Precautions

- Before setting up cross-cloud DR, you need to delegate the DR operation permissions to a system user and connect the two instances properly.
  - No action is required if the two instances are in the same VPC.
  - If the two instances are deployed in different VPCs, create a VPC peering connection on the **Virtual Private Cloud** > **VPC Peering Connections** page of the VPC Endpoint console and configure routes between subnets.
  - In cross-cloud scenarios, create a Direct Connect connection.
  - Ensure that the security group allows IPv4 inbound traffic from port 12016.
- Before establishing the DR relationship, log in to the console and select the cloud where the primary or DR instance is deployed separately, go to the **Basic Information** page of the instance from the **Instances** page and click

Reset Configuration for the **DR IP Address** field in the **Configuration** area. Additionally, configure the CIDR block of the peer instance for both instances. (You can obtain the CIDR block information of an instance from the **Subnet** field of the **Network Information** area on its **Basic Information** page.)

- A DR relationship cannot be established between instances with different CPU architectures.
- The replica consistency protocol and DB engine kernel version of the primary instance must match those of the DR instance.
- The following table lists the supported DR modes.

Table 17-1 Dual-instance streaming DR

Deployment	Primary Instance	DR Instance
Distributed	Combined (Basic edition)	Combined (Basic edition)
Centralized	1 primary + 2 standby	1 primary + 2 standby or Single
	1 primary + 1 standby + 1 log (Paxos)	1 primary + 1 standby + 1 log (Paxos) or Single (Paxos)

Procedure

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 In the navigation pane on the left, choose **Disaster Recovery**. On the displayed page, click **Create DR Task**.

Step 5 Configure required parameters described in **Table 17-2** and select the confirmation information.

Table 17-2 Parameter description

Parameter	Description
DR Type	Type of the disaster recovery. Streaming DR is supported.
Primary Instance	Primary instance in the DR relationship. **NOTICE** • Only available instances that run V2.0-2.0 or later versions and have more than two replicas are displayed. • When **Replica Consistency Protocol** is set to **Paxos**, only 1 primary + 1 standby + 1 log instances can create DR relationships.

Parameter	Description
DR Instance IP Address	To obtain the IP address of the DR instance, log in to the cloud where the DR instance is deployed, go to the **Basic Information** page of the instance from the **Instances** page and view the value of **DR IP Address** in the **Configuration** area. **NOTICE** Before establishing the DR relationship, log in to the console and select the cloud where the primary or DR instance is deployed separately, go to the **Basic Information** page of the instance from the **Instances** page and click **Reset Configuration** for the **DR IP Address** field in the **Configuration** area. Additionally, configure the CIDR block of the peer instance for both instances. (You can obtain the CIDR block information of an instance from the **Subnet** field of the **Network Information** area on its **Basic Information** page.) • DBS will be automatically authorized to access VPC resource information and query IaaS APIs. • DR-related configurations of the current instance will be reset.
DR Instance Administrator	Administrator of the DR instance, which is used to authenticate the primary and DR instances during DR. The default account name is user **root** of the DR instance.
DR Instance Administrator Password	Administrator password of the DR instance.

NOTICE

- Before establishing a cross-region DR relationship, contact Huawei Cloud customer service to enable network connectivity between the primary instance and DR instance.
- After a DR task is created, all databases on the selected DR instance will be cleared. The backup policy will be disabled for the selected DR instance, but historical backup data will be retained. If you want to retain data of the selected DR instance, create a manual backup before the DR task is created.
- DBS will be automatically authorized to access VPC resource information and query IaaS APIs.
- DR-related configurations of the current instance will be reset.

Step 6 Click **OK** to create a DR task. You can view the task status on the **Disaster Recovery** page.

----End

17.3 Checking DR Task Statuses

After the DR relationship is established between the primary and DR instances, you can view the data synchronization status on the DR task details page.

Procedure

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 Choose **Disaster Recovery** in the navigation pane on the left.

Step 5 Click the name of the target DR task to go to the task details page.

View the statuses of the primary and DR instances.

