

Data Warehouse Service

Management Guide

Issue 01
Date 2024-06-13



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

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

Trademarks and Permissions



HUAWEI 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.

Contents

1 Service Overview.....	1
1.1 What Is GaussDB(DWS)?.....	1
1.2 Advantages.....	7
1.3 Application Scenarios.....	9
1.4 Functions.....	11
1.5 Concepts.....	16
1.6 Related Services.....	18
1.7 GaussDB(DWS) Permissions Management.....	19
1.8 GaussDB(DWS) Access.....	23
1.9 Restrictions.....	25
2 Getting Started.....	29
2.1 Step 1: Starting Preparations.....	29
2.2 Step 2: Creating a Cluster.....	29
2.3 Step 3: Connecting to a Cluster.....	31
2.4 Step 4: Viewing Other Documents and Deleting Resources.....	35
3 Process for Using GaussDB(DWS).....	37
4 Preparations.....	40
5 Creating or Deleting a Cluster.....	41
5.1 Accessing the GaussDB(DWS) Management Console.....	41
5.2 Creating a Cluster.....	41
5.3 Deleting a Cluster.....	51
6 Cluster Connection.....	53
6.1 Methods of Connecting to a Cluster.....	53
6.2 Obtaining the Cluster Connection Address.....	54
6.3 Using the Data Studio GUI Client to Connect to a Cluster.....	55
6.4 Using the gsql CLI Client to Connect to a Cluster.....	59
6.4.1 Downloading the Data Studio client.....	59
6.4.2 Using the Linux gsql Client to Connect to a Cluster.....	61
6.4.3 Using the Windows gsql Client to Connect to a Cluster.....	63
6.4.4 Establishing Secure TCP/IP Connections in SSL Mode.....	66
6.5 Using the JDBC and ODBC Drivers to Connect to a Cluster.....	73

6.5.1 Development Specifications.....	73
6.5.2 Downloading the JDBC or ODBC Driver.....	74
6.5.3 Using JDBC to Connect to a Cluster.....	74
6.5.4 Configuring JDBC to Connect to a Cluster (Load Balancing Mode).....	86
6.5.5 Configuring JDBC to Connect to a Cluster (IAM Authentication Mode).....	87
6.5.6 Using ODBC to Connect to a Cluster.....	92
6.6 Using the Third-Party Function Library psycopg2 of Python to Connect to a Cluster.....	98
6.7 Using the Python Library PyGreSQL to Connect to a Cluster.....	108
6.8 Managing Database Connections.....	125
7 Monitoring and Alarms.....	129
7.1 Monitoring Clusters Using Cloud Eye.....	129
7.2 Alarms.....	134
7.2.1 Alarm Management.....	135
7.2.2 Alarm Rules.....	138
7.2.3 Alarm Subscriptions.....	141
7.3 Event Notifications.....	144
7.3.1 Event Notifications Overview.....	144
7.3.2 Viewing Events.....	147
8 Specifications Change and Scaling.....	148
8.1 Managing Nodes.....	148
8.2 Scaling Nodes.....	150
8.2.1 Scaling Out a Cluster.....	150
8.2.2 Cluster Redistribution.....	153
8.2.2.1 Redistributing Data.....	153
8.2.2.2 Viewing Redistribution Details.....	156
8.2.3 Scaling In a Cluster.....	159
8.3 Changing Specifications.....	162
8.3.1 Changing the Node Flavor.....	162
8.3.2 Changing All Specifications.....	164
8.3.3 Disk Capacity Expansion of an EVS Cluster.....	165
9 Backup and Disaster Recovery.....	168
9.1 Snapshots.....	168
9.1.1 Overview.....	168
9.1.2 Manual Snapshots.....	169
9.1.2.1 Creating a Manual Snapshot.....	169
9.1.2.2 Deleting a Manual Snapshot.....	171
9.1.3 Automated Snapshots.....	172
9.1.3.1 Automatic Snapshot Overview.....	172
9.1.3.2 Configuring an Automated Snapshot Policy.....	173
9.1.3.3 Copying Automated Snapshots.....	177
9.1.3.4 Deleting an Automated Snapshot.....	178

9.1.4 Viewing Snapshot Information.....	179
9.1.5 Restoration Using a Snapshot.....	180
9.1.5.1 Constraints on Restoring a Snapshot.....	180
9.1.5.2 Restoring a Snapshot to a New Cluster.....	181
9.1.5.3 Restoring a Snapshot to the Original Cluster.....	182
9.1.6 Configuring a Snapshot.....	183
9.1.7 Stopping Snapshot Creation.....	186
9.2 Cluster DR.....	187
9.2.1 DR Overview.....	187
9.2.2 Creating a DR Task.....	189
9.2.3 Viewing DR Information.....	190
9.2.4 DR Management.....	191
9.2.5 Mutually Exclusive DR Cases.....	195
10 Intelligent O&M.....	196
10.1 Overview.....	196
10.2 O&M Plans.....	197
10.3 O&M Status.....	202
11 Cluster Management.....	204
11.1 Modifying Database Parameters.....	204
11.2 Checking the Cluster Status.....	206
11.3 Viewing Cluster Details.....	210
11.4 Managing Access Domain Names.....	213
11.5 Cluster Topology.....	216
11.6 Managing Tags.....	222
11.6.1 Overview.....	222
11.6.2 Tag Management.....	223
11.7 Managing Enterprise Projects.....	225
11.8 Managing Clusters That Fail to Be Created.....	227
11.9 Removing the Read-only Status.....	227
11.10 Performing a Primary/Standby Switchback.....	228
11.11 Cluster Restart.....	229
11.12 Resetting a Password.....	230
11.13 Cluster Upgrade.....	231
11.14 Associating and Disassociating ELB.....	233
11.15 Managing CNs.....	236
12 Cluster Log Management.....	239
13 Database User Management.....	242
13.1 Managing Users.....	242
13.2 Managing Roles.....	245
14 Audit Logs.....	248

14.1 Audit Log Overview.....	248
14.2 Management Console Audit Logs.....	248
14.3 Database Audit Logs.....	251
14.3.1 Configuring the Database Audit Logs.....	251
14.3.2 Dumping the Database Audit Logs.....	252
14.3.3 Viewing Database Audit Logs.....	259
15 Cluster Security Management.....	262
15.1 Configuring Separation of Permissions.....	262
15.2 Encrypting Databases.....	265
15.2.1 Overview.....	265
15.2.2 Rotating Encryption Keys.....	267
15.3 Permissions.....	267
15.3.1 Syntax of Fine-Grained Permissions Policies.....	267
16 Resource Management.....	287
16.1 Overview.....	287
16.2 Resource Pool.....	289
16.2.1 Feature Description.....	289
16.2.2 Page Overview.....	292
16.2.3 Creating a Resource Pool.....	294
16.2.4 Modifying a Resource Pool.....	296
16.2.5 Deleting a Resource Pool.....	301
16.3 Resource Management Plan.....	302
16.3.1 Managing Resource Management Plans.....	302
16.3.2 Managing Resource Management Plan Stages.....	304
16.3.3 Importing or Exporting a Resource Management Plan.....	308
16.4 Workspace Management.....	309
17 Data Source Management.....	311
17.1 MRS Data Sources.....	311
17.1.1 MRS Data Source Usage Overview.....	311
17.1.2 Creating an MRS Data Source Connection.....	312
17.1.3 Updating the MRS Data Source Configuration.....	315
18 Managing Logical Clusters.....	318
18.1 Logical Cluster Overview.....	318
18.2 Adding Logical Clusters.....	325
18.3 Editing Logical Clusters.....	327
18.4 Managing Resources (in a Logical Cluster).....	328
18.5 Restarting Logical Clusters.....	329
18.6 Scaling Out Logical Clusters.....	329
18.7 Deleting Logical Clusters.....	330
18.8 Tutorial: Converting a Physical Cluster That Contains Data into a Logical Cluster.....	331
18.9 Tutorial: Dividing a New Physical Cluster into Logical Clusters.....	337

19 FAQs.....	340
19.1 General Problems.....	340
19.1.1 Why Do I Need to Use a Data Warehouse?.....	340
19.1.2 What Are the Differences Between Users and Roles?.....	341
19.1.3 When Should I Use GaussDB(DWS) and MRS?.....	342
19.1.4 How Do I Check the Creation Time of a Database User?.....	343
19.1.5 Regions and AZs.....	344
19.1.6 Is My Data Secure in GaussDB(DWS)?.....	344
19.1.7 How Is GaussDB(DWS) Secured?.....	345
19.1.8 Can I Modify the Security Group of a GaussDB(DWS) Cluster?.....	345
19.1.9 How Are Dirty Pages Generated in GaussDB(DWS)?.....	346
19.2 Database Usage.....	347
19.2.1 How Do I Change Distribution Columns?.....	347
19.2.2 How Do I View and Set the Database Character Encoding?.....	349
19.2.3 What Do I Do If Date Type Is Automatically Converted to the Timestamp Type During Table Creation?.....	350
19.2.4 Do I Need to Run VACUUM FULL and ANALYZE on Common Tables Periodically?.....	351
19.2.5 Do I Need to Set a Distribution Key After Setting a Primary Key?.....	353
19.2.6 Is GaussDB(DWS) Compatible with PostgreSQL Stored Procedures?.....	353
19.2.7 What Are Partitioned Tables, Partitions, and Partition Keys?.....	353
19.2.8 How Can I Export the Table Structure?.....	354
19.2.9 How Do I Delete Table Data Efficiently?.....	354
19.2.10 How Do I View Foreign Table Information?.....	355
19.2.11 If No Distribution Column Is Specified, How Will Data Be Stored?.....	355
19.2.12 How Do I Replace the Null Result with 0?.....	356
19.2.13 How Do I Check Whether a Table Is Row-Stored or Column-Stored?.....	357
19.2.14 How Do I Query the Information About GaussDB(DWS) Column-Store Tables?.....	358
19.2.15 Why Sometimes the GaussDB(DWS) Query Indexes Become Invalid?.....	359
19.2.16 How Do I Use a User-Defined Function to Rewrite the CRC32() Function?.....	367
19.2.17 What Are the Schemas Starting with pg_toast_temp* or pg_temp* ?.....	368
19.2.18 Solutions to Inconsistent GaussDB(DWS) Query Results.....	368
19.2.19 Which System Catalogs That the VACUUM FULL Operation Cannot Be Performed on?.....	373
19.2.20 In Which Scenarios Would a Statement Be "idle in transaction"?.....	374
19.2.21 How Does GaussDB(DWS) Implement Row-to-Column and Column-to-Row Conversion?.....	376
19.2.22 What Are the Differences Between Unique Constraints and Unique Indexes?.....	380
19.2.23 What Are the Differences Between Functions and Stored Procedures?.....	381
19.2.24 How Do I Delete Duplicate Table Data?.....	382
19.3 Cluster Management.....	384
19.3.1 What Do I Do If Creating a GaussDB(DWS) Cluster Failed?.....	385
19.3.2 How Can I Clear and Reclaim the Storage Space?.....	385
19.3.3 Why Did the Used Storage Shrink After Scale-out?.....	386
19.3.4 How Do I View Node Metrics (CPU, Memory, and Disk Usage)?.....	386
19.3.5 How Is the Disk Space or Capacity of GaussDB(DWS) Calculated?.....	386

19.3.6 What Are the gaussdb and postgres Databases of GaussDB(DWS)?.....	387
19.3.7 How Do I Set the Maximum Number of Sessions When Adding an Alarm Rule on Cloud Eye?.....	387
19.3.8 When Should I Add CNs or Scale out a cluster?.....	388
19.3.9 How Should I Select from a Small-Flavor Many-Node Cluster and a Large-Flavor Three-Node Cluster with Same CPU Cores and Memory?.....	389
19.3.10 What Are the Differences Between Hot Data Storage and Cold Data Storage?.....	389
19.3.11 What Do I do if the Scale-in Button Is Dimmed?.....	390
19.4 Database Connections.....	390
19.4.1 How Applications Communicate with GaussDB(DWS)?.....	391
19.4.2 Does GaussDB(DWS) Support Third-Party Clients and JDBC and ODBC Drivers?.....	393
19.4.3 Can I Connect to GaussDB(DWS) Cluster Nodes Using SSH?.....	394
19.4.4 What Should I Do If I Cannot Connect to a GaussDB(DWS) Cluster?.....	394
19.4.5 Why Was I Not Notified of Failure Unbinding the EIP When GaussDB(DWS) Is Connected Over the Internet?.....	395
19.4.6 How Do I Configure a Whitelist to Protect Clusters Available Through a Public IP Address?.....	395
19.5 Data Import and Export.....	396
19.5.1 What Are the Differences Between Data Formats Supported by OBS and GDS Foreign Tables?....	396
19.5.2 How Do I Import Incremental Data Using an OBS Foreign Table?.....	396
19.5.3 How Can I Import Data to GaussDB(DWS)?.....	396
19.5.4 How Much Service Data Can a Data Warehouse Store?.....	397
19.5.5 How Do I Use \Copy to Import and Export Data?.....	397
19.5.6 How Do I Implement Fault Tolerance Import Between Different Encoding Libraries.....	398
19.5.7 Can I Import and Export Data to and from OBS Across Regions?.....	399
19.5.8 Can I Import Data over the Public/External Network Using GDS?.....	399
19.5.9 Which Are the Factors That Affect GaussDB(DWS) Import Performance?.....	399
19.6 Account, Password, and Permission.....	400
19.6.1 How Does GaussDB(DWS) Implement Workload Isolation?.....	400
19.6.2 How Do I Change the Password of a Database Account When the Password Expires?.....	404
19.6.3 How Do I Grant Table Permissions to a User?.....	404
19.6.4 How Do I Grant Schema Permissions to a User?.....	408
19.6.5 How Do I Create a Database Read-only User?.....	410
19.6.6 How Do I Create Private Database Users and Tables?.....	411
19.6.7 How Do I Revoke the CONNECT ON DATABASE Permission from a User?.....	412
19.6.8 How Do I View the Table Permissions of a User?.....	414
19.6.9 Who Is User Ruby?.....	416
19.7 Database Performance.....	417
19.7.1 Why Is SQL Execution Slow After Long GaussDB(DWS) Usage?.....	417
19.7.2 Why Does GaussDB(DWS) Perform Worse Than a Single-Server Database in Extreme Scenarios?.....	417
19.7.3 How Can I View SQL Execution Records in a Certain Period When Read and Write Requests Are Blocked?.....	418
19.7.4 What Do I Do If My Cluster Is Unavailable Because of Insufficient Space?.....	418
19.7.5 GaussDB(DWS) CPU Resource Management.....	418

19.7.6 Why the Tasks Executed by an Ordinary User Are Slower Than That Executed by the dbadmin User?.....	421
19.7.7 What Are the Factors Related to the Single-Table Query Performance in GaussDB(DWS)?.....	422
19.8 Snapshot Backup and Restoration.....	423
19.8.1 Why Is Creating an Automated Snapshot So Slow?.....	423
19.8.2 Does a DWS Snapshot Have the Same Function as an EVS Snapshot?.....	423

1 Service Overview

1.1 What Is GaussDB(DWS)?

GaussDB(DWS) is an online data analysis and processing database built on the infrastructure and platform. It offers scalable, ready-to-use, and fully managed analytical database services, and is compatible with ANSI/ISO SQL92, SQL99, and SQL 2003 syntax. Additionally, GaussDB(DWS) is interoperable with other database ecosystems such as PostgreSQL, Oracle, Teradata, and MySQL. This makes it a competitive option for petabyte-scale big data analytics across diverse industries.

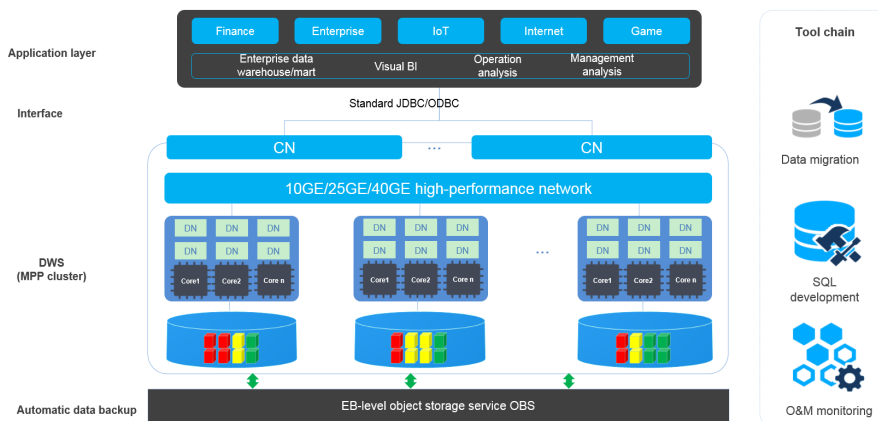
- **Standard Data Warehouse (DWS 2.0):** Oriented to data analysis scenarios, DWS 2.0 provides enterprise-level data warehouse services with high performance, high scalability, high reliability, high security, and easy O&M. It is capable of data analysis at a scale of 2048 nodes and 20 petabytes of data. It supports hot and cold data analysis, elastic scaling of storage and computing resources, and on-demand and pay-per-use pricing, providing users with elastic, flexible, and cost-effective experience. It is applicable to converged analysis services that integrate libraries, warehouses, cities, and lakes.
- **Stream data warehouse:** The stream data warehouse is built on top of the standard data warehouse. It integrates stream and time sequence engines to provide real-time data ingestion and high-concurrency real-time analysis capabilities. It can be used for IoT real-time analysis.
- **Hybrid data warehouse:** It provides high-concurrency, high-performance, and low-latency transaction processing capabilities based on large-scale data query and analysis capabilities. It is suitable for hybrid transaction/analytical processing (HTAP). A database can be used for both production and analysis.

Architecture

GaussDB(DWS) employs the shared-nothing architecture and the massively parallel processing (MPP) engine, and consists of numerous independent logical nodes that do not share the system resources such as CPUs, memory, and storage. In such a system architecture, service data is separately stored on numerous nodes. Data analysis tasks are executed in parallel on the nodes where data is

stored. The massively parallel data processing significantly improves response speed.

Figure 1-1 Architecture



- **Application layer**

Data loading tools, extract, transform, and load (ETL) tools, business intelligence (BI) tools, as well as data mining and analysis tools, can be integrated with GaussDB(DWS) through standard APIs. GaussDB(DWS) is compatible with the PostgreSQL ecosystem, and the SQL syntax is compatible with Oracle and Teradata. Applications can be smoothly migrated to GaussDB(DWS) with few changes.

- **API**

Applications can connect to GaussDB(DWS) through the standard Java Database Connectivity (JDBC) 4.0 and Open Database Connectivity (ODBC) 3.5.

- **GaussDB(DWS)**

A GaussDB(DWS) cluster contains nodes of the same flavor in the same subnet. These nodes jointly provide services. Datanodes (DNs) in a cluster store data on disks. Coordinators (CNs) receive access requests from applications and return the execution results to clients. In addition, a CN splits and distributes tasks to the DN for parallel processing.

- **Automatic data backup**

Cluster snapshots can be automatically backed up to the EB-level Object Storage Service (OBS), which facilitates periodic backup of the cluster during off-peak hours, ensuring data recovery after a cluster exception occurs.

A snapshot is a complete backup of GaussDB(DWS) at a specific time point, including the configuration data and service data of a cluster.

- **Tool chain**

The parallel data loading tool General Data Service (GDS), SQL syntax migration tool Database Schema Converter (DSC), and SQL development tool Data Studio are provided. The cluster O&M can be monitored on a console.

Logical Cluster Architecture

Figure 1-2 shows the logical architecture of a GaussDB(DWS) cluster. For details about instances, see Table 1-1.

Figure 1-2 Logical cluster architecture

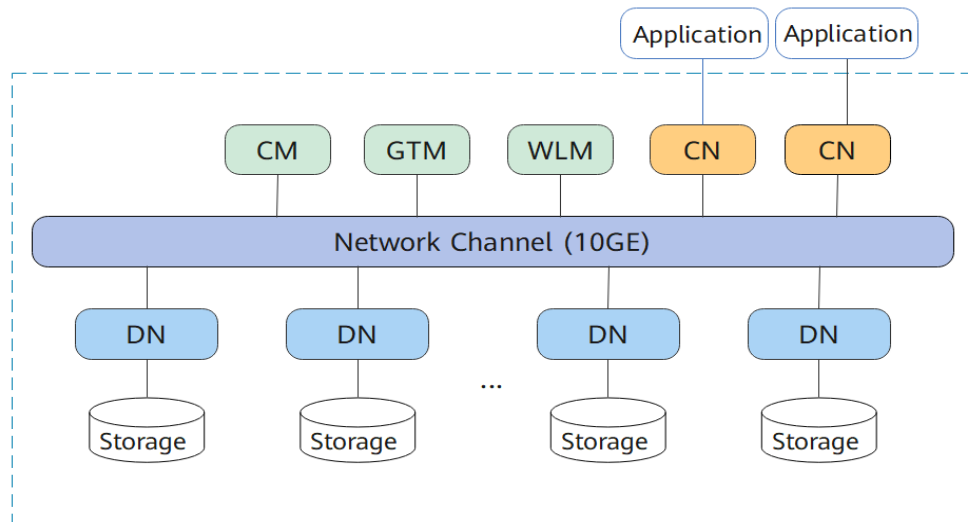


Table 1-1 Cluster architecture description

Name	Function	Description
Cluster Manager (CM)	Cluster Manager. It manages and monitors the running status of functional units and physical resources in the distributed system, ensuring system stability.	<p>The CM consists of CM Agent, OM Monitor, and CM Server.</p> <ul style="list-style-type: none">• CM Agent monitors the running status of primary and standby GTMs, CNs, and primary and standby DNAs on the host, and reports the status to CM Server. In addition, it executes the arbitration instruction delivered by CM Server. A CM Agent process runs on each host.• OM Monitor monitors scheduled tasks of CM Agent and restarts CM Agent when CM Agent stops. If CM Agent cannot be restarted, the host cannot be used. In this case, manually rectify this fault. <p>NOTE CM Agent cannot be restarted probably because of insufficient system resources, which is not a common situation.</p> <ul style="list-style-type: none">• CM Server checks whether the current system is normal according to the instance status reported by CM Agent. In the case of exceptions, CM Server delivers recovery commands to CM Agent. <p>GaussDB(DWS) provides the primary/standby CM Server solution to ensure system HA. CM Agent connects to the primary CM Server. If the primary CM Server is faulty, the standby CM Server is promoted to primary to prevent a single point of failure (SPOF).</p>
Global Transaction Manager (GTM)	Generates and maintains the globally unique information, such as the transaction ID, transaction snapshot, and timestamp.	The cluster includes only one pair of GTMs: one primary GTM and one standby GTM.
Workload Manager (WLM)	Workload Manager. It controls allocation of system resources to prevent service congestion and system crash resulting from excessive workload.	You do not need to specify names of hosts where WLMs are to be deployed, because the installation program automatically installs a WLM on each host.

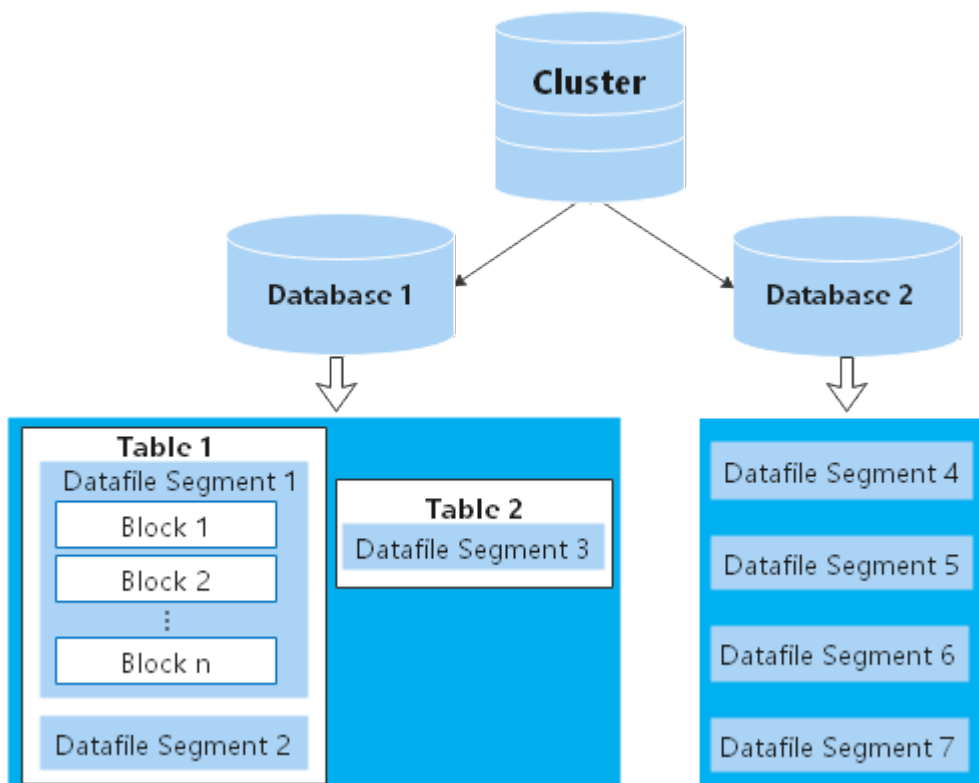
Name	Function	Description
Coordinator (CN)	A CN receives access requests from applications, and returns execution results to the client; splits tasks and allocates task fragments to different DNs for parallel processing.	<p>CNs in a cluster have equivalent roles and return the same result for the same DML statement. Load balancers can be added between CNs and applications to ensure that CNs are transparent to applications. If a CN is faulty, the load balancer automatically connects the application to the other CN. For details, see section "Associating and Disassociating ELB".</p> <p>CNs need to connect to each other in the distributed transaction architecture. To reduce heavy load caused by excessive threads on GTMs, no more than 10 CNs should be configured in a cluster.</p> <p>GaussDB(DWS) handles the global resource load in a cluster using the Central Coordinator (CCN) for adaptive dynamic load management. When the cluster is started for the first time, the CM selects the CN with the smallest ID as the CCN. If the CCN is faulty, CM replaces it with a new one.</p>

Name	Function	Description
Datanode (DN)	A DN stores service data by column or row or in the hybrid mode, executes data query tasks, and returns execution results to CNs.	<p>A cluster consists of multiple DNs and each DN stores part of data. GaussDB(DWS) provides DN high availability: active DN, standby DN, and secondary DN. The working principles of the three are as follows:</p> <ul style="list-style-type: none">• During data synchronization, if the active DN suddenly becomes faulty, the standby DN is switched to the active state.• Before the faulty active DN recovers, the new active DN synchronizes data logs to the secondary DN.• After the faulty active DN recovers, it becomes the standby DN and uses data logs stored on the secondary DN to restore data generated during its faulty period. <p>The secondary DN serves exclusively as a backup, never ascending to active or standby status in case of faults. It conserves storage by only holding Xlog data transferred from the new active DN and data replicated during original active DN failures. This efficient approach saves one-third of the storage space compared to conventional tri-backup methods.</p>
Storage	Functions as the server's local storage resources to store data permanently.	-

DNs in a cluster store data on disks. [Figure 1-3](#) describes the objects on each DN and the relationships among them logically.

- A database manages various data objects and is isolated from other databases.
- A datafile segment stores data in only one table. A table containing more than 1 GB of data is stored in multiple data file segments.
- A table belongs only to one database.
- A block is the basic unit of database management, with a default size of 8 KB.

Data can be distributed in replication, round-robin, or hash mode. You can specify the distribution mode during table creation.

Figure 1-3 Logical database architecture

1.2 Advantages

GaussDB(DWS) uses the GaussDB database kernel and is compatible with PostgreSQL 9.2.4. It transforms from a single OLTP database to an enterprise-level distributed OLAP database oriented to massive data analysis based on the massively parallel processing (MPP) architecture.

Unlike conventional data warehouses, GaussDB(DWS) excels in massive data processing and general platform management with the following benefits:

Ease of use

- Visualized one-stop management
GaussDB(DWS) allows you to easily complete the entire process from project concept to production deployment. With the GaussDB(DWS) management console, you can obtain a high-performance and highly available enterprise-level data warehouse cluster within several minutes. Data warehouse software or data warehouse servers are not required.
With just a few clicks, you can easily connect applications to the data warehouse, back up data, restore data, and monitor data warehouse resources and performance.
- Seamless integration with big data
Without the need to migrate data, you can use standard SQL statements to directly query data on HDFS and OBS.
- Heterogeneous database migration tools

GaussDB(DWS) provides various migration tools to migrate SQL scripts of Oracle and Teradata to GaussDB(DWS).

High performance

- Cloud-based distributed architecture
GaussDB(DWS) adopts the MPP-based database so that service data is separately stored on numerous nodes. Data analysis tasks are executed in parallel on the nodes where data is stored. The massively parallel data processing significantly improves response speed.
- Query response to trillions of data records within seconds
GaussDB(DWS) improves data query performance by executing multi-thread operators in parallel, running commands in registers in parallel with the vectorized computing engine, and reducing redundant judgment conditions using LLVM.
GaussDB(DWS) provides you with a better data compression ratio (column-store), better indexing (column-store), and higher point update and query (row-store) performance.
- Fast data loading
GDS is a tool that helps you with high-speed massively parallel data loading.

High scalability

- On-demand scale-out: With the shared-nothing open architecture, nodes can be added at any time to enhance the data storage, query, and analysis capabilities of the system.
- Enhanced linear performance after scale-out: The capacity and performance increase linearly with the cluster scale. The linear rate is 0.8.
- Service continuity: During scale-out, data can be added, deleted, modified, and queried, and DDL operations (**DROP/TRUNCATE/ALTER TABLE**) can be performed. Online table-level scale-out ensures service continuity.

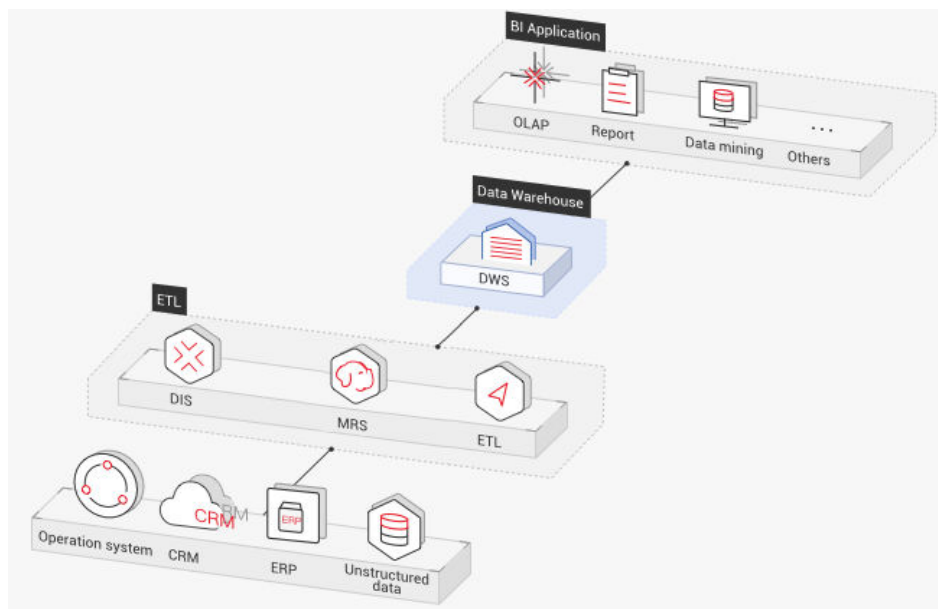
Robust reliability

- ACID
Support for the atomicity, consistency, isolation, and durability (ACID) feature, which ensures strong data consistency for distributed transactions.
- Comprehensive HA design
All software processes of GaussDB(DWS) are in active/standby mode. Logical components such as the CNs and DNPs of each cluster also work in active/standby mode. This ensures data reliability and consistency when any single point of failure (SPOF) occurs.
- High security
GaussDB(DWS) supports transparent data encryption and can interconnect with the Database Security Service (DBSS) to better protect user privacy and data security with network isolation and security group rule setting options. In addition, GaussDB(DWS) supports automatic full and incremental backup of data, improving data reliability.

1.3 Application Scenarios

- Enhanced ETL + Real-time BI analysis

Figure 1-4 ETL+BI analysis



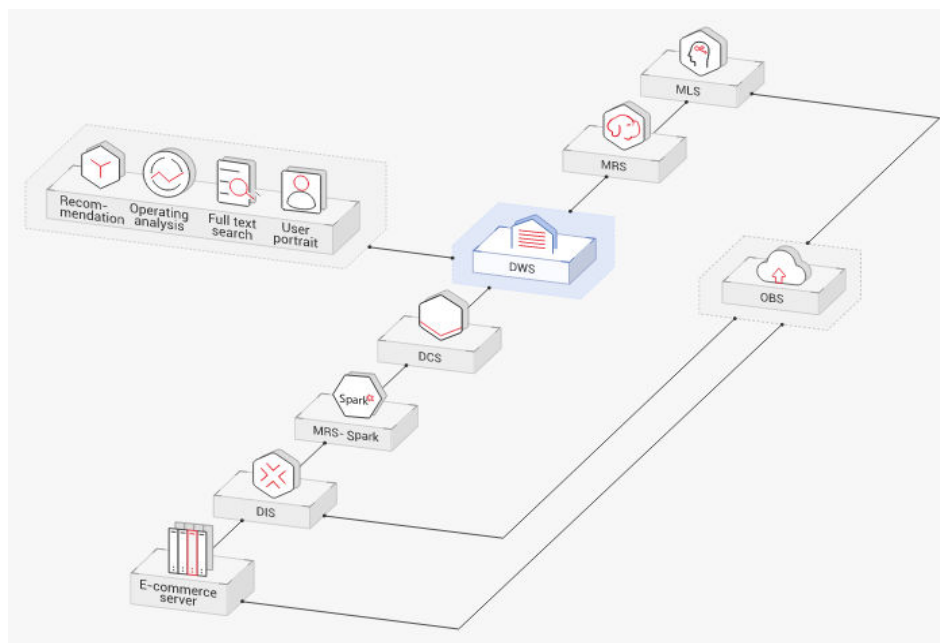
The data warehouse is the pillar of the Business Intelligence (BI) system for collecting, storing, and analyzing massive amounts of data. It provides powerful business analysis support for IoT, mobile Internet, gaming, and Online to Offline (O2O) industries.

Advantages of GaussDB(DWS) are as follows:

- **Data migration**: efficient and real-time data import in batches from multiple data sources
- **High performance**: cost-effective PB-level data storage and second-level response to correlation analysis of trillions of data records
- **Real-time**: real-time consolidation of service data for timely optimization and adjustment of operation decision-making

- **E-commerce**

Figure 1-5 E-commerce

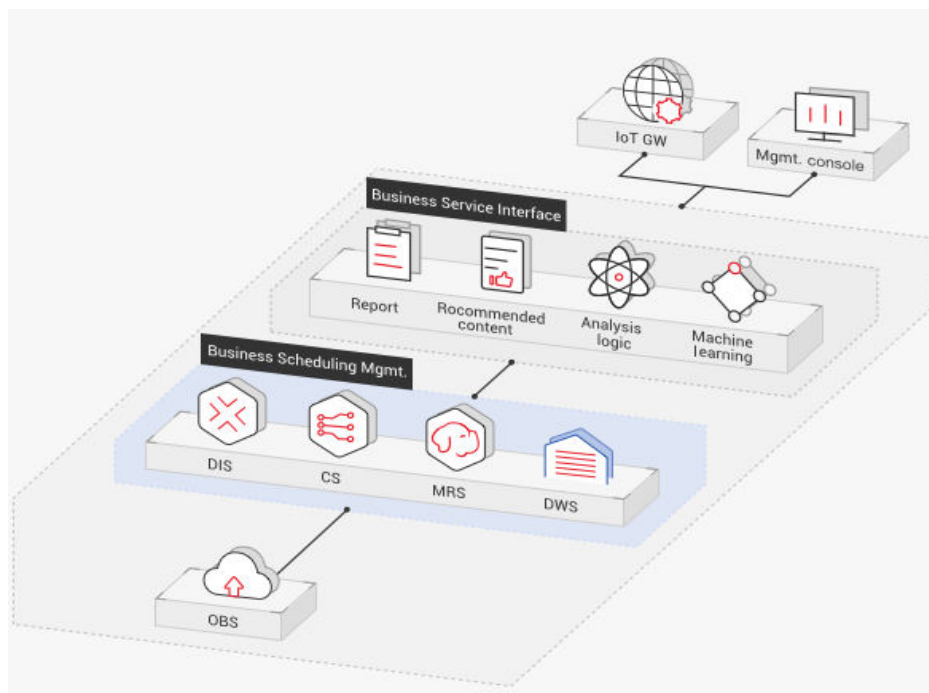


Data of online retailers is mainly used for marketing recommendation, operating and customer analysis, and full text search.

Advantages of GaussDB(DWS) are as follows:

- **Multi-dimensional analysis:** analysis from products, users, operation, regions, and more
- **Scale-out as the business grows:** on-demand cluster scale-out as the business grows
- **High reliability:** long-term stable running of the e-commerce system
- **IoT**

Figure 1-6 IoT



GaussDB(DWS) helps you analyze massive amounts of data from Internet of Things (IoT) in real time and perform optimization based on the results. It is widely used in industrial IoT, O2O service system, and IoV solutions.

Advantages of GaussDB(DWS) are as follows:

- **Device monitoring and prediction:** control, optimization, self-diagnosis, and self-healing based on data analysis, device monitoring, and behavior prediction
- **Information recommendation:** tailored recommendation based on data of users' connected devices

1.4 Functions

GaussDB(DWS) enables you to use this service through various methods, such as the GaussDB(DWS) management console, GaussDB(DWS) client, and REST APIs. This section describes the main functions of GaussDB(DWS).

Enterprise-Level Data Warehouses and Compatibility with Standard SQL

After a data warehouse cluster is created, you can use the SQL client to connect to the cluster and perform operations such as creating a database, managing the database, importing and exporting data, and querying data.

GaussDB(DWS) provides petabyte-level (PB-level) high-performance databases with the following features:

- MPP computing framework, hybrid row-column storage, and vectorized execution, enabling response to billion-level data correlation analysis within seconds

- Optimized in-memory computing based on Hash Join of Bloom Filter, improving the performance by 2 to 10 times
- Supports the symmetrically distributed, active-active multi-node cluster architecture, ensuring no SPOFs.
- Optimized communication between large-scale clusters based on telecommunication technologies, improving data transmission efficiency between compute nodes
- Cost-based intelligent optimizers, helping generate the optimal plan based on the cluster scale and data volume to improve execution efficiency

GaussDB(DWS) has comprehensive SQL capabilities:

- Supports ANSI/ISO SQL 92, SQL99, and SQL 2003 standards, stored procedures, GBK and UTF-8 character sets, and SQL standard functions and OLAP analysis functions.
- Compatible with the PostgreSQL/Oracle/Teradata/MySQL ecosystem and supports interconnection with mainstream database ETL and BI tools provided by third-party vendors.
- Supports roaring bitmaps and common functions used with them, which are widely used for user feature extraction, user profiling, and more applications in the Internet, retail, education, and gaming industries.
- List partitioning (**PARTITION BY LIST** (*partition_key,[...]*)) and range partitioning are supported.
- Read-only HDFS and OBS foreign tables in JSON file format are supported.
- Permissions on system catalogs can be granted to common users. The **VACUUM** permission can be granted separately. Roles with predefined, extensible permissions are supported, including:
 - **ALTER**, **DROP** and **VACUUM** permissions at table level
 - **ALTER** and **DROP** permissions at schema level
 - Preset roles **role_signal_backend** and **role_read_all_stats**

For details about the SQL syntax and database operation guidance, see the *Data Warehouse Service Database Development Guide*.

Cluster Management

A data warehouse cluster contains nodes with the same flavor in the same subnet. These nodes jointly provide services. GaussDB(DWS) provides a professional, efficient, and centralized management console, allowing you to quickly apply for clusters, easily manage data warehouses, and focus on data and services.

Main functions of cluster management are described as follows:

- **Creating Clusters**
To use data warehouse services on the cloud, create a GaussDB(DWS) cluster first. You can select product and node specifications to quickly create a cluster.
- **Managing Snapshots**
A snapshot is a complete backup that records point-in-time configuration data and service data of a GaussDB(DWS) cluster. A snapshot can be used to restore a cluster at a certain time. You can manually create snapshots for a cluster or enable automated snapshot creation (periodic). Automated

snapshots have a limited retention period. You can copy automatic snapshots for long-term retention.

When you restore a cluster from a snapshot, the system can restore the snapshot data to a new cluster or the original cluster.

You can delete snapshots that are no longer needed on the console to release storage space. Automated snapshots cannot be manually deleted.

- Managing nodes

You can check the nodes in a cluster, including the status, specifications, and usage of each node. To prepare for a large scale-out, you can add nodes in batches. For example, if 180 more BMS nodes are needed, add them in three batches (60 for each batch). If some nodes fail to be added, add them again. After all the 180 nodes are successfully added, use the nodes for cluster scale-out. Adding nodes does not affect cluster services.

- Scaling out clusters

As the service volume increases, the current scale of a cluster may not meet service requirements. In this case, you can scale out the cluster by adding compute nodes to it. Services are not interrupted during the scale-out. You can enable online scale-out and automatic redistribution if necessary.

- Managing redistribution

By default, redistribution is automatically started after cluster scale-out. For enhanced reliability, disable the automatic redistribution function and manually start a redistribution task after the scale-out is successful. Data redistribution can accelerate service response. Currently, offline redistribution, online redistribution, and offline scheduling are supported. The default mode is offline redistribution.

- Resource management

When multiple database users query jobs at the same time, some complex queries may occupy cluster resources for a long time, affecting the performance of other queries. For example, a group of database users continuously submit complex and time-consuming queries, while another group of users frequently submit short queries. In this case, short queries may have to wait in the queue for the time-consuming queries to complete. To improve efficiency, you can use the GaussDB(DWS) resource management function to handle such problems. You can create different resource pools for different types of services, and configure different resource ratios for these pools. Then, add database users to the corresponding pools to restrict their resource usages.

- Restarting clusters

Restarting a cluster may cause data loss in running services. If you have to restart a cluster, ensure that there is no running service and all data has been saved.

- Deleting Clusters

You can delete a cluster when you do not need it. Deleting a cluster is risky and may cause data loss. Therefore, exercise caution when performing this operation.

GaussDB(DWS) allows you to manage clusters in either of the following ways:

- Management console

Use the management console to access GaussDB(DWS) clusters. When you have registered an account, log in to the management console and choose **Data Warehouse Service**.

For more information about cluster management, see "Cluster Management" in the *Data Warehouse Service User Guide*.

- REST APIs

Use REST APIs provided by GaussDB(DWS) to manage clusters. In addition, if you need to integrate GaussDB(DWS) into a third-party system for secondary development, use APIs to access the service.

For details, see the *Data Warehouse Service API Reference*.

Diverse Data Import Modes

GaussDB(DWS) supports efficient data import from multiple data sources. The following lists typical data import modes. For details, see "Data Migration to GaussDB(DWS)" in *Data Warehouse Service (DWS) Developer Guide*.

- Importing data from OBS in parallel
- Using GDS to import data from a remote server
- Importing data from MRS to a data warehouse cluster
- Importing data from one GaussDB(DWS) cluster to another
- Using the gsql meta-command `\COPY` to import data
- Running the `COPY FROM STDIN` statement to import data
- Using Database Schema Convertor (DSC) to migrate SQL scripts
- Using `gs_dump` and `gs_dumpall` to export metadata
- Using `gs_restore` to import data

APIs

You can call standard APIs, such as JDBC and ODBC, to access databases in GaussDB(DWS) clusters.

For details, see "Using the JDBC and ODBC Drivers to Connect to a Cluster" in the *Data Warehouse Service (DWS) User Guide*.

High Reliability

- Supports instance and data redundancy, ensuring zero single points of failure (SPOF) in the entire system.
- Supports multiple data backups, and all data can be manually backed up to OBS.
- Automatically isolates the faulty node, uses the backup to restore data, and replaces the faulty node when necessary.
- Automatic snapshots work with OBS to implement intra-region disaster recovery (DR). If the production cluster fails to provide read and write services due to natural disasters in the specified region or cluster internal faults, the DR cluster becomes the production cluster to ensure service continuity.
- In the **Unbalanced** state, the number of primary instances on some nodes increases. As a result, the load pressure is high. In this case, you can perform a

primary/standby switchback for the cluster during off-peak hours to improve performance.