- Statuses of the primary instance
  - **Full synchronizing**: The primary instance is fully synchronizing data to the DR instance after the DR relationship is established.
  - **Full synchronization failed**: The primary instance fails to fully synchronize data to the DR instance after the DR relationship is established.
  - **Demoting to standby**: The roles of the primary and DR instances are being switched. In this case, the status of the primary instance is **Demoting to standby**.
  - **Incremental synchronization in progress**: In the DR relationship, the primary instance synchronizes incremental data to the DR instance.
  - **Incremental synchronization failed**: In the DR relationship, the primary instance fails to synchronize incremental data to the DR instance.
  - **RPO** indicates the time, in seconds, from when a transaction (generally the latest transaction) was submitted to when the transaction was sent to the DR instance.

    NOTE

    Description of special RPO values:

    - If RPO is 0, all transactions have been sent to the DR instance.
    - If RPO is -1, there are transactions that were not synchronized to the DR instance database or the DR instance is stopped.
- Statuses of the DR instance
  - **Full restoration**: The DR instance is in the full restoration status after DR relationship is established.
  - **Full restoration failed**: The DR instance fails to be fully restored.
  - **Incremental restoration**: The DR instance is synchronizing incremental data from the primary instance.

- **Incremental restoration failed**: The DR instance fails to synchronize incremental data from the primary instance.
- **Promoting to primary**: The roles of the primary and DR instances are being switched, or the system is failing over to the DR instance in case the primary instance becomes unavailable. In this case, the status of the DR instance is **Promoting to primary**.
- **Promotion failed**: The DR instance fails to be promoted to primary.
- **RTO** measures the amount of data being transmitted.

  NOTE

  Description of special RTO values:

  - If RTO is 0, all transactions have been transmitted to and successfully executed on the DR instance.
  - If RTO is -1, there are transactions that were not restored to the DR instance or the DR instance is stopped.

----End

17.4 Promoting the DR Instance to Primary

If the primary instance is unavailable, you can manually promote the DR instance to primary.

Precautions

- If the DR instance is available and in the incremental restoration status, you can promote the DR instance to the primary instance.
- If the DR instance is promoted to the primary instance, it can process read and write requests.
- A DR instance can be promoted to primary regardless of whether the primary instance is available. In the streaming DR scenario, if the primary instance is available, you can stop the DR task on the primary instance after the DR instance is promoted to primary.
- If the DR instance node is faulty before the promotion, repair or replace the faulty node after the promotion is complete. For details, contact Huawei Cloud customer service.
- In the streaming DR scenario, you can determine whether to enable **Re-establish DR relationship** when promoting the DR instance to primary. For details, see **Re-creating a DR Task After the Primary Instance Is Faulty**.
- After a DR task is complete, the whitelist configuration set in the **Reset Configuration** dialog box will be automatically cleared.

Procedure

Step 1 **Log in to the management console**.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click ☰ in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 In the navigation pane on the left, choose **Disaster Recovery**. Locate a DR task, and click **Promote DR Instance to Primary** in the **Operation** column.

NOTICE

After the promotion is complete, the DR task cannot be retried. You need to create a new DR task. After the DR instance is promoted to primary, you need to manually stop the DR task on the original primary instance to clear the DR data.

Step 5 Enable **Re-establish DR relationship**.

- If you enable **Re-establish DR relationship**, the system saves the DR relationship record. After the DR instance is promoted to primary, you can re-establish the DR relationship by one click.
- If you disable this option, the DR relationship will be disconnected and cannot be re-established by one click.

Step 6 In the displayed dialog box, select **Promote DR instance to primary** and click **Yes**.

The DR instance becomes the primary instance.

----End

17.5 Stopping a DR Task

This section describes how to stop the DR relationship between the primary and DR instances.

Precautions

- After the DR task is stopped, you need to create a new DR task if needed.
- After the DR task is complete, the network configuration will be automatically cleared. If you need to set up a new DR relationship, go to the instance details page to reset the configuration.
- In the streaming DR scenario, if instance B is promoted to the primary instance and removes the DR relationship with instance A, instance A does not know that the relationship has been deleted and can successfully remove the DR relationship with instance B only when instance B is available. If instance B is abnormal, you need to manually skip the removal step related to instance B when deleting the relationship on the instance A side.

Procedure

Step 1 **Log in to the management console**.

Step 2 Click ⌖ in the upper left corner and select a region and project.

Step 3 Click ☰ in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 In the navigation pane on the left, choose **Disaster Recovery**. Locate a DR task and click **Stop** in the **Operation** column.

Step 5 In the displayed dialog box, select **Stop task** and click **Yes**.

----End

17.6 Deleting a DR Task

After a DR task is stopped or fails to be created, you can delete the DR task record on the console.

Procedure

Step 1 **Log in to the management console**.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 In the navigation pane on the left, choose **Disaster Recovery**. Locate a stopped DR task and click **Delete** in the **Operation** column.

Step 5 In the displayed dialog box, confirm the information and click **Yes**.

----End

17.7 Re-creating a DR Task After the Primary Instance Is Faulty

Scenarios

A DR relationship has been established with instance A as the primary instance and instance B as the DR instance. If primary instance A is faulty, you need to manually promote instance B to primary.

After instance A is recovered, re-establish a DR relationship between instance B and instance A and promote instance A to primary again.

Procedure

Step 1 If the primary instance A is faulty, log in to the console, select the region where the DR instance B is deployed, promote instance B to primary by referring to **Promoting the DR Instance to Primary**, and enable **Re-establish DR Relationship**.

NOTICE

In this case, instance B takes over the role of instance A to provide services.

Step 2 Log in to the console, select the region where instance B is located, and re-establish a DR relationship by referring to **Re-establishing a DR Relationship**.

Step 3 After the fault of instance A is rectified, switch roles of instance A and instance B by referring to **Switching Roles of Primary and DR Instances**.

In this case, the services are taken over by instance A and the DR relationship is restored to the status before the fault occurs.

NOTICE

Instance A provides services. You need to switch services to instance A.

----End

17.8 Switching Roles of Primary and DR Instances

Scenarios

If the primary and DR regions and primary and DR instances are normal, you can log in to either the primary instance or DR instance to switch their roles. This section describes how to switch roles of primary and DR instances.

Procedure

Step 1 Log in to the console and select the region where the primary or DR instance is located.

Step 2 In the navigation pane on the left, choose **Disaster Recovery**. Locate a DR task, and click **Switch Roles** in the **Operation** column.

Step 3 In the dialog box that is displayed, confirm related information.

Step 4 Select **Switch roles** and click **Yes**.

----End

17.9 Re-establishing a DR Relationship

Scenarios

After the DR instance is promoted to primary and the original primary instance restores, you can re-establish the DR relationship between the two instances. If the DR relationship is re-established at the DR instance side, the original primary instance becomes the DR instance, and the original DR instance becomes primary.

Precautions

- Streaming DR is available only for instances whose DB engine version is V2.0-3.200 or later.

Procedure

Step 1 Log in to the console and select the region where the new primary instance is located.

Step 2 In the navigation pane on the left, choose **Disaster Recovery**. Locate the DR task and click **Re-establish DR Relationship** in the **Operation** column.

Step 3 In the displayed dialog box, select the confirmation information, and click **Yes**.

If the DR relationship is re-established at the DR instance side, the original primary instance becomes the DR instance, and the original DR instance becomes primary.

----End

17.10 Performing a DR Drill

Scenarios

If the primary and DR regions and primary and DR instances are normal, you can log in to the DR instance and perform a DR drill. This section describes the precautions and how to perform a DR drill.

Precautions

- This function is supported only when streaming DR is used and the DB engine version is V2.0-3.200 or later.
- After a DR drill starts, data synchronization between the primary and DR instances is suspended. Automated backups for the DR instance are created.
- When a DR drill is being performed on the DR instance, do not remove the DR relationship from the primary instance side.

Procedure

Step 1 Log in to the console and select the region where the DR instance is located.

Step 2 In the navigation pane on the left, choose **Disaster Recovery**. Click **More** in the **Operation** column of the target DR task, and choose **Enable DR Drill**.

Log Cache Space/Primary Instance Space: During the DR drill, the DR instance provides services and the incremental data is temporarily cached in the primary instance. After the drill is complete, the DR instance can perform incremental replay based on the temporarily stored incremental data. If the cached incremental data exceeds the value of this parameter, the DR instance will replay all data after the drill is complete.

Step 3 After the DR drill is enabled, perform DR drill operations on the DR instance.

Step 4 After the drill is complete, log in to the console and select the region where the DR instance is located. On the **Disaster Recovery** page, click **More** in the **Operation** column of the target DR task, and choose **Disable DR Drill**.

Step 5 After the DR drill is complete, view that the DR task is restored to the status before the DR drill is performed. The data generated by the primary instance

during the DR drill is synchronized to the DR instance, and the data generated by the DR instance during the DR drill is cleared.