- If the internal IP address or EIP of a CN is used to connect to a cluster, the failure of this CN will lead to cluster connection failure. To avoid single-CN failures, GaussDB(DWS) uses Elastic Load Balance (ELB). An ELB distributes access traffic to multiple ECSs for traffic control based on forwarding policies. It improves the fault tolerance capability of application programs.
- After a cluster is created, the number of required CNs varies with service requirements. GaussDB(DWS) allows you to add or delete CNs as needed.

Security Management

- Isolates tenants and controls access permissions to protect the privacy and data security of systems and users based on the network isolation and security group rules, as well as security hardening measures.
- Supports SSL network connections, user permission management, and password management, ensuring data security at the network, management, application, and system layers.

Monitoring and Auditing

- **Monitoring Clusters**
GaussDB(DWS) integrates with Cloud Eye, allowing you to monitor compute nodes and databases in the cluster in real time. For details, see "Monitoring Clusters" in the *Data Warehouse Service (DWS) User Guide*.
- **Database Monitoring**
DMS is provided by GaussDB(DWS) to ensure the fast and stable running of databases. It collects, monitors, and analyzes the disk, network, and OS metric data used by the service database, as well as key performance metric data of cluster running. It also diagnoses database hosts, instances, and service SQL statements based on the collected metrics to expose key faults and performance problems in a database in a timely manner, and guides customers to optimize and resolve the problems. For details, see "Database Monitoring" in the *Data Warehouse Service (DWS) User Guide*.
- **Alarms**
Alarm management includes viewing and configuring alarm rules and subscribing to alarm information. Alarm rules display alarm statistics and details of the past week for users to view tenant alarms. In addition to providing a set of default GaussDB(DWS) alarm rules, this feature allows you to modify alarm thresholds based on your own services. For details, see "Alarm Management" in the *Data Warehouse Service (DWS) User Guide*.
- **Notifying Events**
GaussDB(DWS) interconnects with Simple Message Notification (SMN) so that you can subscribe to events and view events that are triggered. For details, see "Event Notifications" in the *Data Warehouse Service (DWS) User Guide*.
- **Audit Logs**
 - GaussDB(DWS) integrates with Cloud Trace Service (CTS), allowing you to audit operations performed on the management console and API invocation operations. For details, see "Viewing Audit Logs of Key Operations on the Management Console".

- GaussDB(DWS) records all SQL operations, including connection attempts, query attempts, and database changes. For details, see "Setting Database Audit Logs" in the *Data Warehouse Service (DWS) User Guide*.

Multiple Database Tools

GaussDB(DWS) provides the following self-developed tools. You can download the tool packages on the GaussDB(DWS) management console. For details about the tools, see the *Data Warehouse Service (DWS) Tool Guide*.

- **gsq**
gsq is a command line SQL client tool running on the Linux operating system. It helps connect to, operate, and maintain the database in a data warehouse cluster.
- **Data Studio**
Data Studio is a Graphical User Interface (GUI) SQL client tool running on the Windows operating system. It is used to connect to the database in a data warehouse cluster, manage the database and database objects, edit, run, and debug SQL scripts, and view the execution plans.
- **GDS**
GDS is a data service tool provided by GaussDB(DWS). It works with the foreign table mechanism to implement high-speed data import and export. The GDS tool package needs to be installed on the server where the data source file is located. This server is called the data server or the GDS server.
- **DSC SQL syntax migration tool**
The DSC is a command-line tool running on the Linux or Windows OS. It is dedicated to providing customers with simple, fast, reliable application SQL script migration services. It parses SQL scripts of source database applications by using the built-in syntax migration logic, and migrates them to be applicable to GaussDB(DWS) databases.
The DSC can migrate SQL scripts of Teradata, Oracle, Netezza, MySQL, and DB2 databases.
- **gs_dump** and **gs_dumpall**
gs_dump exports a single database or its objects. **gs_dumpall** exports all databases or global objects in a cluster.
To migrate database information, you can use a tool to import the exported metadata to a target database.
- **gs_restore**
During database migration, you can export files using **gs_dump tool** and import them to GaussDB(DWS) by using **gs_restore**. In this way, metadata, such as table definitions and database object definitions, can be imported.

1.5 Concepts

GaussDB(DWS) Management Concepts

- Cluster

A cluster is a server group that consists of multiple nodes. GaussDB(DWS) is organized using clusters. A data warehouse cluster contains nodes with the same flavor in the same subnet. These nodes work together to provide services.

- Node

A GaussDB(DWS) cluster can have 3 to 256 nodes. A hybrid data warehouse (standalone) can only have one node. Each node can store and analyze data.

- Type

You need to specify the node flavors when you create a data warehouse cluster. CPU, memory, and storage resources vary depending on node flavors.

- Snapshot

You can create snapshots to back up GaussDB(DWS) cluster data. A snapshot is retained until you delete it on the management console. Automated snapshots cannot be manually deleted. Snapshots will occupy your OBS quotas.

- Project

Projects are used to group and isolate OpenStack resources (computing resources, storage resources, and network resources). A project can be a department or a project team. Multiple projects can be created for one account.

GaussDB(DWS) Database Concepts

- Databases

A data warehouse cluster is an analysis-oriented relational database platform that supports online analysis.

- OLAP

OLAP is a major function of data warehouse clusters. It supports complex analysis, provides decision-making support tailored to analysis results, and delivers intuitive query results.

- MPP

On each node in the data warehouse cluster, memory computing and disk storage systems are independent from each other. With MPP, GaussDB(DWS) distributes service data to different nodes based on the database model and application characteristics. Nodes are connected through the network and collaboratively process computing tasks as a cluster and provide database services that meet service needs.

- Shared-Nothing Architecture

The shared-nothing architecture is a distributed computing architecture. Each node is independent so that nodes do not compete for resources, which improves work efficiency.

- Database Version

Each data warehouse cluster has a specific database version. You can check the version when creating a data warehouse cluster.

- Database Connections

You can use a client to connect to the GaussDB(DWS) cluster. The client can be used for connection on the platform and over the Internet.

- **Database User**
You can add and control users who can access the database of a data warehouse cluster by assigning specific permissions to them. The database administrator generated when you create a cluster is the default database user.

1.6 Related Services

IAM

GaussDB(DWS) uses Identity and Access Management (IAM) for authentication and authorization.

Users who have the **DWS Administrator** permissions can fully utilize GaussDB(DWS). To obtain the permissions, contact a user with the **Security Administrator** permissions or directly create a user with the **DWS Administrator** permissions. Users granted the **DWS Database Access** permissions can generate temporary database user credentials based on IAM users to connect to databases in the data warehouse clusters.

ECS

GaussDB(DWS) uses an ECS as a cluster node.

VPC

GaussDB(DWS) uses the Virtual Private Cloud (VPC) service to provide a network topology for clusters to isolate clusters and control access.

OBS

GaussDB(DWS) uses OBS to convert cluster data and external data, satisfying the requirements for secure, reliable, and cost-effective storage.

CDM

GaussDB(DWS) uses CDM to migrate data from multiple sources to GaussDB(DWS).

DRS

You can use Data Replication Service (DRS) to synchronize stream data to GaussDB(DWS) in real time.

Cloud Eye

GaussDB(DWS) uses Cloud Eye to monitor cluster performance metrics, delivering status information in a concise and efficient manner. Cloud Eye supports alarm customization so that you are notified of the exception instantly.

CTS

GaussDB(DWS) uses Cloud Trace Service (CTS) to audit your non-query operations on the management console to ensure that no invalid or unauthorized operations are performed, enhancing service security management.

LTS

GaussDB(DWS) users can view collected cluster logs or dump logs on the Log Tank Service (LTS) console.

SMN

GaussDB(DWS) uses SMN to actively push notification messages according to your event subscription requirements, so that you can immediately receive a notification when an event occurs (for example, a key cluster operation).

DNS

GaussDB(DWS) uses Domain Name Service (DNS) to provide the cluster IP addresses mapped from domain names.

ELB

With Elastic Load Balance (ELB) health checks, the CN requests of a cluster can be quickly forwarded to normal CNs. If a CN is faulty, the workload can be immediately shifted to a healthy node, minimizing cluster access faults.

1.7 GaussDB(DWS) Permissions Management

If you need to assign different permissions to employees in your enterprise to access your GaussDB(DWS) resources on , IAM is a good choice for fine-grained permissions management. IAM provides identity authentication, permissions management, and access control, helping you secure access to your resources.

With IAM, you can use your account to create IAM users for your employees, and assign permissions to the users to control their access to specific resource types. Assume you want to allow software developers in your enterprise to use GaussDB(DWS) resources, but forbid them from deleting the resources or performing any high-risk operations. To this end, you can create IAM users for these developers and grant them only the permissions required for using GaussDB(DWS) resources.

If your account does not need individual IAM users for permissions management, you may skip this section.

IAM can be used free of charge. You pay only for the resources in your account. For more information about IAM, see "Service Overview" in the *Identity and Access Management User Guide*.

Supported System Policies

By default, new IAM users do not have permissions assigned. You need to add a user to one or more groups, and attach permissions policies or roles to these

groups. Users inherit permissions from the groups to which they are added and can perform specified operations on cloud services.

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

- **Role:** IAM initially provides a coarse-grained authorization mechanism to define permissions based on users' job responsibilities. This mechanism provides only a limited number of service-level roles for authorization. When using roles to grant permissions, you must also assign other roles on which the permissions depend to take effect. However, roles are not an ideal choice for fine-grained authorization and secure access control.
- **Policies:** A type of fine-grained authorization mechanism that defines permissions required to perform operations on specific cloud resources under certain conditions. This mechanism allows for more flexible policy-based authorization, meeting requirements for secure access control. For example, you can grant GaussDB(DWS) users only the permissions for managing a certain type of GaussDB(DWS) resources.

Table 1-2 lists all the system-defined roles and policies supported by GaussDB(DWS).

Table 1-2 GaussDB(DWS) system permissions

Role/Policy Name	Description	Category	Dependencies
DWS ReadOnlyAccess	Read-only permissions for GaussDB(DWS). Users granted these permissions can only view GaussDB(DWS) data.	System-defined policy	N/A
DWS FullAccess	Database administrator permissions for GaussDB(DWS). Users granted these permissions can perform all operations on GaussDB(DWS).	System-defined policy	N/A

Role/Policy Name	Description	Category	Dependencies
DWS Administrator	<p>Database administrator permissions for GaussDB(DWS). Users granted these permissions can perform operations on all GaussDB(DWS) resources.</p> <ul style="list-style-type: none"> • Users granted permissions of the VPC Administrator policy can create VPCs and subnets. • Users granted permissions of the Cloud Eye Administrator policy can view monitoring information of data warehouse clusters. 	System-defined role	Dependent on the Tenant Guest and Server Administrator policies, which must be assigned in the same project as the DWS Administrator policy.
DWS Database Access	GaussDB(DWS) database access permission. Users with this permission can generate the temporary database user credentials based on IAM users to connect to the database in the data warehouse cluster.	System-defined role	Dependent on the DWS Administrator policy, which must be assigned in the same project as the DWS Database Access policy.

Table 1-3 lists the common operations supported by each system-defined policy or role of GaussDB(DWS). Choose appropriate policies or roles as required.

 NOTE

- If you use the EIP for the first time for a project in a region, the system prompts you to create the **DWSAccessVPC** agency to authorize GaussDB(DWS) to access VPC. After the authorization is successful, GaussDB(DWS) can switch to a healthy VM when the VM bound with the EIP is faulty.
- In addition to policy permissions, you may need to grant different operation permissions on resources to users of different roles. For details about operations, such as creating snapshots and restarting clusters, see "Syntax of Fine-Grained Permissions Policies" in *Data Warehouse Service (DWS) User Guide*.
- By default, only accounts or users with Security Administrator permissions can query and create agencies. By default, the IAM users in those accounts cannot query or create agencies. When the users use the EIP, the system makes the binding function unavailable. Contact a user with the **DWS Administrator** permissions to authorize the agency on the current page.

Table 1-3 Common operations supported by each system-defined policy or role of GaussDB(DWS)

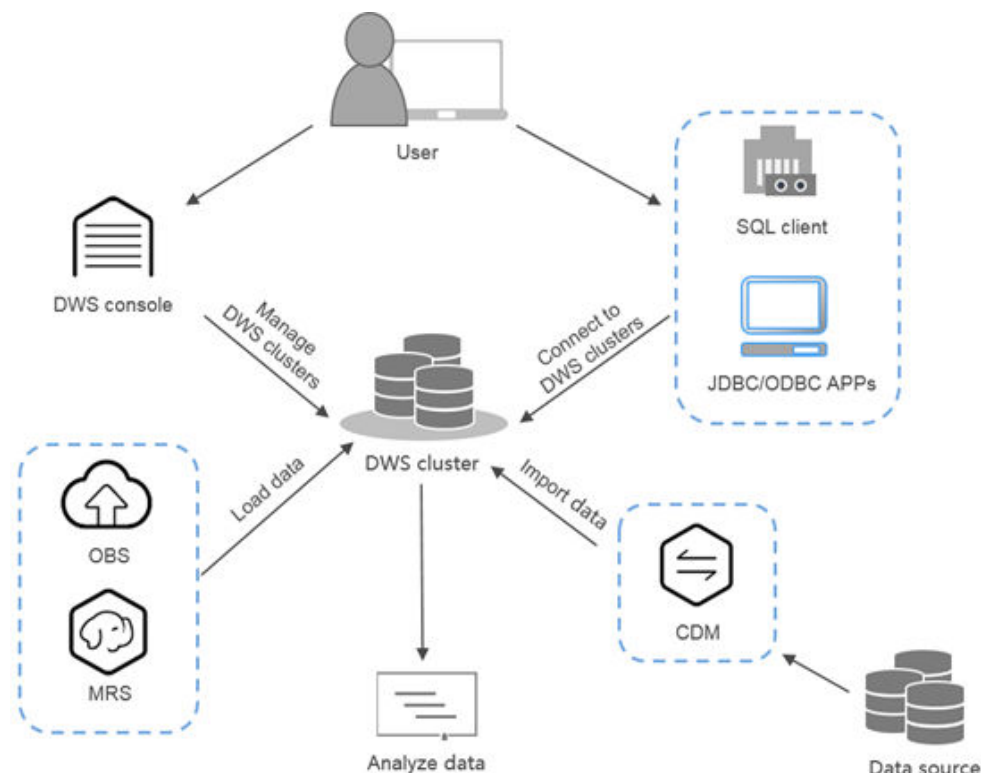
Operation	DWS FullAccess	DWS ReadOnlyAccess	DWS Administrator	DWS Database Access
Creating/Restoring clusters	√	x	√	x
Obtaining the cluster list	√	√	√	x
Obtaining the details of a cluster	√	√	√	x
Setting automated snapshot policy	√	x	√	x
Setting security parameters/parameter groups	√	x	√	x
Restarting clusters	√	x	√	x
Scaling out clusters	√	x	√	x
Resetting passwords	√	x	√	x

Operation	DWS FullAccess	DWS ReadOnlyAccess	DWS Administrator	DWS Database Access
Deleting clusters	√	x	√	x
Configuring maintenance windows	√	x	√	x
Binding EIPs	x	x	√	x
Unbinding EIPs	x	x	√	x
Creating DNS domain names	√	x	√	x
Releasing DNS domain names	√	x	√	x
Modifying DNS domain names	√	x	√	x
Creating snapshots	√	x	√	x
Obtaining the snapshot list	√	√	√	√
Deleting snapshots	√	x	√	x
Copying snapshots	√	x	√	x

1.8 GaussDB(DWS) Access

The following figure shows how to use GaussDB(DWS).

Figure 1-7 Process for using GaussDB(DWS)



Accessing a Cluster

GaussDB(DWS) provides a web-based management console and HTTPS-compliant APIs for you to manage data warehouse clusters.

NOTE

In cluster deployment, if a single node is faulty, the abnormal node is automatically skipped when GaussDB(DWS) is accessed. However, the cluster performance will be affected.

Accessing the Database in a Cluster

GaussDB(DWS) supports database access using the following methods:

- GaussDB(DWS) clients
Access the cluster database using GaussDB(DWS) clients. For details, see "Connecting to a Cluster" in the *Data Warehouse Service (DWS) User Guide*.
- JDBC and ODBC API calling

You can call standard APIs, such as JDBC and ODBC, to access databases in clusters.

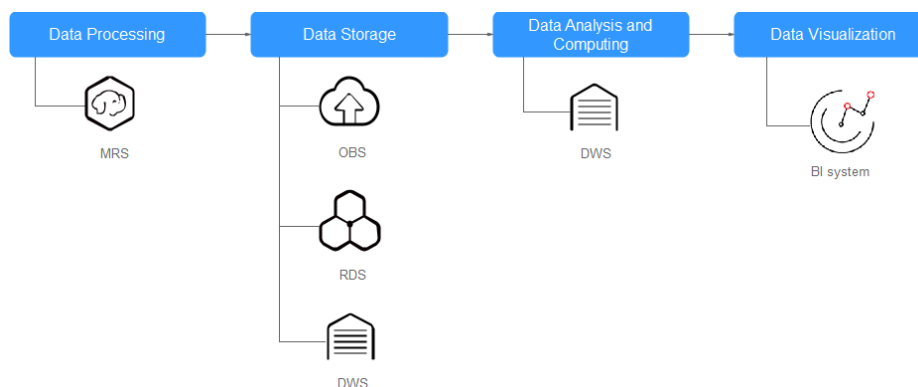
For details, see "Using the JDBC and ODBC Drivers to Connect to a Cluster" in the *Data Warehouse Service (DWS) User Guide*.

End-to-End Data Analysis Process

GaussDB(DWS) has been seamlessly integrated with other services on , helping you rapidly deploy end-to-end data analysis solutions.

The following figure shows the end-to-end data analysis process. Services in use during each process are also displayed.

Figure 1-8 End-to-end data analysis process



1.9 Restrictions

This document describes the constraints and precautions of using the key functions of GaussDB(DWS).

After creating a GaussDB(DWS) cluster, you do not need to perform basic database O&M operations, such as HA and security patch installation. However, you need to pay attention to the following:

Table 1-4 GaussDB (DWS) constraints

Item	Constraint
Creating a Cluster	<ul style="list-style-type: none"> After a cluster is created, its type cannot be changed.
Connecting to a Cluster	<ul style="list-style-type: none"> If you use a client to connect to a cluster, its VPC subnet must be the same as that of the cluster. You can manage clusters only and cannot directly access nodes in a cluster. You can use a cluster's IP address and port to access the database in the cluster.
SQL Syntax Changes	<ul style="list-style-type: none"> The hybrid data warehouse (standalone) does not support full-text search, OBS and HDFS foreign table import and export, automatic partition creation, sequence, and related functions. The hybrid data warehouse (standalone) has only one DN. Therefore, the distribution rule is ignored and cannot be modified. If you copy commands from the document to the operating environment, the text wraps automatically, causing command execution failures. To solve the problem, delete the line break. You are advised to create databases as required. Do not use the default gaussdb database of a cluster.

Item	Constraint
Changing Flavor	<ul style="list-style-type: none"> • Only clusters of version 8.1.1.300 or later support flavor change. • Currently, only offline flavor change is supported. The change takes about 10 minutes. • If your cluster is created using local disks or compute-storage integration, you cannot change the flavor of the cluster. If you need nodes with a higher flavor, create a new cluster. Currently, you can only change the flavors of cloud data warehouse clusters that use SSD cloud disks or hybrid data warehouse clusters. • Currently, changing all specifications can only be performed for standard data warehouses and hybrid data warehouses (except single-node deployment). • A cluster can have up to 240 nodes. The old and new clusters can have up to 480 nodes in total. • Disk capacity expansion can be performed only for standard data warehouses using SSD, hybrid data warehouses, or stream data warehouses. Only version 8.1.1.203 and later are supported. • Disk capacity can be expanded only if the cluster is in Available, To be restarted, Read-only, or Node fault, Unbalanced state.
Scaling Out	<ul style="list-style-type: none"> • When you scale out the standard data warehouse cluster, use the same storage specifications as the cluster. • The cluster redistribution function is supported in cluster versions. • This function can be manually enabled only when the cluster task information displays To be redistributed after scale-out. • Scale-in is supported only by clusters of version 8.1.1.300 and later. • When you scale in a standard data warehouse cluster, you can only modify the same storage specifications as used by the cluster.

Item	Constraint
Backing Up a Cluster	<ul style="list-style-type: none"> • The new GaussDB(DWS) cluster created based on the snapshot must have the same configurations as the original cluster. That is, the number and specifications of nodes, memory, and disks in the new cluster must be the same as those in the original cluster. • If you create a new cluster based on a snapshot without modifying parameters, the parameters of the new cluster will be the same as those of the snapshot. • During snapshot creation, do not perform the VACUUM FULL operation, or the cluster may become read-only. • Snapshot creation affects disk I/O performance. You are advised to create snapshots during off-peak hours. • During the snapshot creation, some intermediate files are retained, which occupy extra disk space. Therefore, avoid peak hours and ensure that the disk capacity usage is less than 70%.
Version Upgrade	<ul style="list-style-type: none"> • Cluster 8.1.1 and later versions allow users to deliver cluster upgrade operations on the console. • If the cluster is interrupted for a long time due to a node fault or system upgrade, contact technical support.
Data Migration	<ul style="list-style-type: none"> • Ensure that no Chinese characters are contained in paths used for importing data to or exporting data from OBS. • Data cannot be imported to or exported from OBS across regions. Ensure that OBS and the GaussDB(DWS) cluster are in the same region.

Item	Constraint
Failover	<ul style="list-style-type: none">• When the DR task is created, the snapshot function of the production cluster is normal, but that of the DR cluster is disabled. Besides, snapshot restoration of both clusters is disabled.• DR does not synchronize data from external sources.• DR management refers to dual-cluster DR under the same tenant.• The DR cluster and the production cluster must be logically homogeneous and in the same type and version.• The production cluster and DR cluster used for intra-region DR must be in the same VPC.• In intra-region DR, after services are switched over from the production cluster to the DR cluster, the bound ELB is automatically switched to the new production cluster. During the switchover, the connection is interrupted for a short period of time. Do not run service statements to write data during the switchover.• During intra-region DR, the EIP, intranet domain name, and connection IP address of the original production cluster are not automatically switched with the cluster switchover. The EIP, domain name, or IP address used for connection in the service system need to be switched to the new cluster.
Hot and Cold Data Management	<ul style="list-style-type: none">• The hybrid data warehouse (standalone) does not support cold and hot partition switchover.• Currently, cold and hot tables support only column-store partitioned tables of version 2.0. Foreign tables do not support cold and hot partitions.• Only hot data can be switched to cold data. Cold data cannot be switched to hot data.• A partition on a DN is either hot or cold. For a partition across DNs, its data on some DNs may be hot, and some may be cold.• Only the cold and hot switchover policies can be modified. The tablespace of cold data in cold and hot tables cannot be modified.
Stream Data Warehouse	<ul style="list-style-type: none">• Time series tables do not support UPDATE, UPSERT, primary keys, or PCKs.• To create a time series table, you must have the USAGE permission on schema cstore.• When you modify the enable_delta parameter of a time series table, other ALTER operations cannot be performed.

2 Getting Started

2.1 Step 1: Starting Preparations

This guide is an introductory tutorial that demonstrates how to create a sample GaussDB(DWS) cluster, connect to the cluster database, import the sample data from OBS, and analyze the sample data. You can use this tutorial to evaluate GaussDB(DWS).

Before creating a GaussDB(DWS) cluster, ensure that the following prerequisites are met:

- [Determining the Cluster Ports](#)

Determining the Cluster Ports

- When creating a GaussDB(DWS) cluster, you need to specify a port for SQL clients or applications to access the cluster.
- If your client is behind a firewall, you need an available port so that you can connect to the cluster and perform query and analysis from the SQL client tool.
- If you do not know an available port, contact the network administrator to specify an open port on your firewall. The ports supported by GaussDB(DWS) range from 8000 to 30000.
- After a cluster is created, its port number cannot be changed. Ensure that the port specified is available.

2.2 Step 2: Creating a Cluster

Before using GaussDB(DWS) to analyze data, create a cluster. A cluster consists of multiple nodes in the same subnet. These nodes jointly provide services. This section describes how to create a GaussDB(DWS) cluster.

Creating a Cluster

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 In the navigation pane, choose **Cluster > Dedicated Clusters**.

Step 3 On the **Dedicated Clusters** page, click **Create Cluster** in the upper right corner.

Step 4 Select the region to which the cluster to be created belongs.

- **Region:** Select the current working area of the cluster.
- **AZ:** Retain the default value.

Step 5 Configure node parameters.


- **Resource:** For example, **Standard**.
- **CPU Architecture:** Select a CPU architecture based on your requirements, for example, **x86**.
- **Node Flavor:** Retain the default value.
- **Nodes:** Retain the default value. At least **3** nodes are required.

Step 6 Configure cluster parameters.

- **Cluster Name:** Enter **dws-demo**.
- **Cluster Version:** The current cluster version is displayed and cannot be changed.
- **Default Database:** The value is **gaussdb**, which cannot be changed.
- **Administrator Account:** The default value is **dbadmin**. Use the default value. After a cluster is created, the client uses this database administrator account and its password to connect to the cluster's database.
- **Administrator Password:** Enter the password.
- **Confirm Password:** Enter the database administrator password again.
- **Database Port:** Use the default port number. This port is used by the client or application to connect to the cluster's database.

Step 7 Configure network parameters.

- **VPC:** You can select an existing VPC from the drop-down list. If no VPC has been configured, click **View VPC** to enter the VPC management console to create one, for example, **vpc-dws**. Then, go back to the page for creating a
- **Subnet:** When you create a VPC, a subnet is created by default. You can select the corresponding subnet.
- **Security Group:** Select **Automatic creation**.

cluster on the GaussDB(DWS) console, click  next to the **VPC** drop-down list, and select the new VPC.

The automatically created security group is named **GaussDB(DWS)-<Cluster name>-<GaussDB(DWS) cluster database port>**. The outbound allows all access requests, while the inbound enables only **Database Port** for access requests from clients or applications.

If you select a custom security group, add an inbound rule to it to enable **Database Port** for client hosts to access GaussDB(DWS). [Table 2-1](#) shows an example. For details about how to add an inbound rule, see "Security > Security Group > Adding a Security Group Rule" in the *Virtual Private Cloud User Guide*.

Table 2-1 Inbound rule example

Parameter	Example Value
Protocol/Application	TCP
Port	8000 NOTE Enter the value of Database Port set when creating the GaussDB(DWS) cluster. This port is used for receiving client connections to GaussDB(DWS). The default port number is 8000.
Source	Select IP address , and enter the IP address and subnet mask of the client host that accesses GaussDB(DWS), for example, 192.168.0.10/16 .

Step 8 Configure the enterprise project to which the cluster belongs. You can configure this parameter only when the Enterprise Project Management service is enabled. The default value is **default**.

An enterprise project facilitates project-level management and grouping of cloud resources and users.

You can select the default enterprise project **default** or other existing enterprise projects. To create an enterprise project, log in to the Enterprise Management console. For details, see *Enterprise Management User Guide*.

Step 9 Select **Default** for **Advanced Settings** in this example.

- **Default:** indicates that the following advanced settings use the default configurations.
 - **CNs:** Three CNs are deployed by default.
- **Custom:** Select this option to configure the following advanced parameters: **Automated Snapshot, CNs**

Step 10 Click **Create Now**. The **Confirm** page is displayed.

Step 11 Click **Submit**.

After the submission is successful, the creation starts. Click **Back to Cluster List**. The **Dedicated Clusters** page is displayed. The initial status of the cluster is **Creating**. Cluster creation takes some time. Wait for a while. Clusters in the **Available** state are ready for use.

----End

2.3 Step 3: Connecting to a Cluster

Scenario


This section describes how to use a database client to connect to the database in a GaussDB(DWS) cluster. In the following example, the Data Studio client tool is used for connection through the public network address. You can also use other SQL clients to connect to the cluster. For more connection methods, see .

1. Obtain the name, username, and password of the database to be connected.
If you use the client to connect to the cluster for the first time, use the administrator username and password set in [Step 2: Creating a Cluster](#) to connect to the default database **gaussdb**.
2. [Obtaining the Public Network Address of the Cluster](#): Connect to the cluster database using the public network address.
3. [Connecting to the Cluster Database Using Data Studio](#): Download and configure the Data Studio client and connect to the cluster database.

Obtaining the Public Network Address of the Cluster

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 In the navigation pane on the left, choose **Clusters > Dedicated Clusters**.

Step 3 In the cluster list, select a created cluster (for example, **dws-demo**) and click  next to the cluster name to obtain the public network address.

The public network address will be used in [Connecting to the Cluster Database Using Data Studio](#).

----End

Connecting to the Cluster Database Using Data Studio

Step 1 GaussDB(DWS) provides a Windows-based Data Studio GUI client. The tool depends on JDK, so you must install Java 1.8.0_141 or later on the client host.

In the Windows operating system, you can download the required JDK version from the official website of JDK, and install it by following the installation guide.

Step 2 Log in to the GaussDB(DWS) management console.

Step 3 Click **Client Connections**.

Step 4 On the **Download Client and Driver** page, download **Data Studio GUI Client**.

- Select **Windows x86** or **Windows x64** based on the operating system type and click **Download** to download the Data Studio tool matching the current cluster version.

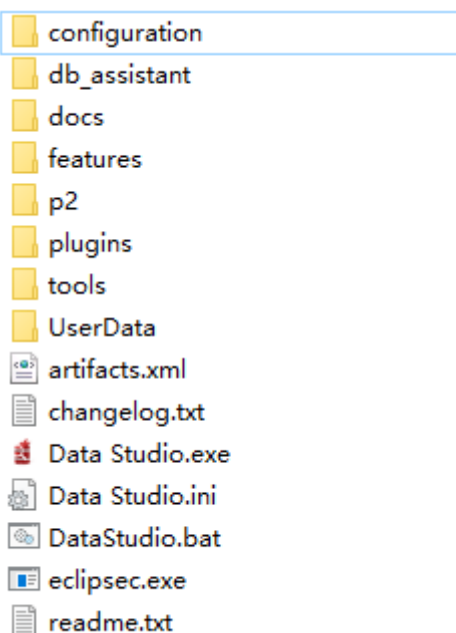
If clusters of different versions are available, you will download the Data Studio tool matching the earliest cluster version after clicking **Download**. If there is no cluster, you will download the Data Studio tool of the earliest version after clicking **Download**. GaussDB(DWS) clusters are compatible with earlier versions of Data Studio tools.

- Click **Historical Version** to download the corresponding Data Studio version. You are advised to download the Data Studio based on the cluster version.

Step 5 Decompress the downloaded client software package (32-bit or 64-bit) to the installation directory.

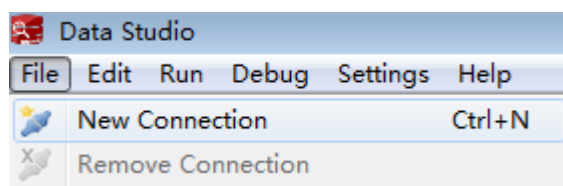
Step 6 Open the installation directory and double-click **Data Studio.exe** to start the Data Studio client. See [Figure 2-1](#).

Figure 2-1 Starting the client



Step 7 Choose **File > New Connection** from the main menu. See [Figure 2-2](#).

Figure 2-2 Creating a connection



Step 8 In the displayed **New Database Connection** window, enter the connection parameters.

Table 2-2 Connection parameters

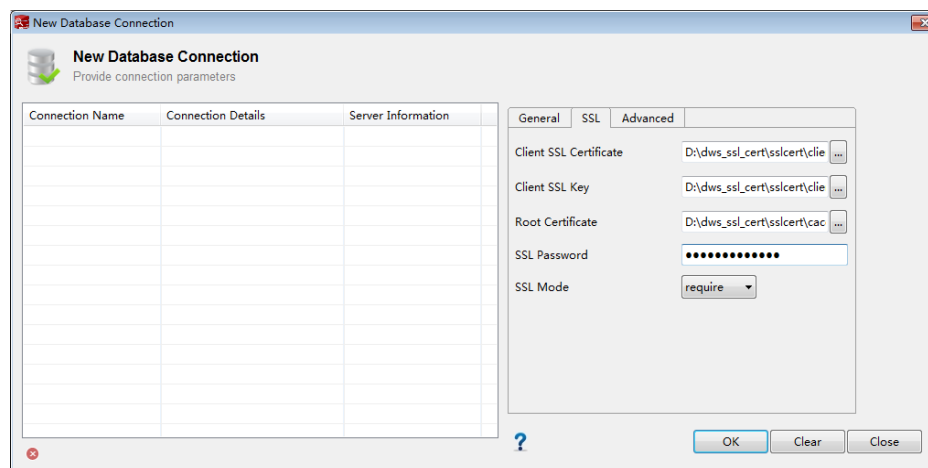
Parameter	Description	Example
Database Type	Select GaussDB A .	GaussDB A
Connection Name	Name of a connection	dws-demo
Host	IP address (IPv4) of the cluster to be connected	-
Host Port	Database port	8000
Database Name	Database name	gaussdb
User Name	Username for connecting to the database	-

Parameter	Description	Example
Password	Password for logging in to the database to be connected	-
Save Password	Select an option from the drop-down list: <ul style="list-style-type: none"> ● Current Session Only: The password is saved only in the current session. ● Do Not Save: The password is not saved. 	-
Enable SSL	If Enable SSL is selected, the client can use SSL to encrypt connections. The SSL mode is more secure than common modes, so you are advised to enable SSL connection.	-

If **Enable SSL** is selected, download and decompress the SSL certificate. For details, see . Click the **SSL** tab and configure the following parameters:

Table 2-3 Configuring SSL parameters

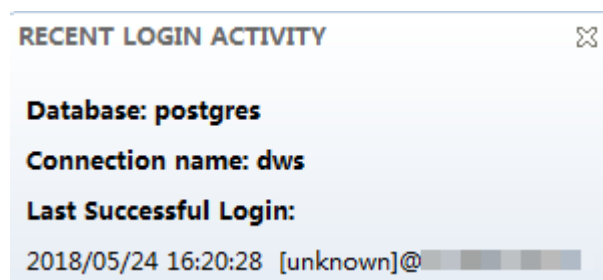
Parameter	Description
Client SSL Certificate	Select the sslcert\client.crt file in the decompressed SSL certificate directory.
Client SSL Key	Only the PK8 format is supported. Select the sslcert\client.key.pk8 file in the directory where the SSL certificate is decompressed.
Root Certificate	When SSL Mode is set to verify-ca , the root certificate must be configured. Select the sslcert\cacert.pem file in the decompressed SSL certificate directory.
SSL Password	Set the password for the client SSL key in PK8 format.
SSL Mode	Supported SSL modes include: <ul style="list-style-type: none"> ● require ● verify-ca GaussDB(DWS) does not support the verify-full mode.

Figure 2-3 Configuring SSL parameters

Step 9 Click **OK** to establish the database connection.

If SSL is enabled, click **Continue** in the displayed **Connection Security Alert** dialog box.

After the login is successful, the **RECENT LOGIN ACTIVITY** dialog box is displayed, indicating that Data Studio is connected to the database. You can run the SQL statement in the **SQL Terminal** window on the Data Studio page.

Figure 2-4 Successful login

For details about how to use other functions of Data Studio, press **F1** to view the Data Studio user manual.

----End

2.4 Step 4: Viewing Other Documents and Deleting Resources

Viewing Other Relevant Documents

After performing the steps in preceding sections, you can refer to the documentation listed as follows for more information about GaussDB(DWS):

- *Data Warehouse Service User Guide*: This guide provides detailed information about the concepts and operations related to creating, managing, monitoring, and connecting to clusters.

- *Data Warehouse Service Database Development Guide*: This guide provides comprehensive and detailed information for database developers to understand how to build, manage, and query GaussDB(DWS) databases, including SQL syntax, user management, and data import and export.

Deleting Resources

After performing the steps in "Getting Started," if you do not need to use the sample data, clusters, ECSs, or VPCs, delete the resources so that your service quotas will not be wasted or occupied.

Step 1 Delete a GaussDB(DWS) cluster.

On the GaussDB(DWS) management console, click **Cluster > Dedicated Cluster**, locate the row that contains **dws-demo** in the cluster list, and choose **More > Delete**. In the dialog box that is displayed, select **Release the EIP bound with the cluster** and click **OK**.

If the cluster to be deleted uses an automatically created security group that is not used by other clusters, the security group is automatically deleted when the cluster is deleted.

Step 2 Delete a subnet. Before deleting the subnet, ensure that it is not bound to other resources.

Log in to the VPC management console. In the navigation tree on the left, click **Virtual Private Cloud**. In the VPC list, click **vpc-dws**. In the subnet list, locate the row that contains **subnet-dws** and click **Delete**.

Step 3 Delete a VPC. Before deleting the VPC, ensure that it is not bound to other resources.

Log in to the VPC management console, locate the row that contains **vpc-dws** in the VPC list, and click **Delete**.

For details, see "VPC and Subnet > Deleting a VPC" in the *Virtual Private Cloud User Guide*.

----End

3 Process for Using GaussDB(DWS)

GaussDB(DWS) is an online data processing database that uses the infrastructure to provide scalable, fully-managed, and out-of-the-box analytic database service, freeing you from complex database management and monitoring. It is a native cloud service based on the converged data warehouse GaussDB, and is fully compatible with the standard ANSI SQL 99 and SQL 2003, as well as the PostgreSQL and Oracle ecosystems. GaussDB(DWS) provides competitive solutions for PB-level big data analysis in various industries.

GaussDB(DWS) provides an easy-to-use management console, allowing you to quickly create clusters and easily manage data warehouses.

Process Description

Figure 3-1 Process for using GaussDB(DWS)

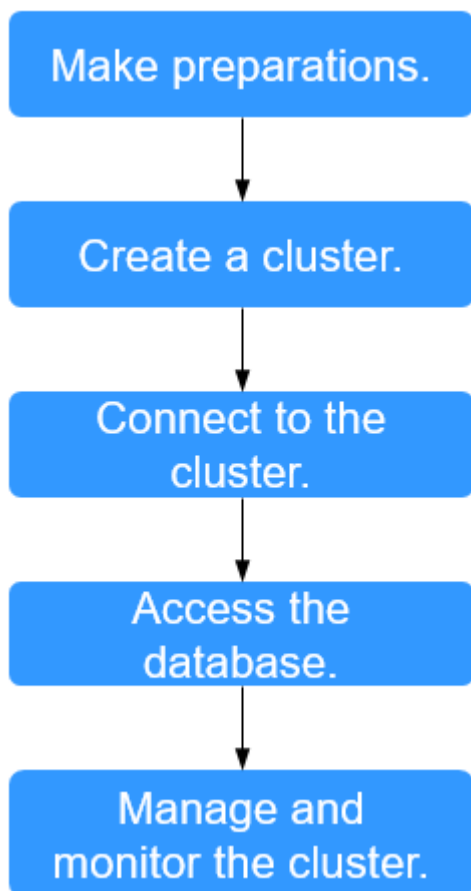


Table 3-1 Process description

Process	Task	Description	Operation Instruction
Make preparations.	-	Before using GaussDB(DWS), select an open port on your firewall as the database port of your data warehouse cluster.	Preparations
Create a cluster.	-	Create a cluster before using GaussDB(DWS) to execute data analysis tasks. A GaussDB(DWS) cluster contains nodes in the same subnet. These nodes jointly provide services. During cluster creation, the system creates a default database.	Creating a Cluster

Process	Task	Description	Operation Instruction
Connect to the cluster.	-	After the data warehouse cluster is successfully created, use the SQL client tool or a third-party driver such as JDBC or ODBC to connect to the database in the cluster. You can download the SQL client tool and JDBC/ODBC driver on the Client Connections page of the GaussDB(DWS) management console.	Methods of Connecting to a Cluster
Access the database.	-	After connecting to the cluster, you can create and manage databases, manage users and permissions, import and export data, and query and analyze data.	Data Warehouse Service (DWS) Developer Guide
Manage and monitor the cluster.	Manage the cluster.	View the cluster status, modify cluster configurations, add cluster tags, and scale out, restart, and delete the cluster.	Managing clusters
	Manage the snapshot.	Create snapshots to back up and restore the cluster.	Snapshots
	Perform O&M and monitoring.	View the running status and performance of the cluster through monitoring, log auditing, event notification, and resource load management.	<ul style="list-style-type: none"> • Monitoring Clusters Using Cloud Eye • Audit Logs • Resource Management

4 Preparations

Before using GaussDB(DWS), make the following preparations:

- **Determining the Cluster Ports**

Determining the Cluster Ports

- When creating a GaussDB(DWS) cluster, you need to specify a port for SQL clients or applications to access the cluster.
- If your client is behind a firewall, you need an available port so that you can connect to the cluster and perform query and analysis from the SQL client tool.
- If you do not know an available port, contact the network administrator to specify an open port on your firewall. The ports supported by GaussDB(DWS) range from 8000 to 30000.
- After a cluster is created, its port number cannot be changed. Ensure that the port specified is available.

5 Creating or Deleting a Cluster

5.1 Accessing the GaussDB(DWS) Management Console

Scenario

This section describes how to log in to the GaussDB(DWS) management console and use GaussDB(DWS).

Procedure

Step 1 Log in to the management console.

Step 2 Choose > **GaussDB(DWS)** to enter the GaussDB(DWS) management console.

----End

5.2 Creating a Cluster

To use GaussDB(DWS), create a data warehouse cluster first.

This section describes how to create a data warehouse cluster on the GaussDB(DWS) management console.

Preparations Before Creating a Cluster

- You have evaluated the flavor of cluster nodes.
You can select the number of nodes by data volume, service load, and performance. More nodes bring you stronger storage and compute capabilities.
When first using GaussDB(DWS), you can create a cluster with a smaller flavor. Then, you can adjust the cluster scale and node flavor based on the data volume and service load changes without interrupting services. For details, see [Scaling Out a Cluster](#).

- Determine the number of nodes that can be used by users.

The number of nodes that can be used by users must meet the following requirements. Otherwise, the system displays a message indicating that the cluster cannot be created.

The number of nodes that can be used by a user depends on the product type you select. A hybrid data warehouse cluster (standalone mode) has only one node. For other types of clusters, the number of nodes can be greater than or equal to 3. You can view the number of available nodes on the **Clusters > Dedicated Clusters** page.

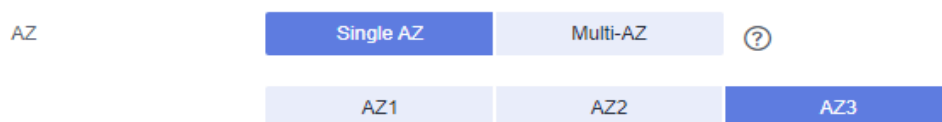
Creating a Cluster

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane on the left, choose **Clusters > Dedicated Clusters**.
- Step 3** On the **Dedicated Clusters** page, click **Create GaussDB(DWS) Cluster**.
- Step 4** Select **Region**.

Table 5-1 Region parameters

Parameter	Description	Example Value
Region	Select the actual region where the cluster nodes run.	-
AZ	Select an AZ associated with the cluster region. For more information, see Regions and AZs .	-

- Step 5** Select an AZ. You can select **Single AZ** or **Multi-AZ** as required.



NOTE

- Multi-AZ clusters are supported only by clusters of version 8.2.0.100 or later.
- The **Multi-AZ** option is displayed only if the number of AZs in the selected region is greater than or equal to 3. If this condition is not met, only a single-AZ cluster can be created.
- For a multi-AZ cluster, only three AZs can be selected at a time so far. Server nodes are evenly distributed among the three AZs.
- The multi-AZ cluster supports only GaussDB(DWS) 2.0 standard data warehouses.
- The numbers of nodes in a multi-AZ cluster must be a multiple of 3.
- In a multi-AZ cluster, the number of DNs must be less than or equal to 2.

- Step 6** Configure **Resource**, **CPU Architecture**, and **Node Flavor**.

 NOTE

- The number of nodes in a new cluster cannot exceed the quota that can be used by a user or 256. If the node quota is insufficient, click **Increase quota** to submit a service ticket and apply for higher node quota.