----End

17.11 Enabling or Disabling Log Cache

Scenarios

If the primary and standby regions are disconnected for a long time due to network problems, you can enable log cache to temporarily store incremental data in the primary instance storage space. When the network recovers, the incremental data, instead of full data, is synchronized to the DR instance.

Precautions

- This function is supported only when streaming DR is used and the DB engine version is V2.0-3.200 or later.
- During the log caching period in the cross-cloud DR scenario, the DR relationship cannot be removed.

Procedure

Step 1 Log in to the console and select the region where the DR instance is located.

Step 2 In the navigation pane on the left, choose **Disaster Recovery**. Click **More** in the **Operation** column of the target DR task, and choose **Enable Log Cache**.

Step 3 After the log cache is enabled, the task status of the primary instance changes to **Caching logs**. The incremental data temporarily stores in the primary instance storage space.

Step 4 After the DR relationship is normal, log in to the console, select the region where the primary instance is located, and click **Disable Log Cache** in the **Operation** column.

Step 5 After log cache is disabled, check that the DR task is restored to the status before the log cache is enabled. The data generated during the log cache on the primary instance is synchronized to the DR instance.

----End

17.12 Restrictions on the DR Instance

After a DB instance serves as a DR instance, some basic functions are unavailable and some monitoring metrics cannot be reported. This section describes the restrictions on the DR instance.

Restrictions on the Primary Instance

After a DR relationship is established, the DR instance only allows for storage scaling, specifications change, rebooting, and parameter modification. The basic functions of the primary instance have the following restrictions:

Function	Supported by the Primary Instance During DR
Rebooting an instance	Yes
Modifying parameters	Yes
Applying parameters	Yes
Resetting a password	Yes
Performing a full backup	Yes
Creating a differential backup	Yes
Deleting a backup	Yes
Modifying the backup policy	Yes
Restoring data to the original instance	No
Restoring data to a new instance	Yes
Scaling up storage space	Yes
Adding nodes (only for distributed instances)	No
Changing instance specifications (only for centralized instances)	Yes
Upgrading a minor version	Yes
Viewing slow query logs	Yes
Viewing metrics	Yes
Deleting an instance	No
Deleting an automated backup	N/A
Rebuilding a deleted instance from the recycle bin	N/A
Backing up and restoring data in a single table	Yes
Balancing an instance	Yes
Querying the disk usage	Yes
Creating a database	Yes
Querying a database	Yes
Creating a schema and user	Yes
Deleting a schema and user	Yes
Performing database operations	Yes

Function	Supported by the Primary Instance During DR
Using DAS for database management	Yes
Repairing a node	Yes
Replacing a node	Yes
Forcibly starting a minority of AZs	No
Resuming AZs	No
Switching AZs	Yes
Resetting configurations (for DR)	Yes
Adding replicas	No
Managing tags	Yes
Binding an EIP	Yes

Restrictions on Monitoring Metrics

After the DR relationship is established, some monitoring metrics of the DR instance are unavailable. The following table lists the monitoring metrics of the DR instance.

NOTICE

If the deployment model of an instance is **1 primary + 1 standby + 1 log**, the log node does not have a data disk, and the following node-level metrics related to data disks cannot be displayed for the node:

- Disk IOPS
- Disk Write Throughput
- Disk Read Throughput
- Time Required for per Data Disk Write
- Time Required for per Data Disk Read

Category	Sub-category	Metric	Displayed for Distributed DR Instance	Displayed for Centralized DR Instance
Instance level	-	Used Instance Disk Size	Yes	No
		Total Instance Disk Size	Yes	No
		Instance Disk Usage	Yes	No
		Deadlocks	No	No

Category	Sub-category	Metric	Displayed for Distributed DR Instance	Displayed for Centralized DR Instance
		Response Time of 80% SQL Statements	No	No
		Response Time of 95% SQL Statements	No	No
		Buffer Hit Rate	No	No
Node level	-	CPU Usage	Yes	Yes
		Memory Usage	Yes	Yes
		Data Write Volume	Yes	Yes
		Outgoing Data Volume	Yes	Yes
		Disk IOPS	Yes	Yes
		Disk Write Throughput	Yes	Yes
		Disk Read Throughput	Yes	Yes
		Time Required for per Data Disk Write	Yes	Yes
		Time Required for per Data Disk Read	Yes	Yes
		Swap Memory Usage	Yes	Yes
		Total Swap Memory	Yes	Yes
		IOPS Usage (This metric applies only to nodes that use EVS disks.)	Yes	Yes
		Disk I/O Bandwidth Usage (This metric applies only to nodes that use EVS disks.)	Yes	Yes

Category	Sub-category	Metric	Displayed for Distributed DR Instance	Displayed for Centralized DR Instance
Component level	CN	User Logins, User Logouts, Lock Waiting Session Rate, Active Session Rate, CN Connections, User Committed Transactions, User Rollback Transactions, Background Committed Transactions, Background Rollback Transactions, Average Response Time of User Transactions, User Transaction Rollback Rate, Background Transaction Rollback Rate, Data Definition Language, Data Manipulation Language, Data Control Language, DDL and DCL Rate, Data Volume to Be Flushed to Disks, Physical Reads per Second, Physical Writes per Second, Online Sessions, Active Sessions, Online Session Rate, Used Dynamic Memory, Dynamic Memory Usage, Maximum Execution Duration of Database Transactions, Idle Transactions, Slow SQL Statements in the System Database, Slow SQL Statements in the User Database, and Xlog Rate	No	N/A
	Primary DN	Used Disk Size, Total Disk Size, Disk Usage, Data Volume to Be Flushed to Disks, Physical Reads per Second, and Physical Writes per Second	N/A	N/A

Category	Sub-category	Metric	Displayed for Distributed DR Instance	Displayed for Centralized DR Instance
		Standby Redo Progress, User Logins, User Logouts, Lock Waiting Session Rate, Active Session Rate, User Committed Transactions, User Rollback Transactions, Background Committed Transactions, Background Rollback Transactions, Average Response Time of User Transactions, User Transaction Rollback Rate, Background Transaction Rollback Rate, Data Definition Language, Data Manipulation Language, Data Control Language, DDL and DCL Rate, Online Sessions, Active Sessions, Online Session Rate, Used Dynamic Memory, Dynamic Memory Usage, Size of WALs Retained in the Replication Slot, and Xlog Rate	N/A	N/A
	Standby DN	Used Disk Size	Yes	Yes
		Total Disk Size	Yes	Yes
		Disk Usage	Yes	Yes
		Primary Node Flow Control Duration	No	No
		Standby Node RTO	No	No
		Standby Node Redo Progress	No	N/A
		Lock-Waiting Session Rate	No	No
		Active Session Rate	No	No
		Online Session Rate	No	No

Category	Sub-category	Metric	Displayed for Distributed DR Instance	Displayed for Centralized DR Instance
		Online Sessions	No	No
		Dynamic Memory Usage	No	No

Restrictions on Major Version Upgrade in the DR Mode

Only gray upgrade is supported.

Step 1 Upgrade the primary instance. Ensure the upgrade is not submitted.

Step 2 Upgrade the DR instance.

Step 3 Submit the upgrade of the DR instance first and then the primary instance until the DR instance upgrade is complete

----End

Restrictions on Minor Version Upgrade in the DR Mode

Only gray upgrade is supported.

Step 1 Upgrade the primary and DR instances at the same time.

Step 2 Submit the upgrade of the DR instance first and then the primary instance.

----End

NOTICE

- Before submitting the upgrade of the DR instance, ensure that the upgrade of primary instance is complete.
- Submit the upgrade of the DR instance first and then the primary instance.
- If the DR instance upgrade has been submitted, the upgrade of the primary instance cannot be rolled back.
- During the upgrade of the primary and DR instances, do not perform a switchover.

18 Managing GaussDB Tasks

You can view the progresses and results of tasks on the **Task Center** page.

NOTE

You can view and manage the following tasks:

- Creating a GaussDB instance
- Creating a manual backup
- Restoring data to a new DB instance
- Adding shards
- Adding coordinator nodes
- Restoring data to an existing instance
- Restoring data to the original DB instance
- Scaling up storage space
- Changing instance specifications
- Deleting a GaussDB instance
- Stopping a backup
- Changing the deployment model of an instance
- Rolling upgrade
- Upgrade commit
- Upgrade auto-commit
- In-place upgrade
- Upgrade rollback

Viewing a Task

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 Choose **Task Center** in the navigation pane on the left. On the displayed page, view the task details.