Table 5-2 Node configuration parameters


Parameter	Description	Example Value
Resource	Product type. It can be: <ul style="list-style-type: none">• Standard data warehouse: It can analyze hot and cold data and is highly cost-effective. Its storage and computing resources are not limited, and can be elastically scaled and billed per use. It is suitable for the converged analysis that requires integrated databases, warehouses, marts, and lakes. It is most suitable for OLAP workloads.	Standard
Compute Resource	It can be: <ul style="list-style-type: none">• ESC: Scalable, reliable, and high-throughput virtual block storage is provided in a distributed architecture. This ensures that data can be quickly migrated and restored if any data replica is unavailable, preventing data from being lost because of a single hardware fault. Backup and restoration can be performed on ECSs and EVS disks. You can configure automatic backup policies for them.	-
Storage Type	It can be: <ul style="list-style-type: none">• Cloud SSD	-
CPU Architecture	The CPU architecture includes: <ul style="list-style-type: none">• x86	-
Node Flavor	Select the desired node flavor based on service requirements. Each node flavor displays the vCPU, memory, and recommended application scenario.	dws2.m6.4xlarge.8


Parameter	Description	Example Value
Hot storage	Available storage capacity of each node. NOTE <ul style="list-style-type: none">The storage capacity you apply for has the necessary file system overhead, which includes index nodes and the space required for database running. The storage space must be an integer multiple of 100.200 GB per node is the actual storage capacity for service data. For example, if the number of nodes is set to 3, the total resource capacity is 600 GB.By default, tablespaces are automatically created when you configure cold and hot data storage. You do not need to manually create tablespaces. This feature is supported only in clusters of 8.1.3 and later versions.	-
Cold storage	You are advised to store cold data in OBS.	-
Nodes	Specify the number of nodes in the cluster. The number of nodes ranges from 3 to 256.	3
Total	Displays the total capacity of a cluster. The storage capacity of each flavor is the actual database space used for storing data. The displayed storage capacity has deducted the disk space consumed by backups and RAIDs.	-

Step 7 Click **Next: Configure Network**.

Step 8 Configure the network.

Table 5-3 Network parameters

Parameter	Description	Example Value
VPC	<p>Specify a virtual private network for nodes in a cluster to isolate networks of different services.</p> <p>If you create a data warehouse cluster for the first time and have not configured the VPC, click View VPC. On the VPC management console that is displayed, create a VPC that satisfies your needs.</p> <p>For details about how to create a VPC, see "VPC and Subnet > Creating a VPC" in the <i>Virtual Private Cloud User Guide</i>.</p> <p>After selecting a VPC from the drop-down list, click View VPC to enter the VPC management console and view the detailed information about the VPC.</p> <p>You can click  to refresh the options in the VPC drop-down list.</p>	vpc-dws
Subnet	<p>Specify a VPC subnet.</p> <p>A subnet provides dedicated network resources that are isolated from other networks, improving network security.</p> <p>NOTE</p> <p>After a cluster is created, the subnet cannot be modified. If you need to modify the subnet, you can restore the snapshot of the cluster to a new cluster. The data of the new cluster is the same as that of the old cluster, and the subnet can be modified when the new cluster is created.</p>	subnet-dws

Parameter	Description	Example Value
Security Group	<p>Specify a VPC security group.</p> <p>A security group restricts access rules to enhance security when GaussDB(DWS) and other services access each other.</p> <ul style="list-style-type: none"> Automatic creation If Automatic creation is selected, the system automatically creates a default security group. This option is selected by default. The rule of the default security group is as follows: The outbound allows all access requests, while the inbound is open only to the database port that you set to connect to the GaussDB(DWS) cluster. The format of the default security group name is <code>dws-<i>Cluster_name</i>-<i>Cluster_database_port</i></code>, for example, dws-dws-demo-8000. <p>NOTE If the quotas of the security group and the security group rule are insufficient, an error message will be displayed after you submit the cluster creation application. Select an existing group and retry.</p> <ul style="list-style-type: none"> Manual creation You can also log in to the VPC management console to manually create a security group. Then, go back to the page for creating data warehouse clusters, click the  button next to the Security Group drop-down list to refresh the page, and select the new security group. To enable the GaussDB(DWS) client to connect to the cluster, you need to add an inbound rule to the new security group to grant the access permission to the database port of the GaussDB(DWS) cluster. The following is an example of an inbound rule.. <ul style="list-style-type: none"> - Protocol: TCP - Port: 8000. Use the database port set when creating the GaussDB(DWS) cluster. This port is used for receiving client connections to GaussDB(DWS). - Source: Select IP address and use the host IP address of the client host, for example, 192.168.0.10/32. 	Automatic creation

Parameter	Description	Example Value
	<p>The security group of a cluster cannot be changed but can be modified. For details, see Modifying a Security Group.</p>	
Public Network Access	<p>Specify whether users can use a client to connect to a cluster's database over the Internet. The following methods are supported:</p> <ul style="list-style-type: none">• Do not use: The EIP is not required.• : Users specify the bandwidth of the EIP and the system automatically assigns an EIP that exclusively uses bandwidth to each cluster so that users can use the EIP to access the cluster over the Internet. The bandwidth name of an automatically assigned EIP starts with the cluster name.• Specify: A specified EIP is bound to the cluster. If no available EIPs are displayed in the drop-down list, click Create EIP to go to the Elastic IP page and create an EIP that satisfies your needs. You can set the bandwidth as needed. <p>NOTE</p> <ul style="list-style-type: none">• If you use the EIP binding function for the first time in each project of each region, the system prompts you to create the DWSAccessVPC agency to authorize GaussDB(DWS) to access VPC. After the authorization is successful, GaussDB(DWS) can switch to a healthy VM when the VM bound with the EIP becomes faulty.• By default, only accounts or users with Security Administrator permissions can query and create agencies. By default, the IAM users in those accounts cannot query or create agencies. When the users use the EIP, the system makes the binding function unavailable. Contact a user with the DWS Administrator permissions to authorize the agency on the current page.	Automatically assign
ELB	<p>Specifies whether ELB is bound. With ELB health checks, CN requests of a cluster can be quickly forwarded to normal CNs. If a CN is faulty, the workload can be immediately shifted to a healthy node, minimizing cluster access faults.</p> <ul style="list-style-type: none">• Do not use: The load balancer is not used.• Specify: Specify an ELB to be bound to the cluster. If no available load balancers are displayed in the drop-down list, click Create ELB to go to the Elastic Load Balance page and create a load balancer as needed.	Specify

Parameter	Description	Example Value
Bandwidth	you need to specify the bandwidth of the EIP, which ranges from 1 Mbit/s to 100 Mbit/s.	50Mbit/s

Step 9 Click **Next: Configure Advanced Settings**.

Step 10 Configure cluster parameters.

Table 5-4 Cluster parameters

Parameter	Description	Example Value
Cluster Name	Set the name of the data warehouse cluster. The cluster name contains 4 to 64 case-insensitive characters and must start with a letter. Only letters, digits, hyphens (-), and underscores (_) are allowed. NOTE Only in 8.3.1 and later versions, you can change the cluster name after a cluster is created.	dws-demo
Cluster Version	Displays the version of the database instance installed in the cluster. The figure is for reference only.	-
Default Database	The default database name of the cluster is gaussdb . NOTE This name cannot be changed.	gaussdb
Administrator or Account	Set the database administrator name. The administrator username must: <ul style="list-style-type: none">• Consist of lowercase letters, digits, or underscores.• Start with a lowercase letter or an underscore.• Contain 6 to 64 characters.• Cannot be a keyword of the GaussDB(DWS) database. For details about the keywords of the GaussDB(DWS) database, see "SQL Reference > Keyword" in the <i>Data Warehouse Service (DWS) Developer Guide</i>.	dbadmin

Parameter	Description	Example Value
Administrator Password	<p>Set the password of the database administrator account.</p> <p>The password complexity requirements are as follows:</p> <ul style="list-style-type: none">• Consists of 12 to 32 characters.• Cannot be the username or the username spelled backwards.• Must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters (~!`?.,;:_'"(){}[]/<>@#%^&*+ \=)• Passes the weak password check. <p>NOTE Change the password regularly and keep it secure.</p>	-
Confirm Password	Enter the database administrator password again.	-
Database Port	<p>Specify the port used when the client or application connects to the database in the cluster.</p> <p>The port number ranges from 8000 to 30000.</p> <p>NOTE The database port of a created cluster cannot be changed. You can specify the database port only when creating a cluster.</p>	8000

Step 11 Configure the enterprise project to which the cluster belongs. You can configure this parameter only when the Enterprise Project Management service is enabled. The default value is **default**.

An enterprise project facilitates project-level management and grouping of cloud resources and users.

You can select the default enterprise project **default** or other existing enterprise projects.

Step 12 Configure advanced settings. Select **Default** to keep the default values of the advanced parameters. You can also select **Custom** to modify the values.

- **CNs**

CNs receive access requests from the clients and return the execution results. In addition, a CN splits and distributes tasks to the DNs for parallel execution. The value ranges from 3 to the number of cluster nodes. The maximum value is **20** and the default value is **3**. In a large-scale cluster, you are advised to deploy multiple CNs.

- **Tag**

A tag is a key-value pair used to identify a cluster. For details about the keys and values, see [Table 5-5](#). By default, no tag is added to the cluster.

For more information about tags, see [Overview](#).

Table 5-5 Tag parameters

Parameter	Description	Example Value
Key	<p>You can select:</p> <ul style="list-style-type: none">Select a predefined tag key or an existing resource tag key from the drop-down list of the text box. <p>NOTE To add a predefined tag, you need to create one on TMS and select it from the drop-down list of Tag key. You can click View predefined tags to enter the Predefined Tags page of TMS. Then, click Create Tag to create a predefined tag. For more information, see Management > Predefined Tags > Creating Predefined Tags in the <i>Tag Management Service User Guide</i>.</p> <ul style="list-style-type: none">Enter a tag key in the text box. A tag key can contain a maximum of 36 characters. It cannot be an empty string or start or end with a space. The value cannot contain the following characters: =*<>\\, / <p>NOTE A key must be unique in a given cluster.</p>	key01
Value	<p>You can select:</p> <ul style="list-style-type: none">Select a predefined tag value or resource tag value from the drop-down list of the text box.Enter a tag value in the text box. A tag value can contain a maximum of 43 characters, which can be an empty string. It cannot start or end with a space. The value cannot contain the following characters: =*<>\\, /	value01

- **Encrypt DataStore**



indicates that database encryption is disabled. This function is disabled by default.



indicates that database encryption is enabled. After this function is enabled, Key Management Service (KMS) encrypts the cluster and the cluster's snapshot data.

NOTICE

Only users with the Tenant Admin permission can view and toggle the **Encrypt DataStore** switch.

When you enable database encryption for each project in each region for the first time, the system displays a **Create Agency** dialog box. Click **Yes** to create **DWSAccessKMS** to authorize GaussDB(DWS) to access KMS. If you click **No**, the encryption function is not enabled. Select the created KMS key from the **KMS Key Name** drop-down list.

By default, only Huawei Cloud accounts or users with Security Administrator permissions can query and create agencies. IAM users under an account do not have the permission to query or create agencies by default. Contact a user with that permission and complete the authorization on the current page.

NOTICE

- The database encryption function cannot be disabled once it is enabled.
 - After **Encrypt DataStore** is enabled, the key cannot be disabled, deleted, or frozen when being used. Otherwise, the cluster becomes abnormal and the database becomes unavailable.
 - Snapshots created after the database encryption function is enabled cannot be restored using open APIs.
-

Step 13 Click **Next: Confirm**.

Step 14 Click **Create Now**.

After the submission is successful, the creation starts. Click **Back to Cluster List** to go back to the **Dedicated Clusters** page. The initial status of the cluster is **Creating**. Cluster creation takes some time. Clusters in the **Available** state are ready for use.

----End

5.3 Deleting a Cluster


If you do not need to use a cluster, perform the operations in this section to delete it.

Impact on the System

Deleted clusters cannot be recovered. Additionally, you cannot access user data and automated snapshots in a deleted cluster because the data and snapshots are automatically deleted. If you delete a cluster, its manual snapshots will not be deleted.

Deleting a Cluster

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 Click  in the upper left corner of the management console to select a region.

Step 3 On the **Clusters > Dedicated Clusters** page, locate the cluster to be deleted.

Step 4 In the row of a cluster, choose **More > Delete**.

Step 5 In the displayed dialog box, confirm the deletion. You can determine whether to perform the following operations:

- Create a snapshot for the cluster.

If the cluster status is normal, you can click **Create Snapshot**. In the dialog box that is displayed, enter the snapshot name and click **OK** to create a snapshot for the cluster to be deleted. After the snapshot is created, go back to the **Clusters > Dedicated Clusters** page to delete the cluster.

- Resource

- Release the EIP bound to the cluster.

If an EIP is bound to the cluster, you are advised to select **EIP** to release the EIP.

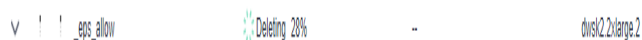
- Automated snapshots

- Manual snapshots

If a manual snapshot has been created, you can select **Manual Snapshot** to delete it.

Step 6 After confirming that the information is correct, enter **DELETE** and click **OK** to delete the cluster. The cluster status in the cluster list will change to **Deleting**, and the cluster deletion progress will be displayed.

If a cluster to be deleted uses an automatically created security group that is not used by other clusters, the security group will be automatically deleted with the cluster.

A screenshot of a cluster deletion progress bar. The progress bar is partially filled with a blue color, indicating the deletion progress. The text 'Deleting 28%' is displayed above the progress bar. To the left of the progress bar, there is a small icon of a cluster and the text '_eps_allow'. To the right of the progress bar, there is a small icon of a cluster and the text 'disk2.2xlarge.2'.

----End

6 Cluster Connection

6.1 Methods of Connecting to a Cluster

If you have created a GaussDB(DWS) cluster, you can use the SQL client tool or a third-party driver such as JDBC or ODBC to connect to the cluster and access the database in the cluster.

The procedure for connecting to a cluster is as follows:

1. [Obtaining the Cluster Connection Address](#)
2. If SSL encryption is used, perform the operations in [Establishing Secure TCP/IP Connections in SSL Mode](#).
3. Connect to the cluster and access the database in the cluster. You can choose any of the following methods to connect to a cluster:
 - Use the SQL client tool to connect to the cluster.
 - [Using the Linux gsql Client to Connect to a Cluster](#)
 - [Using the Windows gsql Client to Connect to a Cluster](#)
 - [Using the Data Studio GUI Client to Connect to a Cluster](#)
 - Use a JDBC, psycopg2, or PyGreSQL driver to connect to the cluster.
 - [Using JDBC to Connect to a Cluster](#)
 - [Using ODBC to Connect to a Cluster](#)
 - [Using the Third-Party Function Library psycopg2 of Python to Connect to a Cluster](#)
 - [Using the Python Library PyGreSQL to Connect to a Cluster](#)

6.2 Obtaining the Cluster Connection Address

Scenario

You can access GaussDB(DWS) clusters by different methods and the connection address of each connection method varies. This section describes how to view and obtain the private network address on the Huawei Cloud platform, public network address on the Internet, and JDBC connection strings.

To obtain the cluster connection address, use either of the following methods:

- [Obtaining the cluster connection address on the Client Connections Page](#)
- [Obtaining the Cluster Access Addresses on the Cluster Information Page](#)

Obtaining the cluster connection address on the Client Connections Page

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 In the navigation tree on the left, choose **Client Connections**.

Step 3 In the **Data Warehouse Connection Information** area, select an available cluster.
You can only select clusters in the **Available** state.

Step 4 View and obtain the cluster connection information.

- **Private Network IP Address**
- **Public Network IP Address**
- **ELB Address**
- **JDBC URL (Private Network)**
- **JDBC URL (Public Network)**
- **ODBC URL**

NOTE

- If no EIP is automatically assigned during cluster creation, **Public Network Address** is empty. If you want to use a public network address (consisting of an EIP and the database port) to access the cluster from the Internet, click **Bind EIP** to bind one.
- If an EIP is bound during cluster creation but you do not want to use the public network address to access the cluster, click **Unbind EIP** to unbind the EIP. After the EIP is unbound, **Public Network Address** is empty.

----End

Obtaining the Cluster Access Addresses on the Cluster Information Page

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 In the navigation pane on the left, choose **Clusters > Dedicated Clusters**.

Step 3 In the cluster list, click the name of the target cluster. The **Cluster Information** page is displayed.

Step 4 In the **Connection** area, view and obtain the cluster's access address information, including the private network address and public network address.

----End

6.3 Using the Data Studio GUI Client to Connect to a Cluster

Data Studio is a SQL client tool running on the Windows operating system. It provides various GUIs for you to manage databases and database objects, as well as edit, run, and debug SQL scripts, and view execution plans. Download the Data Studio software package from the GaussDB(DWS) management console. The package can be used without installation after being decompressed.

Data Studio versions include **Windows x86** (32-bit Windows system) and **Windows x64** (64-bit Windows system).

Preparations Before Connecting to a Cluster

- You have obtained the administrator username and password for logging in to the database in the data warehouse cluster.
- You have obtained the public network address, including the IP address and port number in the data warehouse cluster. For details, see [Obtaining the Cluster Connection Address](#).
- You have configured the security group of the GaussDB(DWS) cluster and added an inbound rule that allows users' IP addresses to access ports using the TCP.

For details, see "Security > Security Group > Adding a Security Group Rule" in the *Virtual Private Cloud User Guide*.

Connecting to the Cluster Database Using Data Studio

Step 1 GaussDB(DWS) provides a Windows-based Data Studio client and the tool depends on the JDK. You need to install the JDK on the client host first.

NOTICE

Only JDK 1.8 is supported.

In the Windows operating system, you can download the required JDK version from the official website of JDK, and install it by following the installation guide.

Step 2 Log in to the GaussDB(DWS) management console.

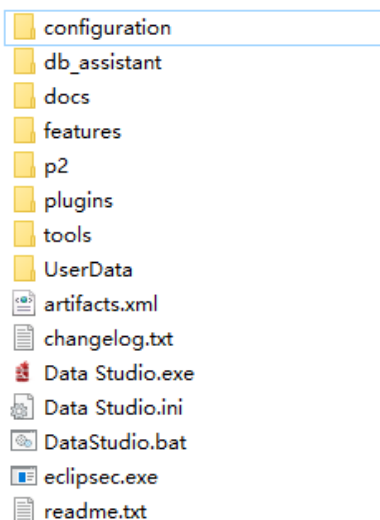
Step 3 Click **Client Connections**.

Step 4 On the **Download Client and Driver** page, download **Data Studio GUI Client**.

- Select **Windows x86** or **Windows x64** based on the operating system type and click **Download** to download the Data Studio tool matching the current cluster version.
- Click **Historical Version** to download the corresponding Data Studio version. You are advised to download the Data Studio based on the cluster version.

- Step 5** Decompress the downloaded client software package (32-bit or 64-bit) to the installation directory.
- Step 6** Open the installation directory and double-click **Data Studio.exe** to start the Data Studio client. See [Figure 6-1](#).

Figure 6-1 Starting the client

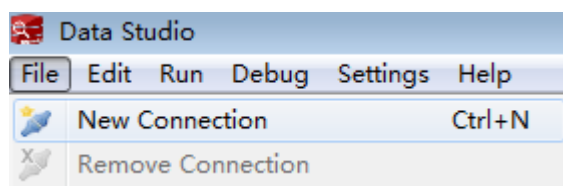


NOTE

If your computer blocks the running of the application, you can unlock the **Data Studio.exe** file to start the application.

- Step 7** Choose **File > New Connection** from the main menu. See [Figure 6-2](#).

Figure 6-2 Creating a connection



- Step 8** In the displayed **New Database Connection** window, enter the connection parameters.

Table 6-1 Connection parameters

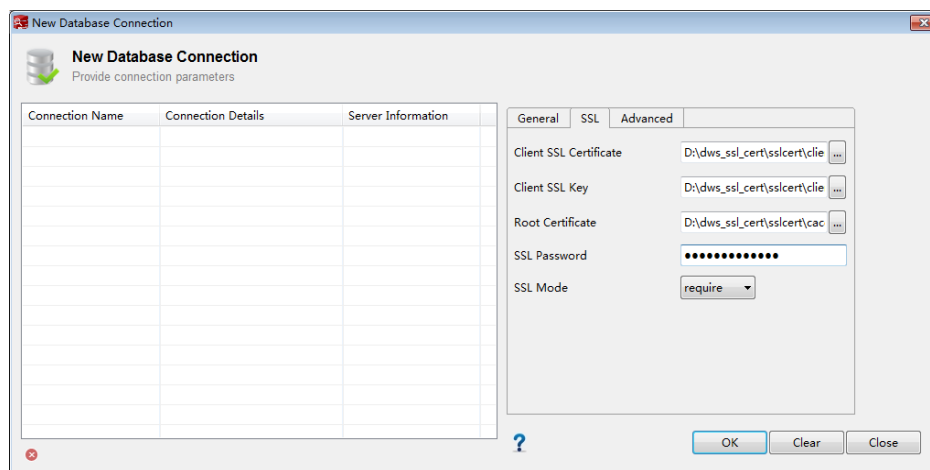
Parameter	Description	Example
Database Type	Select GaussDB A .	GaussDB A
Name	Name of a connection	dws-demo
Host	IP address (IPv4) of the cluster to be connected	-

Parameter	Description	Example
Host Port	Database port	8000
Database	Database name	gaussdb
User	Username for connecting to the database	-
Password	Password for logging in to the database to be connected	-
Save Password	Select an option from the drop-down list: <ul style="list-style-type: none">● Current Session Only: The password is saved only in the current session.● Do Not Save: The password is not saved.	-
Enable SSL	If Enable SSL is selected, the client can use SSL to encrypt connections. The SSL mode is more secure than common modes, so you are advised to enable SSL connection.	-

When **Enable SSL** is selected, download the SSL certificate and decompress it by referring to [Downloading SSL Certificate](#). Click the **SSL** tab and configure the following parameters:

Table 6-2 Configuring SSL parameters

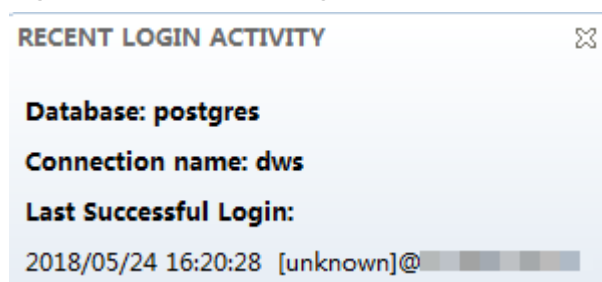
Parameter	Description
Client SSL Certificate	Select the sslcert\client.crt file in the decompressed SSL certificate directory.
Client SSL Key	Only the PK8 format is supported. Select the sslcert\client.key.pk8 file in the directory where the SSL certificate is decompressed.
Root Certificate	When SSL Mode is set to verify-ca , the root certificate must be configured. Select the sslcert\cacert.pem file in the decompressed SSL certificate directory.
SSL Password	Set the password for the client SSL key in PK8 format.
SSL Mode	Supported SSL modes include: <ul style="list-style-type: none">● require● verify-ca GaussDB(DWS) does not support the verify-full mode.

Figure 6-3 Configuring SSL parameters

Step 9 Click **OK** to establish the database connection.

If SSL is enabled, click **Continue** in the displayed **Connection Security Alert** dialog box.

After the login is successful, the **RECENT LOGIN ACTIVITY** dialog box is displayed, indicating that Data Studio is connected to the database. You can run the SQL statement in the **SQL Terminal** window on the Data Studio page.

Figure 6-4 Successful login

For details about how to use other functions of Data Studio, press **F1** to view the Data Studio user manual.

NOTE

- Data cannot be rolled back after being added, deleted, modified, or queried in Data Studio.
- Data Studio can save connection information, excluding passwords.
- DDL/DDI and data cannot be exported in batches for the following objects:
 - **Export DDL:**
Connection, database, foreign table, sequence, column, index, constraint, partition, function/procedure group, regular tables group, views group, schemas group, and system catalog group.
 - **Export DDL and Data**
Connection, database, namespace, foreign table, sequence, column, index, constraint, partition, function/procedure, view, regular tables group, schemas group, and system catalog group.

----End

6.4 Using the gsql CLI Client to Connect to a Cluster

6.4.1 Downloading the Data Studio client

GaussDB(DWS) provides client tool packages that match the cluster versions. You can download the desired client tool package on the GaussDB(DWS) management console.

The client tool package contains the following:

- **Linux database connection tool gsql and the script for testing sample data**

Linux gsql is a Linux command line client running in Linux. It is used to connect to the database in a data warehouse cluster.

The script for testing sample data is used to execute the introductory example.

- **Windows gsql**

Windows gsql is a command line client running on the Windows OS. It is used to connect to the database in a data warehouse cluster.

NOTE

Only 8.1.3.101 and later cluster versions can be downloaded from the console.

Downloading the Data Studio client

Step 1 Log in to the GaussDB(DWS) console. For details, see [Accessing the GaussDB\(DWS\) Management Console](#).

Step 2 In the navigation tree on the left, choose **Client Connections**.

Step 3 Select the GaussDB(DWS) client of the corresponding version from the drop-down list of **gsql CLI Client**.

Choose a corresponding client version according to the cluster version and operating system to which the client is to be installed.

Step 4 Click **Download** to download the gsql tool matching the 8.1.x cluster version. Click **Historical Version** to download the gsql tool corresponding to the cluster version.

- You are advised to download the gsql tool that matches the cluster version. That is, use gsql 8.1.x for clusters of 8.1.0 or later, and use gsql 8.2.x for clusters of 8.2.0 or later.
- The following table describes the files and folders in the Linux gsql tool package.

Table 6-3 Files and folders in the Linux gsql tool package

File or Folder	Description
bin	This folder contains the executable files of gsql in Linux. It contains the gsql client tool, GDS parallel data loading tool, and gs_dump, gs_dumpall, and gs_restore tools. For details, see section Server Tools in the <i>Data Warehouse Service Tool Guide</i> .
gds	This folder contains the files of the GDS data service tool. The GDS tool is used for parallel data loading and can import the data files stored in a common file system to a GaussDB(DWS) database.
lib	This folder contains the lib library required for executing the gsql client.
sample	This folder contains the following directories and files: <ul style="list-style-type: none">– setup.sh: script file for configuring the AK/SK before using gsql to import sample data– tpcds_load_data_from_obs.sql: script file for importing the TPC-DS sample data using the gsql client– query_sql directory: script file for querying the TPC-DS sample data
gsql_env.sh	Script file for configuring environment variables before running the gsql client.

- The following table describes the files and folders in the Windows gsql tool package.

Table 6-4 Files and folders in the Windows gsql tool package

File or Folder	Description
x64	This folder contains the 64-bit Windows gsql execution binary file and the dynamic library.
x86	This folder contains the 32-bit Windows gsql execution binary file and the dynamic library.

 **NOTE**

In the cluster list on the **Clusters > Dedicated Clusters** page, click the name of the specified cluster to go to the **Cluster Information** page and view the cluster version.

----End

6.4.2 Using the Linux gsql Client to Connect to a Cluster

This section describes how to connect to a database through an SQL client after you create a data warehouse cluster and before you use the cluster's database. GaussDB(DWS) provides the Linux gsql client that matches the cluster version for you to access the cluster through the cluster's public or private network address.

The gsql command line client provided by GaussDB(DWS) runs on Linux. Before using it to remotely connect to a GaussDB(DWS) cluster, you need to prepare a Linux server for installing and running the gsql client. If you use a public network address to access the cluster, you can install the Linux gsql client on your own Linux server. Ensure that the Linux server has a public network address. If no EIPs are configured for your GaussDB(DWS) cluster, you are advised to create a Linux ECS for convenience purposes. For more information, see [\(Optional\) Preparing an ECS as the gsql Client Server](#).

(Optional) Preparing an ECS as the gsql Client Server

For details about how to create an ECS, see "Getting Started > Creating an ECS" in the *Elastic Cloud Server User Guide*.

The created ECS must meet the following requirements:

- The ECS and data warehouse cluster must belong to the same region and AZ.
- If you use the gsql client provided by GaussDB(DWS) to connect to the GaussDB(DWS) cluster, the ECS image must meet the following requirements:

The image's OS must be one of the following Linux OSs supported by the gsql client:

– The **Redhat x86_64** client can be used on the following OSs:

- RHEL 6.4~7.6
- CentOS 6.4~7.4
- EulerOS 2.3

– The **SUSE x86_64** client can be used on the following OSs:

- SLES 11.1~11.4
- SLES 12.0~12.3

- If the client accesses the cluster using the private network address, ensure that the created ECS is in the same VPC as the GaussDB(DWS) cluster.

For details about VPC operations, see "VPC and Subnet" in the *Virtual Private Cloud User Guide*.

- If the client accesses the cluster using the public network address, ensure that both the created ECS and GaussDB(DWS) cluster have an EIP.

When creating an ECS, set **EIP** to **Automatically assign** or **Specify**.

- The security group rules of the ECS must enable communication between the ECS and the port that the GaussDB(DWS) cluster uses to provide services.

For details about security group operations, see "Security Group" in the *Virtual Private Cloud User Guide*.

Ensure that the security group of the ECS contains rules meeting the following requirements. If the rules do not exist, add them to the security group:

- **Transfer Direction: Outbound**
 - Protocol: The protocol must contain TCP. For example, **TCP** or **All**.
 - **Port:** The value must contain the database port that provides services in the GaussDB(DWS) cluster. For example, set this parameter to **1-65535** or a specific GaussDB(DWS) database port.
 - Destination: The IP address set here must contain the IP address of the GaussDB(DWS) cluster to be connected. **0.0.0.0/0** indicates any IP address.
- The security group rules of the data warehouse cluster must ensure that GaussDB(DWS) can receive network access requests from clients.

Ensure that the cluster's security group contains rules meeting the following requirements. If the rules do not exist, add them to the security group:

- **Transfer Direction: Inbound**
- Protocol: The protocol must contain TCP. For example, **TCP** or **All**.
- **Port:** Set this parameter to the database port that provides services in the data warehouse cluster, for example, **8000**.
- **Source:** The IP address set here must contain the IP address of the GaussDB(DWS) client server, for example, **192.168.0.10/32**.

Downloading the Linux gsql Client and Connecting to a Cluster

- Step 1** Download the Linux gsql client by referring to [Downloading the Data Studio client](#), and use an SSH file transfer tool (such as WinSCP) to upload the client to a target Linux server.

You are advised to download the gsql tool that matches the cluster version. That is, use gsql 8.1.x for clusters of 8.1.0 or later, and use gsql 8.2.x for clusters of 8.2.0 or later. To download gsql 8.2.x, replace **dws_client_8.1.x_redhat_x64.zip** with **dws_client_8.2.x_redhat_x64.zip**. The **dws_client_8.1.x_redhat_x64.zip** is used as an example.

The user who uploads the client must have the full control permission on the target directory on the host to which the client is uploaded.

- Step 2** Use the SSH tool to remotely manage the host where the client is installed.

For details about how to log in to an ECS, see "ECSs> Logging In to a Linux ECS > Login Using an SSH Password" in the *Elastic Cloud Server User Guide*.

- Step 3** (Optional) To connect to the cluster in SSL mode, configure SSL authentication parameters on the host where the client is installed. For details, see [Establishing Secure TCP/IP Connections in SSL Mode](#).

NOTE

The SSL connection mode is more secure than the non-SSL mode. You are advised to connect the client to the cluster in SSL mode.

- Step 4** Run the following commands to decompress the client:

```
cd <Path for saving the client>  
unzip dws_client_8.1.x_redhat_x64.zip
```

In the preceding commands:

- `<Path_for_storing_the_client>`: Replace it with the actual path.
- `dws_client_8.1.x_redhat_x64.zip`: This is the client tool package name of **RedHat x86**. Replace it with the actual name.

Step 5 Run the following command to configure the GaussDB(DWS) client:

```
source gsql_env.sh
```

If the following information is displayed, the gsql client is successfully configured:

```
All things done.
```

Step 6 Connect to the database in the GaussDB(DWS) cluster using the gsql client. Replace the values of each parameter with actual values.

```
gsql -d <Database_name> -h <Cluster_address> -U <Database_user> -p <Database_port> -W  
<Cluster_password> -r
```

The parameters are described as follows:

- `Database_name`: Enter the name of the database to be connected. If you use the client to connect to the cluster for the first time, enter the default database **gaussdb**.
- `Cluster_address`: For details about how to obtain this address, see [Obtaining the Cluster Connection Address](#). If a public network address is used for connection, set this parameter to **Public Network Address**. If a private network address is used for connection, set this parameter to **Private Network Address**.
- `Database_user`: Enter the username of the cluster's database. If you use the client to connect to the cluster for the first time, set this parameter to the default administrator configured during cluster creation, for example, **dbadmin**.
- `Database_port`: Enter the database port set during cluster creation.

For example, run the following command to connect to the default database **gaussdb** in the GaussDB(DWS) cluster:

```
gsql -d gaussdb -h 10.168.0.74 -U dbadmin -p 8000 -W password -r
```

If the following information is displayed, the connection succeeded:

```
gaussdb=>
```

```
----End
```

gsql Command Reference

For more information about the gsql commands, see the *Data Warehouse Service (DWS) Tool Guide*.

6.4.3 Using the Windows gsql Client to Connect to a Cluster

This section describes how to connect to a database through an SQL client after you create a data warehouse cluster and before you use the cluster's database. GaussDB(DWS) provides the Windows gsql client that matches the cluster version for you to access the cluster through the cluster's public or private network address.

Procedure

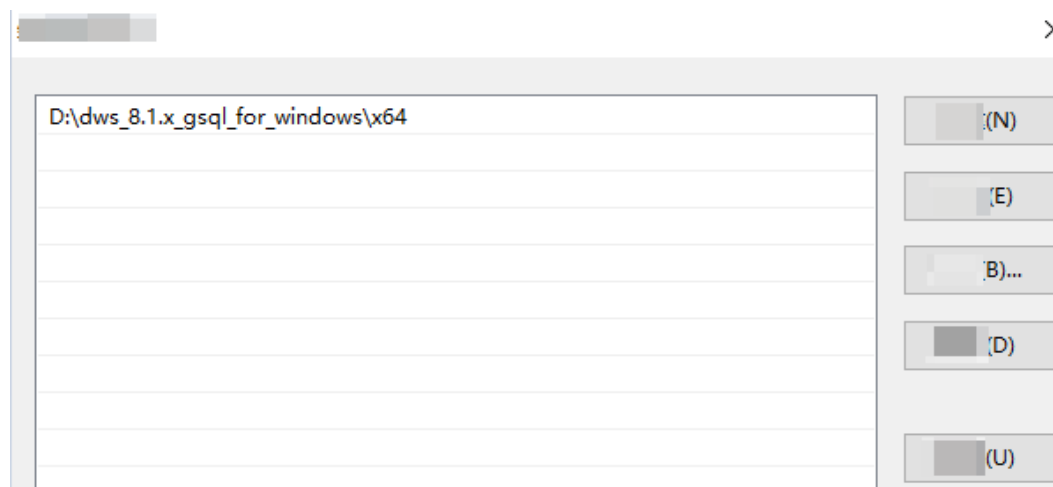
- Step 1** Install and run the gsql client on the local Windows server (in Windows CLI). Windows Server 2008/Windows 7 and later are supported.
- Step 2** Download the Windows gsql client by referring to [Downloading the Data Studio client](#) and decompress the package to a local folder.
- Step 3** On the local server, click **Start**, search for **cmd**, and run the program as the administrator. Alternatively, press **Win+R** to open the Windows CLI.
- Step 4** Set environment variables. For a 32-bit OS, select the **x86** folder. For a 64-bit OS, select the **x64** folder.

Method 1: Configure environment variables in the Windows CLI. Open the command prompt and run the **set path=<window_gsql>;%path%** command, where *<window_gsql>* indicates the folder path where the Windows gsql client was decompressed to in the previous step. For example:

```
set path=C:\Users\xx\Desktop\dws_8.1.x_gsql_for_windows\x64;%path%
```

Method 2: In the **Control Panel** window, search for **System** and click **View advanced system settings**. Click the **Advanced** tab, and click **Environment Variables**. Select the **Path** parameter and click **Edit**. Add the gsql path in the parameter value. For example:

Figure 6-5 Configuring Windows environment variables



- Step 5** (Optional) To connect to the cluster in SSL mode, configure SSL authentication parameters on the server where the client is installed. For details, see [Establishing Secure TCP/IP Connections in SSL Mode](#).

NOTE

The SSL connection mode is more secure than the non-SSL mode. You are advised to connect the client to the cluster in SSL mode.

- Step 6** In the Windows CLI, run the following command to connect to the database in the GaussDB(DWS) cluster using the gsql client:

```
gsql -d <Database_name> -h <Cluster_address> -U <Database_user> -p <Database_port> -W <Cluster_password> -r
```

The parameters are as follows:

- **Database name:** Enter the name of the database to be connected. If you use the client to connect to the cluster for the first time, enter the default database **gaussdb**.
- **Cluster address:** For details about how to obtain this address, see [Obtaining the Cluster Connection Address](#). If a public network address is used for connection, set this parameter to the public network domain name. If a private network address is used for connection, set this parameter to the private network domain name.
- **Database user:** Enter the username of the cluster's database. If you use the client to connect to the cluster for the first time, set this parameter to the default administrator configured during cluster creation, for example, **dbadmin**.
- **Database port:** Enter the database port set during cluster creation.

For example, run the following command to connect to the default database **gaussdb** in the GaussDB(DWS) cluster:

```
gsql -d gaussdb -h 10.168.0.74 -U dbadmin -p 8000 -W password -r
```

If the following information is displayed, the connection succeeded:

```
gaussdb=>
```

----End

Precautions

1. The default character encoding of the Windows command prompt is GBK, and the default value of **client_encoding** of Windows gsql is **GBK**. Some characters encoded using UTF-8 cannot be displayed in Windows gsql.
Suggestion: Ensure the file specified using **-f** uses UTF-8 encoding, and set the default encoding format to **UTF-8 (set client_encoding='utf-8')**;

```
gaussdb=> \i D:\test.sql
D:: Permission denied
postgres=> \i D:\test.sql
id
----
1
(1 row)
```
2. Paths in Windows gsql must be separated by slashes (/), or an error will be reported. In a meta-command, the backslash (\) indicates the start of a meta-command. If the backslash is enclosed in single quotation marks ('\'), it is used for escape.

```
gaussdb=> \i D:\test.sql
D:: Permission denied
postgres=> \i D:\test.sql
id
----
1
(1 row)
```
3. To use the **\!** metacommand to run a system command in Windows gsql, be sure to use the path separator required by the system command. Generally, the path separator is a backslash (\).

```
gaussdb=> \! type D:/test.sql
Incorrect syntax.
gaussdb=> \! type D:\test.sql
select 1 as id;
```
4. Windows gsql does not support the **\parallel** meta-command.

```
gaussdb=> \parallel
ERROR: "\parallel" is not supported in Windows.
```
5. In Linux shell, single quotation marks (') and double quotation marks (") can be used to enclose strings. In Windows, only double quotation marks can be used.

```
gsql -h 192.168.233.189 -p 8109 -d postgres -U odbcuser -W password -c "select 1 as id"
id
----
 1
(1 row)
```

If single quotation marks are used, an error will be reported and the input will be ignored.

```
gsql -h 192.168.233.189 -p 8109 -d postgres -U odbcuser -W password -c 'select 1 as id'
gsql: warning: extra command-line argument "1" ignored
gsql: warning: extra command-line argument "as" ignored
gsql: warning: extra command-line argument "id" ignored
ERROR: unterminated quoted string at or near "'select"
LINE 1: 'select
```

6. If Windows gsql is idle for a long time after a connection is established, the connection session times out, and an SSL error is reported. In this case, you need to log in again. The following error is reported:

```
SSL SYSCALL error: Software caused connection abort (0x00002745/10053), remote datanode <NULL>, error: Result too large
```

7. In Windows, press **Ctrl+C** to exit gsql. If **Ctrl+C** are pressed during input, the input will be ignored and you will be forced to exit gsql.

Enter **as** and press **Ctrl+C**. After **\q** is displayed, exit gsql.

```
gaussdb=> select 1
gaussdb=> as \q
```

8. Windows gsql cannot connect to a database using the LATIN1 character encoding. The error information is as follows:

```
gsql: FATAL: conversion between GBK and LATIN1 is not supported
```

9. The location of the **gsqlrc.conf** file:

The default **gsqlrc** path is **%APPDATA%/postgresql/gsqlrc.conf**. You can also set the path using the **PSQLRC** variable.

```
set PSQLRC=C:\Users\xx\Desktop\dws_8.1.x_gsql_for_windows\x64\gsqlrc.conf
```

gsql Command Reference

For more information about the gsql commands, see the *Data Warehouse Service (DWS) Tool Guide*.

6.4.4 Establishing Secure TCP/IP Connections in SSL Mode

GaussDB(DWS) supports the standard SSL. As a highly secure protocol, SSL authenticates bidirectional identification between the server and client using digital signatures and digital certificates to ensure secure data transmission. To support SSL connection, GaussDB(DWS) has obtained the formal certificates and keys for the server and client from the CA certification center. It is assumed that the key and certificate for the server are **server.key** and **server.crt** respectively; the key and certificate for the client are **client.key** and **client.crt** respectively, and the name of the CA root certificate is **cacert.pem**.

The SSL connection mode is more secure. By default, the SSL feature in a cluster allows SSL and non-SSL connections from the client. For security purposes, you are advised to connect to the cluster via SSL from the client. Ensure the certificate, private key, and root certificate of the GaussDB(DWS) server have been configured by default. To forcibly use an SSL connection, configure the **require_ssl** parameter in the **Require SSL Connection** area of the cluster's **Security Settings** page on the GaussDB(DWS) management console. Require SSL Connection on the Security Settings page of the cluster. For more information, see [Configuring SSL](#).

Connection and Combinations of SSL Connection Parameters on the Client and Server.

The client or JDBC/ODBC driver needs to use SSL connection. Configure related SSL connection parameters in the client or application code. The GaussDB(DWS) management console provides the SSL certificate required by the client. The SSL certificate contains the default certificate, private key, root certificate, and private key password encryption file required by the client. Download the SSL certificate to the host where the client is installed, and specify the path of the certificate on the client. For more information, see [Configuring Digital Certificate Parameters Related to SSL Authentication on the gsql Client](#) and [SSL Authentication Modes and Client Parameters](#).

NOTE

Using the default certificate may pose security risks. To improve system security, you are advised to periodically change the certificate to prevent password cracking. If you need to replace the certificate, contact the .

Configuring SSL Connection

Prerequisites

- After you have modified the security parameters and the modifications take effect, the cluster may be restarted, which makes the cluster unavailable temporarily.
- To modify the cluster's security configuration, ensure that the following conditions are met:
 - The cluster status is **Available** or **Unbalanced**.
 - The **Task Information** cannot be set to **Creating snapshot**, **Scaling out**, **Configuring**, or **Restarting**.

Procedure

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 In the navigation pane on the left, choose **Clusters > Dedicated Clusters**.

Step 3 In the cluster list, click the name of a cluster. On the page that is displayed, click **Security Settings**.

By default, **Configuration Status** is **Synchronized**, which indicates that the latest database result is displayed.

Step 4 In the **SSL Connection** area, enable **Require SSL Connection** (recommended).

indicates the function is enabled. The `require_ssl` is set to **1**, indicating that the server forcibly requires the SSL connection.



indicates the function is disabled (default value). The `require_ssl` parameter is set to **0**, indicating that the server does not require SSL connections. For details about how to configure the `require_ssl` parameter, see [require_ssl \(Server\)](#).

 NOTE

- If the gsql client or ODBC driver provided by GaussDB(DWS) is used, GaussDB(DWS) supports the TLSv1.2 SSL protocol.
- If the JDBC driver provided by GaussDB(DWS) is used, GaussDB(DWS) supports SSL protocols, such as SSLv3, TLSv1, TLSv1.1, and TLSv1.2. The SSL protocol used between the client and the database depends on the Java Development Kit (JDK) version used by the client. Generally, JDK supports multiple SSL protocols.

Step 5 Click **Apply**.

The system automatically saves the SSL connection settings. On the **Security Settings** page, **Configuration Status** is **Applying**. After **Configuration Status** changes to **Synchronized**, the settings have been saved and taken effect.

----End

Configuring Digital Certificate Parameters Related to SSL Authentication on the gsql Client

After a data warehouse cluster is deployed, the SSL authentication mode is enabled by default. The server certificate, private key, and root certificate have been configured by default. You need to configure the client parameters.

Step 1 Log in to the GaussDB(DWS) management console. In the navigation pane, choose **Client Connections**.

Step 2 In the **Driver** area, click **download an SSL certificate**.

Step 3 Use a file transfer tool (such as WinSCP) to upload the SSL certificate to the host where the client is installed.

For example, save the downloaded certificate **dws_ssl_cert.zip** to the **/home/dbadmin/dws_ssl/** directory.

Step 4 Use an SSH remote connection tool (such as PuTTY) to log in to the host where the gsql client is installed and run the following commands to go to the directory where the SSL certificate is stored and decompress the SSL certificate:

```
cd /home/dbadmin/dws_ssl/  
unzip dws_ssl_cert.zip
```

Step 5 Run the export command and configure digital certificate parameters related to SSL authentication on the host where the gsql client is installed.