- To identify a task, you can use the task name/ID or instance name/ID, or simply select a task name in the search box displayed in the upper pane of the page.
- You can view the progress and status of tasks in a specific period. The default period is seven days.

  The task list can only show up to 30 days of past tasks.
- You can view tasks in the following statuses:
  - Running
  - Completed
  - Failed
- You can view the task creation and completion time.

----End

Deleting a Task Record

You can delete the task records that no longer need to be displayed. The deletion only deletes the task records, and does not delete the DB instances or terminate the tasks that are being executed.

NOTICE

Deleted task records cannot be recovered. Exercise caution when performing this operation.

Step 1 **Log in to the management console**.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 Choose **Task Center** in the navigation pane on the left. On the displayed page, locate the task record to be deleted and click **Delete** in the **Operation** column. In the displayed dialog box, click **OK**.

You can delete tasks in the following statuses:

- Completed
- Failed

----End

19 Managing GaussDB Tags

Scenarios

Tag Management Service (TMS) enables you to use tags on the management console to manage resources. TMS works with other cloud services to manage tags. TMS manages tags globally, and other cloud services manage their own tags.

- You are advised to set predefined tags on the TMS console.
- A tag consists of a key and value. You can add only one value for each key.
- A maximum of 20 tags can be added for a DB instance.

Editing Tags

Step 1 **Log in to the management console**.

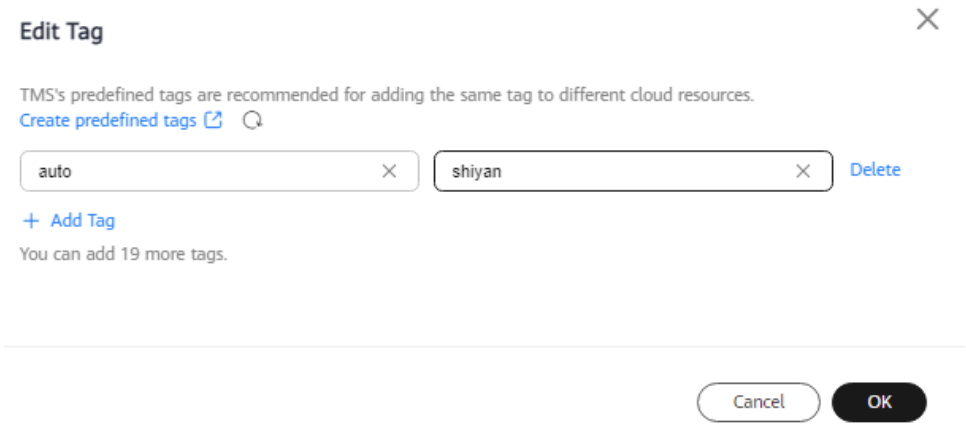
Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 5 In the navigation pane, choose **Tags**. On the displayed page, click **Edit Tag**. In the displayed dialog box, click **Add Tag**, enter a tag key and value, and click **OK**.

Figure 19-1 Editing a tag



- When you enter a tag key and value, the system automatically displays all tags (including predefined tags and resource tags) associated with all DB instances except the current one.
- A tag key can contain up to 128 characters. It cannot start with _**sys**_ or a space, and cannot end with a space. Only letters, digits, spaces, and the following special characters are allowed: _.:=+-@
- A tag value can contain up to 255 characters. Only letters, digits, spaces, and the following special characters are allowed: _.:/=+-@

Step 6 View and manage the tag on the **Tags** page.

----End

Deleting a Tag

Step 1 **Log in to the management console**.

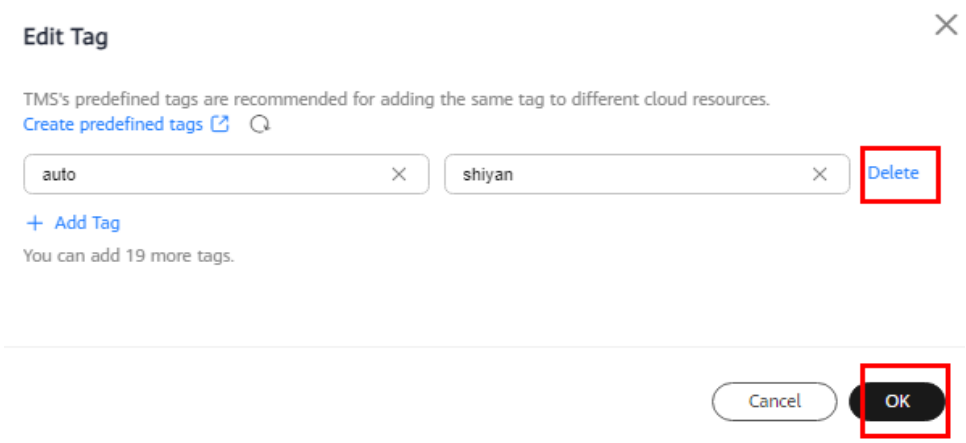
Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, click the instance name to go to the **Basic Information** page.

Step 5 In the navigation pane on the left, choose **Tags**. On the displayed page, click **Edit Tag**, locate the tag to be deleted, and click **Delete**. Then, click **OK**.

Figure 19-2 Deleting a tag



Step 6 Check that the tag is no longer displayed on the **Tags** page.

----End

20 Resetting the Administrator Password of a GaussDB Instance

Scenarios

If you forget the password of your **root** account when using GaussDB, you can reset the password.

Precautions

- If the password you provide is regarded as a weak password by the system, you will be prompted to enter a stronger password.
- If the DB instance is abnormal, the administrator password cannot be reset.
- The volume of data being processed by the instance determines how long it takes for the new password to take effect.
- To prevent brute force cracking and ensure system security, change your password periodically.
- You cannot reset the administrator password when the account is frozen.
- If you log in to the database as the **root** user, resetting the password may interrupt services. Exercise caution when performing this operation.

Procedure

Step 1 Log in to the management console.

Step 2 Click  in the upper left corner and select a region and project.

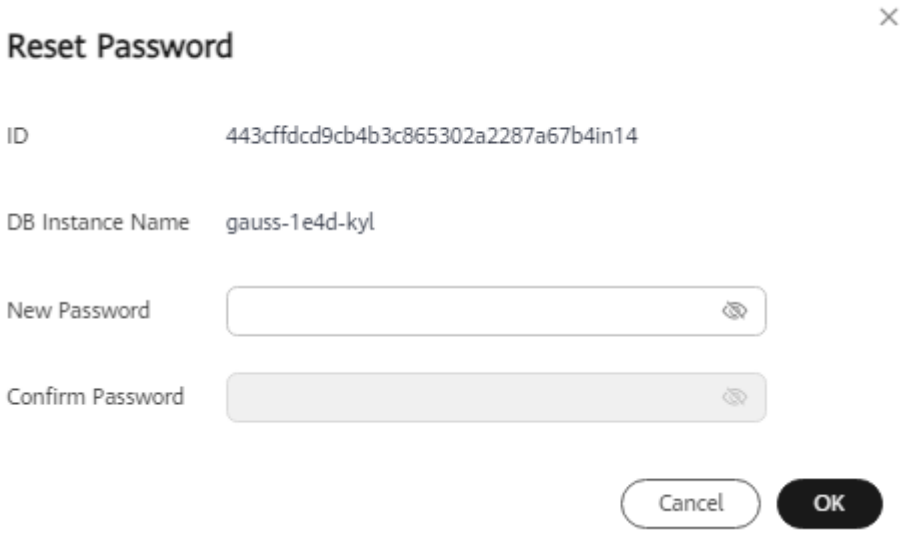
Step 3 Click  in the upper left corner of the page and choose **Databases** > **GaussDB**.

Step 4 On the **Instances** page, locate the instance that you want to reset password for and click **More** > **Reset Password** in the **Operation** column.

Alternatively, click the instance name to go to the **Basic Information** page. In the **Basic Information** area, click **Reset Password** next to the **Administrator** field.

Step 5 Enter a new password and confirm the password.

Figure 20-1 Resetting a password



NOTICE

The new password must:

- Contain 8 to 32 characters.
- Contain at least three types of the following: uppercase letters, lowercase letters, digits, and special characters. Supported characters: ~!@#%^*-_=+?,
- Be different from the old password or the old password written backwards.

Step 6 If you have enabled operation protection, click **Start Verification** in the displayed dialog box. On the displayed page, click **Send Code**, enter the obtained verification code, and click **Verify** to close the page.

Two-factor authentication improves the security of your account. For details about how to enable operation protection, see ***Identity and Access Management User Guide***.

----End