There are two SSL authentication modes: bidirectional authentication and unidirectional authentication. The client environment variables to be configured vary according to the authentication mode. For details, see [SSL Authentication Modes and Client Parameters](#).

The following parameters must be configured for bidirectional authentication:

```
export PGSSLCERT="/home/dbadmin/dws_ssl/sslcert/client.crt"  
export PGSSLKEY="/home/dbadmin/dws_ssl/sslcert/client.key"  
export PGSSLMODE="verify-ca"  
export PGSSLROOTCERT="/home/dbadmin/dws_ssl/sslcert/cacert.pem"
```

The following parameters must be configured for unidirectional authentication:

```
export PGSSLMODE="verify-ca"  
export PGSSLROOTCERT="/home/dbadmin/dws_ssl/sslcert/cacert.pem"
```

NOTICE

- You are advised to use bidirectional authentication for security purposes.
- The environment variables configured for a client must contain the absolute file paths.

Step 6 Change the client private key permissions.

The permissions on the client's root certificate, private key, certificate, and encrypted private key file must be **600**. If the permissions do not meet the requirement, the client cannot connect to the cluster in SSL mode.

```
chmod 600 client.key
chmod 600 client.crt
chmod 600 client.key.cipher
chmod 600 client.key.rand
chmod 600 cacert.pem
```

----End

SSL Authentication Modes and Client Parameters

There are two SSL authentication modes: bidirectional authentication and unidirectional authentication. Table [Table 6-5](#) shows the differences between these two modes. You are advised to use bidirectional authentication for security purposes.

Table 6-5 Authentication modes

Authentication Mode	Description	Environment Variables Configured on a Client	Maintenance
Bidirectional authentication (recommended)	The client verifies the server's certificate and the server verifies the client's certificate. The connection can be set up only after the verifications are successful.	Set the following environment variables: <ul style="list-style-type: none">• PGSSLCERT• PGSSLKEY• PGSSLROTCERT• PGSSLMODE	This authentication mode is applicable to scenarios that require high data security. When using this mode, you are advised to set the PGSSLMODE client variable to verify-ca for network data security purposes.

Authentication Mode	Description	Environment Variables Configured on a Client	Maintenance
Unidirectional authentication	The client verifies the server's certificate, whereas the server does not verify the client's certificate. The server loads the certificate information and sends it to the client. The client verifies the server's certificate according to the root certificate.	Set the following environment variables: <ul style="list-style-type: none"> PGSSLOTCERT PGSSLMODE 	To prevent TCP-based link spoofing, you are advised to use the SSL certificate authentication. In addition to configuring the client root certificate, you are advised to set the PGSSLMODE variable to verify-ca on the client.

Configure environment variables related to SSL authentication on the client. For details, see [Table 6-6](#).

 **NOTE**

The path of environment variables is set to `/home/dbadmin/dws_ssl/` as an example. Replace it with the actual path.

Table 6-6 Client parameters

Environment Variable	Description	Value Range
PGSSLCERT	Specifies the certificate files for a client, including the public key. Certificates prove the legal identity of the client and the public key is sent to the remote end for data encryption.	The absolute path of the files must be specified, for example: <pre>export PGSSLCERT='/home/dbadmin/dws_ssl/sslcert/client.crt'</pre> (No default value)
PGSSLKEY	Specifies the client private key file used to decrypt the digital signatures and the data encrypted using the public key.	The absolute path of the files must be specified, for example: <pre>export PGSSLKEY='/home/dbadmin/dws_ssl/sslcert/client.key'</pre> (No default value)

Environment Variable	Description	Value Range
PGSSLMODE	Specifies whether to negotiate with the server about SSL connection and specifies the priority of the SSL connection.	<p>Values and meanings:</p> <ul style="list-style-type: none"> • disable: only tries to establish a non-SSL connection. • allow: tries to establish a non-SSL connection first, and then an SSL connection if the first attempt fails. • prefer: tries to establish an SSL connection first, and then a non-SSL connection if the first attempt fails. • require: only tries to establish an SSL connection. If there is a CA file, perform the verification according to the scenario in which the parameter is set to verify-ca. • verify-ca: tries to establish an SSL connection and check whether the server certificate is issued by a trusted CA. • verify-full: GaussDB(DWS) does not support this mode. <p>Default value: prefer</p> <p>NOTE When an external client accesses a cluster, the error message "ssl SYSCALL error" is displayed on some nodes. In this case, run export PGSSLMODE="allow" or export PGSSLMODE="prefer".</p>
PGSSLROOTCERT	Specifies the root certificate file for issuing client certificates. The root certificate is used to verify the server certificate.	<p>The absolute path of the files must be specified, for example: <code>export PGSSLROOTCERT='/home/dbadmin/dws_ssl/sslcert/certca.pem'</code></p> <p>Default value: null</p>
PGSSLCRL	Specifies the certificate revocation list file, which is used to check whether a server certificate is in the list. If the certificate is in the list, it is invalid.	<p>The absolute path of the files must be specified, for example: <code>export PGSSLCRL='/home/dbadmin/dws_ssl/sslcert/sslcert-file.crt'</code></p> <p>Default value: null</p>

Combinations of SSL Connection Parameters on the Client and Server

Whether the client uses the SSL encryption connection mode and whether to verify the server certificate depend on client parameter **sslmode** and server (cluster) parameters **ssl** and **require_ssl**. The parameters are as follows:

- **ssl (Server)**

The **ssl** parameter indicates whether to enable the SSL function. **on** indicates that the function is enabled, and **off** indicates that the function is disabled.

- The default value is **on** and you cannot set this parameter on the GaussDB(DWS) management console.

- **require_ssl (Server)**

The **require_ssl** parameter specifies whether the server forcibly requires SSL connection. This parameter is valid only when **ssl** is set to **on**. **on** indicates that the server forcibly requires SSL connection. **off** indicates that the server does not require SSL connection.

- The default value is **off**. You can set the **require_ssl** parameter in the **Require SSL Connection** area of the cluster's **Security Settings** page on the GaussDB(DWS) management console.

- **sslmode (Client)**

You can set this parameter in the SQL client tool.

- In the gsql command line client, this parameter is the **PGSSLMODE** parameter.
- On the Data Studio client, this parameter is the **SSL Mode** parameter.

The combinations of client parameter **sslmode** and server parameters **ssl** and **require_ssl** are as follows.

Table 6-7 Combinations of SSL connection parameters on the client and server

ssl (Server)	sslmode (Client)	require_ssl (Server)	Result
on	disable	on	The server requires SSL, but the client disables SSL for the connection. As a result, the connection cannot be set up.
	disable	off	The connection is not encrypted.
	allow	on	The connection is encrypted.
	allow	off	The connection is not encrypted.
	prefer	on	The connection is encrypted.
	prefer	off	The connection is encrypted.
	require	on	The connection is encrypted.
	require	off	The connection is encrypted.

ssl (Server)	sslmode (Client)	require_ssl (Server)	Result
	verify-ca	on	The connection is encrypted and the server certificate is verified.
	verify-ca	off	The connection is encrypted and the server certificate is verified.
off	disable	on	The connection is not encrypted.
	disable	off	The connection is not encrypted.
	allow	on	The connection is not encrypted.
	allow	off	The connection is not encrypted.
	prefer	on	The connection is not encrypted.
	prefer	off	The connection is not encrypted.
	require	on	The client requires SSL, but SSL is disabled on the server. Therefore, the connection cannot be set up.
	require	off	The client requires SSL, but SSL is disabled on the server. Therefore, the connection cannot be set up.
	verify-ca	on	The client requires SSL, but SSL is disabled on the server. Therefore, the connection cannot be set up.
verify-ca	off	The client requires SSL, but SSL is disabled on the server. Therefore, the connection cannot be set up.	

6.5 Using the JDBC and ODBC Drivers to Connect to a Cluster

6.5.1 Development Specifications

If the connection pool mechanism is used during application development, the following specifications must be met. Otherwise, connections in the connection pool have statuses, which will affect the correctness of subsequent operations on the connection pool.

- If the GUC parameter is set in a connection, you must execute **SET SESSION AUTHORIZATION DEFAULT;RESET ALL;** to clear the connection status before returning the connection to the connection pool.
- If a temporary table is used, it must be deleted before the connection is returned to the connection pool.

6.5.2 Downloading the JDBC or ODBC Driver

The JDBC or ODBC driver is used to connect to data warehouse clusters. You can download the JDBC or ODBC driver provided by GaussDB(DWS) from the management console or use the open-source JDBC or ODBC driver.

Open-Source JDBC or ODBC Driver

GaussDB(DWS) also supports open-source JDBC driver: PostgreSQL JDBC 9.3-1103 or later.

GaussDB(DWS) also supports open-source ODBC driver: PostgreSQL ODBC 09.01.0200 or later.

Downloading the JDBC or ODBC Driver

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 In the navigation tree on the left, choose **Client Connections**.

Step 3 In the **Driver** area, choose a driver that you want to download.

- **JDBC Driver**

Select **DWS JDBC Driver** and click **Download** to download the JDBC driver matching the current cluster version. The driver package name is **dws_8.1.x_jdbc_driver.zip**.

If clusters of different versions are available, you will download the JDBC driver matching the earliest cluster version after clicking **Download**. If there is no cluster, you will download the JDBC driver of the earliest version after clicking **Download**. GaussDB(DWS) clusters are compatible with earlier versions of JDBC drivers.

Click **Historical Version** to download the corresponding JDBC driver version. You are advised to download the JDBC driver based on the cluster version.

The JDBC driver can be used on all platforms and depends on JDK 1.6 or later.

- **ODBC Driver**

Select a corresponding version and click **Download** to download the ODBC driver matching the current cluster version.

Click **Historical Version** to download the corresponding ODBC driver version. You are advised to download the ODBC driver based on the cluster version.

 **NOTE**

- The ODBC driver is incompatible with Windows Server 2016.

----End

6.5.3 Using JDBC to Connect to a Cluster

In GaussDB(DWS), you can use a JDBC driver to connect to a database on Linux or Windows. The driver can connect to the database through an ECS on the platform or over the Internet.

When using the JDBC driver to connect to the data warehouse cluster, determine whether to enable SSL authentication. SSL authentication is used to encrypt

communication data between the client and the server. It safeguards sensitive data transmitted over the Internet. You can download a self-signed certificate file on the GaussDB(DWS) management console. To make the certificate take effect, you must configure the client program using the OpenSSL tool and the Java keytool.

NOTE

The SSL mode delivers higher security than the common mode. You are advised to enable SSL connection when using JDBC to connect to a GaussDB(DWS) cluster.

For details about how to use the JDBC API, see the official documentation.

Prerequisites

- You have installed JDK 1.6 or later and configured environment variables.
- You have downloaded the JDBC driver. For details, see [Downloading the JDBC or ODBC Driver](#).

GaussDB(DWS) also supports open-source JDBC driver: PostgreSQL JDBC 9.3-1103 or later.

- You have downloaded the SSL certificate file. For details, see [Downloading an SSL Certificate](#).

Using a JDBC Driver to Connect to a Database

The procedure for connecting to the database using a JDBC driver in a Linux environment is similar to that in a Windows environment. The following describes the connection procedure in a Windows environment.

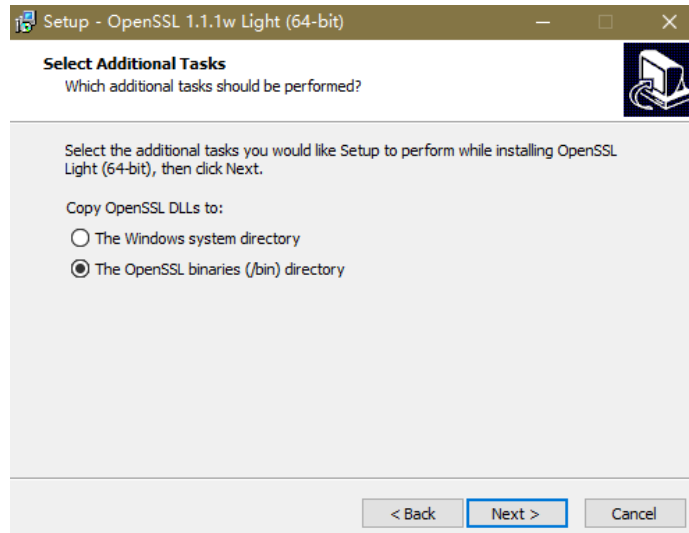
Step 1 Determine whether you want to use the SSL mode to connect to the GaussDB(DWS) cluster.

- If yes, enable SSL connection by referring to [Configuring SSL Connection](#). SSL connection is enabled by default. Then go to [Step 2](#).
- If no, disable SSL connection by referring to [Configuring SSL Connection](#) and go to [Step 4](#).

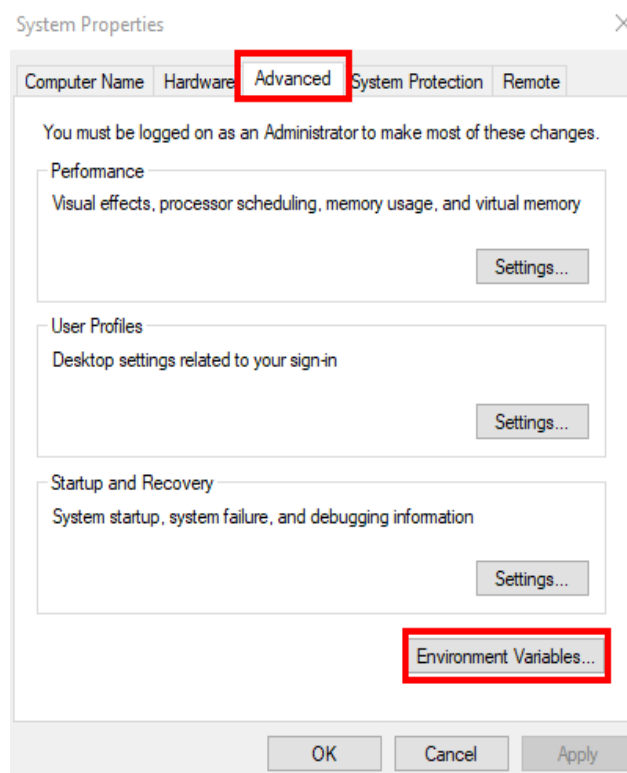
Step 2 (Optional) On Linux, use WinSCP to upload the downloaded SSL certificate file to the Linux environment.

Step 3 Configure the certificate to enable SSL connection.

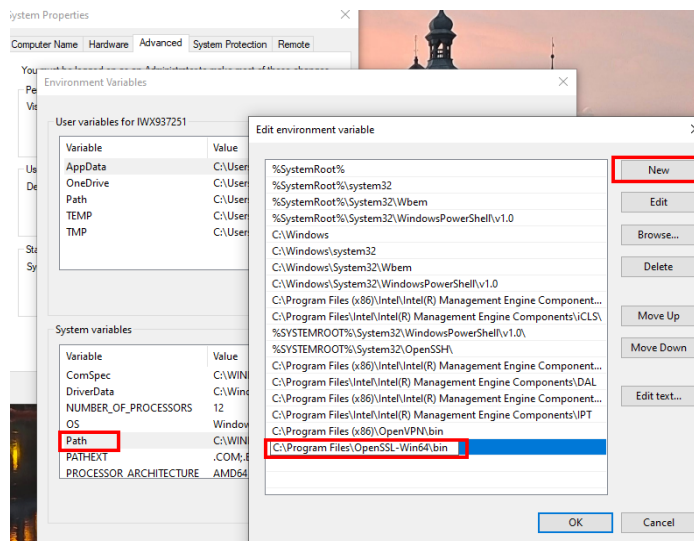
1. Download the OpenSSL tool for Windows. Download address: <https://slproweb.com/products/Win32OpenSSL.html>. Currently, OpenSSL 3.0.0 is not supported. Download **Win64 OpenSSL v1.1.1w Light**.
2. Double-click the installation package **Win64OpenSSL_Light-1_1_1w.exe** and install it to the default path on drive C. Copy the DLLs to the OpenSSL directory, as shown in the following figure. Retain the default settings in the remaining steps until the installation is successful.



3. Install an environment variable. Click **Start** in the lower left corner of the local PC, right-click **This PC**, choose **More > Properties > View advanced system settings**. Switch to the **Advanced** tab and click **Environment Variables**.



4. In the **System variables** area, double-click **Path** and click **New** in the window displayed. Add the OpenSSL **bin** path to the last line, for example, **C:\Program Files\OpenSSL-Win64\bin**, and click **OK**. Click **OK** again and the variable is configured successfully.



- Decompress the package to obtain the certificate file. Decompression path `C:\` is used as an example.

You are advised to store the certificate file in a path of the English version and can specify the actual path when configuring the certificate. If the path is incorrect, a message stating that the file does not exist will be prompted.

- Open **Command Prompt** and switch to the `C:\dws_ssl_cert\sslcert` path. Run the following commands to import the root license to the truststore:

```
openssl x509 -in cacert.pem -out cacert.crt.der -outform der
keytool -keystore mytruststore -alias cacert -import -file cacert.crt.der
```

- `cacert.pem` indicates the root certificate obtained after decompression.
- `cacert.crt.der` indicates the generated intermediate file. You can store the file to another path and change the file name to your desired one.
- `mytruststore` indicates the generated truststore name and `cacert` indicates the alias name. Both parameters can be modified.

Enter the truststore password as prompted and answer **y**.

- Convert the format of the client private key.
`openssl pkcs12 -export -out client.pkcs12 -in client.crt -inkey client.key`

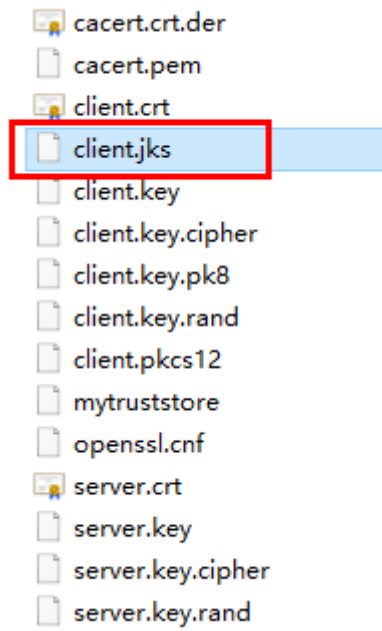
Enter the client private key password **Gauss@MppDB**. Then enter and confirm the self-defined private key password.

- Import the private key to the keystore.
`keytool -importkeystore -deststorepass Gauss@MppDB -destkeystore client.jks -srckeystore client.pkcs12 -srcstorepass Password -srcstoretype PKCS12 -alias 1`

 NOTE

- In the preceding command, *Password* is an example. Replace it with the actual password.
- If information similar to the following is displayed and no error is reported, the import is successful. The target key file **client.jks** will be generated in **C:\dws_ssl_cert\sslcert**.

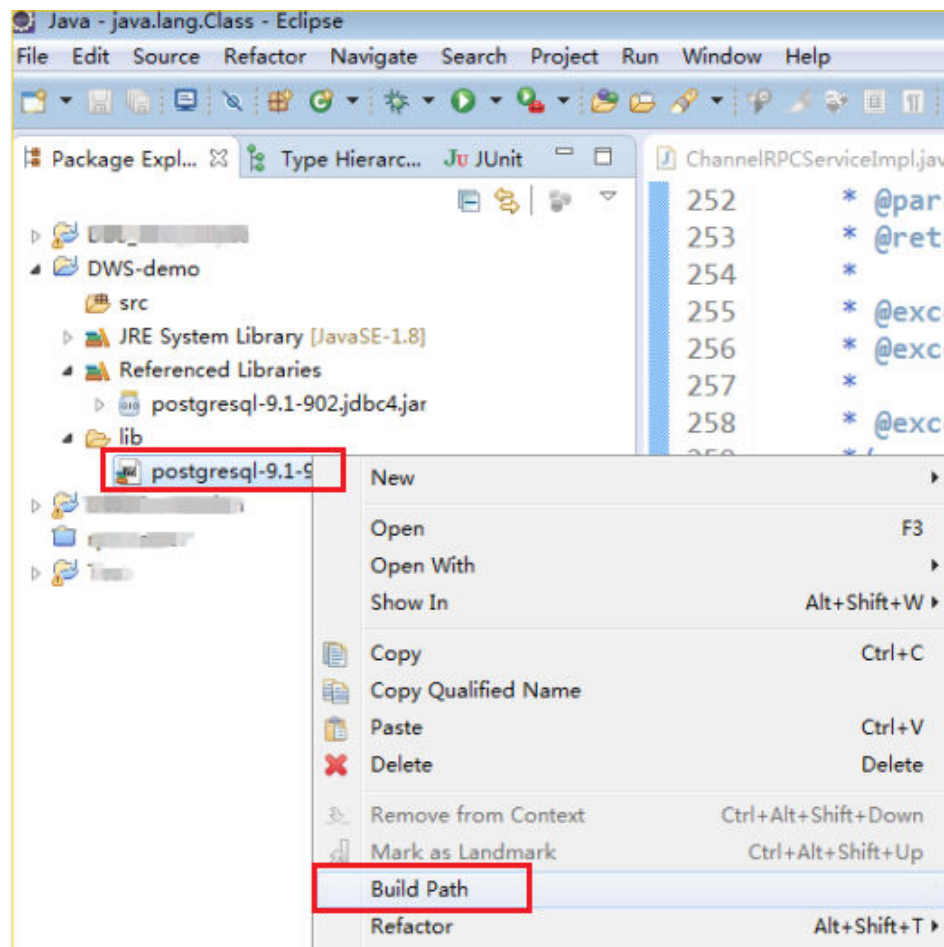
```
C:\dws_ssl_cert\sslcert>keytool -importkeystore -deststorepass Gauss@Opd0 -destkeystore client.jks -srckeystore client.pkcs12 -srcstorepass key123 -srcstoretype PKCS12 -alias 1  
Importing keystore client.pkcs12 to client.jks...
```



Step 4 Download the driver package **dws_8.1.x_jdbc_driver.zip** and decompress it. There will be two JDBC drive JAR packages, **gsjdbc4.jar** and **gsjdbc200.jar**. Use either of them as required.

Step 5 Add the JAR file to the application project so that applications can reference the JAR file.

Take the Eclipse project as an example. Store the JAR file to the project directory, for example, the **lib** directory in the project directory. In the Eclipse project, right-click the JAR file in the **lib** directory and choose **Build Path** to reference the JAR file.

Figure 6-6 Referencing a JAR file**Step 6** Load the driver.

The following methods are available:

- Using a code: **`Class.forName("org.postgresql.Driver");`**
- Using a parameter during the JVM startup: **`java -Djdbc.drivers=org.postgresql.Driver jdbctest`**

NOTE

The JDBC driver package downloaded on GaussDB(DWS) contains **`gsjdbc.jar`**.

- **`gsjdbc4.jar`**: The **`gsjdbc4.jar`** driver package is compatible with PostgreSQL. Its class names and class structures are the same as those of the PostgreSQL driver. Applications that run in PostgreSQL can be directly migrated to the current system.

Step 7 Call the **`DriverManager.getConnection()`** method of JDBC to connect to GaussDB(DWS) databases.

The JDBC API does not provide the connection retry capability. You need to implement the retry processing in the service code.

`DriverManager.getConnection()` methods:

- **`DriverManager.getConnection(String url);`**
- **`DriverManager.getConnection(String url, Properties info);`**

- DriverManager.getConnection(String url, String user, String password);

Table 6-8 Database connection parameters

Parameter	Description
url	<p>Specifies the database connection descriptor, which can be viewed on the management console. For details, see Obtaining the Cluster Connection Address.</p> <p>The URL format is as follows:</p> <ul style="list-style-type: none">• jdbc:postgresql:database• jdbc:postgresql://host/database• jdbc:postgresql://host:port/database• jdbc:postgresql://host:port[,host:port][...]/database <p>NOTE</p> <ul style="list-style-type: none">• If gsjdbc200.jar is used, change jdbc:postgresql to jdbc:gaussdb.<ul style="list-style-type: none">– database indicates the name of the database to be connected.– host indicates the name or IP address of the database server. If an ELB is bound to the cluster, set host to the IP address of the ELB.– port indicates the port number of the database server. By default, the database running on port 8000 of the local host is connected.– Multiple IP addresses and ports can be configured. JDBC balances load by random access and failover, and will automatically ignore unreachable IP addresses. Separate multiple pairs of IP addresses and ports by commas (,). Example: jdbc:postgresql://10.10.0.13:8000,10.10.0.14:8000/database• If JDBC is used to connect to a cluster, only JDBC connection parameters can be configured in a cluster address. Variables cannot be added.

Parameter	Description
info	<p>Specifies database connection properties. Common properties include the following:</p> <ul style="list-style-type: none">• user: a string type. It indicates the database user who creates the connection task.• password: a string type. It indicates the password of the database user.• ssl: a boolean type. It indicates whether to use the SSL connection.• loggerLevel: string type. It indicates the volume of log data sent to the LogStream or LogWriter specified in the DriverManager. Currently, OFF, DEBUG, and TRACE are supported. DEBUG indicates that only logs of DEBUG or a higher level are printed, generating little log information. TRACE indicates that logs of the DEBUG and TRACE levels are displayed, generating detailed log information. The default value is OFF, indicating that no logs will be displayed.• prepareThreshold: integer type. It indicates the number of PreparedStatement executions required before requests are converted to prepared statements in servers. The default value is 5.• batchMode: boolean type. It indicates whether to connect the database in batch mode.• fetchsize: integer type. It indicates the default fetch size for statements in the created connection.• ApplicationName: string type. It indicates an application name. The default value is PostgreSQL JDBC Driver.• allowReadOnly: boolean type. It indicates whether to enable the read-only mode for connection. The default value is false. If the value is not changed to true, the execution of connection.setReadOnly does not take effect.• blobMode: string type. It is used to set the setBinaryStream method to assign values to different data types. The value on indicates that values are assigned to the BLOB data type and off indicates that values are assigned to the BYTEA data type. The default value is on.• currentSchema: string type. It specifies the schema used for connecting to the database.• defaultQueryMetaData: Boolean. It specifies whether to query SQL metadata by default. The default value is false. After this function is enabled, raw data operations are supported. However, it is incompatible with the create table as and select into operations in PrepareStatement.• connectionExtraInfo: boolean type. This parameter indicates whether the JDBC driver reports the driver deployment path and process owner to the database.

Parameter	Description
	NOTE The value can be true or false . The default value is true . If connectionExtraInfo is set to true , the JDBC driver reports the driver deployment path and process owner to the database and displays the information in the connection_info parameter. In this case, you can query the information from PG_STAT_ACTIVITY or PGXC_STAT_ACTIVITY .
user	Specifies the database user.
password	Specifies the password of the database user.

The following describes the sample code used to encrypt the connection using the SSL certificate:

// The following code obtains the database SSL connection operation and encapsulates the operation as an API.

```
public static Connection GetConnection(String username, String passwd) {  
    // Define the driver class.  
    String driver = "org.postgresql.Driver";  
    //Set keyStore.  
    System.setProperty("javax.net.ssl.trustStore", "mytruststore");  
    System.setProperty("javax.net.ssl.keyStore", "client.jks");  
    System.setProperty("javax.net.ssl.trustStorePassword", "password");  
    System.setProperty("javax.net.ssl.keyStorePassword", "password");  
  
    Properties props = new Properties();  
    props.setProperty("user", username);  
    props.setProperty("password", passwd);  
    props.setProperty("ssl", "true");  
  
    String url = "jdbc:postgresql://" + "10.10.0.13" + ':' + "8000" + '/' + "postgresgaussdb";  
    Connection conn = null;  
  
    try {  
        // Load the driver.  
        Class.forName(driver);  
    } catch (Exception e) {  
        e.printStackTrace();  
        return null;  
    }  
    try {  
        // Create a connection.  
        conn = DriverManager.getConnection(url, props);  
        System.out.println("Connection succeed!");  
    } catch (SQLException throwables) {  
        throwables.printStackTrace();  
        return null;  
    }  
    return conn;  
}
```

Step 8 Run SQL statements.

1. Run the following command to create a statement object:
Statement stmt = con.createStatement();
2. Run the following command to execute the statement object:
int rc = stmt.executeUpdate("CREATE TABLE tab1(id INTEGER, name VARCHAR(32));");
3. Run the following command to release the statement object:
stmt.close();

Step 9 Call `close()` to close the connection.

----End

Sample Code

This code sample illustrates how to develop applications based on the JDBC API provided by GaussDB(DWS).

NOTE

Before completing the following example, you need to create a stored procedure. For details, see "Guide: JDBC- or ODBC-Based Development" in the *Data Warehouse Service Developer Guide*.

```
create or replace procedure testproc
(
  psv_in1 in integer,
  psv_in2 in integer,
  psv_inout in out integer
)
as
begin
  psv_inout := psv_in1 + psv_in2 + psv_inout;
end;
/
```

```
//DBtest.java
//gsjdbc4.jar is used as an example.
//Demonstrate the main steps for JDBC development, including creating databases, creating tables, and
inserting data.

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

import java.sql.Statement;
import java.sql.CallableStatement;
import java.sql.Types;

public class DBTest {
  //Create a database connection. Replace the following IP address and database with the actual database
  connection address and database name.
  public static Connection GetConnection(String username, String passwd) {
    String driver = "org.postgresql.Driver";
    String sourceURL = "jdbc:postgresql://10.10.0.13:8000/database";
    Connection conn = null;
    try {
      // Load the database driver.
      Class.forName(driver).newInstance();
    } catch (Exception e) {
      e.printStackTrace();
      return null;
    }

    try {
      //Create a database connection.
      conn = DriverManager.getConnection(sourceURL, username, passwd);
      System.out.println("Connection succeed!");
    } catch (Exception e) {
      e.printStackTrace();
      return null;
    }

    return conn;
  };
}
```

```
//Run the common SQL statements to create table customer_t1.
public static void CreateTable(Connection conn) {
    Statement stmt = null;
    try {
        stmt = conn.createStatement();

        //Run the common SQL statements.
        int rc = stmt
            .executeUpdate("CREATE TABLE customer_t1(c_customer_sk INTEGER, c_customer_name
VARCHAR(32));");

        stmt.close();
    } catch (SQLException e) {
        if (stmt != null) {
            try {
                stmt.close();
            } catch (SQLException e1) {
                e1.printStackTrace();
            }
        }
        e.printStackTrace();
    }
}

//Run the prepared statements and insert data in batches.
public static void BatchInsertData(Connection conn) {
    PreparedStatement pst = null;

    try {
        //Generate the prepared statements.
        pst = conn.prepareStatement("INSERT INTO customer_t1 VALUES (?,?)");
        for (int i = 0; i < 3; i++) {
            //Add parameters.
            pst.setInt(1, i);
            pst.setString(2, "data " + i);
            pst.addBatch();
        }
        //Execute batch processing.
        pst.executeBatch();
        pst.close();
    } catch (SQLException e) {
        if (pst != null) {
            try {
                pst.close();
            } catch (SQLException e1) {
                e1.printStackTrace();
            }
        }
        e.printStackTrace();
    }
}

//Run the precompiled statement to update the data.
public static void ExecPreparedSQL(Connection conn) {
    PreparedStatement pstmt = null;
    try {
        pstmt = conn
            .prepareStatement("UPDATE customer_t1 SET c_customer_name = ? WHERE c_customer_sk = 1");
        pstmt.setString(1, "new Data");
        int rowcount = pstmt.executeUpdate();
        pstmt.close();
    } catch (SQLException e) {
        if (pstmt != null) {
            try {
                pstmt.close();
            } catch (SQLException e1) {
                e1.printStackTrace();
            }
        }
    }
}
```

```
e.printStackTrace();
}
}

//Execute the storage procedure.
public static void ExecCallableSQL(Connection conn) {
    CallableStatement cstmt = null;
    try {

        cstmt=conn.prepareCall("{? = CALL TESTPROC(?,?,?)}");
        cstmt.setInt(2, 50);
        cstmt.setInt(1, 20);
        cstmt.setInt(3, 90);
        cstmt.registerOutParameter(4, Types.INTEGER); //Register a parameter of the out type. Its value is an
integer.
        cstmt.execute();
        int out = cstmt.getInt(4); //Obtain the out parameter.
        System.out.println("The CallableStatment TESTPROC returns:"+out);
        cstmt.close();
    } catch (SQLException e) {
        if (cstmt != null) {
            try {
                cstmt.close();
            } catch (SQLException e1) {
                e1.printStackTrace();
            }
        }
        e.printStackTrace();
    }
}

/**
 * Main program, which gradually invokes each static method.
 * @param args
 */
public static void main(String[] args) {
    //Create a database connection. Replace User and Password with the actual database user name and
password.
    Connection conn = GetConnection("User", "Password");

    //Create a table.
    CreateTable(conn);

    //Insert data in batches.
    BatchInsertData(conn);

    //Run the precompiled statement to update the data.
    ExecPreparedSQL(conn);

    //Execute the storage procedure.
    ExecCallableSQL(conn);

    //Close the database connection.
    try {
        conn.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}
```

6.5.4 Configuring JDBC to Connect to a Cluster (Load Balancing Mode)

Context

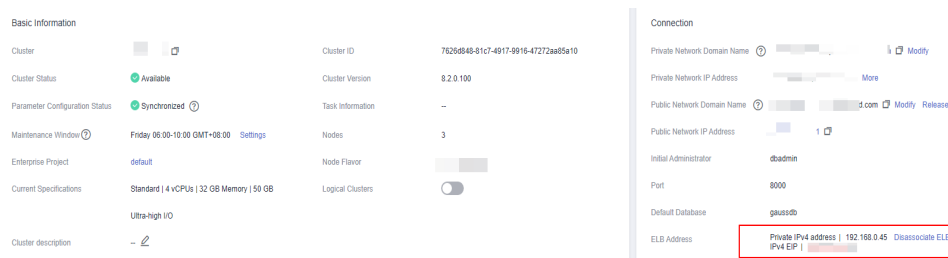
If you use JDBC to connect to only one CN in the cluster, this CN may be overloaded and other CN resources wasted. It also incurs single-node failure risks.

To avoid these problems, you can use JDBC to connect to multiple CNs. Two modes are available:

- Connection using ELB: An ELB distributes access traffic to multiple ECSs for traffic control based on forwarding policies. It improves the fault tolerance capability of application programs.
- Connection in multi-host mode: Use JDBC to configure multiple nodes, which is similar to ELB.

Method 1: Using ELB to Connect to a Cluster

Step 1 Obtain the Elastic Load Balance address. On the console, go to the details page of a cluster and obtain the ELB IP address.



Step 2 Configure the driver. For details, see [Downloading the JDBC or ODBC Driver](#).

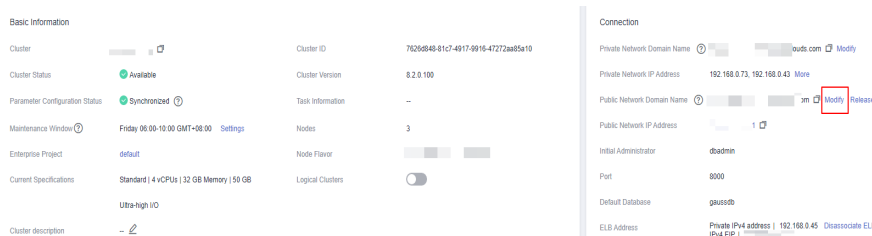
Step 3 Obtain the database connection.

```
private static final String USER_NAME = "dbadmin";
private static final String PASSWORD = "password";
// jdbc:postgresql://ELB_IP:PORT/dbName"
private static final String URL = "jdbc:postgresql://100.95.153.169:8000/gaussdb";
private static Properties properties = new Properties();
static {
    properties.setProperty("user", USER_NAME);
    properties.setProperty("password", PASSWORD);
}
/**
 * Obtain the database connection.
 */
public static Connection getConnection() {
    Connection connection = null;
    try {
        connection = DriverManager.getConnection(URL, properties);
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return connection;
}
```

----End

Method 2: Connecting to the Cluster in Multi-host Mode

Step 1 Obtain the EIP. Go to the details page of a cluster on the console and obtain the EIP.



Step 2 Configure the driver. For details, see [Downloading the JDBC or ODBC Driver](#).

Step 3 Obtain the database connection.

```
private static final String USER_NAME = "dbadmin";
private static final String PASSWORD = "password";
// jdbc:postgresql://host1:port1,host2:port2/dbName
private static final String URL = "jdbc:postgresql://
100.95.146.194:8000,100.95.148.220:8000,100.93.0.221:8000/gaussdb?loadBalanceHosts=true";
private static Properties properties = new Properties();
static {
    properties.setProperty("user", USER_NAME);
    properties.setProperty("password", PASSWORD);
}
/**
 * Obtain the database connection.
 */
public static Connection getConnection() {
    Connection connection = null;
    try {
        connection = DriverManager.getConnection(URL, properties);
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return connection;
}
```

----End

6.5.5 Configuring JDBC to Connect to a Cluster (IAM Authentication Mode)

Overview

GaussDB(DWS) allows you to access databases using IAM authentication. When you use the JDBC application program to connect to a cluster, set the IAM username, credential, and other information as you configure the JDBC URL. After doing this, when you try to access a database, the system will automatically generate a temporary credential and a connection will be set up.

NOTE

- Currently, only clusters 1.3.1 and later versions and their corresponding JDBC drivers can access the databases in IAM authentication mode. Download the JDBC driver. For details, see [Downloading the JDBC or ODBC Driver](#).

IAM supports two types of user credential: password and Access Key ID/Secret Access Key (AK/SK). JDBC connection requires the latter.

The IAM account you use to access a database must be granted with the **DWS Database Access** permission. Only users with both the **DWS Administrator** and **DWS Database Access** permissions can connect to GaussDB(DWS) databases using the temporary database user credentials generated based on IAM users.

The **DWS Database Access** permission can only be granted to user groups. Ensure that your IAM account is in a user group with this permission.

On IAM, only users in the **admin** group have the permissions to manage users. This requires that your IAM account be in the **admin** user group. Otherwise, contact the IAM account administrator to grant your IAM account this permission.

The process of accessing a database is as follows:

1. [Granting an IAM Account the GaussDB\(DWS\) Database Access Permission](#)
2. [Creating an IAM User Credential](#)
3. [Configuring the JDBC Connection to Connect to a Cluster Using IAM Authentication](#)

Granting an IAM Account the GaussDB(DWS) Database Access Permission

Step 1 Log in to the management console. In the service list, choose **Management & Governance > Identity and Access Management** to enter the IAM management console.

Step 2 Modify the user group to which your IAM user belongs. Set a policy for, grant the **DWS Database Access** permission to, and add your IAM user to it.

Only users in the **admin** user group of IAM can perform this step. In IAM, only users in the **admin** user group can manage users, including creating user groups and users and setting user group rights.

For details, see "User and User Group Management > Viewing or Modifying User Group Information" in the *Identity and Access Management User Guide*.

You can also create an IAM user group, and set a policy for, grant the **DWS Administrator** and **DWS Database Access** permissions to, and add your IAM user to it. For details, see "User and User Group Management > Creating a User Group" in the *Identity and Access Management User Guide*.

----End

Creating an IAM User Credential

You can log in to the management console to create an AK/SK pair or use an existing one.

Step 1 Log in to the management console.

Step 2 Move the cursor to the username in the upper right corner and choose **My Credentials**.

Step 3 Choose **Access Keys** to view the existing access keys. You can also click **Create Access Key** to create a new one.

The AK/SK pair is so important that you can download the private key file containing the AK/SK information only when you create the pair. On the

management console, you can only view the AKs. If you have not downloaded the file, obtain it from your administrator or create an AK/SK pair again.

 **NOTE**

Each user can create a maximum of two AK/SK pairs, which are valid permanently. To ensure account security, change your AK/SK pairs periodically and keep them safe.

----End

Configuring the JDBC Connection to Connect to a Cluster Using IAM Authentication

Configuring JDBC Connection Parameters

Table 6-9 Database connection parameters

Parameter	Description
url	<p>gsjdbc4.jar/gsjdbc200.jar database connection descriptor. The JDBC API does not provide the connection retry capability. You need to implement the retry processing in the service code. The URL example is as follows:</p> <pre>jdbc:dws:iam://dws-IAM-demo:/gaussdb? AccessKeyID=XXXXXXXXXXXXXXXXXXXX&SecretAccessKey=XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXX&DbUser=user_test&AutoCreate=true</pre> <p>JDBC URL parameters:</p> <ul style="list-style-type: none"> • jdbc:dws:iam is a prefix in the URL format. • dws-IAM-demo indicates the name of the cluster containing the database. • indicates the region where the cluster resides. JDBC accesses the GaussDB(DWS) cluster in the corresponding region and delivers the IAM certificate to the cluster for IAM user authentication. The GaussDB(DWS) service address has been recorded in the JDBC configuration file. For details about GaussDB(DWS) regions, visit Regions and Endpoints. • gaussdb indicates the name of the database to which you want to connect. • AccessKeyID and SecretAccessKey are the access key ID and secret access key corresponding to the IAM user specified by DbUser. • Set DbUser to the IAM username. Note that the current version does not support hyphens (-) in the IAM username. <ul style="list-style-type: none"> – If the user specified by DbUser exists in the database, the temporary user credential has the same permissions as the existing user. – If the user specified by DbUser does not exist in the database and the value of AutoCreate is true, a new user named by the value of DbUser is automatically created. The created user is a common database user by default. • Parameter AutoCreate is optional. The default value is false. This parameter indicates whether to automatically create a database user named by the value of DbUser in the database. <ul style="list-style-type: none"> – The value true indicates that a user is automatically created. If the user already exists, the user will not be created again. – The value false indicates that a user is not created. If the username specified by DbUser does not exist in the database, an error is returned.

Parameter	Description
info	<p>Database connection properties. Common properties include the following:</p> <ul style="list-style-type: none">• ssl: a boolean type. It indicates whether the SSL connection is used.• loglevel: an integer type. It sets the log amount recorded in DriverManager for LogStream or LogWriter. Currently, org.postgresql.Driver.DEBUG and org.postgresql.Driver.INFO logs are supported. If the value is 1, only org.postgresql.Driver.INFO (little information) is recorded. If the value is greater than or equal to 2, org.postgresql.Driver.DEBUG and org.postgresql.Driver.INFO logs are printed, and detailed log information is generated. Its default value is 0, which indicates that no logs are printed.• charSet: a string type. It indicates character sets used when data is sent from the database or the database receives data.• prepareThreshold: an integer type. It is used to determine the execution times of PreparedStatement before the information is converted into prepared statements on the server. The default value is 5.

Example

```
//The following uses gsjdbc4.jar as an example.  
// The following code encapsulates the database connection obtaining operations into an API. You can  
connect to the database by specifying the region where the cluster is located, cluster name, access key ID,  
secret access key, and the corresponding IAM username.  
public static Connection GetConnection(String clustername, String regionname, String AK, String SK,  
String username) {  
    String driver = "org.postgresql.Driver";  
    // Driver class.  
    String driver = "org.postgresql.Driver";  
    // Database connection descriptor.  
    String sourceURL = "jdbc:dws:iam://" + clustername + ":" + regionname + "/postgresgaussdb?" +  
"AccessKeyID=" +  
    AK + "&SecretAccessKey=" + SK + "&DbUser=" + username + "&autoCreate=true";  
  
    Connection conn = null;  
  
    try {  
        // Load the driver.  
        Class.forName(driver);  
    } catch (ClassNotFoundException e) {  
        return null;  
    }  
    try {  
        // Create a connection.  
        conn = DriverManager.getConnection(sourceURL);  
        System.out.println("Connection succeed!");  
    } catch (SQLException e) {  
        return null;  
    }  
    return conn;  
}
```

6.5.6 Using ODBC to Connect to a Cluster

In GaussDB(DWS), you can use an ODBC driver to connect to the database. The driver can connect to the database through an ECS on the Huawei Cloud platform or over the Internet.

For details about how to use the ODBC API, see the official document.

Prerequisites

- You have downloaded ODBC driver packages **dws_x.x.x_odbc_driver_for_xxx.zip** (for Linux) and **dws_odbc_driver_for_windows.zip** (for Windows). For details, see [Downloading the JDBC or ODBC Driver](#).
GaussDB(DWS) also supports open-source ODBC driver: PostgreSQL ODBC 09.01.0200 or later.
- You have downloaded the open-source unixODBC code file 2.3.0 from <https://sourceforge.net/projects/unixodbc/files/unixODBC/2.3.0/unixODBC-2.3.0.tar.gz/download>.
- You have downloaded the SSL certificate file. For details, see [Downloading an SSL Certificate](#).

Using an ODBC Driver to Connect to a Database (Linux)

Step 1 Upload the ODBC package and code file to the Linux environment and decompress them to the specified directory.

Step 2 Log in to the Linux environment as user **root**.

Step 3 Prepare **unixODBC**.

- Decompress the **unixODBC** code file.

```
tar -xvf unixODBC-2.3.0.tar.gz
```
- Compile the code file and install the driver.

```
cd unixODBC-2.3.0
./configure --enable-gui=no
make
make install
```

NOTE

- After the unixODBC is compiled and installed, the ***.so.2** library file will be in the installation directory. To create the ***.so.1** library file, change **LIB_VERSION** in the configure file to **1:0:0**.

```
LIB_VERSION="1:0:0"
```
- This driver dynamically loads the **libodbcinst.so.*** library files. If one of the library files is successfully loaded, the library file is loaded. The loading priority is **libodbcinst.so > libodbcinst.so.1 > libodbcinst.so.1.0.0 > libodbcinst.so.2 > libodbcinst.so.2.0.0**.

For example, a directory can be dynamically linked to **libodbcinst.so.1**, **libodbcinst.so.1.0.0**, and **libodbcinst.so.2**. The driver file loads **libodbcinst.so** first. If **libodbcinst.so** cannot be found in the current environment, the driver file searches for **libodbcinst.so.1**, which has a lower priority. After **libodbcinst.so.1** is loaded, the loading is complete.

Step 4 Replace the driver file. (This document uses the **dws_8.1.x_odbc_driver_for_x86_redhat.zip** package of Red Hat as an example.)

1. Decompress the **dws_8.1.x_odbc_driver_for_x86_redhat.zip** package.
unzip dws_8.1.x_odbc_driver_for_x86_redhat.zip
2. Copy all files in the **lib** directory to **/usr/local/lib**. If there are files with the same name, overwrite them.
3. Copy **psqlodbcw.la** and **psqlodbcw.so** in the **odbc/lib** directory to **/usr/local/lib**.

Step 5 Run the following command to modify the configuration of the driver file:

```
vi /usr/local/etc/odbcinst.ini
```

Copy the following content to the file:

```
[DWS]
Driver64=/usr/local/lib/psqlodbcw.so
```

The parameters are as follows:

- **[DWS]**: indicates the driver name. You can customize the name.
- **Driver64** or **Driver**: indicates the path where the dynamic library of the driver resides. For a 64-bit operating system, search for **Driver64** first. If **Driver64** is not configured, search for **Driver**.

Step 6 Run the following command to modify the data source file:

```
vi /usr/local/etc/odbc.ini
```

Copy the following content to the configuration file, save the modification, and exit.

```
[DWSODBC]
Driver=DWS
Servername=10.10.0.13
Database=gaussdb
Username=dbadmin
Password=password
Port=8000
Sslmode=allow
```

Parameter	Description	Example Value
[DSN]	Data source name.	[DWSODBC]
Driver	Driver name, corresponding to DriverName in odbcinst.ini .	Driver=DWS
Servername	IP address of the server. When the cluster is bound to an ELB, set this parameter to the IP address of the ELB.	Servername=10.10.0.13
Database	Name of the database to be connected to.	Database=gaussdb
Username	Database username.	Username=dbadmin
Password	Database user password.	Password= <i>password</i>
Port	Port number of the server.	Port=8000

Parameter	Description	Example Value
Sslmode	<p>SSL certification mode. This parameter is enabled for the cluster by default.</p> <p>Values and meanings:</p> <ul style="list-style-type: none">• disable: only tries to establish a non-SSL connection.• allow: tries establishing a non-SSL connection first, and then an SSL connection if the attempt fails.• prefer: tries establishing an SSL connection first, and then a non-SSL connection if the attempt fails.• require: only tries establishing an SSL connection. If there is a CA file, perform the verification according to the scenario in which the parameter is set to verify-ca.• verify-ca: tries establishing an SSL connection and checks whether the server certificate is issued by a trusted CA.• verify-full: not supported by GaussDB(DWS) <p>NOTE The SSL mode delivers higher security than the common mode. By default, the SSL function is enabled in a cluster to allow SSL or non-SSL connections from the client. You are advised to use the SSL mode when using ODBC to connect to a GaussDB (DWS) cluster.</p>	Sslmode=allow

 **NOTE**

You can view the values of **Servname** and **Port** on the GaussDB(DWS) management console. Log in to the GaussDB(DWS) management console and click **Client Connections**. In the **Data Warehouse Connection String** area, select the target cluster and obtain **Private Network Address** or **Public Network Address**. For details, see [Obtaining the Cluster Connection Address](#).

Step 7 Configure environment variables.

```
vi ~/.bashrc
```

Add the following information to the configuration file:

```
export LD_LIBRARY_PATH=/usr/local/lib/:$LD_LIBRARY_PATH
export ODBCYSINI=/usr/local/etc
export ODBCINI=/usr/local/etc/odbc.ini
```

Step 8 Import environment variables.

```
source ~/.bashrc
```

Step 9 Run the following commands to connect to the database:

```
/usr/local/bin/isql -v DWSODBC
```

If the following information is displayed, the connection is successful:

```
+-----+
| Connected!          |
|                    |
| sql-statement      |
| help [tablename]   |
| quit               |
|                    |
+-----+
SQL>
```

----End

Using an ODBC Driver to Connect to a Database (Windows)

Step 1 Decompress ODBC driver package **dws_odbc_driver_for_windows.zip** (for Windows) and install **psqlodbc.msi**.

Step 2 Decompress the SSL certificate package to obtain the certificate file.

You can choose to automatically or manually deploy the certificate based on your needs.

Automatic deployment:

Double-click the **sslcert_env.bat** file. The certificate is automatically deployed to a default location.

 **NOTE**

The **sslcert_env.bat** file ensures the purity of the certificate environment. When the **%APPDATA%\postgresql** directory exists, a message will be prompted asking you whether you want to remove related directories. If you want to remove related directories, back up files in the directory.

Manual deployment:

1. Create a new folder named **postgresql** in the **%APPDATA%** directory.
2. Copy files **client.crt**, **client.key**, **client.key.cipher**, and **client.key.rand** to the **%APPDATA%\postgresql** directory and change **client** in the file name to **postgres**. For example, change the name of **client.key** to **postgres.key**.
3. Copy **cacert.pem** to **%APPDATA%\postgresql** and change the name of **cacert.pem** to **root.crt**.

Step 3 Open Driver Manager.

GaussDB(DWS) provides 32-bit and 64-bit ODBC drivers. Choose the version suitable for your system when configuring the data source. (Assume the Windows system drive is drive C. If another disk drive is used, modify the path accordingly.)

- If you want to develop 32-bit programs in the 64-bit OS and have installed the 32-bit driver, open the 32-bit Driver Manager at **C:\Windows\SysWOW64\odbcad32.exe**.

Do not choose **Control Panel > System and Security > Administrative Tools > Data Sources (ODBC)** directly.

 **NOTE**

WOW64 is the acronym for Windows 32-bit on Windows 64-bit. **C:\Windows\SysWOW64** stores the 32-bit environment on a 64-bit system.

- If you want to develop 64-bit programs in the 64-bit OS and have installed the 64-bit driver, open the 64-bit Driver Manager at **C:\Windows\System32\odbcad32.exe**.

Do not choose **Control Panel > System and Security > Administrative Tools > Data Sources (ODBC)** directly.

 **NOTE**

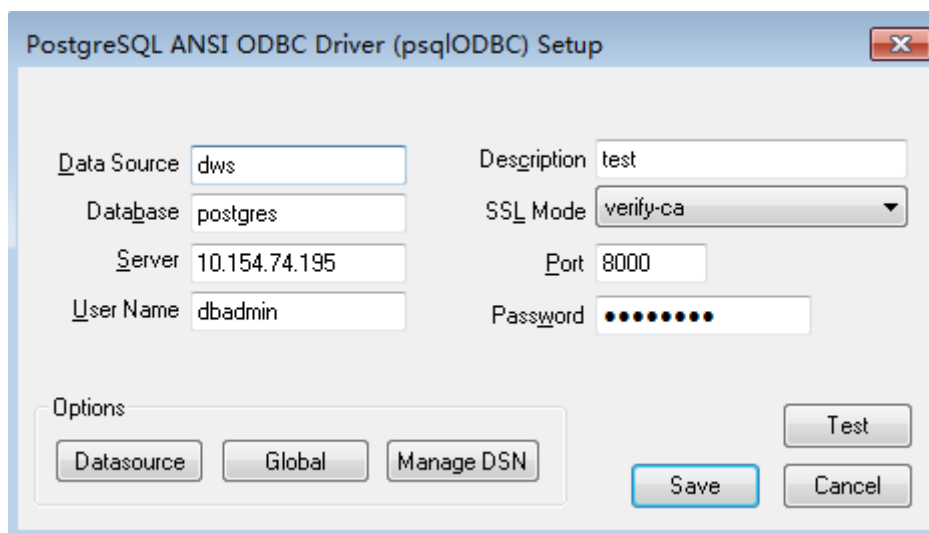
C:\Windows\System32 stores the environment consistent with the current OS. For technical details, see Windows technical documents.

- In a 32-bit OS, open **C:\Windows\System32\odbcad32.exe**.
Alternatively, click **Computer**, and choose **Control Panel**. Click **Administrative Tools** and click **Data Sources (ODBC)**.

Step 4 Configure a data source to be connected to.

1. On the **User DSN** tab, click **Add** and choose **PostgreSQL Unicode** for setup.

Figure 6-7 Configuring a data source to be connected to



You can view the values of **Server** and **Port** on the GaussDB(DWS) management console. Log in to the GaussDB(DWS) management console and click **Client Connections**. In the **Data Warehouse Connection String** area, select the target cluster and obtain **Private Network Address** or **Public**

Network Address. For details, see [Obtaining the Cluster Connection Address](#).

2. Click **Test** to verify that the connection is correct. If **Connection successful** is displayed, the connection is correct.

Step 5 Compile an ODBC sample program to connect to the data source.

The ODBC API does not provide the database connection retry capability. You need to implement the connection retry processing in the service code.

The sample code is as follows:

```
// This example shows how to obtain GaussDB(DWS) data through the ODBC driver.
// DBtest.c (compile with: libodbc.so)
#include <stdlib.h>
#include <stdio.h>
#include <sqlext.h>
#ifdef WIN32
#include <windows.h>
#endif
SQLHENV    V_OD_Env;    // Handle ODBC environment
SQLHSTMT   V_OD_hstmt; // Handle statement
SQLHDBC    V_OD_hdbc;  // Handle connection
char       typename[100];
SQLINTEGER value = 100;
SQLINTEGER V_OD_erg,V_OD_buffer,V_OD_err,V_OD_id;
int main(int argc,char *argv[])
{
    // 1. Apply for an environment handle.
    V_OD_erg = SQLAllocHandle(SQL_HANDLE_ENV,SQL_NULL_HANDLE,&V_OD_Env);
    if ((V_OD_erg != SQL_SUCCESS) && (V_OD_erg != SQL_SUCCESS_WITH_INFO))
    {
        printf("Error AllocHandle\n");
        exit(0);
    }
    // 2. Set environment attributes (version information).
    SQLSetEnvAttr(V_OD_Env, SQL_ATTR_ODBC_VERSION, (void*)SQL_OV_ODBC3, 0);
    // 3. Apply for a connection handle.
    V_OD_erg = SQLAllocHandle(SQL_HANDLE_DBC, V_OD_Env, &V_OD_hdbc);
    if ((V_OD_erg != SQL_SUCCESS) && (V_OD_erg != SQL_SUCCESS_WITH_INFO))
    {
        SQLFreeHandle(SQL_HANDLE_ENV, V_OD_Env);
        exit(0);
    }
    // 4. Set connection attributes.
    SQLSetConnectAttr(V_OD_hdbc, SQL_ATTR_AUTOCOMMIT, SQL_AUTOCOMMIT_ON, 0);
    // 5. Connect to a data source. You do not need to enter the username and password if you have
    // configured them in the odbc.ini file. If you have not configured them, specify the name and password of
    // the user who wants to connect to the database in the SQLConnect function.
    V_OD_erg = SQLConnect(V_OD_hdbc, (SQLCHAR*) "gaussdb", SQL_NTS,
        (SQLCHAR*) "", SQL_NTS, (SQLCHAR*) "", SQL_NTS);
    if ((V_OD_erg != SQL_SUCCESS) && (V_OD_erg != SQL_SUCCESS_WITH_INFO))
    {
        printf("Error SQLConnect %d\n",V_OD_erg);
        SQLFreeHandle(SQL_HANDLE_ENV, V_OD_Env);
        exit(0);
    }
    printf("Connected !\n");
    // 6. Set statement attributes.
    SQLSetStmtAttr(V_OD_hstmt,SQL_ATTR_QUERY_TIMEOUT,(SQLPOINTER *)3,0);
    // 7. Apply for a statement handle.
    SQLAllocHandle(SQL_HANDLE_STMT, V_OD_hdbc, &V_OD_hstmt);
    // 8. Executes an SQL statement directly.
    SQLExecDirect(V_OD_hstmt,"drop table IF EXISTS testtable",SQL_NTS);
    SQLExecDirect(V_OD_hstmt,"create table testtable(id int)",SQL_NTS);
    SQLExecDirect(V_OD_hstmt,"insert into testtable values(25)",SQL_NTS);
    // 9. Prepare for execution.
    SQLPrepare(V_OD_hstmt,"insert into testtable values(?)",SQL_NTS);
```

```
// 10. Bind parameters.
SQLBindParameter(V_OD_hstmt,1,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_INTEGER,0,0,
                &value,0,NULL);
// 11. Execute the ready statement.
SQLExecute(V_OD_hstmt);
SQLExecDirect(V_OD_hstmt,"select id from testtable",SQL_NTS);
// 12. Obtain the attributes of a certain column in the result set.
SQLColAttribute(V_OD_hstmt,1,SQL_DESC_TYPE,typename,100,NULL,NULL);
printf("SQLColAttribute %s\n",typename);
// 13. Bind the result set.
SQLBindCol(V_OD_hstmt,1,SQL_C_SLONG, (SQLPOINTER)&V_OD_buffer,150,
           (SQLLEN *)&V_OD_err);
// 14. Collect data using SQLFetch.
V_OD_erg=SQLFetch(V_OD_hstmt);
// 15. Obtain and return data using SQLGetData.
while(V_OD_erg != SQL_NO_DATA)
{
    SQLGetData(V_OD_hstmt,1,SQL_C_SLONG,(SQLPOINTER)&V_OD_id,0,NULL);
    printf("SQLGetData ----ID = %d\n",V_OD_id);
    V_OD_erg=SQLFetch(V_OD_hstmt);
};
printf("Done !\n");
// 16. Disconnect from the data source and release handles.
SQLFreeHandle(SQL_HANDLE_STMT,V_OD_hstmt);
SQLDisconnect(V_OD_hdbc);
SQLFreeHandle(SQL_HANDLE_DBC,V_OD_hdbc);
SQLFreeHandle(SQL_HANDLE_ENV, V_OD_Env);
return(0);
}
```

----End

6.6 Using the Third-Party Function Library psycopg2 of Python to Connect to a Cluster

After creating a data warehouse cluster and using the third-party function library psycopg2 to connect to the cluster, you can use Python to access GaussDB(DWS) and perform various operations on data tables.

Preparations Before Connecting to a Cluster

- An EIP has been bound to the data warehouse cluster.
- You have obtained the administrator username and password for logging in to the database in the data warehouse cluster.

MD5 algorithms may be vulnerable to collision attacks and cannot be used for password verification. Currently, GaussDB(DWS) uses the default security design. By default, MD5 password verification is disabled, and this may cause failures of connections from open source clients. You are advised to set **password_encryption_type** to **1**. For details, see "Modifying Database Parameters" in *User Guide*.

 NOTE

- For security purposes, GaussDB(DWS) no longer uses MD5 to store password digests by default. As a result, the open-source drivers and clients may fail to connect to the database. To use the MD5 algorithm used in an open-source protocol, you must modify your password policy and create a new user, or change the password of an existing user.
- The database stores the hash digest of passwords instead of password text. During password verification, the system compares the hash digest with the password digest sent from the client (salt operations are involved). If you change your cryptographic algorithm policy, the database cannot generate a new hash digest for your existing password. For connectivity purposes, you must manually change your password or create a new user. The new password will be encrypted using the hash algorithm and stored for authentication in the next connection.
- You have obtained the public network address, including the IP address and port number in the data warehouse cluster. For details, see [Obtaining the Cluster Connection Address](#).
- You have installed the third-party function library psycopg2. Download address: <https://pypi.org/project/psycopg2/>. For details about installation and deployment, see <https://www.psycopg.org/install/>.

 NOTE

- In CentOS and Red Hat OS, run the following **yum** command:
yum install python-psycopg2
- psycopg2 depends on the libpq dynamic library of PostgreSQL (32-bit or 64-bit version, whichever matches the psycopg2 bit version). In Linux, you can run the **yum** command and do not need to install the library. Before using psycopg2 in Windows, you need to install libpq in either of the following ways:
 - Install PostgreSQL and configure the libpq, ssl, and crypto dynamic libraries in the environment variable **PATH**.
 - Install psqldb and use the libpq, ssl, and crypto dynamic libraries carried by the PostgreSQL ODBC driver.

Constraints

psycopg2 is a PostgreSQL-based client interface, and its functions are not fully supported by GaussDB(DWS). For details, see [Table 6-10](#).

 NOTE

The following APIs are supported based on Python 3.8.5 and psycopg 2.9.1.

Table 6-10 psycopg2 APIs supported by DWS

Class Name	Usage	Function/Member Variable	Yes	Remarks
connections	basic	<i>cursor(name=None, cursor_factory=None, scrollable=None, withhold=False)</i>	Y	-
		commit()	Y	-
		rollback()	Y	-

Class Name	Usage	Function/Member Variable	Yes	Remarks
		close()	Y	-
	Two-phase commit support methods	xid(<i>format_id, gtrid, bqual</i>)	Y	-
		tpc_begin(<i>xid</i>)	Y	-
		tpc_prepare()	N	The kernel does not support explicit PREPARE TRANSACTION .
		tpc_commit(<i>[xid]</i>)	Y	-
		tpc_rollback(<i>[xid]</i>)	Y	-
		tpc_recover()	Y	-
		closed	Y	-
		cancel()	Y	-
		reset()	N	DISCARD ALL is not supported.
		dsn	Y	-
	Transaction control methods and attributes.	set_session(<i>isolation_level=None, readonly=None, deferrable=None, autocommit=None</i>)	Y	The database does not support the setting of default_transaction_read_only in a session.
		autocommit	Y	-
		isolation_level	Y	-
		readonly	N	The database does not support the setting of default_transaction_read_only in a session.
		deferrable	Y	-

Class Name	Usage	Function/Member Variable	Yes	Remarks
		set_isolation_level(<i>level</i>)	Y	-
		encoding	Y	-
		set_client_encoding(enc)	Y	-
		notices	N	The database does not support listen/notify .
		notifies	Y	-
		cursor_factory	Y	-
		info	Y	-
		status	Y	-
		lobjct	N	The database does not support operations related to large objects.
	Methods related to asynchronous support	poll()	Y	-
		fileno()	Y	-
		isexecuting()	Y	-
	Interoperation with other C API modules	pgconn_ptr	Y	-
		get_native_connection()	Y	-
	informative methods of the native connection	get_transaction_status()	Y	-
protocol_version		Y	-	
server_version		Y	-	
get_backend_pid()		Y	The obtained PID is not the background PID, but the ID of the logical connection.	
get_parameter_status(parameter)		Y	-	

Class Name	Usage	Function/Member Variable	Yes	Remarks
		get_dsn_parameters()	Y	-
cursor	basic	description	Y	-
		close()	Y	-
		closed	Y	-
		connection	Y	-
		name	Y	-
		scrollable	N	The database does not support SCROLL CURSOR .
		withhold	N	The withhold cursor needs to be closed before the commit operation.
	Command s execution methods	execute(<i>query</i> , <i>vars=None</i>)	Y	-
		executemany(<i>query</i> , <i>vars_list</i>)	Y	-
		callproc(<i>procname</i> [, <i>parameters</i>])	Y	-
		mogrify(<i>operation</i> [, <i>parameters</i>])	Y	-
		setinputsizes(<i>sizes</i>)	Y	-
		fetchone()	Y	-
		fetchmany([<i>size=cursor.arraysize</i>])	Y	-
		fetchall()	Y	-
		scroll(<i>value</i> [, <i>mode='relative'</i>])	N	The database does not support SCROLL CURSOR .
		arraysize	Y	-
		itersize	Y	-
		rowcount	Y	-

Class Name	Usage	Function/Member Variable	Yes	Remarks
		rownumber	Y	-
		lastrowid	Y	-
		query	Y	-
		statusmessage	Y	-
		cast(<i>oid</i> , <i>s</i>)	Y	-
		tzinfo_factory	Y	-
		nextset()	Y	-
		setoutputsize(<i>size</i> [, <i>column</i>])	Y	-
	COPY-related methods	copy_from(<i>file</i> , <i>table</i> , <i>sep</i> =' t', <i>null</i> =' N', <i>size</i> =8192, <i>columns</i> =None)	Y	-
		copy_to(<i>file</i> , <i>table</i> , <i>sep</i> =' t', <i>null</i> =' N', <i>columns</i> =None)	Y	-
		copy_expert(<i>sql</i> , <i>file</i> , <i>size</i> =8192)	Y	-
	Interoperation with other C API modules	pgresult_ptr	Y	-

Using the Third-Party Function Library psycopg2 to Connect to a Cluster (Linux)

Step 1 Log in to the Linux environment as user **root**.

Step 2 Run the following command to create the **python_dws.py** file:

```
vi python_dws.py
```

Copy and paste the following content to the **python_dws.py** file:

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

from __future__ import print_function

import psycopg2

def create_table(connection):
    print("Begin to create table")
    try:
        cursor = connection.cursor()
        cursor.execute("drop table if exists test;"
            "create table test(id int, name text);")
```



```
        connection.commit()
    except psycopg2.ProgrammingError as e:
        print(e)
    else:
        print("Table created successfully")
        cursor.close()

def insert_data(connection):
    print("Begin to insert data")
    try:
        cursor = connection.cursor()
        cursor.execute("insert into test values(1,'number1');")
        cursor.execute("insert into test values(2,'number2');")
        cursor.execute("insert into test values(3,'number3');")
        connection.commit()
    except psycopg2.ProgrammingError as e:
        print(e)
    else:
        print("Insert data successfully")
        cursor.close()

def update_data(connection):
    print("Begin to update data")
    try:
        cursor = connection.cursor()
        cursor.execute("update test set name = 'numberupdated' where id=1;")
        connection.commit()
        print("Total number of rows updated :", cursor.rowcount)
        cursor.execute("select * from test order by 1;")
        rows = cursor.fetchall()
        for row in rows:
            print("id = ", row[0])
            print("name = ", row[1], "\n")
    except psycopg2.ProgrammingError as e:
        print(e)
    else:
        print("After Update, Operation done successfully")

def delete_data(connection):
    print("Begin to delete data")
    try:
        cursor = connection.cursor()
        cursor.execute("delete from test where id=3;")
        connection.commit()
        print("Total number of rows deleted :", cursor.rowcount)
        cursor.execute("select * from test order by 1;")
        rows = cursor.fetchall()
        for row in rows:
            print("id = ", row[0])
            print("name = ", row[1], "\n")
    except psycopg2.ProgrammingError as e:
        print(e)
    else:
        print("After Delete, Operation done successfully")

def select_data(connection):
    print("Begin to select data")
    try:
        cursor = connection.cursor()
        cursor.execute("select * from test order by 1;")
        rows = cursor.fetchall()
        for row in rows:
            print("id = ", row[0])
            print("name = ", row[1], "\n")
    except psycopg2.ProgrammingError as e:
```

```
print(e)
print("select failed")
else:
    print("Operation done successfully")
    cursor.close()

if __name__ == '__main__':
    try:
        conn = psycopg2.connect(host='10.154.70.231',
                                port='8000',
                                database='gaussdb', # Database to be connected
                                user='dbadmin',
                                password='password') # Database user password
    except psycopg2.DatabaseError as ex:
        print(ex)
        print("Connect database failed")
    else:
        print("Opened database successfully")
        create_table(conn)
        insert_data(conn)
        select_data(conn)
        update_data(conn)
        delete_data(conn)
        conn.close()
```

- Step 3** Change the public network address, cluster port number, database name, database username, and database password in the `python_dws.py` file based on the actual cluster information.

The `psycopg2` API does not provide the connection retry capability. You need to implement the retry processing in the service code.

```
conn = psycopg2.connect(host='10.154.70.231',
                        port='8000',
                        database='gaussdb', # Database to be connected
                        user='dbadmin',
                        password='password') # Database user password
```

- Step 4** Run the following command to connect to the cluster using the third-party function library `psycopg`:

```
python python_dws.py
```

----End

Using the Third-Party Function Library `psycopg2` to Connect to a Cluster (Windows)

- Step 1** In the Windows operating system, click the **Start** button, enter `cmd` in the search box, and click `cmd.exe` in the result list to open the command-line interface (CLI).

- Step 2** In the CLI, run the following command to create the `python_dws.py` file:

```
type nul> python_dws.py
```

Copy and paste the following content to the `python_dws.py` file:

```
#!/usr/bin/python
# -*- coding:UTF-8 -*-

from __future__ import print_function

import psycopg2

def create_table(connection):
```

```
print("Begin to create table")
try:
    cursor = connection.cursor()
    cursor.execute("drop table if exists test;"
                  "create table test(id int, name text);")
    connection.commit()
except psycopg2.ProgrammingError as e:
    print(e)
else:
    print("Table created successfully")
    cursor.close()

def insert_data(connection):
    print("Begin to insert data")
    try:
        cursor = connection.cursor()
        cursor.execute("insert into test values(1,'number1');")
        cursor.execute("insert into test values(2,'number2');")
        cursor.execute("insert into test values(3,'number3');")
        connection.commit()
    except psycopg2.ProgrammingError as e:
        print(e)
    else:
        print("Insert data successfully")
        cursor.close()

def update_data(connection):
    print("Begin to update data")
    try:
        cursor = connection.cursor()
        cursor.execute("update test set name = 'numberupdated' where id=1;")
        connection.commit()
        print("Total number of rows updated :", cursor.rowcount)
        cursor.execute("select * from test order by 1;")
        rows = cursor.fetchall()
        for row in rows:
            print("id = ", row[0])
            print("name = ", row[1], "\n")
    except psycopg2.ProgrammingError as e:
        print(e)
    else:
        print("After Update, Operation done successfully")

def delete_data(connection):
    print("Begin to delete data")
    try:
        cursor = connection.cursor()
        cursor.execute("delete from test where id=3;")
        connection.commit()
        print("Total number of rows deleted :", cursor.rowcount)
        cursor.execute("select * from test order by 1;")
        rows = cursor.fetchall()
        for row in rows:
            print("id = ", row[0])
            print("name = ", row[1], "\n")
    except psycopg2.ProgrammingError as e:
        print(e)
    else:
        print("After Delete, Operation done successfully")

def select_data(connection):
    print("Begin to select data")
    try:
        cursor = connection.cursor()
        cursor.execute("select * from test order by 1;")
```

```
rows = cursor.fetchall()
for row in rows:
    print("id = ", row[0])
    print("name = ", row[1], "\n")
except psycopg2.ProgrammingError as e:
    print(e)
    print("select failed")
else:
    print("Operation done successfully")
    cursor.close()

if __name__ == '__main__':
    try:
        conn = psycopg2.connect(host='10.154.70.231',
                                port='8000',
                                database='postgresgaussdb', # Database to be connected
                                user='dbadmin',
                                password='password') # Database user password
    except psycopg2.DatabaseError as ex:
        print(ex)
        print("Connect database failed")
    else:
        print("Opened database successfully")
        create_table(conn)
        insert_data(conn)
        select_data(conn)
        update_data(conn)
        delete_data(conn)
        conn.close()
```

Step 3 Change the public network address, cluster port number, database name, database username, and database password in the `python_dws.py` file based on the actual cluster information.

```
conn = psycopg2.connect(host='10.154.70.231',
                        port='8000',
                        database='gaussdb', # Database to be connected
                        user='dbadmin',
                        password='password') # Database user password
```

Step 4 On the CLI, run the following command to use `psycopg2` to connect to the cluster:

```
python python_dws.py
```

----End

Why CN Retry Is Not Supported When `psycopg2` Is Connected to a Cluster?

With the CN retry feature, GaussDB(DWS) retries a statement that failed to be executed and identifies the failure type. However, in a session connected using `psycopg2`, a failed SQL statement will report an error and stop to be executed. In a primary/standby switchover, if a failed SQL statement is not retried, the following error will be reported. If the switchover is complete during an automatic retry, the correct result will be returned.

```
psycopg2.errors.ConnectionFailure: pooler: failed to create 1 connections, Error Message: remote node dn_6003_6004, detail: could not connect to server: Operation now in progress
```

Error causes:

1. `psycopg2` sends the **BEGIN** statement to start a transaction before sending an SQL statement.
2. CN retry does not support statements in transaction blocks.

Solution:

- In synchronous connection mode, end the transaction started by the driver.
cursor = conn.cursor()
End the transaction started by the driver.
cursor.execute("end; select * from test order by 1;")
rows = cursor.fetchall()
- Start a transaction in an asynchronous connection. For details, visit the PyScopg official website at: <https://www.psycopg.org/docs/advanced.html?highlight=async>

```
#!/usr/bin/env python3
# -*- encoding=utf-8 -*-

import psycopg2
import select

# Wait function provided by psycopg2 in asynchronous connection mode
#For details, see https://www.psycopg.org/docs/advanced.html?highlight=async.
def wait(conn):
    while True:
        state = conn.poll()
        if state == psycopg2.extensions.POLL_OK:
            break
        elif state == psycopg2.extensions.POLL_WRITE:
            select.select([], [conn.fileno()], [])
        elif state == psycopg2.extensions.POLL_READ:
            select.select([conn.fileno()], [], [])
        else:
            raise psycopg2.OperationalError("poll() returned %s" % state)

def psycopg2_cnretry_sync():
    # Create a connection.
    conn = psycopg2.connect(host='10.154.70.231',
                           port='8000',
                           database='gaussdb', # Database to be connected
                           user='dbadmin',
                           password='password', # Database user password
                           async=1) # Use the asynchronous connection mode.

    wait(conn)

    # Execute a query.
    cursor = conn.cursor()
    cursor.execute("select * from test order by 1;")
    wait(conn)
    rows = cursor.fetchall()
    for row in rows:
        print(row[0], row[1])

    # Close the connection.
    conn.close()

if __name__ == '__main__':
    psycopg2_cnretry_async()
```

6.7 Using the Python Library PyGreSQL to Connect to a Cluster

After creating a data warehouse cluster and using the third-party function library PyGreSQL to connect to the cluster, you can use Python to access GaussDB(DWS) and perform various operations on data tables.

Preparations Before Connecting to a Cluster

- An EIP has been bound to the data warehouse cluster.

- You have obtained the administrator username and password for logging in to the database in the data warehouse cluster.

MD5 algorithms may be vulnerable to collision attacks and cannot be used for password verification. Currently, GaussDB(DWS) uses the default security design. By default, MD5 password verification is disabled, and this may cause failures of connections from open source clients. You are advised to set **password_encryption_type** to **1**. For details, see "Modifying Database Parameters" in *User Guide*.

NOTE

- For security purposes, GaussDB(DWS) no longer uses MD5 to store password digests by default. As a result, the open-source drivers and clients may fail to connect to the database. To use the MD5 algorithm used in an open-source protocol, you must modify your password policy and create a new user, or change the password of an existing user.
- The database stores the hash digest of passwords instead of password text. During password verification, the system compares the hash digest with the password digest sent from the client (salt operations are involved). If you change your cryptographic algorithm policy, the database cannot generate a new hash digest for your existing password. For connectivity purposes, you must manually change your password or create a new user. The new password will be encrypted using the hash algorithm and stored for authentication in the next connection.
- You have obtained the public network address, including the IP address and port number in the data warehouse cluster. For details, see [Obtaining the Cluster Connection Address](#).
- You have installed the third-party function library PyGreSQL.
Download address: <http://www.pygresql.org/download/index.html>
- For details about the installation and deployment operations, see <http://www.pygresql.org/contents/install.html>

NOTE

- In CentOS and Red Hat OS, run the following **yum** command:

```
yum install PyGreSQL
```
- PyGreSQL depends on the libpq dynamic library of PostgreSQL (32-bit or 64-bit version, whichever matches the PyGreSQL bit version). In Linux, you can run the **yum** command and do not need to install the library. Before using PyGreSQL in Windows, you need to install libpq in either of the following ways:
 - Install PostgreSQL and configure the libpq, ssl, and crypto dynamic libraries in the environment variable **PATH**.
 - Install **psqlodbc** and use the **libpq**, **ssl**, and **crypto** dynamic libraries carried by the PostgreSQL ODBC driver.

Constraints

PyGreSQL is a PostgreSQL-based client interface, and its functions are not fully supported by GaussDB(DWS). For details, see [Table 6-11](#).

NOTE

The following APIs are supported based on Python 3.8.5 and PyGreSQL 5.2.4.

Table 6-11 PyGreSQL APIs supported by DWS

PyGreSQL		Yes	Remarks
Module functions and constants	connect – Open a PostgreSQL connection	Y	-
	get_pqlib_version – get the version of libpq	Y	-
	get/set_defhost – default server host [DV]	Y	-
	get/set_defport – default server port [DV]	Y	-
	get/set_defopt – default connection options [DV]	Y	-
	get/set_defbase – default database name [DV]	Y	-
	get/set_defuser – default database user [DV]	Y	-
	get/set_defpasswd – default database password [DV]	Y	-
	escape_string – escape a string for use within SQL	Y	-
	escape_bytea – escape binary data for use within SQL	Y	-
	unescape_bytea – unescape data that has been retrieved as text	Y	-
	get/set_namedresult – conversion to named tuples	Y	-
	get/set_decimal – decimal type to be used for numeric values	Y	-
	get/set_decimal_point – decimal mark used for monetary values	Y	-
	get/set_bool – whether boolean values are returned as bool objects	Y	-
	get/set_array – whether arrays are returned as list objects	Y	-
	get/set_bytea_escaped – whether bytea data is returned escaped	Y	-
	get/set_jsondecode – decoding JSON format	Y	-
	get/set_cast_hook – fallback typecast function	Y	-
	get/set_datestyle – assume a fixed date style	Y	-
get/set_typecast – custom typecasting	Y	-	
cast_array/record – fast parsers for arrays and records	Y	-	
Type helpers	Y	-	

PyGreSQL		Yes	Remarks
	Module constants	Y	-
Connection - The connection object	query - execute a SQL command string	Y	-
	send_query - executes a SQL command string asynchronously	Y	-
	query_prepared - execute a prepared statement	Y	-
	prepare - create a prepared statement	Y	-
	describe_prepared - describe a prepared statement	Y	-
	reset - reset the connection	Y	-
	poll - completes an asynchronous connection	Y	-
	cancel - abandon processing of current SQL command	Y	-
	close - close the database connection	Y	-
	transaction - get the current transaction state	Y	-
	parameter - get a current server parameter setting	Y	-
	date_format - get the currently used date format	Y	-
	fileno - get the socket used to connect to the database	Y	-
	set_non_blocking - set the non-blocking status of the connection	Y	-
is_non_blocking - report the blocking status of the connection	Y	-	
getnotify - get the last notify from the server	N	The database does not support listen / notify .	

PyGreSQL		Yes	Remarks
	inserttable – insert a list into a table	Y	Use double quotation marks ("") to quote \n in the copy command .
	get/set_notice_receiver – custom notice receiver	Y	-
	putline – write a line to the server socket [DA]	Y	-
	getline – get a line from server socket [DA]	Y	-
	endcopy – synchronize client and server [DA]	Y	-
	locreate – create a large object in the database [LO]	N	Operations related to large objects
	getlo – build a large object from given oid [LO]	N	Operations related to large objects
	loimport – import a file to a large object [LO]	N	Operations related to large objects
	Object attributes	Y	-

PyGreSQL		Yes	Remarks
The DB wrapper class	Initialization	Y	-
	pkey – return the primary key of a table	Y	-
	get_databases – get list of databases in the system	Y	-
	get_relations – get list of relations in connected database	Y	-
	get_tables – get list of tables in connected database	Y	-
	get_attnames – get the attribute names of a table	Y	-
	has_table_privilege – check table privilege	Y	-
	get/set_parameter – get or set run-time parameters	Y	-
	begin/commit/rollback/savepoint/release – transaction handling	Y	-
	get – get a row from a database table or view	Y	-
	insert – insert a row into a database table	Y	-
	update – update a row in a database table	Y	-
	upsert – insert a row with conflict resolution	Y	-
	query – execute a SQL command string	Y	-
	query_formatted – execute a formatted SQL command string	Y	-
	query_prepared – execute a prepared statement	Y	-
	prepare – create a prepared statement	Y	-
	describe_prepared – describe a prepared statement	Y	-
	delete_prepared – delete a prepared statement	Y	-
	clear – clear row values in memory	Y	-
delete – delete a row from a database table	Y	A tuple must have unique key or primary key.	

PyGreSQL		Yes	Remarks
	truncate – quickly empty database tables	Y	-
	get_as_list/dict – read a table as a list or dictionary	Y	-
	escape_literal/identifier/string/bytea – escape for SQL	Y	-
	unescape_bytea – unescape data retrieved from the database	Y	-
	encode/decode_json – encode and decode JSON data	Y	-
	use_regtypes – determine use of regular type names	Y	-
	notification_handler – create a notification handler	N	The database does not support listen / notify .
	Attributes of the DB wrapper class	Y	-
Query methods	getresult – get query values as list of tuples	Y	-
	dictresult/dictiter – get query values as dictionaries	Y	-
	namedresult/namediter – get query values as named tuples	Y	-
	scalarresult/scalariter – get query values as scalars	Y	-
	one/onedict/onenamed/onescalar – get one result of a query	Y	-
	single/singledict/singlenamed/singlescalar – get single result of a query	Y	-
	listfields – list fields names of previous query result	Y	-
	fieldname, fieldnum – field name/number conversion	Y	-
	fieldinfo – detailed info about query result fields	Y	-
	ntuples – return number of tuples in query object	Y	-

PyGreSQL		Yes	Remarks
	memsize – return number of bytes allocated by query result	Y	-
LargeObject – Large Objects	open – open a large object	N	Operations related to large objects
	close – close a large object	N	Operations related to large objects
	read, write, tell, seek, unlink – file-like large object handling	N	Operations related to large objects
	size – get the large object size	N	Operations related to large objects
	export – save a large object to a file	N	Operations related to large objects

PyGreSQL		Yes	Remarks
	Object attributes	N	Operations related to large objects
The Notification Handler	Instantiating the notification handler	N	The database does not support listen / notify .
	Invoking the notification handler	N	The database does not support listen / notify .
	Sending notifications	N	The database does not support listen / notify .

PyGreSQL		Yes	Remarks
	Auxiliary methods	N	The database does not support listen / notify .
pgdb			
Module functions and constants	connect - Open a PostgreSQL connection	Y	-
	get/set/reset_typecast - Control the global typecast functions	Y	-
	Module constants	Y	-
	Errors raised by this module	Y	-
Connection - The connection object	close - close the connection	Y	-
	commit - commit the connection	Y	-
	rollback - roll back the connection	Y	-
	cursor - return a new cursor object	Y	-
	Attributes that are not part of the standard	Y	-
Cursor - The cursor object	description - details regarding the result columns	Y	-
	rowcount - number of rows of the result	Y	-
	close - close the cursor	Y	-
	execute - execute a database operation	Y	-
	executemany - execute many similar database operations	Y	-
	callproc - Call a stored procedure	Y	-
	fetchone - fetch next row of the query result	Y	-
	fetchmany - fetch next set of rows of the query result	Y	-
	fetchall - fetch all rows of the query result	Y	-
	arraysize - the number of rows to fetch at a time	Y	-

PyGreSQL		Yes	Remarks
	Methods and attributes that are not part of the standard	Y	-
Type – Type objects and construct ors	Type constructors	Y	-
	Type objects	Y	-

Using the Third-Party Function Library PyGreSQL to Connect to a Cluster (Linux)

Step 1 Log in to the Linux environment as user **root**.

Step 2 Run the following command to create the **python_dws.py** file:

```
vi python_dws.py
```

Copy and paste the following content to the **python_dws.py** file:

```
#!/usr/bin/env python3
# -*- encoding:utf-8 -*-

from __future__ import print_function

import pg

def create_table(connection):
    print("Begin to create table")
    try:
        connection.query("drop table if exists test;"
                        "create table test(id int, name text);")
    except pg.InternalError as e:
        print(e)
    else:
        print("Table created successfully")

def insert_data(connection):
    print("Begin to insert data")
    try:
        connection.query("insert into test values(1,'number1');")
        connection.query("insert into test values(2,'number2');")
        connection.query("insert into test values(3,'number3');")
    except pg.InternalError as e:
        print(e)
    else:
        print("Insert data successfully")

def update_data(connection):
    print("Begin to update data")
    try:
        result = connection.query("update test set name = 'numberupdated' where id=1;")
        print("Total number of rows updated :", result)
        result = connection.query("select * from test order by 1;")
        rows = result.getresult()
        for row in rows:
```

```
        print("id = ", row[0])
        print("name = ", row[1], "\n")
    except pg.InternalError as e:
        print(e)
    else:
        print("After Update, Operation done successfully")

def delete_data(connection):
    print("Begin to delete data")
    try:
        result = connection.query("delete from test where id=3;")
        print("Total number of rows deleted :", result)
        result = connection.query("select * from test order by 1;")
        rows = result.getresult()
        for row in rows:
            print("id = ", row[0])
            print("name = ", row[1], "\n")
    except pg.InternalError as e:
        print(e)
    else:
        print("After Delete, Operation done successfully")

def select_data(connection):
    print("Begin to select data")
    try:
        result = connection.query("select * from test order by 1;")
        rows = result.getresult()
        for row in rows:
            print("id = ", row[0])
            print("name = ", row[1])
    except pg.InternalError as e:
        print(e)
        print("select failed")
    else:
        print("Operation done successfully")

if __name__ == '__main__':
    try:
        conn = pg.DB(host='10.154.70.231',
                    port=8000,
                    dbname='gaussdb', # Database to be connected
                    user='dbadmin',
                    passwd='password') # Database user password
    except pg.InternalError as ex:
        print(ex)
        print("Connect database failed")
    else:
        print("Opened database successfully")
        create_table(conn)
        insert_data(conn)
        select_data(conn)
        update_data(conn)
        delete_data(conn)
        conn.close()
```

Alternatively, use the dbapi interface.

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

from __future__ import print_function

import pg
import pgdb

def create_table(connection):
```



```
print("Begin to create table")
try:
    cursor = connection.cursor()
    cursor.execute("drop table if exists test;"
                  "create table test(id int, name text);")
    connection.commit()
except pg.InternalError as e:
    print(e)
else:
    print("Table created successfully")
    cursor.close()

def insert_data(connection):
    print("Begin to insert data")
    try:
        cursor = connection.cursor()
        cursor.execute("insert into test values(1,'number1');")
        cursor.execute("insert into test values(2,'number2');")
        cursor.execute("insert into test values(3,'number3');")
        connection.commit()
    except pg.InternalError as e:
        print(e)
    else:
        print("Insert data successfully")
        cursor.close()

def update_data(connection):
    print("Begin to update data")
    try:
        cursor = connection.cursor()
        cursor.execute("update test set name = 'numberupdated' where id=1;")
        connection.commit()
        print("Total number of rows updated :", cursor.rowcount)
        cursor.execute("select * from test;")
        rows = cursor.fetchall()
        for row in rows:
            print("id = ", row[0])
            print("name = ", row[1], "\n")
    except pg.InternalError as e:
        print(e)
    else:
        print("After Update, Operation done successfully")

def delete_data(connection):
    print("Begin to delete data")
    try:
        cursor = connection.cursor()
        cursor.execute("delete from test where id=3;")
        connection.commit()
        print("Total number of rows deleted :", cursor.rowcount)
        cursor.execute("select * from test;")
        rows = cursor.fetchall()
        for row in rows:
            print("id = ", row[0])
            print("name = ", row[1], "\n")
    except pg.InternalError as e:
        print(e)
    else:
        print("After Delete, Operation done successfully")

def select_data(connection):
    print("Begin to select data")
    try:
        cursor = connection.cursor()
        cursor.execute("select * from test;")
```

```
rows = cursor.fetchall()
for row in rows:
    print("id = ", row[0])
    print("name = ", row[1], "\n")
except pg.InternalError as e:
    print(e)
    print("select failed")
else:
    print("Operation done successfully")
    cursor.close()

if __name__ == '__main__':
    try:
        conn = pgdb.connect(host='10.154.70.231',
                            port='8000',
                            database='gaussdb', # Database to be connected
                            user='dbadmin',
                            password='password') # Database user password
    except pg.InternalError as ex:
        print(ex)
        print("Connect database failed")
    else:
        print("Opened database successfully")
        create_table(conn)
        insert_data(conn)
        select_data(conn)
        update_data(conn)
        delete_data(conn)
        conn.close()
```

- Step 3** Change the public network address, cluster port number, database name, database username, and database password in the `python_dws.py` file based on the actual cluster information.

 **NOTE**

The PyGreSQL API does not provide the connection retry capability. You need to implement the retry processing in the service code.

```
conn = pgdb.connect(host='10.154.70.231',
                    port='8000',
                    database='gaussdb', # Database to be connected
                    user='dbadmin',
                    password='password') # Database user password
```

- Step 4** Run the following command to connect to the cluster using the third-party function library PyGreSQL:

```
python python_dws.py
```

----End

Using the Third-Party Function Library PyGreSQL to Connect to a Cluster (Windows)

- Step 1** In the Windows operating system, click the **Start** button, enter **cmd** in the search box, and click **cmd.exe** in the result list to open the command-line interface (CLI).

- Step 2** In the CLI, run the following command to create the `python_dws.py` file:

```
type nul> python_dws.py
```

Copy and paste the following content to the `python_dws.py` file:

```
#!/usr/bin/env python3
# *_ encoding:utf-8 *_
```

```
from __future__ import print_function

import pg

def create_table(connection):
    print("Begin to create table")
    try:
        connection.query("drop table if exists test;"
                          "create table test(id int, name text);")
    except pg.InternalError as e:
        print(e)
    else:
        print("Table created successfully")

def insert_data(connection):
    print("Begin to insert data")
    try:
        connection.query("insert into test values(1,'number1');")
        connection.query("insert into test values(2,'number2');")
        connection.query("insert into test values(3,'number3');")
    except pg.InternalError as e:
        print(e)
    else:
        print("Insert data successfully")

def update_data(connection):
    print("Begin to update data")
    try:
        result = connection.query("update test set name = 'numberupdated' where id=1;")
        print("Total number of rows updated :", result)
        result = connection.query("select * from test order by 1;")
        rows = result.getresult()
        for row in rows:
            print("id = ", row[0])
            print("name = ", row[1], "\n")
    except pg.InternalError as e:
        print(e)
    else:
        print("After Update, Operation done successfully")

def delete_data(connection):
    print("Begin to delete data")
    try:
        result = connection.query("delete from test where id=3;")
        print("Total number of rows deleted :", result)
        result = connection.query("select * from test order by 1;")
        rows = result.getresult()
        for row in rows:
            print("id = ", row[0])
            print("name = ", row[1], "\n")
    except pg.InternalError as e:
        print(e)
    else:
        print("After Delete, Operation done successfully")

def select_data(connection):
    print("Begin to select data")
    try:
        result = connection.query("select * from test order by 1;")
        rows = result.getresult()
        for row in rows:
            print("id = ", row[0])
            print("name = ", row[1])
    except pg.InternalError as e:
```

```
print(e)
print("select failed")
else:
    print("Operation done successfully")

if __name__ == '__main__':
    try:
        conn = pg.DB(host='10.154.70.231',
                    port=8000,
                    dbname='gaussdb', # Database to be connected
                    user='dbadmin',
                    passwd='password') # Database user password
    except pg.InternalError as ex:
        print(ex)
        print("Connect database failed")
    else:
        print("Opened database successfully")
        create_table(conn)
        insert_data(conn)
        select_data(conn)
        update_data(conn)
        delete_data(conn)
        conn.close()
```

Alternatively, use the dbapi interface.

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

from __future__ import print_function

import pg
import pgdb

def create_table(connection):
    print("Begin to create table")
    try:
        cursor = connection.cursor()
        cursor.execute("drop table if exists test;"
                      "create table test(id int, name text);")
        connection.commit()
    except pg.InternalError as e:
        print(e)
    else:
        print("Table created successfully")
        cursor.close()

def insert_data(connection):
    print("Begin to insert data")
    try:
        cursor = connection.cursor()
        cursor.execute("insert into test values(1,'number1');")
        cursor.execute("insert into test values(2,'number2');")
        cursor.execute("insert into test values(3,'number3');")
        connection.commit()
    except pg.InternalError as e:
        print(e)
    else:
        print("Insert data successfully")
        cursor.close()

def update_data(connection):
    print("Begin to update data")
    try:
        cursor = connection.cursor()
        cursor.execute("update test set name = 'numberupdated' where id=1;")
```

```
connection.commit()
print("Total number of rows updated :", cursor.rowcount)
cursor.execute("select * from test;")
rows = cursor.fetchall()
for row in rows:
    print("id = ", row[0])
    print("name = ", row[1], "\n")
except pg.InternalError as e:
    print(e)
else:
    print("After Update, Operation done successfully")

def delete_data(connection):
    print("Begin to delete data")
    try:
        cursor = connection.cursor()
        cursor.execute("delete from test where id=3;")
        connection.commit()
        print("Total number of rows deleted :", cursor.rowcount)
        cursor.execute("select * from test;")
        rows = cursor.fetchall()
        for row in rows:
            print("id = ", row[0])
            print("name = ", row[1], "\n")
    except pg.InternalError as e:
        print(e)
    else:
        print("After Delete, Operation done successfully")

def select_data(connection):
    print("Begin to select data")
    try:
        cursor = connection.cursor()
        cursor.execute("select * from test;")
        rows = cursor.fetchall()
        for row in rows:
            print("id = ", row[0])
            print("name = ", row[1], "\n")
    except pg.InternalError as e:
        print(e)
        print("select failed")
    else:
        print("Operation done successfully")
        cursor.close()

if __name__ == '__main__':
    try:
        conn = pgdb.connect(host='10.154.70.231',
                            port='8000',
                            database='gaussdb', # Database to be connected
                            user='dbadmin',
                            password='password') # Database user password
    except pg.InternalError as ex:
        print(ex)
        print("Connect database failed")
    else:
        print("Opened database successfully")
        create_table(conn)
        insert_data(conn)
        select_data(conn)
        update_data(conn)
        delete_data(conn)
        conn.close()
```

Step 3 Change the public network address, cluster port number, database name, database username, and database password in the `python_dws.py` file based on the actual cluster information.

The PyGreSQL API does not provide the connection retry capability. You need to implement the retry processing in the service code.

```
conn = pgdb.connect(host='10.154.70.231',
                    port='8000',
                    database='gaussdb', # Database to be connected
                    user='dbadmin',
                    password='password') # Database user password
```

Step 4 Run the following command to connect to the cluster using the third-party function library PyGreSQL:

```
python python_dws.py
```

----End

6.8 Managing Database Connections

Scenario

By default, a database supports a certain number of connections. Administrators can manage database connections to learn about the connection performance of the current database or increase the connection limit so that more users or applications can connect to the database at the same time.

Maximum Number of Connections

The number of connections supported by a cluster depends on its node flavor.

Table 6-12 Number of supported connections

Parameter	Description	Number of CN Connections	Number of DN Connections
max_connections	Specifies the maximum number of concurrent connections to the database.	800	Max (Number of vCPU cores/Number of DNs on a single node x 120 + 24, 5000)
max_pool_size	Specifies the maximum number of connections between the connection pool of a CN and another CN or DN.		

Parameter	Description	Number of CN Connections	Number of DN Connections
max_prepared_transactions	Specifies the maximum number of transactions that can stay in the prepared state simultaneously.		

Viewing the Maximum Number of Connections

Step 1 Use the SQL client tool to connect to the database in a cluster.

Step 2 Run the following command:

```
SHOW max_connections;
```

Information similar to the following is displayed, showing that the maximum number of database connections is **200** by default.

```
max_connections
-----
200
(1 row)
```

----End

Viewing the Number of Used Connections

Step 1 Use the SQL client tool to connect to the database in a cluster.

Step 2 View the number of connections in scenarios described in [Table 6-13](#).

NOTICE

Except for database and user names that are enclosed with double quotation marks (") during creation, uppercase letters are not allowed in the database and user names in the commands in the following table.

Table 6-13 Viewing the number of connections

Description	Command
View the maximum number of sessions connected to a specific user.	<p>Run the following command to view the maximum number of sessions connected to user dbadmin. <code>SELECT ROLNAME,ROLCONNLIMIT FROM PG_ROLES WHERE ROLNAME='dbadmin';</code></p> <p>Information similar to the following is displayed. -1 indicates that the number of sessions connected to user dbadmin is not limited.</p> <pre>rolname rolconnlimit -----+----- dwsadmin -1 (1 row)</pre>
View the number of session connections that have been used by a user.	<p>Run the following command to view the number of session connections that have been used by dbadmin. <code>SELECT COUNT(*) FROM V\$SESSION WHERE USERNAME='dbadmin';</code></p> <p>Information similar to the following is displayed. 1 indicates the number of session connections used by user dbadmin.</p> <pre>count ----- 1 (1 row)</pre>
View the maximum number of sessions connected to a specific database.	<p>Run the following command to view the upper limit of connections used by the database: <code>SELECT DATNAME,DATCONNLIMIT FROM PG_DATABASE WHERE DATNAME='';</code></p> <p>Information similar to the following is displayed. -1 indicates that the number of sessions connected to the database is not limited.</p> <pre>datname datconnlimit -----+----- -1 (1 row)</pre>
View the number of session connections that have been used by a database.	<p>Run the following command to view the number of session connections that have been used by the database: <code>SELECT COUNT(*) FROM PG_STAT_ACTIVITY WHERE DATNAME='';</code></p> <p>Information similar to the following is displayed. 1 indicates the number of session connections used by the database.</p> <pre>count ----- 1 (1 row)</pre>

Description	Command
View the number of session connections that have been used by all users.	Run the following command to view the number of session connections that have been used by all users: <pre>SELECT COUNT(*) FROM PG_STAT_ACTIVITY; count ----- 10 (1 row)</pre>

----End

7 Monitoring and Alarms

7.1 Monitoring Clusters Using Cloud Eye

Function

This section describes how to check cluster metrics on Cloud Eye. By monitoring cluster running metrics, you can identify the time when the database cluster is abnormal and analyze potential activity problems based on the database logs, improving database performance. This section describes the metrics that can be monitored by Cloud Eye as well as their namespaces and dimensions. You can use the management console or APIs provided by Cloud Eye to query the monitoring metrics and alarms generated by GaussDB(DWS). For details, see the *User Guide* and *API Reference* of Cloud Eye.

This section is organized as follows:

Namespace

SYS.DWS

Cluster Monitoring Metrics

With the GaussDB(DWS) monitoring metrics provided by Cloud Eye, you can obtain information about the cluster running status and performance. This information will provide a better understanding of the node-level information.

[Table 7-1](#) describes GaussDB(DWS) monitoring metrics.

Table 7-1 GaussDB(DWS) monitoring metrics

Metric ID	Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
dws001_shared_buffer_hit_ratio	Cache Hit Ratio	Ratio of requested data that already exists in the cache. It is the ratio of the amount of data that already exists in the cache to the total amount of requested data. A higher cache hit ratio means higher cache usage of the system, fewer times that data needs to be read from the disk or network, and faster system response speed. Unit: Percent	0% to 100%	Data warehouse cluster	4 minutes
dws002_in_memory_sort_ratio	In-memory Sort Ratio	Ratio of the extra memory space used by the sorting algorithm to the memory space occupied by the sorted data. In a merge sort, for example, the size of the merge buffer is often proportional to the size of the sorted data, so the in-memory ratio is usually between 10% and 50%. Unit: Percent	0% to 100%	Data warehouse cluster	4 minutes
dws003_physical_reads	File Reads	Total number of database file reads	> 0	Data warehouse cluster	4 minutes

Metric ID	Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
dws004_physical_writes	File Writes	Total number of database file writes	> 0	Data warehouse cluster	4 minutes
dws005_physical_reads_per_second	File Reads per Second	Number of database file reads per second	≥ 0	Data warehouse cluster	4 minutes
dws006_physical_writes_per_second	File Writes per Second	Number of database file writes per second	≥ 0	Data warehouse cluster	4 minutes
dws007_db_size	Data Volume	Total data volume of the database Unit: MB	≥ 0 MB	Data warehouse cluster	4 minutes
dws008_active_sql_count	Active SQL Count	Number of active SQLs in the database	≥ 0	Data warehouse cluster	4 minutes
dws009_session_count	Session Count	Number of sessions that access the database	≥ 0	Data warehouse cluster	4 minutes
dws010_cpu_usage	CPU Usage	CPU usage of each node in a cluster, in percentage	0% to 100%	Data warehouse node	1 minute
dws011_memory_usage	Memory Usage	Memory usage of each node in a cluster, in percentage	0% to 100%	Data warehouse node	1 minute
dws012_iops	IOPS	Number of I/O requests processed by each node in the cluster per second	≥ 0	Data warehouse node	1 minute

Metric ID	Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
dws013_bytes_in	Network Input Throughput	Data input to each node in the cluster per second over the network Unit: byte/s	≥ 0 bytes/s	Data warehouse node	1 minute
dws014_bytes_out	Network Output Throughput	Data sent to the network per second from each node in the cluster Unit: byte/s	≥ 0 bytes/s	Data warehouse node	1 minute
dws015_disk_usage	Disk Usage	Disk usage of each node in a cluster, in percentage	0% to 100%	Data warehouse node	1 minute
dws016_disk_total_size	Total Disk Size	Total disk space of each node in the cluster Unit: GB	100 to 2000 GB	Data warehouse node	1 minute
dws017_disk_used_size	Used Disk Space	Used disk space of each node in the cluster Unit: GB	0 to 3600 GB	Data warehouse node	1 minute
dws018_disk_read_throughput	Disk Read Throughput	Data volume read from each disk in the cluster per second Unit: byte/s	≥ 0 bytes/s	Data warehouse node	1 minute
dws019_disk_write_throughput	Disk Write Throughput	Data volume written to each disk in the cluster per second Unit: byte/s	≥ 0 bytes/s	Data warehouse node	1 minute
dws020_avg_disk_sec_per_read	Average Time per Disk Read	Average time used each time when a disk reads data Unit: second	> 0s	Data warehouse node	1 minute

Metric ID	Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
dws021_avg_disk_sec_per_write	Average Time per Disk Write	Average time used each time when data is written to a disk Unit: second	> 0s	Data warehouse node	1 minute
dws022_avg_disk_queue_length	Average Disk Queue Length	Average I/O queue length of a disk	≥ 0	Data warehouse node	1 minute
dws_024_dn_diskio_util	DN I/O usage	Average disk I/O usage of DNs in a cluster	0% to 100%	Data warehouse instance	1 minute

Dimensions


Key	Value
datastore_id	Data warehouse cluster ID
dws_instance_id	Data warehouse node ID

Cluster and Node Monitoring Information

Step 1 Log in to the GaussDB(DWS) management console and choose **Clusters > Dedicated Clusters**.

Step 2 View the cluster information. In the cluster list, click **View Metric** in the **Operation** column where a specific cluster resides. The Cloud Eye management console is displayed. By default, the cluster monitoring information on the Cloud Eye management console is displayed.

Additionally, you can specify a specific monitoring metric and the time range to view the performance curve.

Step 3 View the node information. Click  to return to the Cloud Eye management console. On the **Data Warehouse Nodes** tab page in the right pane, you can view metrics of each node in the cluster.

Additionally, you can specify a specific monitoring metric and the time range to view the performance curve.

Cloud Eye also supports the ability to compare the monitoring metrics of multiple nodes. For details, see [Comparing the Monitoring Metrics of Multiple Nodes](#).

----End

Comparing the Monitoring Metrics of Multiple Nodes

- Step 1** In the left navigation pane of the Cloud Eye management console, choose **Dashboard > Panels**.
- Step 2** On the page that is displayed, click **Create Panel**. In the displayed dialog box, enter the name and click **OK**.
- Step 3** Click **Add Graph** in the upper right corner.
- Step 4** In the displayed dialog box, configure the title and monitoring metrics.

NOTE


You can add multiple monitoring metrics by clicking **Add Metric**.

The following describes how to set parameters if you want to compare CPU usage of two nodes.

Table 7-2 Configuration example

Parameter	Example Value
Resource Type	DWS
Dimension	Data Warehouse Node
Monitored Object	dws-demo-dws-cn-cn-2-1 dws-demo-dws-cn-cn-1-1 dws-demo-dws-dn-1-1
Metric	CPU Usage

- Step 5** Click **OK**.

Then you can view the corresponding monitoring graph on the **Panels** page. Move the cursor to the graph and click  in the upper right corner to zoom in the graph and view detailed metric comparison data.

----End

7.2 Alarms

7.2.1 Alarm Management

Overview

Alarm management includes viewing and configuring alarm rules and subscribing to alarm information. Alarm rules display alarm statistics and details of the past week for users to view tenant alarms. In addition to providing a set of default GaussDB(DWS) alarm rules, this feature allows you to modify alarm thresholds based on your own services. GaussDB(DWS) alarm notifications are sent using the SMN service.

NOTE

- This feature supports only the database kernel of 8.1.1.200 and later.
- Currently, alarms cannot be categorized and managed by enterprise project.

Visiting the Alarms Page

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 In the navigation pane on the left, click **Alarms**.

Step 3 Go to the data warehouse alarm page. This page is divided into three areas:

- **Existing Alarm Statistics**

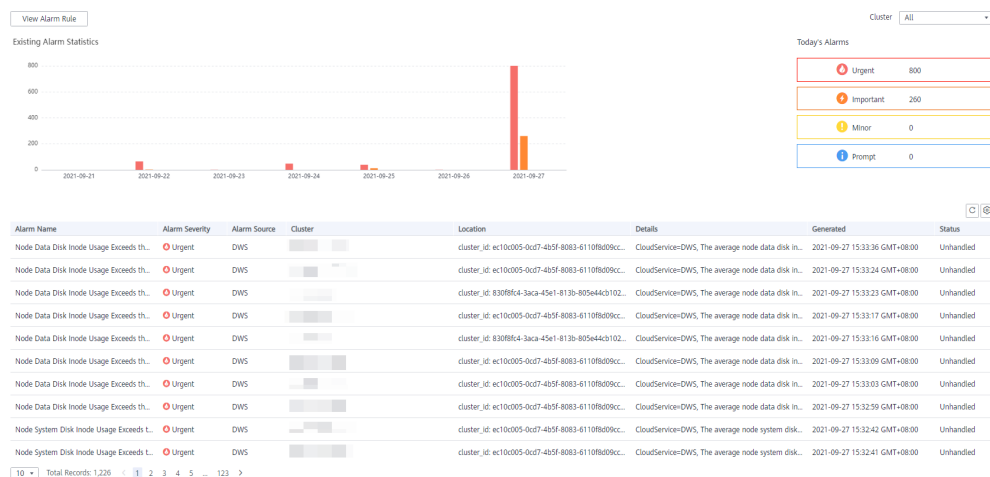
Statistics of the existing alarms in the past seven days are displayed by alarm severity in a bar chart. In this way, you can see clearly the number and category of the alarms generated in the past week.

- **Today's Alarms**

Statistics of the existing alarms on the current day are displayed by alarm severity in a list. In this way, you can see clearly the number and category of the unhandled alarms generated on the day.

- **Alarm details**

Details about all alarms, handled and unhandled, in the past seven days are displayed in a table for you to quickly locate faults, including the alarm name, alarm severity, cluster name, location, description, generation date, and status.



 NOTE

The alarm data displayed (a maximum of 30 days) is supported by the Event Service microservice.

----End

Alarm Types and Alarms

Table 7-3 Threshold alarms of DMS alarm sources

Type	Name	Severity	Description
Default	Node CPU Usage Exceeds the Threshold	Urgent	This alarm is generated if the threshold of CPU usage (system + user) of any node in the cluster is exceeded within the specified period and the constraint is not met. The alarm will be cleared when the CPU usage (system + user) is lower than the threshold and the constraint is not met.
Default	Node Data Disk Usage Exceeds the Threshold	Urgent: > 85%; Important: > 80%	This alarm is generated if the threshold of data disk (<code>/var/chroot/DWS/data[n]</code>) usage of any node in the cluster is exceeded within the specified period and the constraint is not met. The alarm will be cleared when the data disk (<code>/var/chroot/DWS/data[n]</code>) usage is lower than the threshold and the constraint is not met.
Default	Node Data Disk I/O Usage Exceeds the Threshold	Urgent	This alarm is generated if the threshold of data disk (<code>/var/chroot/DWS/data[n]</code>) I/O usage (<code>util</code>) of any node in the cluster is exceeded within the specified period and the constraint is not met. The alarm will be cleared when the data disk (<code>/var/chroot/DWS/data[n]</code>) I/O usage (<code>util</code>) is lower than the threshold and the constraint is not met.
Default	Node Data Disk Latency Exceeds the Threshold	Important	This alarm is generated if the threshold of data disk (<code>/var/chroot/DWS/data[n]</code>) I/O latency (<code>await</code>) of any node in the cluster is exceeded within the specified period and the constraint is not met. The alarm will be cleared when the data disk (<code>/var/chroot/DWS/data[n]</code>) I/O latency (<code>await</code>) is lower than the threshold and the constraint is not met.

Type	Name	Severity	Description
Default	Node Data Disk Inode Usage Exceeds the Threshold	Urgent: > 95%; important: > 90%	This alarm is generated if the threshold of data disk (<code>/var/chroot/DWS/data/n/</code>) inode usage of any node in the cluster is exceeded within the specified period and the constraint is not met. The alarm will be cleared when the data disk (<code>/var/chroot/DWS/data/n/</code>) inode usage is lower than the threshold and the constraint is not met.
Default	Data Flushed to Disks of the Query Statement Exceeds the Threshold	Urgent	This alarm is generated if the threshold of data flushed to disks of the SQL statement in the cluster is exceeded within the specified period and the constraint is not met. The alarm can be cleared only after you handle the SQL statement.
Default	Number of Queuing Query Statements Exceeds the Threshold	Urgent	This alarm is generated if the threshold of the number of queuing SQL statements is exceeded within the specified period. The alarm will be cleared when the number of queuing SQL statements is less than the threshold.
Default	Queue congestion in the cluster default resource pool	Urgent	This alarm is generated if the queue in the default resource pool of a cluster is congested and no alarm suppression conditions are met. This alarm will be cleared if the queue is not congested.
Default	The packet loss retransmission rate on the cluster network exceeds the threshold.	Urgent	This alarm is generated if the DMS alarm module detects a high retransmission rate on a server and no alarm suppression conditions are met. If the retransmission rate decreases, the alarm will be automatically cleared.
Default	Long SQL Probe Execution Duration in a Cluster	Urgent	This alarm is generated if the DMS alarm module detects a SQL probe execution duration on a server and no alarm suppression conditions are met. If no execution duration exceeds the threshold, the alarm will be automatically cleared. NOTE The alarm is supported only in 8.1.1.300 and later cluster versions. For earlier versions, contact technical support.

Type	Name	Severity	Description
Default	A vacuum full operation that holds a table lock for a long time exists in the cluster.	Important	In a specified period, the DMS alarm module detects that VACUUM FULL has been running for a long time in the cluster and blocks other operations. This alarm is generated if there are other SQL statements in the lock wait state and no suppression conditions are met. This alarm will be cleared if VACUUM FULL in the cluster did not cause lock wait. NOTE If this alarm is generated, contact technical support engineers.
Custom	<i>Name of the user-defined threshold alarm</i>	<i>User-defined alarm severity</i>	<i>Alarm description</i>

7.2.2 Alarm Rules

Overview

- Concepts related to threshold alarms
 - Alarm rule: consists of the alarm rule name, rule description, clusters associated with the rule, alarm policy triggering relationship, and alarm policy. An alarm rule can apply to one or all clusters, and can consist of one or more policies. The relationship between alarm policies can be selected in **Triggered Policies**. Each alarm policy consists of the triggers and constraint of each alarm rule.
 - Alarm policy: consists of the triggers, constraint, and alarm severity for an alarm metric.
 - Alarm metric: indicates a database cluster metric, which is generally time series data, for example, node CPU usage and amount of data flushed to disks.
- Alarm rule types
 - Default rule: best practices of GaussDB(DWS) threshold alarms.
 - User-defined rule: personalized alarm rules by configuring or combining monitoring metrics. (The current version supports only user-defined alarm rules of schema usage.)
- Alarm rule operations
 - Modify: modifies an alarm rule. All alarm rules apply (all items of user-defined alarm rules but only some items of the default alarm rules).
 - Enable/Disable: enables or disables an alarm rule. All alarm rules can be enabled or disabled. After an alarm rule is enabled, it is added to the check list by the alarm engine and can be triggered normally. Disabled

alarm rules will be removed from the check list by the alarm engine and will not be triggered.

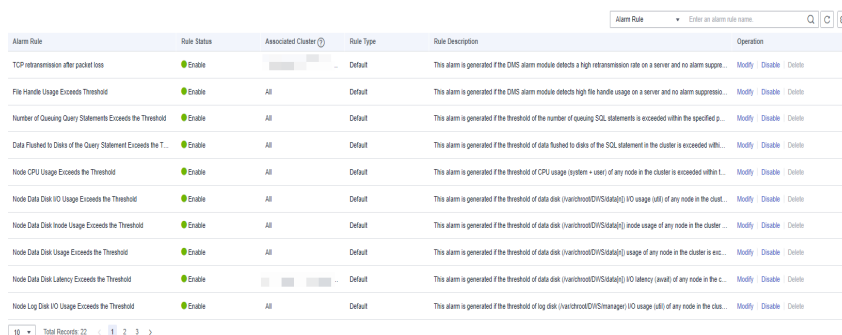
- **Delete:** deletes an alarm rule. You can delete only user-defined rules. Default alarm rules cannot be deleted.

Precautions

After a cluster is migrated, to monitor alarms of the new cluster, change the cluster bound to the alarm rule to the new cluster. You can also create an alarm rule for the new cluster.

Viewing Alarm Rules

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane on the left, choose **Alarms**.
- Step 3** Click **View Alarm Rule** in the upper left corner. On the page that is displayed, you can see the threshold alarm rules of database cluster monitoring metrics, as shown in the following figure.



Alarm Rule	Rule Status	Associated Cluster	Rule Type	Rule Description	Operation
TCP retransmission after packet loss	Enable		Default	This alarm is generated if the DWS alarm module detects a high retransmission rate on a server and no alarm suppression.	Modify Disable Delete
File Handle Usage Exceeds Threshold	Enable	All	Default	This alarm is generated if the DWS alarm module detects high file handle usage on a server and no alarm suppression.	Modify Disable Delete
Number of Querying Query Statements Exceeds the Threshold	Enable	All	Default	This alarm is generated if the threshold of the number of querying SQL statements is exceeded within the specified period.	Modify Disable Delete
Data Flushed to Disks of the Query Statement Exceeds the Threshold	Enable	All	Default	This alarm is generated if the threshold of data flushed to disks of the SQL statement in the cluster is exceeded within the specified period.	Modify Disable Delete
Node CPU Usage Exceeds the Threshold	Enable	All	Default	This alarm is generated if the threshold of CPU usage (system + user) of any node in the cluster is exceeded within the specified period.	Modify Disable Delete
Node Data I/O Usage Exceeds the Threshold	Enable	All	Default	This alarm is generated if the threshold of data disk (vnode/dnode) I/O usage (all) of any node in the cluster is exceeded within the specified period.	Modify Disable Delete
Node Data Disk I/O Usage Exceeds the Threshold	Enable	All	Default	This alarm is generated if the threshold of data disk (vnode/dnode) I/O usage of any node in the cluster is exceeded within the specified period.	Modify Disable Delete
Node Data Disk Usage Exceeds the Threshold	Enable	All	Default	This alarm is generated if the threshold of data disk (vnode/dnode) usage of any node in the cluster is exceeded within the specified period.	Modify Disable Delete
Node Data Disk Latency Exceeds the Threshold	Enable		Default	This alarm is generated if the threshold of data disk (vnode/dnode) I/O latency (avg) of any node in the cluster is exceeded within the specified period.	Modify Disable Delete
Node Log I/O Usage Exceeds the Threshold	Enable	All	Default	This alarm is generated if the threshold of log disk (vnode/dnode) I/O usage (all) of any node in the cluster is exceeded within the specified period.	Modify Disable Delete

----End

Modifying an Alarm Rule

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane on the left, click **Alarms**.
- Step 3** Click **View Alarm Rule** in the upper left corner.
- Step 4** On the **Alarm Rules** page that is displayed, click **Modify** in the **Operation** column of the target alarm rule.
 - **Alarm rule name:** The rule name contains 6 to 64 characters (letters, digits, Chinese characters, slashes) and must start with a non-digit character.
 - **Description**
 - **Associated Cluster:** From the drop-down list, select the current tenant's clusters to which the alarm rule applies.
 - **Triggered Policies**
 - **Independent:** Alarm policies are triggered independently of each other.
 - **Priority:** Alarm policies are triggered by priority. Policies of a lower priority will be automatically triggered after those of a higher priority.

- **Alarm Policy**
 - **Metric:** GaussDB(DWS) monitoring metric, which is the data source used by the alarm engine for threshold determination.
 - **Trigger:** calculation rule for threshold determination of a monitoring metric. Select the average value within a period of time of a metric to reduce the probability of alarm oscillation.
 - **Constraint:** suppresses the repeated triggering and clearance of alarms of the same type within the specified period.
 - **Alarm Severity:** includes **Urgent**, **Important**, **Minor**, and **Prompt**.

* Alarm Rule: Node CPU Usage Exceeds the Threshold

Description: This alarm is generated if the threshold of CPU usage (system + user) of any node in the cluster is exceeded within the specified period. 287/400

* Associated Cluster: A.

* Triggered Policies: Independent

* Alarm Policy

Metric	Trigger	Constraint	Alarm Severity
Node CPU Usage	Average > 0 %	None	Urgent

Confirm Cancel

NOTE

You can modify only some items of the default rules (associated cluster, alarm policy threshold, time period, and alarm constraint). User-defined rules support modification of all items.

Step 5 Confirm the information and click **OK**.

----End

Creating an Alarm Rule

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 In the navigation pane on the left, click **Alarms**.

Step 3 Click **View Alarm Rule** in the upper left corner.

Step 4 Click **Create Alarm Rule** in the upper right corner. You can configure items, such as the alarm rule name, rule description, clusters associated with the rule, and alarm policy.

- **Alarm rule name:** The rule name contains 6 to 64 characters (letters, digits, Chinese characters, slashes) and must start with a non-digit character.
- **Description**
- **Associated Cluster:** From the drop-down list, select the current tenant's clusters to which the alarm rule applies.
- **Triggered Policies**
 - **Independent:** Alarm policies are triggered independently of each other.
 - **Priority:** Alarm policies are triggered by priority. Policies of a lower priority will be automatically triggered after those of a higher priority.
- **Alarm Policy**
 - **Metric:** GaussDB(DWS) monitoring metric, which is the data source used by the alarm engine for threshold determination.

- **Alarm Object:** databases in the selected cluster and schemas in the selected databases.
- **Trigger:** calculation rule for threshold determination of a monitoring metric. Select the average value within a period of time of a metric to reduce the probability of alarm oscillation.
- **Constraint:** suppresses the repeated triggering and clearance of alarms of the same type within the specified period.
- **Alarm Severity:** includes **Urgent**, **Important**, **Minor**, and **Prompt**.

Figure 7-1 Creating an alarm rule

NOTE

Currently, only alarm rules of schema usage metrics can be created on GaussDB(DWS).

----End

7.2.3 Alarm Subscriptions

You can subscribe to GaussDB(DWS) alarm notifications to receive notifications by SMS message, email, or application when an alarm of a specified severity is generated.

Creating a Subscription

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane on the left, choose **Alarms > Subscriptions**.
- Step 3** Click **Create Subscription** in the upper left corner of the page.
- Step 4** In the **Subscription Settings** area, configure the basic information and alarm severity of the subscription.

Subscription Settings

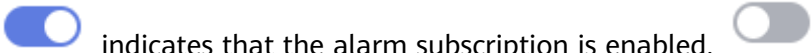
Edit subscription information and select alarm severities

* Status ?

* Subscription Name ?

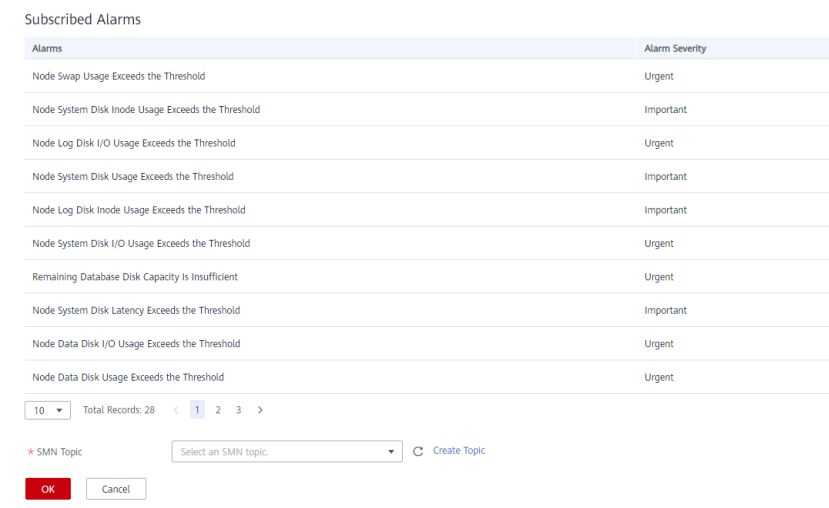
Alarm Severity

Table 7-4 Subscription parameters

Parameter	Description
Status	Whether to enable the alarm subscription.  indicates that the alarm subscription is enabled. indicates that the alarm subscription is disabled. When you disable a subscription, you will not receive the corresponding alarm notifications, but the subscription will not be deleted.
Subscription Name	Name of the alarm subscription: <ul style="list-style-type: none"> Contains only letters, digits, hyphens (-), and underscores (_), and must start with a letter or digit. Contains 1 to 256 characters.
Alarm Severity	Severity of the alarm you want to subscribe to: Urgent, Important, Minor, or Prompt

Step 5 The **Subscribed Alarms** area displays the subscribed alarms by subscription settings. Select an SMN topic from the drop-down list.

To create a topic, click **Create Topic** to switch to the SMN console page. For details, .



NOTE

The selected topic must have granted GaussDB(DWS) the permission for publishing messages to the topic. To grant permissions, configure topic policies on the SMN management console. When configuring the topic policy, select **DWS** as the service that can publish messages to this topic.

Step 6 Confirm the information and click **OK**.

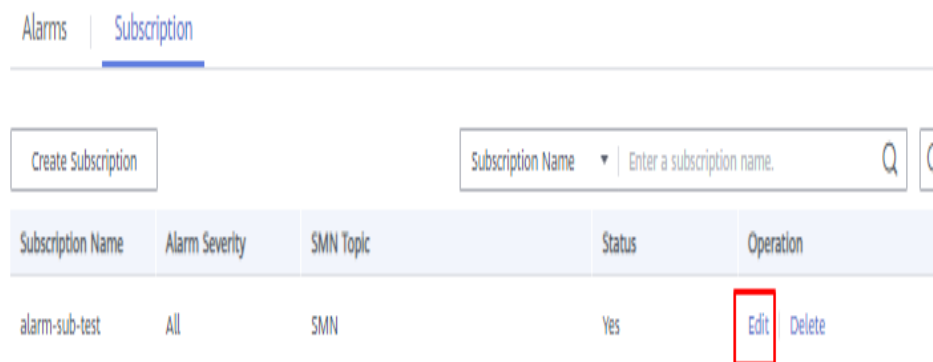
----End

Modifying a Subscription

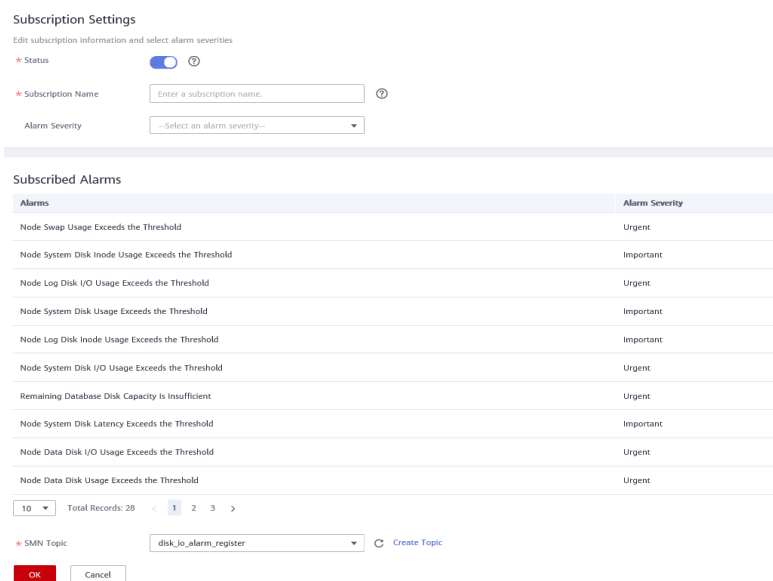
Step 1 Log in to the GaussDB(DWS) management console.

Step 2 In the navigation pane on the left, choose **Alarms** > **Subscriptions**.

Step 3 In the **Operation** column of the target subscription, click **Edit**.



Step 4 On the **Edit Subscription** page displayed, modify the parameters. For details, see [Step 4](#) to [5](#).



Step 5 Click **OK**.

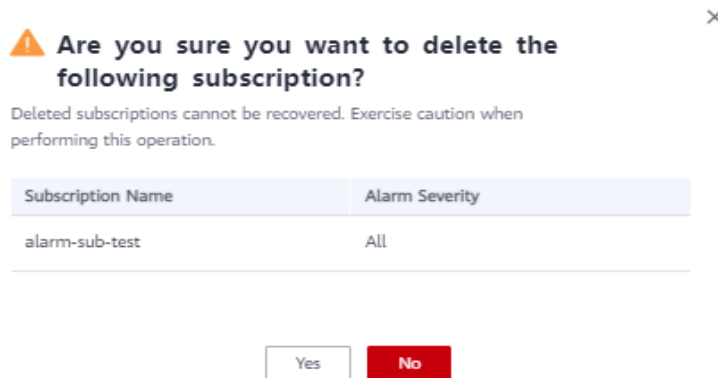
----End

Deleting a Subscription

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 In the navigation pane on the left, choose **Alarms** > **Subscriptions**.

Step 3 In the **Operation** column of the target subscription, click **Delete**. A confirmation dialog box is displayed.



Step 4 Click **Yes** to delete the subscription.

----End

7.3 Event Notifications

7.3.1 Event Notifications Overview

Supported Event Types and Events

Events are records of changes in the user's cluster status. Events can be triggered by user operations (such as audit events), or may be caused by cluster service status changes (for example, cluster repaired successfully or failed to repair the cluster). The following tables list the events and event types supported by GaussDB(DWS).

- The following table lists the events whose **Event Source Category** is **Cluster**.

Table 7-5 Events whose **Event Source Category** is **Cluster**

Event Type	Event Name	Event Severity	Event
Management	createClusterFail	Warning	Failed to create the cluster.
Management	createClusterSuccess	Normal	Cluster created successfully.
Management	createCluster	Normal	Cluster creation started.

Event Type	Event Name	Event Severity	Event
Management	extendCluster	Normal	Cluster scale-out started.
Management	extendClusterSuccess	Normal	Cluster scaled out successfully.
Management	extendClusterFail	Warning	Failed to scale out the cluster.
Management	deleteClusterFail	Warning	Failed to delete the cluster.
Management	deleteClusterSuccess	Normal	Cluster deleted successfully.
Management	deleteCluster	Normal	Cluster deletion started.
Management	restoreClusterFail	Warning	Failed to restore the cluster.
Management	restoreClusterSuccess	Normal	Cluster restored successfully.
Management	restoreCluster	Normal	Cluster restoration started.
Management	restartClusterFail	Warning	Failed to restart the cluster.
Management	restartClusterSuccess	Normal	Cluster restarted successfully.
Management	restartCluster	Normal	Cluster restarted.
Management	bindEipToCluster	Normal	Bound an EIP to the cluster.

Event Type	Event Name	Event Severity	Event
Management	bindEipToClusterFail	Warning	Failed to bind an EIP to the cluster.
Management	unbindEipToCluster	Normal	Unbound an EIP from the cluster.
Management	unbindEipToClusterFail	Warning	Failed to unbind an EIP from the cluster.
Management	refreshEipToCluster	Normal	Refreshed the cluster's EIP.
Management	refreshEipToClusterFail	Warning	Failed to refresh the cluster's EIP.
Security	resetPasswordFail	Warning	Failed to reset the password.
Security	resetPasswordSuccess	Normal	Password of the cluster reset successfully.
Security	updateConfiguration	Normal	Updating security parameters of the cluster started.
Security	updateConfigurationFail	Warning	Failed to update security parameters of the cluster.
Security	updateConfigurationSuccess	Normal	Security parameters of the cluster updated successfully.
Monitoring	repairCluster	Normal	The node is faulty. Repairing the cluster starts.
Monitoring	repairClusterFail	Warning	Failed to repair the cluster.
Monitoring	repairClusterSuccess	Normal	Cluster repaired successfully.

- The following table lists the events whose **Event Source Category** is **Snapshot**.

Table 7-6 Events whose **Event Source Category** is **Snapshot**

Event Type	Event Name	Event Severity	Event
Management	deleteBackup	Normal	Snapshot deleted successfully.
Management	deleteBackupFail	Warning	Failed to delete the snapshot.
Management	createBackup	Normal	Snapshot creation started.
Management	createBackupSuccess	Normal	Snapshot created successfully.
Management	createBackupFail	Warning	Failed to create the snapshot.


7.3.2 Viewing Events


This section describes how to search for events that occur in a cluster or snapshot.

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 In the navigation tree on the left, click **Events**.

On the **Events** tab page, all events that occur in the clusters or snapshots are displayed by default.

You can sort the events in descending or ascending order by clicking  next to **Time**.

You can filter the events by clicking  next to a field (except **Time**) and selecting the criteria.


----End

8 Specifications Change and Scaling

8.1 Managing Nodes

Overview

On the **Nodes** tab page, you can view the node list of the current cluster, add new nodes to or remove nodes from it, and view the node usage, status, flavor, and AZ.

To modify an alias, click  next to it.

Add		Remove		Resource Status			All
Node Name	Node Alias Name	AZ	Resource Status	Node Status	Node Flavor		
		AZ3			d2x.xlarge.4		
		AZ3			d2x.xlarge.4		
		AZ3			d2x.xlarge.4		

NOTE

- This feature is supported only in 8.1.1.200 or later cluster versions.

Adding Nodes

This function is more suited for large-scale scale-out. Nodes can be added in batches in advance without interrupting services. For example, if 180 more BMS nodes are needed, add them in three batches (60 for each batch). If some nodes fail to be added, add them again. After all the 180 nodes are successfully added, use the nodes for cluster scale-out.

Precautions

- Nodes can be added only when no other task is running on the management side.
- The storage size of a new node must be the same as that of each of the existing nodes in the cluster.
- A node that is successfully added, usually for scale-out purposes, is called an idle node. You are advised to add nodes only when necessary and use them for scale-out in a timely manner once they are added.

- The anti-affinity rule dictates that the number of nodes to be added at a time must be an integer multiple of the cluster ring size. For example, if the cluster ring size is 3, the number of nodes to be added must be an integer multiple of 3.
- In the anti-affinity deployment mode, when a node is idle and fails due to power-off or other causes, it makes other nodes in its server group unavailable. In this case, you should remove and re-add the failed node.
- The anti-affinity rule dictates that, if a node fails to be added and is rolled back, other nodes that are being added in the same server group will also be rolled back.

Procedure

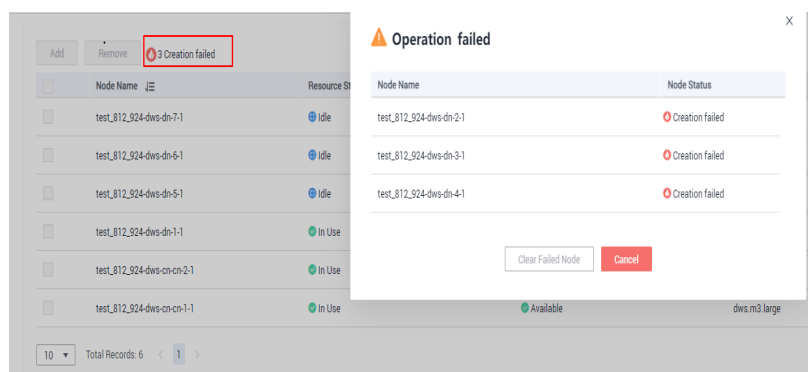
- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** Choose **Clusters > Dedicated Clusters**. All clusters are displayed by default.
- Step 3** Click the name of the target cluster. On the **Cluster Information** page that is displayed, choose **Nodes**.
- Step 4** Click **Add**, enter the number of nodes to be added, read the note, and select **Confirmed**. Click **Next: Confirm**.
- Step 5** Click **Submit**. The nodes will start to be added, as shown in the following figure.

Node Name	Node Alias Name	AZ	Resource Status	Node Status	Node Flavor
[Redacted]	[Redacted]	AZ3	In Use	Available	dws.xlarge.4
[Redacted]	[Redacted]	AZ3	In Use	Available	dws.xlarge.4
[Redacted]	[Redacted]	AZ3	In Use	Available	dws.xlarge.4

----End

NOTE

The nodes that fail to be added will be automatically rolled back and recorded in the displayed list, as shown in the following figure.



Removing Nodes

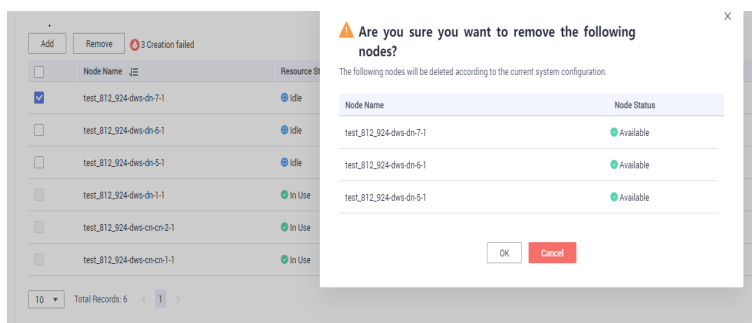
Precautions

- Nodes can be removed only when no other task is running on the management side.

- Only nodes whose resource status is **Idle** can be removed. Nodes that are in use cannot be removed.
- In anti-affinity deployment, nodes are removed by cluster ring. For example, when you remove a node, other nodes in the same ring will be automatically selected and displayed.

Procedure

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** Choose **Clusters > Dedicated Clusters**. All clusters are displayed by default.
- Step 3** Click the name of the target cluster. On the **Cluster Information** page that is displayed, choose **Nodes**.
- Step 4** On the **Nodes** page, select the nodes to be removed, click **Remove**, and click **Yes** to submit the task.



- Step 5** The nodes that are successfully removed will not be displayed on the **Nodes** page.
----End

8.2 Scaling Nodes

8.2.1 Scaling Out a Cluster

When you need more compute and storage resources, add more nodes for cluster scale-out on the management console.

After the data in a data warehouse is deleted, the occupied disk space may not be released, resulting in dirty data and disk waste. Therefore, if you need to scale out your cluster due to insufficient storage capacity, run the **VACUUM** command to reclaim the storage space first. If the used storage capacity is still high after you run the **VACUUM** command, you can scale out your cluster. For details about the **VACUUM** syntax, see "SQL References > SQL Syntax > VACUUM" in the *Data Warehouse Service (DWS) Developer Guide*.

Impact on the System

- Before the scale-out, disable the client connections that have created temporary tables because temporary tables created before or during the scale-out will become invalid and operations performed on these temporary tables will fail. Temporary tables created after the scale-out will not be affected.

- During the scale-out, functions such as cluster restart, scale-out, snapshot creation, database administrator password resetting, and cluster deletion are disabled.
- During an offline scale-out, the cluster automatically restarts. Therefore, the cluster stays **Unavailable** for a period of time. After the cluster is restarted, the status becomes **Available**. After scale-out, the system dynamically redistributes user data among all nodes in the cluster.

Prerequisites

- The cluster to be scaled out is in the **Available** or **Unbalanced** state.
- The number of nodes to be added must be less than or equal to the available nodes. Otherwise, system scale-out is not allowed.

Scaling Out a Cluster

NOTE

- A cluster becomes read-only during scale-out. Exercise caution when performing this operation.
- To ensure data security, you are advised to create a snapshot before the scale-out. For details about how to create a snapshot, see [Manual Snapshots](#).

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 Choose **Clusters** > **Dedicated Clusters**.

All clusters are displayed by default.

Step 3 In the **Operation** column of the target cluster, choose **More** > **Scale Node** > **Scale Out**. The scale-out page is displayed.

Step 4 Specify the number of nodes to be added.

- The number of nodes after scale-out must be at least three nodes more than the original number. The maximum number of nodes that can be added depends on the available quota. In addition, the number of nodes after the scale-out cannot exceed 32.
- Flavor of the new nodes must be the same as that of existing nodes in the cluster.
- The VPC, subnet, and security group of the cluster with new nodes added are the same as those of the original cluster.

Step 5 Configure advanced parameters.

- If you choose **Default**, **Scale Online** will be disabled, **Auto Redistribution** will be enabled, and **Redistribution Mode** will be **Offline** by default.
- If you choose **Custom**, you can configure the following advanced configuration parameters for online scale-out:
 - **Scale Online**: Online scale-out can be enabled. During online scale-out, data can be added, deleted, modified, and queried in the database; and some DDL syntaxes are supported. Errors will be reported for unsupported syntaxes.
 - **Auto Redistribution**: Automatic redistribution can be enabled. If automatic redistribution is enabled, data will be redistributed

immediately after the scale-out is complete. If this function is disabled, only the scale-out is performed. In this case, to redistribute data, select a cluster and choose **More > Scale Node > Redistribute**.

- **Redistribution Concurrency:** If automatic redistribution is enabled, you can set the number of concurrent redistribution tasks. The value range is 1 to 32. The default value is 4.
- **Redistribution Mode:** It can be set to **Online** or **Offline**. After confirming that the information is correct, click **OK** in the displayed dialog box.

Step 6 Click **Next: Confirm**.

Step 7 Click **Submit**.

- After you submit the scale-out application, task information of the cluster changes to **Scaling out** and the process will take several minutes. During the scale-out, the cluster automatically restarts. Therefore, the cluster status will stay **Unavailable** for a while. After the cluster is restarted, the status will change to **Available**. In the last phase of scale-out, the system dynamically redistributes user data in the cluster, during which the cluster is in the **Read-only** state.
- A cluster is successfully scaled out only when the cluster is in the **Available** state and task information **Scaling out** is not displayed. Then you can use the cluster.
- If **Scale-out failed** is displayed, the cluster fails to be scaled out.

----End

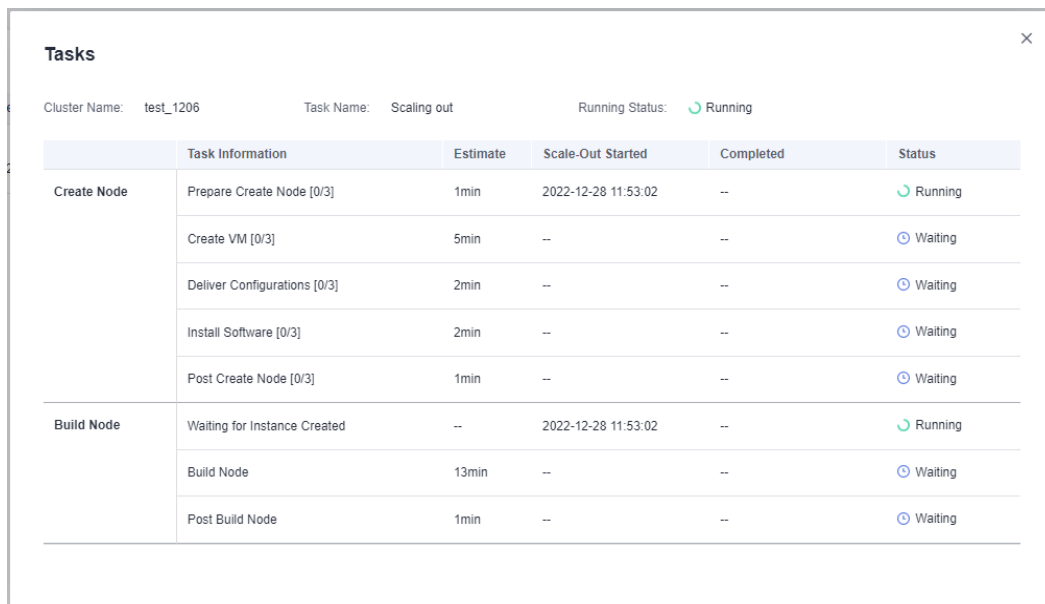
Viewing Scaling Details

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 Choose **Clusters > Dedicated Clusters**.

Step 3 In the **Task Information** column of a cluster, click **View Details**.

Step 4 Check the scale-out status of the cluster on the scaling details page.



	Task Information	Estimate	Scale-Out Started	Completed	Status
Create Node	Prepare Create Node [0/3]	1min	2022-12-28 11:53:02	--	Running
	Create VM [0/3]	5min	--	--	Waiting
	Deliver Configurations [0/3]	2min	--	--	Waiting
	Install Software [0/3]	2min	--	--	Waiting
	Post Create Node [0/3]	1min	--	--	Waiting
Build Node	Waiting for Instance Created	--	2022-12-28 11:53:02	--	Running
	Build Node	13min	--	--	Waiting
	Post Build Node	1min	--	--	Waiting

----End

8.2.2 Cluster Redistribution

8.2.2.1 Redistributing Data

Data redistribution, where data in existing nodes is evenly allocated to the new nodes after you scale out a cluster, is a time-consuming yet crucial task that accelerates service response.

By default, redistribution is automatically started after cluster scale-out. For enhanced reliability, disable the automatic redistribution function and manually start a redistribution task after the scale-out is successful. In this way, both scale-out and redistribution can be retried upon failures.

Currently, **offline redistribution** and **online redistribution** are supported. The default mode is offline redistribution.

Before redistribution starts or when redistribution is paused, you can set redistribution priorities for the tables that have not been redistributed by schema or table.

NOTICE

- The cluster redistribution function is supported in 8.1.1.200 or later cluster versions.
- This function can be manually enabled only when the cluster task information displays **To be redistributed** after scale-out.
- You can also select the redistribution mode when you configure cluster scale-out (see [Configure advanced parameters](#)).
- Redistribution queues are sorted based on the relpage size of tables. To ensure that the relpage size is correct, you are advised to perform the **ANALYZE** operation on the tables to be redistributed.

Offline Redistribution

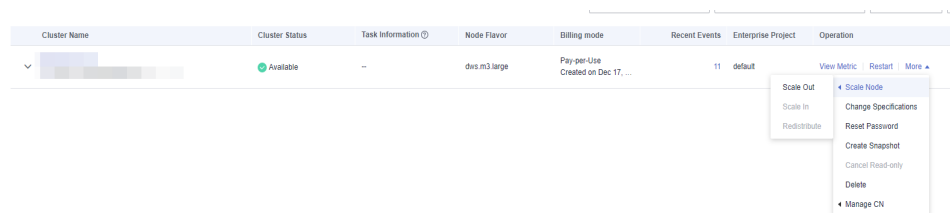
Precautions

- In offline redistribution mode, the database does not support DDL and DCL operations. Tables that are being redistributed support only simple DQL operations.
- During table redistribution, a shared lock is added to tables. All insert, update, and delete operations as well as DDL operations on the tables are blocked for a long time, which may cause a lock wait timeout. Do not perform queries that take more than 20 minutes during the redistribution (the default time for applying for the write lock during redistribution is 20 minutes). Otherwise, data redistribution may fail due to lock wait timeout.

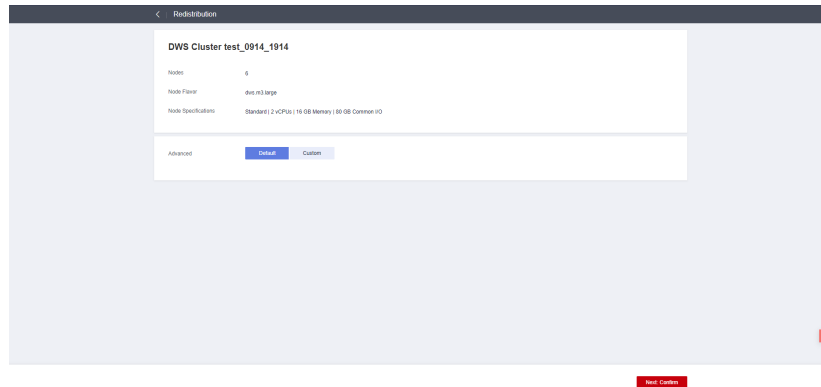
Procedure

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** Choose **Clusters > Dedicated Cluster**. All clusters are displayed by default.
- Step 3** In the **Operation** column of the target cluster, choose **More > Scale Node > Redistribute**, as shown in the following figure.

The **Redistribution** page is displayed.



- Step 4** On the **Redistribute** page that is displayed, keep the default **offline** redistribution mode and click **Next: Confirm** to submit the task.



----End

Online Redistribution

Precautions

In online redistribution mode, the database supports partial DDL and DCL operations.

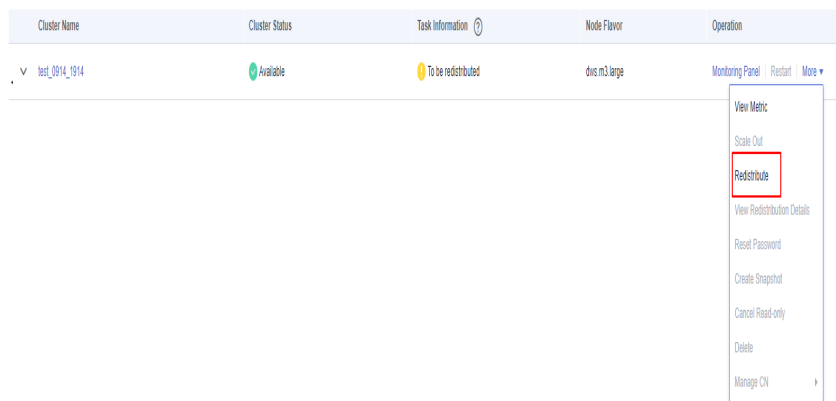
- Tables that are being redistributed support insert, delete and update operations and some DDL operations. The following functions are supported:
 - **INSERT, DELETE, UPDATE, MERGE INTO, OVERWRITE, UPSERT**
 - Join queries across node groups
 - Local table renaming, schema modification, **DROP, TRUNCATE, TRUNCATE-PARTITION**
- The following operations cannot be performed on tables that are being redistributed:
 - Run **ALTER TABLE** statements (except for **TRUNCATE PARTITION**), including adding or deleting columns or partitions.
 - Create, modify, or delete indexes.
 - **VACUUM FULL** or **CLUSTER** can not be executed on tables during table redistribution.
 - Modify the sequence objects on which a column depends, including creating and modifying them. Typical statements are **CREATE** and **ALTER SEQUENCE ... OWNED BY**.
 - During the redistribution of a table with more than 996 columns, **UPDATE** and **DELETE** statements cannot be executed. **SELECT** and **INSERT** statements are allowed.
 - Database and tablespace objects cannot be created, deleted, or modified during redistribution.
 - A partition swap can be performed only if the redistribution is complete for both of the tables to be swapped. The two tables belong to different node groups and do not allow partition swap if either of them is being redistributed.

Procedure

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 Choose **Clusters > Dedicated Clusters**. All clusters are displayed by default.

Step 3 In the **Operation** column of the target cluster, choose **More > Scale Node > Redistribute**, as shown in the following figure.



Step 4 On the **Redistribute** page that is displayed, set **Advanced to Custom**, set the redistribution mode to **Online mode**, and click **Next: Confirm** to submit the task.

----End

8.2.2.2 Viewing Redistribution Details

On the **View Redistribution Details** page, you can check the monitoring information, including the redistribution mode, redistribution progress, and table redistribution details of the current cluster. You can pause and resume redistribution, set the redistribution priority, and change the number of concurrent redistribution tasks.

NOTE

The function of viewing redistribution details is supported by 8.1.1.200 and later cluster versions. Details about the data table redistribution progress are supported only by 8.2.1 and later cluster versions.

Precautions

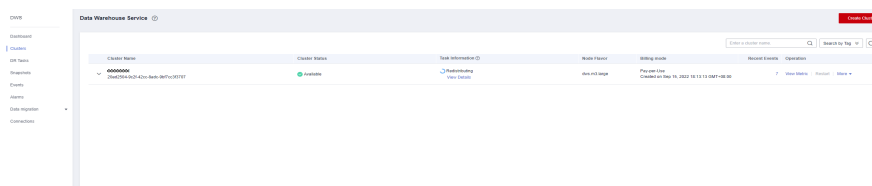
You can check redistribution details only if the cluster is being redistributed, failed to be redistributed, or is suspended. There may be a delay in the statistics update.

Procedure

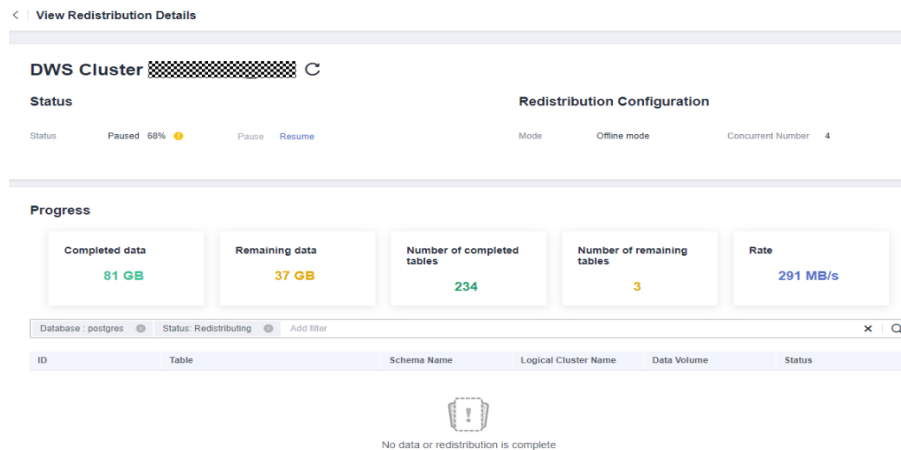
Step 1 Log in to the GaussDB(DWS) management console.

Step 2 Choose **Clusters > Dedicated Clusters**. All clusters are displayed by default.

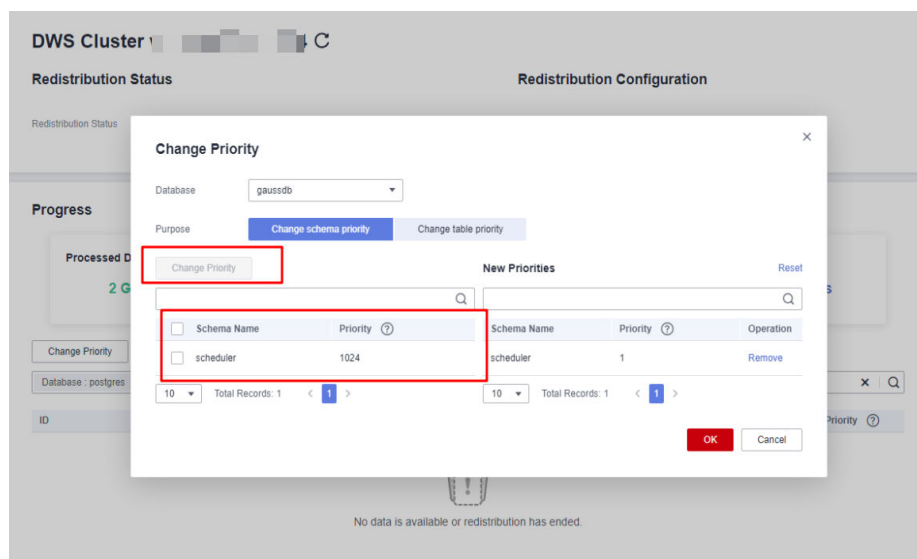
Step 3 In the **Task Information** column of a cluster, click **View Details**.

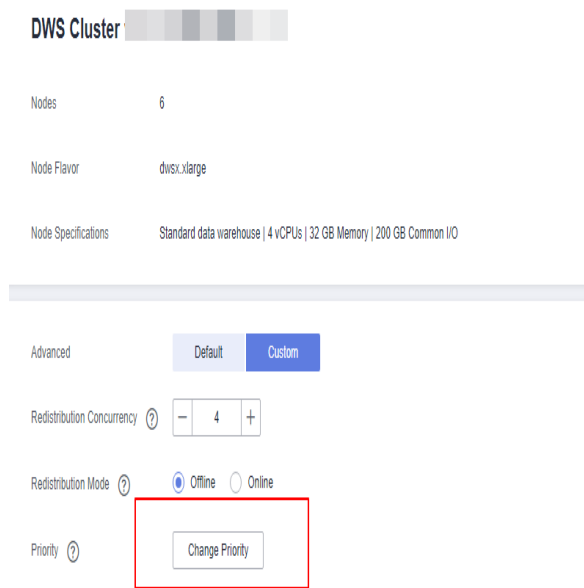


Step 4 Check the redistribution status, configuration, progress, and redistribution details of all the tables in a specified database. Specify a database that and can be searched by table redistribution status and table name. If all the tables in a database have completed redistribution, no data will be displayed for the database.



Step 5 When redistribution is paused, you can set the redistribution priority (in schema or table dimension), and redistribution will be performed based on the configured redistribution sequence. You can also set the redistribution priority before the redistribution starts.

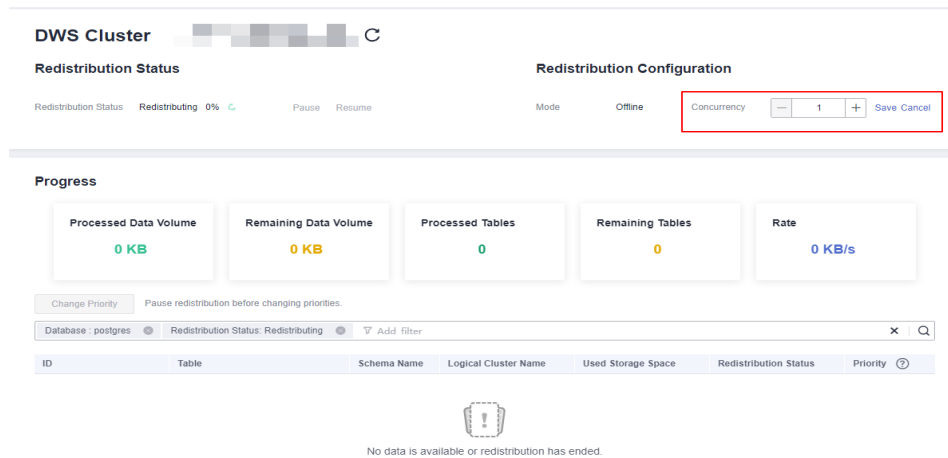




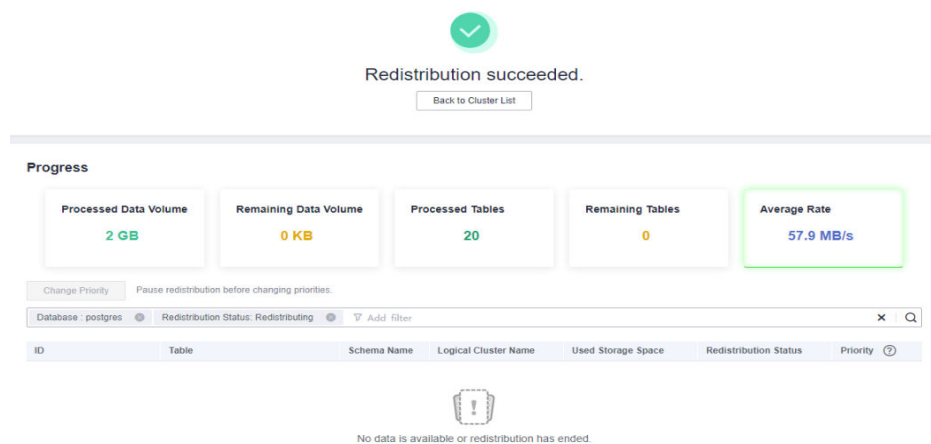
Step 6 The number of concurrent redistribution tasks can be adjusted during redistribution.

NOTE

Cluster 8.1.0 and earlier versions do not support dynamic adjustment. To change redistribution concurrency, suspend redistribution first.



Step 7 Check the redistribution progress. After the redistribution is complete, the amount of completed data, amount of remaining data, number of completed tables, number of remaining tables, and average rate during redistribution are displayed.



----End

8.2.3 Scaling In a Cluster

You can scale in your clusters on the console to release unnecessary computing and storage resources provided by GaussDB(DWS).

Impact on the System

- Before the scale-in, exit the client connections that have created temporary tables, because temporary tables created before or during the scale-in will become invalid and operations performed on these temporary tables will fail. Temporary tables created after the scale-in will not be affected.
- If you start a scale-in, an automatic snapshot will be created for the cluster before scale-in. If you do not need the snapshot, you can disable the automated backup function on the scale-in page.
- Before scale-in, ensure that the skew rate does not exceed 10%. There is no general requirement for the dirty page rate. However, for a large table whose size is greater than 50 GB, ensure that the dirty page rate does not exceed 20% to 30%.
- In a cluster that is being scaled in, the following functions are disabled: cluster restart, cluster scale-out, snapshot creation, node management, intelligent O&M, resource management, parameter modification, security configurations, log service, database administrator password resetting, and cluster deletion.
- During offline scale-in, stop all services or run only a few query statements. During table redistribution, a shared lock is added to tables. All insert, update, and delete operations as well as DDL operations on the tables are blocked for a long time, which may cause a lock wait timeout. After a table is redistributed, you can access the table. During redistribution, avoid querying data for more than 20 minutes. The default time for applying a write lock during redistribution is 20 minutes. Exceeding this duration may lead to redistribution failure due to lock waiting timeout.
- During online scale-in, you can perform insert, update, and delete operations on tables, but data updates may still be blocked for a short period of time. Redistribution consumes lots of CPU and I/O resources, which will greatly impact job performance. Therefore, perform redistribution when services are stopped or during periods of light load.

- During offline scale-in, if a node is deleted while DDL statements are executed (to create a schema or function), these statements may report errors, because the DN cannot be found. In this case, you simply need to retry the statements.
- If a cluster scale-in fails, the database does not automatically roll back the scale-in operation, and no O&M operations can be performed. In this case, you need to click the **Scale In** on the console to try again.

Prerequisites

- The cluster is in **Available** state, is not read-only, and there is no data being redistributed in the cluster.
- A cluster configuration file has been generated, and configuration information is consistent with the current cluster configuration.
- Before the scale-in operation starts, the value of **default_storage_nodegroup** is **installation**.
- The cluster is configured in the ring mode. A ring is the smallest unit for scale-in. Four or five hosts form a ring. The primary, standby, and secondary DNs are deployed in this ring.
- The scale-in host does not contain the GTM, ETCD, or CM Server component.
- There are no CNs on the nodes to be scaled in.
- Scale-in does not support rollback but supports retry. A data redistribution failure after a scale-in does not affect services. You can complete scale-in at other appropriate time. Otherwise, unbalanced data distribution will persist for a long time.
- Before redistribution, ensure that the **data_redis** schema in the corresponding database is reserved for redistribution and that no user operation on it or its tables is allowed. During redistribution, **data_redis** is used. After the operation is complete, the schema will be deleted. User tables (if any) in the schema will also be deleted.
- **gs_cgroup** cannot be used during scale-in.
- Before the scale-in, check the remaining capacity of the cluster. The nodes remaining in a scale-in must have sufficient space to store the data of the entire cluster. Otherwise, the scale-in cannot be properly performed.
 - The used physical disk space on each node is less than 80%.
 - All the users and roles use less than 80% of resource quota in total.
 - The estimated space usage after scale-in must be less than 80%.
 - The available space is 1.5 times larger than the maximum size of a single table.

NOTE

To check the maximum size of a single table, use the following inspection tool:

```
gs_check -i CheckBiggestTable -L
```

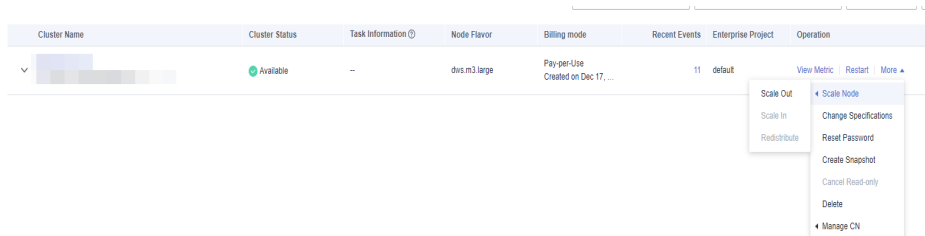
- Automatic removal of faulty CNs is disabled during the scale-in and is enabled after the scale-in is complete.

Procedure

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 Choose **Clusters > Dedicated Clusters**.

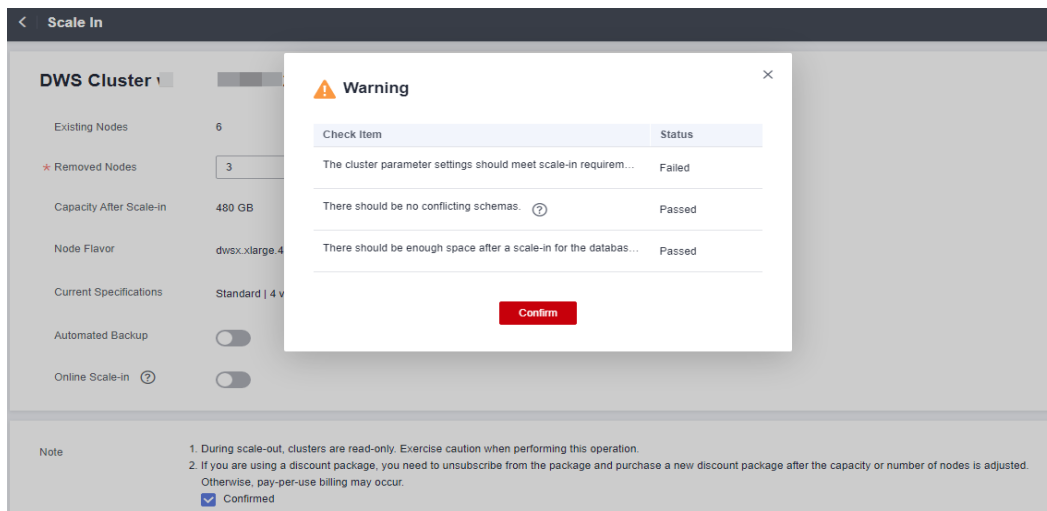
Step 3 In the **Operation** column of the target cluster, choose **More > Scale Node > Scale In**.



Step 4 The scale-in page is displayed. You can select the number of nodes to be scaled in. The automated backup function is enabled by default.



Step 5 Click **Next: Confirm**. The system will check the cluster status before scale-in. If your cluster fails the check, an error message will be displayed.



Step 6 After the check is passed, click **Confirm** to return to the cluster list. The cluster status is **Scaling in**. Wait for a while.

Cluster Name	Cluster Status	Task Information [?]	Node Flavor	Recent Events	Operation
▼ [Cluster Name]	Available	Scaling in 99%	dws.m3.large	8	View Metric Restart More ▼

----End

NOTE

- If the cluster parameters fail the check, the scale-in will fail. To avoid this problem, ensure your parameter settings are correct.
- If schemas fail the check, the scale-in will fail. To avoid this problem, check whether any schema that conflicts with the scale-in exists.
- If the disk space fails the check, the scale-in may fail or the cluster may become read-only after the scale-in. To avoid this problem, increase your cluster disk capacity.

8.3 Changing Specifications

8.3.1 Changing the Node Flavor

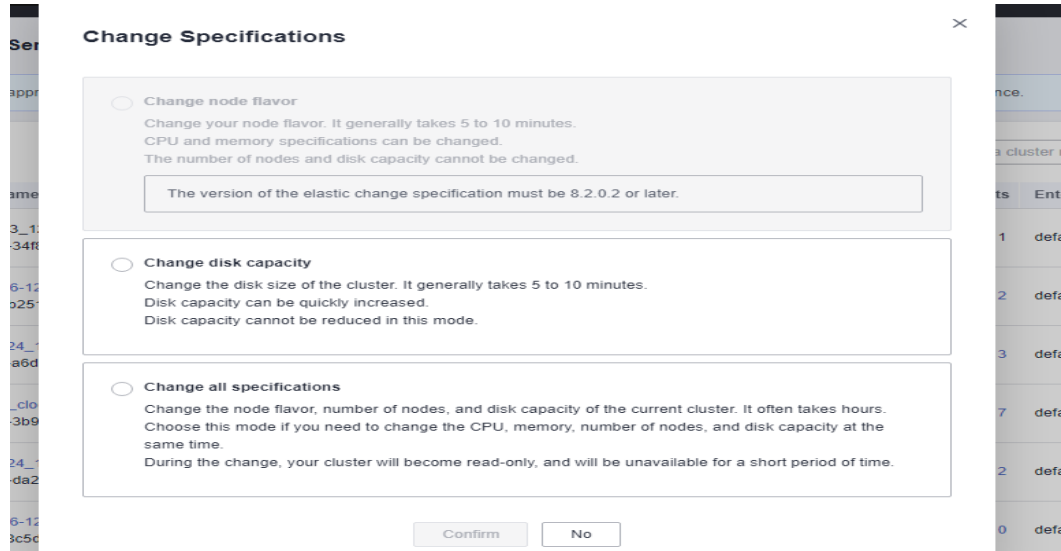
If you only need to cope with occasional service peaks or only increase computing capabilities, you are advised to modify cluster specifications instead of adding nodes. Before a service peak, you can modify cluster specifications to quickly improve computing capabilities. After the service peak, you can quickly reduce cluster configurations to minimize costs.

NOTE

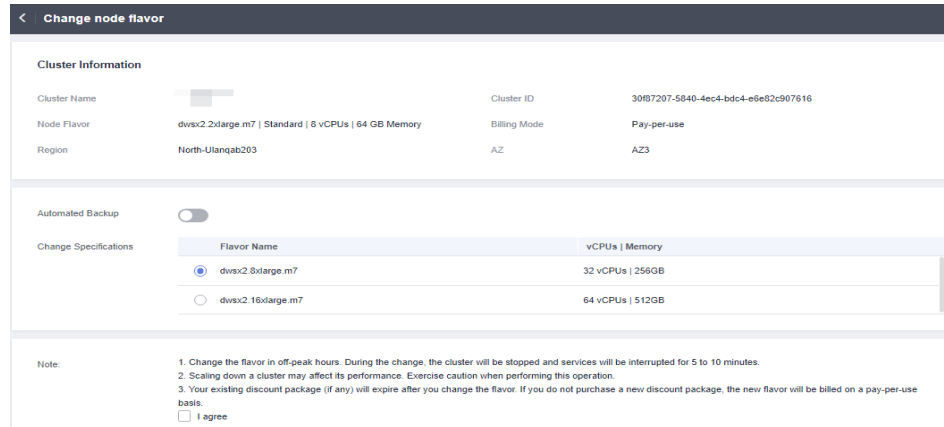
- Only cluster versions 8.1.1.300 and later support elastic flavor change. For an earlier version, contact technical support to upgrade it first.
- Currently, specifications can be modified only for offline clusters. The modification takes about 10 minutes.

Procedure

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** Choose **Clusters > Dedicated Clusters**. All clusters are displayed by default.
- Step 3** In the row of a cluster, choose **More > Change Specifications** in the **Operation** column and click **Change node flavor**.



- Step 4** Configure the flavor. Enable automatic backup as needed.



NOTICE

Decreasing the specifications of a cluster is to select the target specifications that are lower than the current specifications of the cluster. This operation may affect the cluster performance. Therefore, evaluate service impact before performing this operation.

- Step 5** Click **Next: confirm**.

Step 6 Return to the cluster list. The cluster status will change to **Changing node flavor**. Wait for about 10 minutes.

----End

8.3.2 Changing All Specifications

If you want to change your cluster topology or capacity but the **Change node flavor** option is grayed out, you can select **Change all specifications** to increase or decrease the nodes and their capacities on the GaussDB(DWS) console. First, you need to configure the new specifications you want, and a cluster with these specifications will be created. Then, data will be migrated from the old cluster to the new one. In case you need to restore data, a full snapshot will be taken for the old cluster, and the old cluster will be retained for a period of time.

NOTE

- To use this feature, contact technical support engineers to upgrade your version first.
- A cluster can have up to 240 nodes. The old and new clusters can have up to 480 nodes in total.

Impact of Changing All Specifications

- Before the change, you need to exit the client connections that have created temporary tables, because temporary tables created before or during the change will become invalid and operations performed on these temporary tables will fail. The temporary tables created after the change are not affected.
- The change involves data redistribution, during which the cluster is read-only.
- After the specifications are changed, the private IP address changes, which should be updated for connection.
- After the specifications are changed, the domain name remains unchanged, and the IP address bound to the domain name is switched. During the switchover, the connection is interrupted for a short period of time. Therefore, avoid writing service statements in the switchover. If the service side uses a domain name for connection, you need to update the cache information corresponding to the domain name to prevent connection failure after the change.
- If an ELB is bound to the cluster, the connection address on the service side remains unchanged after the specifications are changed, while the internal server address of the ELB is changed to the new connection address.
- In case you need to restore data, a full snapshot will be taken for the old cluster (on condition that your cluster support snapshot creation). You can check it in the snapshot list and manually delete it if it is no longer necessary.
- During the change, the cluster is read-only, affecting intelligent O&M tasks. You are advised to start these tasks after the change or pause them before the change.

Prerequisites

- The cluster to be changed is in the **Available**, **Read-only**, or **Unbalanced** state.

- The number of nodes after resizing must be smaller than or equals the available node quotas, or the resizing will fail.
- The total capacity of the new cluster after the change must be at least 1.2 times greater than the used capacity of the old cluster.

Changing All Specifications

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 Choose **Clusters > Dedicated Cluster**. All clusters are displayed by default.

Step 3 In the row of a cluster, choose **More > Change Specifications** in the **Operation** column and click **Change all specifications**.

- For the **Node Flavor** parameter, select a flavor.

NOTE

The VPC, subnet, and security group of the new cluster are the same as those of the original cluster.

- For the **Set to** parameter, set the number of nodes you want for the new cluster.

Step 4 (Optional) If the cluster storage can be modified, you can set the storage type and the available storage for each node.

Step 5 Read the nodes and select **Confirmed**. Click **Resize Cluster Now**.

Step 6 Click **Submit**.

- After the change request is submitted, **Task Status** of the cluster changes to **Changing all specifications**. The process will take several minutes.
- During the change, the cluster automatically restarts, and **Cluster Status** is **Unavailable** for a period of time. After the restart is complete, **Cluster Status** changes to **Available**. Data is redistributed during resizing. During the redistribution, **Cluster Status** is **Read-only**.
- The resizing succeeds only when **Cluster Status** is **Available** and the **Change all specifications** task in **Task Information** is complete. Then the cluster begins providing services.
- If **Change all specifications failed** is displayed, the cluster failed to be changed.
- If change fails, and a message requiring retry is displayed when you click **Resize**, the failure is probably caused by abnormal cluster status or network problems. In this case, contact technical support to troubleshoot the problem and try again.

----End

8.3.3 Disk Capacity Expansion of an EVS Cluster

Context

In conventional scaling, compute and storage resources are coupled. If a company scales out disks, it has to add unnecessary CPUs and memory at the same time. The scaling takes a long time and interrupts services. Disk capacity expansion can

quickly increase storage without service interruption. You can increase disk space without having to stop services.

NOTE

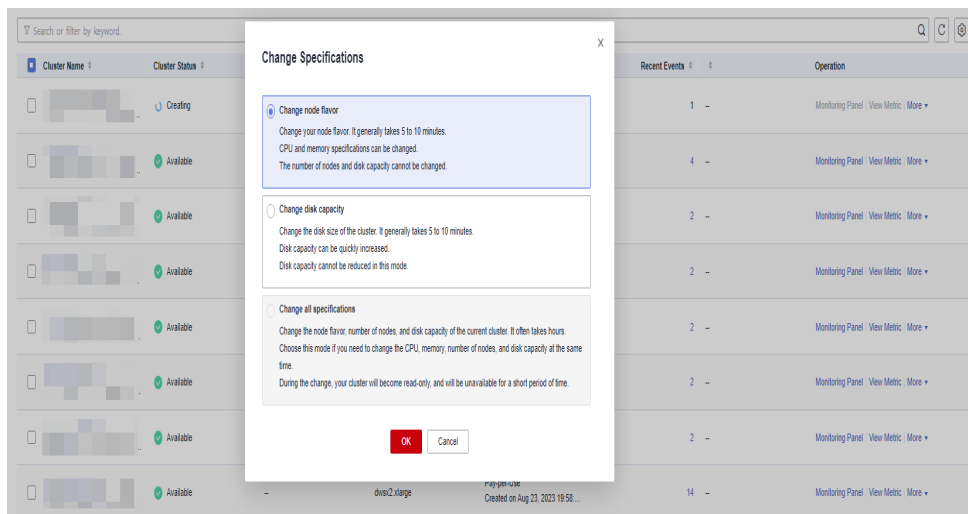
- Disk capacity expansion can be performed only for standard data warehouses using SSD. Only version 8.1.1.203 and later are supported.
- Disk capacity can be expanded only if the cluster is in **Available**, **To be restarted**, **Read-only**, or **Node fault**, **Unbalanced** state.

Precautions

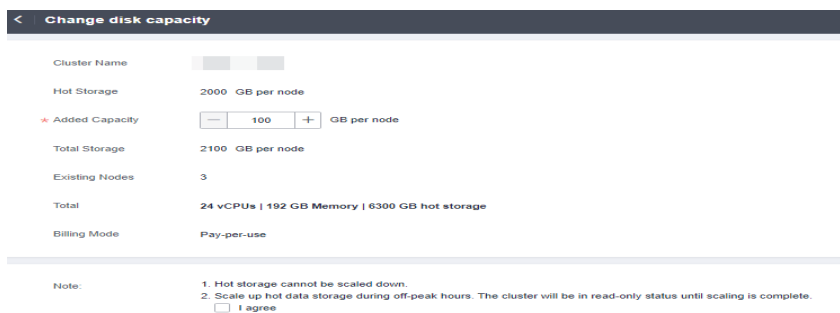
- Hot storage disks cannot be scaled down.
- Scale up hot data storage during off-peak hours.
- If the cluster is in the read-only state, a message will be displayed after you click **Expand Disk Capacity**. After you start expansion, wait until it is completed and the cluster changes to the available state.

Procedure

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** Choose **Clusters > Dedicated Cluster**. All clusters are displayed by default.
- Step 3** In the **Operation** column of the target cluster, choose **More > Change Specifications** and click **Change disk capacity**. The **Expand Disk Capacity** page is displayed.



- Step 4** Set the capacity and click **Resize Cluster Now**.



Step 5 Confirm the settings and click **Submit**.

Step 6 Return to the cluster list and check the disk capacity expansion progress.

Cluster Name	Cluster Status	Task Information ⓘ	Node Flavor	Recent Events	Operation
v [] 302	Read-only	Expanding disk capacity	dwsl2.2xlarge	3	View Metric Restart More ▾
v [] 11	Available	--	dwsl2.2xlarge	8	View Metric Restart More ▾

----End

9 Backup and Disaster Recovery

9.1 Snapshots

9.1.1 Overview

A snapshot is a full or incremental backup of a GaussDB(DWS) cluster at a specific point in time. It records the current database data and cluster information, including the number of nodes, node specifications, and database administrator name. Snapshots can be created manually or automatically. For details, see [Manual Snapshots](#) and [Automated Snapshots](#).

If you restore a snapshot to a new cluster, GaussDB(DWS) creates a new cluster based on the cluster information recorded in the snapshot, and then restores data from the snapshot. For more information, see [Restoring a Snapshot to a New Cluster](#).

If you restore a snapshot to the original cluster, GaussDB(DWS) clears the existing data in the cluster, and then restores the database information from the snapshot to the cluster. For more information, see [Restoring a Snapshot to the Original Cluster](#).

The snapshot backup and restoration rates are listed below. (The rates are obtained from the test environment with local SSDs as the backup media. The rates are for reference only.) The actual rate depends on your disk, network, and bandwidth resources.)

- Backup rate: 200 MB/s/DN
- Restoration rate: 125 MB/s/DN

 NOTE

- Snapshot storage space
 - The cluster storage is provided by GaussDB(DWS) free of charge. Cluster storage = Storage space per node x Number of nodes
- The dependency of the snapshot service is as follows:
 - Only the snapshots stored in OBS can be used to restore data to a new cluster.
- The new GaussDB(DWS) cluster created based on the snapshot must have the same configurations as the original cluster. That is, the number and specifications of nodes, memory, and disks in the new cluster must be the same as those in the original cluster.
- If you create a new cluster based on a snapshot without modifying parameters, the parameters of the new cluster will be the same as those of the snapshot.
- During snapshot creation, do not perform the VACUUM FULL operation, or the cluster may become read-only.
- Snapshot creation affects disk I/O performance. You are advised to create snapshots during off-peak hours.
- During the snapshot creation, some intermediate files are retained, which occupy extra disk space. Therefore, create snapshots in off-peak hours and ensure that the disk capacity usage is less than 70%.

9.1.2 Manual Snapshots

9.1.2.1 Creating a Manual Snapshot

Prerequisites

A cluster snapshot is a complete backup that records point-in-time configuration data and service data of a GaussDB(DWS) cluster. This section describes how to create a snapshot on the **Snapshots** page to back up cluster data.

A manual snapshot can be created at any time. It will be retained until it is deleted from the GaussDB(DWS) console. Manual snapshots are full backup data, which takes a long time to create.

 NOTE

- Manual cluster snapshots can be backed up to OBS.
- To create a manual snapshot of a cluster, the cluster state must be **Available**, **To be restarted**, or **Unbalanced**. In cluster versions earlier than 8.1.3.101, you can also create a snapshot of a cluster in the **Read-only** state.

Impact on the System

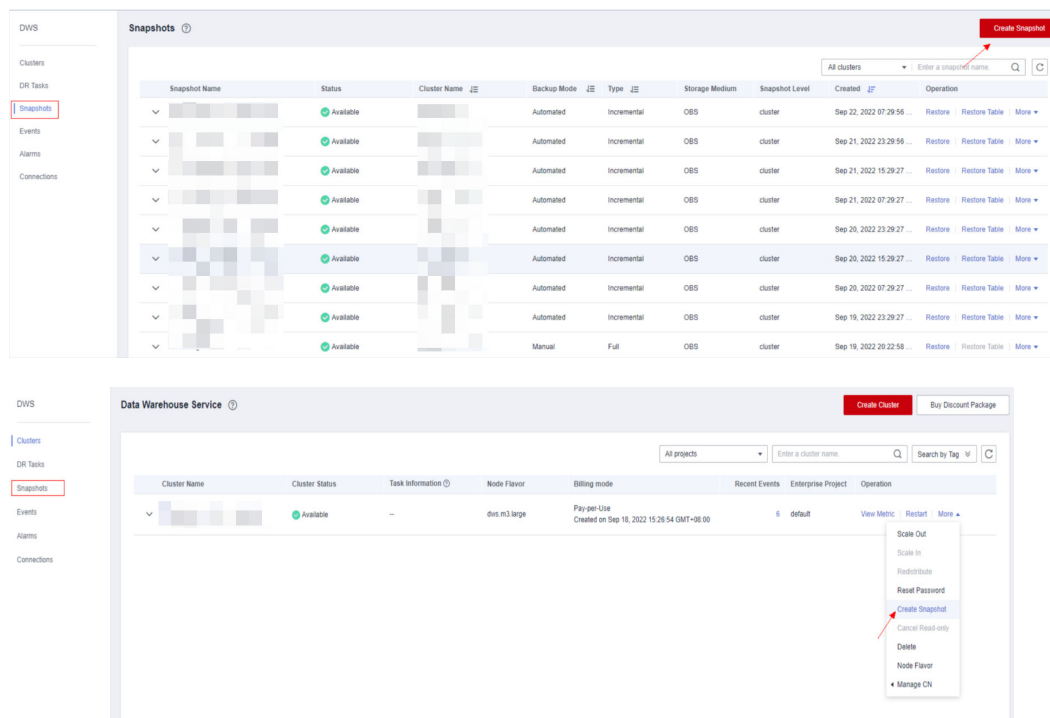
If a snapshot is being created for a cluster, the cluster cannot be restarted, scaled, its password cannot be reset, and its configurations cannot be modified.

 NOTE

To ensure the integrity of snapshot data, do not write data during snapshot creation.

Procedure

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane, choose **Snapshots**.
- Step 3** In the navigation pane, choose **Snapshots**. Click **Create Snapshot** in the upper right corner. Alternatively, choose **More > Create Snapshot** in the **Operation** column on the **Dedicated Clusters** page.



- Step 4** Configure the following snapshot information:
 - **Cluster Name:** Select a GaussDB(DWS) cluster from the drop-down list. The drop-down list only displays clusters that are in the **Available** state.
 - **Snapshot Name:** Enter a snapshot name. The snapshot name must be 4 to 64 characters in length and start with a letter. It is case-insensitive and contains only letters, digits, hyphens (-), and underscores (_).
 - **Snapshot Level:** Select **cluster**.
 - **Snapshot Description:** Enter the snapshot information. This parameter is optional. Snapshot information contains 0 to 256 characters and does not support the following special characters: !<>!=&"

* Cluster Name ? C

* Snapshot Name ?

* Snapshot Level

Snapshot Description ?

0/256

Step 5 Click **Create**.

Task status of the cluster for which you are creating a snapshot is **Creating snapshot**. The status of the snapshot that is being created is **Creating**. After the snapshot is created, its status changes to **Available**.

NOTE

If the snapshot size is much greater than that of the data stored in the cluster, the data is possibly labeled with a deletion tag, but is not cleared and reclaimed. In this case, clear the data and recreate a snapshot. For details, see [How Can I Clear and Reclaim the Storage Space?](#)

----End

9.1.2.2 Deleting a Manual Snapshot

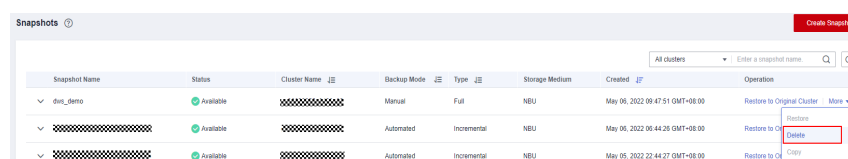
On the **Snapshot Management** page of the GaussDB(DWS) management console, you can delete an unwanted snapshot in the **Unavailable** state or delete an available snapshot to release the storage space.

CAUTION

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

Procedure

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane, choose **Snapshots**. All snapshots are displayed by default.
- Step 3** In the **Operation** column of the snapshot that you want to delete, choose **More > Delete**.



NOTE

You can only delete snapshots that were manually created.

Step 4 If the information is correct, enter **DELETE** and click **OK** to delete the snapshot.

----End

9.1.3 Automated Snapshots

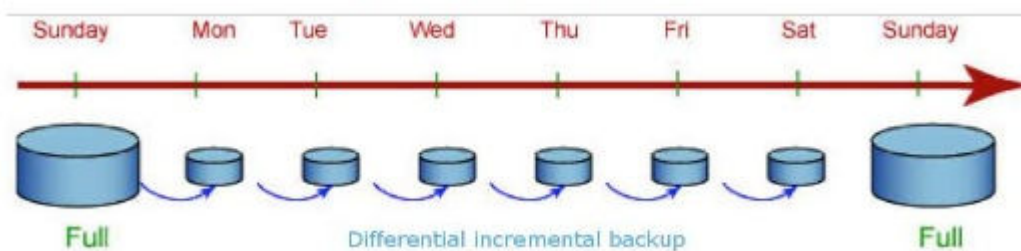
9.1.3.1 Automatic Snapshot Overview

Automated snapshots adopt differential incremental backups. The automated snapshot created for the first time is a full backup (base version), and then the system creates full backups at a specified interval. Incremental backups are generated between two full backups. The incremental backup records change based on the previous backup.

During snapshot restoration, GaussDB(DWS) uses all backups between the latest full backup and the current incremental backup to restore the cluster. Therefore, no data loss occurs.

If the retention period of an incremental snapshot exceeds the maximum retention period, GaussDB(DWS) does not delete the snapshot immediately. Instead, GaussDB(DWS) retains it until the next full backup is completed, when the deletion of the snapshot will not hinder incremental data backup and restoration.

Figure 9-1 Snapshot backup process



Automated snapshots are enabled by default when you create a cluster. If automated snapshots are enabled for a cluster, GaussDB(DWS) periodically takes snapshots of that cluster based on the time and interval you set, usually every eight hours. You can configure one or more automated snapshot policies for the cluster as required. For details, see [Configuring an Automated Snapshot Policy](#).

The retention period of an automated snapshot can be set to 1 to 31 days. The default retention period is 3 days. The system deletes the snapshot at the end of the retention period. If you want to keep an automated snapshot for a longer period, you can create a copy of it as a manual snapshot. The automated snapshot is retained until the end of the retention period, whereas the corresponding manual snapshot is retained until you manually delete it. For details about how to copy an automated snapshot, see [Copying Automated Snapshots](#).

9.1.3.2 Configuring an Automated Snapshot Policy

You can select a snapshot type and set one or more automated snapshot policies for a cluster. After an automated snapshot policy is enabled, the system automatically creates snapshots based on the time, period, and snapshot type you configured.

Procedure



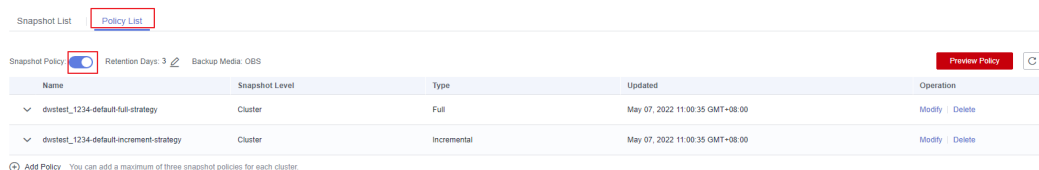
- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane on the left, choose **Clusters > Dedicated Clusters**.
- Step 3** In the cluster list, click the name of the target cluster. The **Cluster Information** page is displayed.
- Step 4** Click the **Snapshots** tab page and click **Policy List**. All policies of the current cluster are displayed on the **Policy List** page. Toggle on **Snapshot Policy**.
 -  indicates that the policy is enabled (default). The default retention period is three days.
 -  indicates that the automatic snapshot function is disabled.

Figure 9-2 Policy list



- Step 5** After this function is enabled, you can set the retention mode for automated snapshots. For more information, see [Table 9-1](#).

Table 9-1 Automated snapshot parameters

Parameter	Description
Retention Days	Retention days of the snapshots that are automatically created. The value ranges from 1 to 31 days. NOTE Snapshots that are automatically created cannot be deleted manually. The system automatically deletes these snapshots when their retention duration exceeds the threshold.

- Step 6** After automated snapshot is enabled, you can configure its parameters. For more information, see [Table 9-2](#).

NOTE

The snapshot creation time is UTC, which may be different from your local time.

- If the snapshot type is set to **Full**, you can choose either **Periodic** or **One-time**, as shown in the following figures.
 - **Periodic**: Specify the days for every week/month and the exact time on the days.


WARNING


Choosing the days in red (29th/30th/31st) may skip some monthly backups.

- **One-time**: Specify a day and the exact time on the day.

- Incremental snapshots can be set only to **Periodic**, as shown in the first figure below.

When configuring a periodic incremental snapshot policy, you can specify the days for every week/month and the exact time on the days. You can also specify the start time and interval for the snapshots.



Snapshot Policy  No full snapshot policy is configured for the current cluster. The default policy is used, that is, a full snapshot is taken every 14 incremental snapshots. You can set full snapshot policies as required.

Name 



Type Full Incremental

Policy Periodic

Periodic Policy Configurations

Days Weekly  Monthly 

Sunday Monday Tuesday Wednesday Thursday Friday Saturday

Time Daily  Interval 

Create a backup at UTC

Note: The UTC time is used by default. Set the policy based on the time zone and time difference as required.

Table 9-2 Snapshot policy parameters

Parameter	Description
Name	The policy name must be unique, consist of 4 to 92 characters, and start with a letter. It is case-insensitive and can contain only letters, digits, hyphens (-), and underscores (_).
Type	You can choose either full or incremental snapshots. NOTE <ul style="list-style-type: none"> A full snapshot is created after every fifteen incremental snapshots are created. Incremental snapshot restoration is based on full snapshots. Incremental snapshots are used to restore all data to the time point when they were created. An incremental snapshot records the changes made after the previous snapshot was created. A full snapshot backs up the data of an entire cluster. It takes a short time to create an incremental snapshot, and a long time to create a full snapshot. When restoring a snapshot to a new cluster, GaussDB(DWS) uses all snapshots between the latest full backup and the current snapshot.
Policy	You can choose either periodic or one-time snapshots. NOTE One-time can be selected only for full snapshots.
One-time	You can create a full snapshot at a specified time in the future. The UTC time is used.

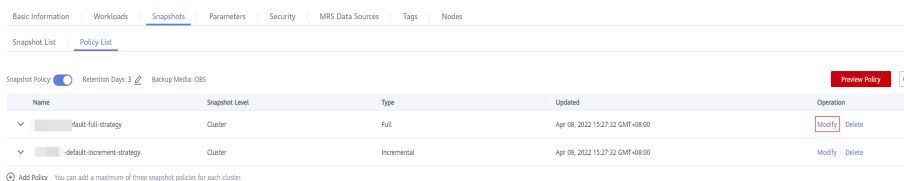
Parameter	Description
Periodic Policy Configurations	<p>You can create automated snapshots on a daily, weekly, or monthly basis:</p> <ul style="list-style-type: none"> • Days: Specify days for every week or every month. Weekly and Monthly cannot be selected at the same time. For Monthly, the specified days are applicable only to months that contain the dates. For example, if you select 29, no automated snapshot will be created on February, 2022. • Time: Specify the exact time on the selected days. For incremental snapshots, you can specify the start time and interval. The interval can be 4 to 24 hours, indicating that a snapshot is created at an interval of 4 to 24 hours. <p>NOTICE If the incremental data is large and the execution period is long, the backup will be slow. In this case, increase the backup frequency.</p>

Step 7 Click **OK**.

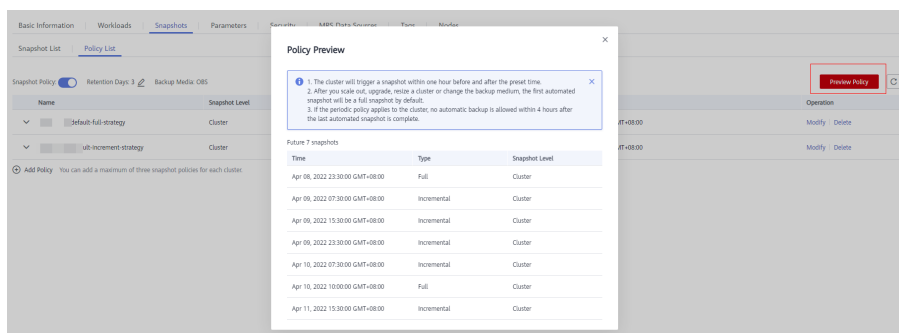
 **NOTE**

A maximum of three snapshot policies can be set for a cluster.

Step 8 (Optional) To modify an automated snapshot policy, click **Modify** in the **Operation** column.



Step 9 (Optional) To preview a policy, click **Preview Policy**. The next seven snapshots of the cluster will be displayed. If no full snapshot policy is configured for the cluster, the default policy is used, that is, a full snapshot is taken after every 14 incremental snapshots.



NOTICE

Implementation of the same policy varies according to operations in the cluster. For example:

- The policy preview time is for your reference only. The cluster triggers a snapshot within one hour before and after the preset time.
- The next automated snapshots after cluster scale-out, upgrade, resize, and media modification are full snapshots by default.
- If a periodic policy is used for a cluster, no automatic backup is allowed within 4 hours after the last automated snapshot is complete.
- If the time for triggering snapshots of multiple policies conflicts, the priorities of the policies are as follows: one-time > periodic > full > incremental.
- You can use any backup, full or incremental, to restore the full data of a resource.

----End

9.1.3.3 Copying Automated Snapshots

This section describes how to copy snapshots that are automatically created for long-term retention.

Copying an Automated Snapshot

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 In the navigation pane, choose **Snapshots**.

All snapshots are displayed by default. You can copy the snapshots that were automatically created.

Step 3 In the **Operation** column of the snapshot that you want to copy, choose **More > Copy**.

- **New Snapshot Name:** Enter a new snapshot name.

The snapshot name must be 4 to 64 characters in length and start with a letter. It is case-insensitive and contains only letters, digits, hyphens (-), and underscores (_).

- **Snapshot Description:** Enter the snapshot information.

This parameter is optional. Snapshot information contains 0 to 256 characters and does not support the following special characters: !<>'=&"

Figure 9-3 Copying a snapshot

Copy Snapshot

* Source Snapshot Name dws-3n-20200120081439

* New Snapshot Name ?

Snapshot Description ?

0/256

Step 4 Click **OK**. The system starts to copy the snapshot for the cluster.

The system displays a message indicating that the snapshot is successfully copied and delivered. After the snapshot is copied, the status of the copied snapshot is **Available**.

NOTE

If the snapshot size is much greater than that of the data stored in the cluster, the data is possibly labeled with a deletion tag, but is not cleared and reclaimed. In this case, clear the data and recreate a snapshot. For details, see [How Can I Clear and Reclaim the Storage Space?](#)

----End

9.1.3.4 Deleting an Automated Snapshot

Only GaussDB(DWS) can delete automated snapshots; you cannot delete them manually.

GaussDB(DWS) deletes an automated snapshot if:

- The retention period of the snapshot ends.
- The cluster is deleted.

CAUTION

To help users restore a cluster deleted by mistake, GaussDB(DWS) provides the following policies (supported only in 8.2.0 and later) for cluster snapshots:

- If the latest snapshot is an automated snapshot, it will be retained for one day.
 - If the latest snapshot is a manual snapshot, the automated snapshot of the cluster will be deleted.
-


9.1.4 Viewing Snapshot Information

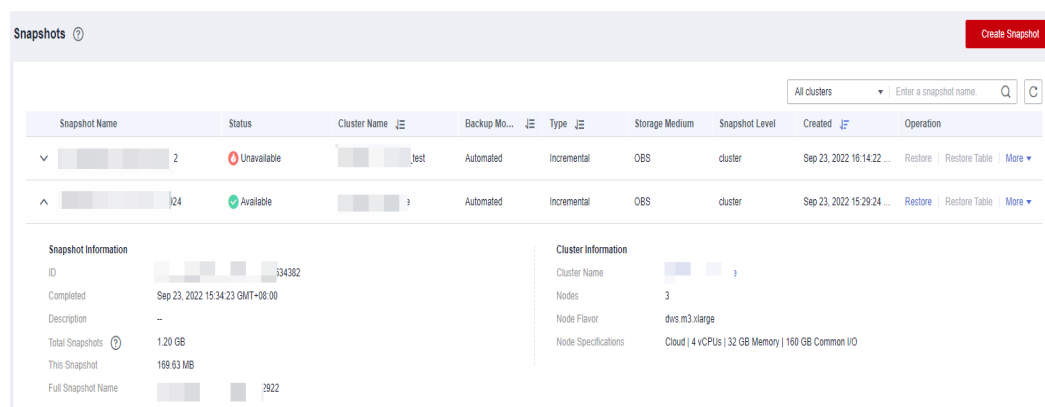
This section describes how to view snapshot information on the **Snapshots** page.

Viewing Snapshot Information

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 In the navigation pane on the left, choose **Snapshots**.

In the snapshot list, all snapshots are displayed by default. Click  next to the snapshot name to check the snapshot details.



Step 3 You can view the **Snapshot Name**, **Snapshot Status**, **Cluster Name**, **Backup Mode**, **Snapshot Type**, **Storage Medium**, and creation time of snapshots.


You can also enter a snapshot name or cluster name in the upper right corner of the snapshot list and click  to search for the specified snapshot. GaussDB(DWS) supports fuzzy search.

Table 9-3 describes snapshot status.

Table 9-3 Snapshot status

Status	Description
Available	Indicates that the existing snapshot works properly.
Creating	Indicates that a snapshot is being created.
Unavailable	Indicates that the existing snapshot cannot provide services.

The following table describes the backup modes.

Table 9-4 Backup modes

Type	Description
Manual	Indicates the snapshot that you manually create through the GaussDB(DWS) management console or using APIs. You can delete the snapshots that are manually created.
Automated	Indicates the snapshot that is automatically created after the automated snapshot backup policy is enabled. You cannot delete the snapshots that are automatically created. The system automatically deletes the snapshots whose retention duration expires.

The following table describes the snapshot types.

Table 9-5 Type

Type	Description
Full	The snapshot is a full backup.
Incremental	The snapshot is an incremental backup.

The following table describes the snapshot media.

Table 9-6 Storage media

Storage Medium	Description
OBS	The created snapshot is an OBS snapshot and the backup data is stored on the OBS server.

----End

9.1.5 Restoration Using a Snapshot

9.1.5.1 Constraints on Restoring a Snapshot

Cluster-Level Snapshot Restoration

Cluster-level restoration consists of two steps:

1. Data restoration: Restores data in the backup set to the data directory of each primary DN/CN instance in parallel.
2. Rebuilding the standby DN: After the primary DN is restored, standby DNs are rebuilt with full data in parallel.

 NOTE

- The restoration process takes 1.5 to 2 times longer than the backup process.
- The parameters after cluster-level restoration are the same as those before backup. When restoring data to a new cluster, ensure that the flavor of the new cluster is the same as that of the original cluster. If the flavor of the new cluster is smaller, the restoration may fail.

9.1.5.2 Restoring a Snapshot to a New Cluster

Scenario

This section describes how to restore a snapshot to a new cluster when you want to check point-in-time snapshot data of the cluster.

When a snapshot is restored to a new cluster, the restoration time is determined by the amount of data backed up by the snapshot. If a snapshot contains a large amount of data, the restoration will be slow. A small snapshot can be quickly restored.

Automatic snapshots are incremental backups. When restoring a snapshot to a new cluster, GaussDB(DWS) uses all snapshots between the latest full backup and the current snapshot. You can set the backup frequency. If snapshots are backed up only once a week, the backup will be slow if the incremental data volume is large. You are advised to increase the backup frequency.

NOTICE

- Currently, you can only use the snapshots stored in OBS to restore data to a new cluster.
 - By default, the new cluster created during restoration has the same specifications and node quantity as the original cluster.
 - Restoring data to a new cluster does not affect the services running in the original cluster.
 - If cold and hot tables are used, snapshots cannot be used to restore cold data to a new cluster.
 - Fine-grained restoration does not support tables in absolute or relative tablespace.
-

Prerequisites

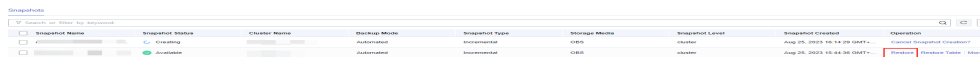
- The resources required for restoring data to a new cluster do not exceed your available resource quota.
- The snapshot is in the **Available** state.

Procedure

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 In the navigation pane, choose **Snapshots**. All snapshots are displayed by default.

Step 3 In the **Operation** column of a snapshot, click **Restore**.



Step 4 On the **Restore Snapshot** page, configure the parameters of the new cluster, as shown in the following figure.

1. Restore to a single-AZ cluster.

You can modify cluster parameters. For details, see [Table 9-7](#). By default, other parameters are the same as those in the snapshot. For details, see [Table 9-2](#).

Table 9-7 Parameters for the new cluster

Category	Operation
Basic settings	Region, AZ, node flavor, cluster name, database port, VPC, subnet, security group, public access, and enterprise project
Advanced settings	If Custom is selected, configure the following parameters: <ul style="list-style-type: none"> • Tag: If encryption is enabled for the original cluster, you can configure a key name.

Step 5 Click **Restore** to go to the confirmation page.

Step 6 Click **Submit** to restore the snapshot to the new cluster.

When the status of the new cluster changes to **Available**, the snapshot is restored.

After the snapshot is restored, the private network address and EIP (if **EIP** is set to **Automatically assign**) are automatically assigned.

----End

9.1.5.3 Restoring a Snapshot to the Original Cluster

Scenario

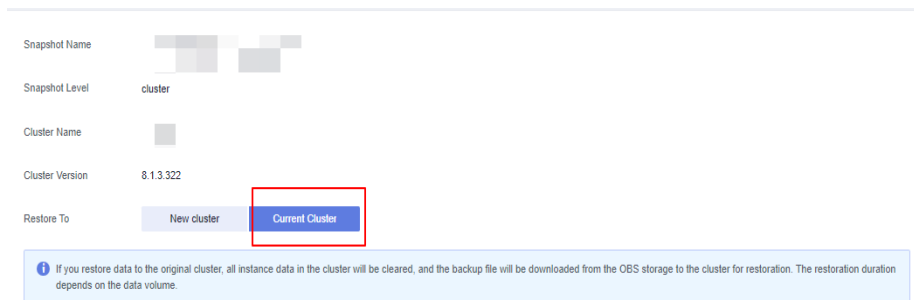
You can use a snapshot to restore data to the original cluster. This function is used when a cluster is faulty or data needs to be rolled back to a specified snapshot version.

NOTICE

- This function is supported only by clusters of version 8.1.3.200 or later.
- Snapshots whose backup device is OBS can be backed up.
- Only a snapshot in the **Available** state can be used for restoration.
- Logical clusters and resource pools cannot be restored to the current cluster.

Procedure

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane, choose **Snapshots**. All snapshots are displayed by default.
- Step 3** In the snapshot list, locate the row that contains the target snapshot and click **Restore** in the **Operation** column. The **Restore Snapshot to New Cluster** page is displayed.
- Step 4** Restore the snapshot to the current cluster.



NOTE

If you use a snapshot to restore data to the original cluster, the cluster will be unavailable during the restoration.

----End

9.1.6 Configuring a Snapshot

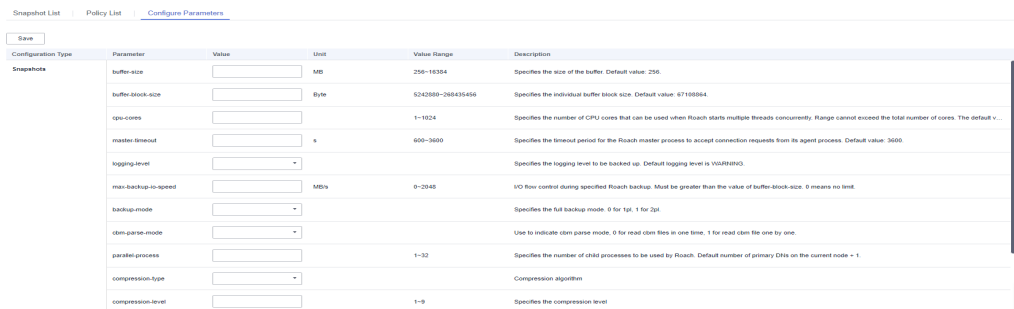
You can configure the parameters for creating and restoring a snapshot.

NOTE

- This feature applies only to clusters of 8.2.0 or later. (For clusters of versions earlier than 8.2.0, only some parameters can be configured.)
- The parameters take effect on all the snapshot creation and restoration tasks.

Procedure

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane on the left, choose **Clusters** > **Dedicated Clusters**.
- Step 3** In the cluster list, click the name of the target cluster. The **Cluster Information** page is displayed.
- Step 4** Click the **Snapshots** tab page and click **Configure Parameters**. All the configurable parameters of the current cluster will be displayed.
- Step 5** Configure parameters as required. For details, see [Table 9-8](#).



Step 6 Click **Save**.

----End

Snapshot parameters

Table 9-8 Snapshot information

Parameter	Type	Description	Default Value
parallel-process	Backup parameter	Number of concurrent processes on each node during Roach backup. NOTE This parameter can be configured for clusters earlier than 8.2.0.	The value is the number of DNs on the current node.
compression-type	Backup parameter	Compression algorithm. <ul style="list-style-type: none"> zlib LZ4 NOTE This parameter can be configured for clusters earlier than 8.2.0.	LZ4
compression-level	Backup parameter	Compression level. The value range is 0 to 9. <ul style="list-style-type: none"> 0: fast backup and no compression 9: slow backup and maximum compression NOTE This parameter can be configured for clusters earlier than 8.2.0.	6
buffer-size	Backup parameter	Buffer size of the Roach upload media. The value range is 256 to 16,384, in MB.	256

Parameter	Type	Description	Default Value
buffer-block-size	Backup parameter	Data block size of the data file to be read by Roach. The value range is 5,242,880 to 268,435,456, in bytes.	67108864
cpu-cores	Backup parameter	Number of CPU cores that can be used when Roach starts multiple threads concurrently	1/2 of the total number of logical CPU cores on the node
master-timeout	Backup parameter	Timeout period for the communication between the Roach master and agent nodes. The value range is 600 to 3600, in seconds.	3600
max-backup-io-speed	Backup parameter	I/O flow control during Roach backup. The value range is 0 to 2048, in MB/s. The value must be greater than the value of buffer-block-size . The value 0 indicates no limit.	0
backup-mode	Backup parameter	Full backup mode. <ul style="list-style-type: none">● 0: phase-1 backup● 1: phase-2 backup	0
cbm-parse-mode	Backup parameter	Incremental backup mode. <ul style="list-style-type: none">● 0: one-time CBM scan (high memory usage and high performance)● 1: multiple CBM scans (stable memory usage and low performance)	0
parallel-process	Restoration parameter	Number of concurrent processes on each node during Roach backup. By default, the value is the number of primary DNs on the current node plus 1.	1
cpu-cores	Restoration parameter	Number of CPU cores that can be used when Roach starts multiple threads concurrently	The default value is 1/2 of the number of CPU cores.

Parameter	Type	Description	Default Value
logging-level	Restoration parameter	Log levels: <ul style="list-style-type: none">● FATAL: Unrecoverable faults that cause the system suspension. This is the most severe level.● ERROR: Major errors.● WARNING: Exceptions. In this case, the system may continue to process tasks.● INFO: Notes.● DEBUG: Debugging details.● DEBUG2: Detailed debugging information, which is generally not displayed. This is the least severe level.	INFO

9.1.7 Stopping Snapshot Creation

You can stop snapshot creation on the **Snapshots** page.

NOTE

- This feature is supported only in version 8.1.3.200 and later.
- If the snapshot is ready to complete, the command for stopping the snapshot will not take effect and the snapshot will end normally.

Precautions

Only the snapshots in the **Creating** state can be stopped. A snapshot creation task that just started or is about to complete cannot be stopped.

Procedure

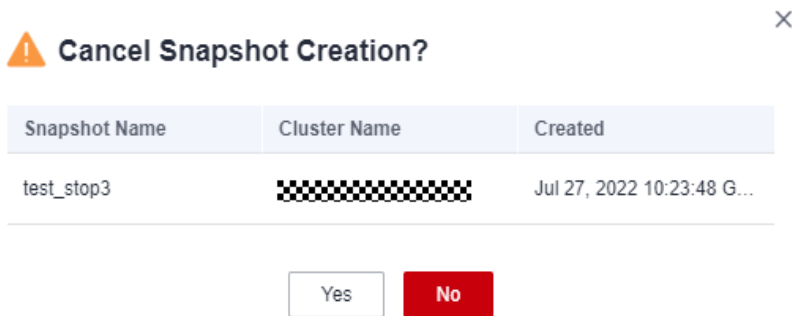
Step 1 Log in to the GaussDB(DWS) management console.

Step 2 In the navigation pane on the left, choose **Snapshots**.

In the **Operation** column of a snapshot that is being created, and click **Cancel Creation**.

Snapshot Name	Status	Cluster Name	Backup Mode	Type	Storage Medium	Snapshot Level	Created	Operation
...	Creating 14%	...	Automated	Full	OBS	cluster	Sep 13, 2022 10:44:26 GMT...	Cancel Creation
...	Available	...	Manual	Full	OBS	cluster	Sep 13, 2022 10:19:21 GMT...	Restore Restore Table More

Step 3 In the dialog box that is displayed, click **Yes** to stop the snapshot. The snapshot state will change to **Unavailable**.



Snapshot Name	Status	Cluster Name	Backup Mode	Type	Storage Medium	Snapshot Level	Created	Operation
...	Unavailable	...	Automated	Full	OBS	cluster	Sep 13, 2022 10:44:26 GMT...	Restore Restore Table More
...	Available	...	Manual	Full	OBS	cluster	Sep 13, 2022 10:19:21 GMT...	Restore Restore Table More

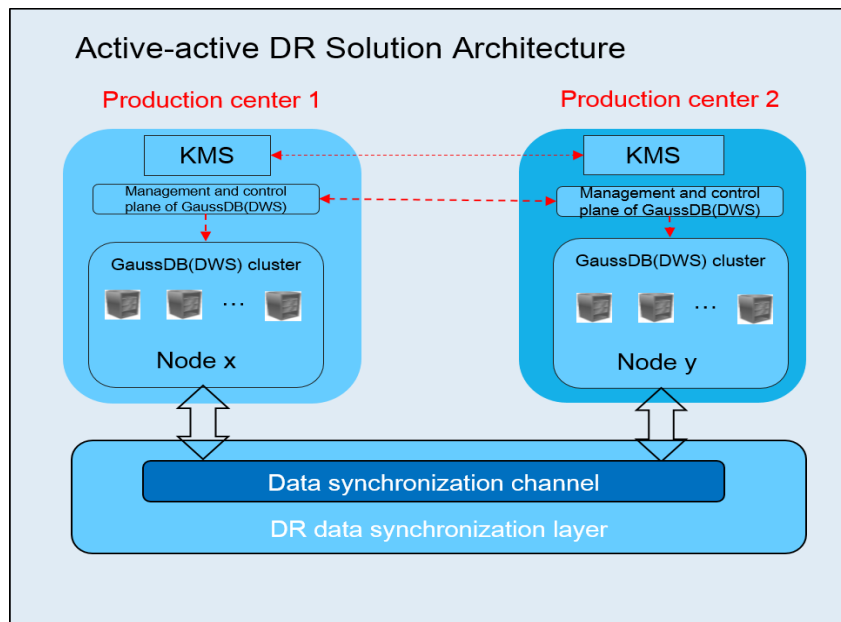
----End

9.2 Cluster DR

9.2.1 DR Overview

Overview

A homogeneous GaussDB(DWS) disaster recovery (DR) cluster is deployed in the same region. If the production cluster fails to provide read and write services due to natural disasters in the specified region or cluster internal faults, the DR cluster becomes the production cluster to ensure service continuity. The following figure shows the architecture.



DR Features

- Multi-form DR
 - Intra-region DR
 - Multiple data synchronization modes: synchronization layer based on mutual trust
- Low TCO
 - Heterogeneous deployment (logical homogeneity)
 - Cluster-level DR
- Visual console
 - Automatic and one-click DR drills

Constraints and Limitations

- During data synchronization, a non-fine-grained DR cluster cannot provide read or write services.
- When the DR task is stopped or abnormal but the DR cluster is normal, the DR cluster can provide the read service. After the DR switchover is successful, the DR cluster can provide the read and write services.
- When the DR task is created, the snapshot function of the production cluster is normal, but that of the DR cluster is disabled. Besides, snapshot restoration of both clusters is disabled.
- Resource pools are not supported.
- If cold and hot tables are used, cold data is synchronized using OBS.
- DR does not synchronize data from external sources.
- DR management refers to dual-cluster DR under the same tenant.
- The DR cluster and the production cluster must be logically homogeneous and in the same type and version.
- The production cluster and DR cluster used for intra-region DR must be in the same VPC.

- In intra-region DR, after services are switched over from the production cluster to the DR cluster, the bound ELB is automatically switched to the new production cluster. During the switchover, the connection is interrupted for a short period of time. Do not run service statements to write data during the switchover.
- During intra-region DR, the EIP, intranet domain name, and connection IP address of the original production cluster are not automatically switched with the cluster switchover. The EIP, domain name, or IP address used for connection in the service system need to be switched to the new cluster.

9.2.2 Creating a DR Task

Creating an Intra-Region Cluster-Level DR Task

Prerequisites

You can create a DR task only when the cluster is in the **Available** or **Unbalanced** state.

Procedure

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane on the left, choose **DR Tasks**.
- Step 3** On the displayed page, click **Create**.
- Step 4** Select the type and enter the name of the DR task to be created.
 - **Type: Intra-region DR**
 - **Name:** Enter 4 to 64 case-insensitive characters, starting with a letter. Only letters, digits, hyphens (-), and underscores (_) are allowed.
- Step 5** Configure the production cluster.
 - Select a created production cluster from the drop-down list.
 - After a production cluster is selected, the system automatically displays its AZ.
- Step 6** Configure the DR cluster.
 - Select the AZ associated with the region where the DR cluster resides.

NOTE

- The AZ of the DR cluster can be the same as that of the production cluster. In a 3-AZ cluster, any of the three AZs can be selected for DR.
- After you select an AZ for the DR cluster, homogeneous DR clusters will be displayed. If no DR cluster is available, create a cluster with the same configurations as the production cluster.

DR Cluster Information

AZ AZ1 AZ2 AZ3

Cluster Name [Create DR Cluster](#)

No DR clusters available in the current AZ. Create a DR cluster with the same configurations as the production cluster. The configurations are as follows:
AZ: AZ1 | Cluster Type: Standard | Node Flavor: dwsx.xlarge | Nodes: 3 | VPC: vpc-caoyan-test-ipv6

Step 7 Configure advanced parameters. Select **Default** to keep the default values of the advanced parameters. You can also select **Custom** to modify the values.

- The DR synchronization period indicates the interval for synchronizing incremental data from the production cluster to the DR cluster. Set this parameter based on the actual service data volume.

 **NOTE**

The default DR synchronization period is 30 minutes.

Step 8 Click **OK**.

The DR status will then change to **Creating**. Wait until the creation is complete, and the DR status will change to **Not Started**.

----End

9.2.3 Viewing DR Information

Step 1 Log in to the GaussDB(DWS) console.

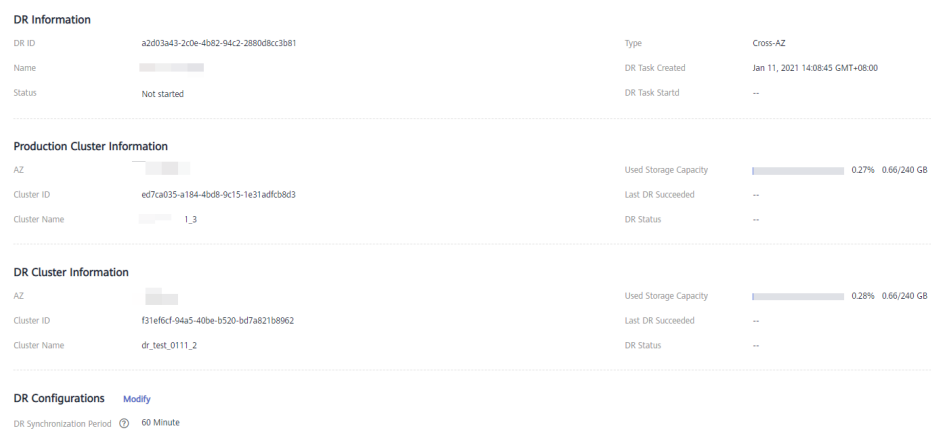
Step 2 In the navigation pane on the left, choose **DR Tasks**.

Step 3 In the DR list, click the name of a DR task.

On the page that is displayed, view the following information:

- **DR Information:** You can view the DR ID, DR name, DR creation time, and DR status.
- **Production Cluster Information:** You can view the production cluster ID, cluster name, AZ, used storage capacity, cluster DR status, and the time of the latest successful DR task.
- **DR Cluster Information:** You can view the DR cluster ID, cluster name, AZ, used storage capacity, cluster DR status, and the time of the latest successful DR task.
- **DR Configuration:** Users can view and modify the DR synchronization period.

Figure 9-4 Viewing the DR information



----End

9.2.4 DR Management

Starting a DR Task

- Step 1** Log in to the GaussDB(DWS) console.
- Step 2** In the navigation pane on the left, choose **DR Tasks**.
- Step 3** Click **Start** in the **Operation** column of the target DR task.



- Step 4** In the dialog box that is displayed, click **OK**.

The DR status will change to **Starting**. The process will take some time. After the task is started, the DR status will change to **Running**.

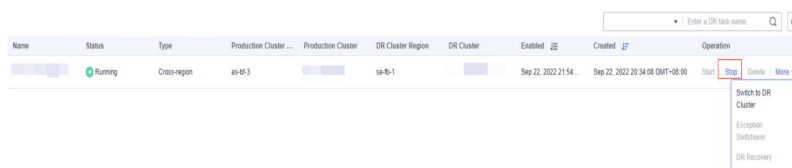
NOTE

- You can start a DR task that is in the **Not started/Startup failed/Stopped** state.
- After you start the DR task, you cannot perform operations, including restoration, scale-out, upgrade, restart, node replacement, and password update, on the production cluster or DR cluster. Backup is also not allowed on the DR cluster. Exercise caution when performing this operation.

----End

Stopping the DR Task

- Step 1** Log in to the GaussDB(DWS) console.
- Step 2** In the navigation pane on the left, choose **DR Tasks**.
- Step 3** Click **Stop** in the **Operation** column of the target DR task.



- Step 4** In the dialog box that is displayed, click **OK**.

The DR status will change to **Stopping**. The process will take some time. After the DR task is stopped, the status will change to **Stopped**.

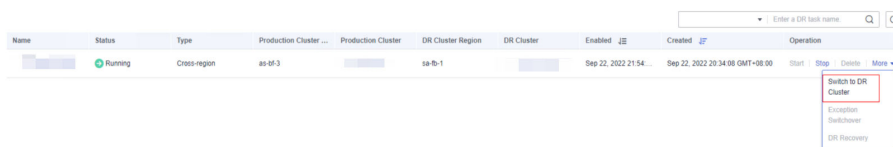
NOTE

- Only DR tasks in the **Running** or **Stop failed** state can be stopped.
- Data cannot be synchronized after a DR task is stopped.

----End

Switching to the DR Cluster

- Step 1** Log in to the GaussDB(DWS) console.
- Step 2** In the navigation pane on the left, choose **DR Tasks**.
- Step 3** Click **Switch to DR Cluster** in the **Operation** column of the target DR task.



- Step 4** In the dialog box that is displayed, click **OK**.

The DR status will change to **DR switching**.

After the switchover is successful, the DR status will change to the original status.

NOTE

- To perform a switchover when the DR cluster is running properly, click **Switch to DR Cluster**.
- You can perform a DR switchover when the DR task is in the **Running** state.
- During a switchover, the original production cluster is not available.
- Recovery Point Object (RPO) refers to the point in time to which a system and data must be restored after a disaster occurs. Its value varies by cluster status.
 - Production cluster in the **Available** state: RPO = 0
 - Production cluster in the **Unavailable** state: A zero RPO may not be achieved, but data can at least be restored to that of the latest successful DR synchronization (**Last DR Succeeded**). For details, see [Viewing DR Information](#).

----End

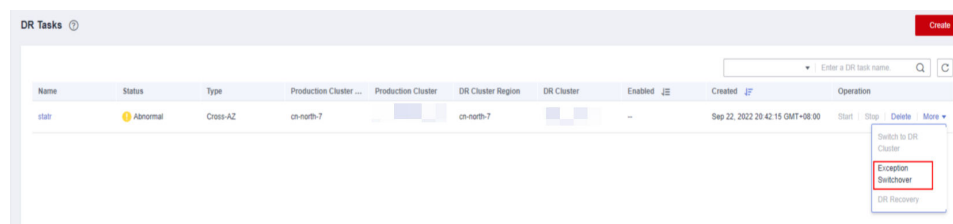
Exception Switchover

Scenario

The production cluster is unavailable, the DR cluster is normal, and the DR status is **Abnormal**.

Procedure

- Step 1** Log in to the GaussDB(DWS) console.
- Step 2** In the navigation pane on the left, choose **DR Tasks**.
- Step 3** Choose **More > Exception Switchover** in the **Operation** column of the target DR task.



Step 4 In the dialog box that is displayed, click **OK**.

The **Status** will change to **Switchover in progress**.

After the switchover is successful, the DR status will change to the original status. In this procedure, the DR status will change back to **Abnormal**.

NOTE

- To perform a switchover when the DR cluster is abnormal or the production cluster is faulty, click **Exception Switchover**.
- DR exception switchover is supported only by clusters of version 8.1.2 or later.
- Before a switchover, check the latest synchronization time in the DR cluster. The DR cluster will serve as a production cluster after an abnormal switchover, but the data that failed to be synchronized from the original production cluster to the DR cluster will not exist in the DR cluster.
- If the DR type is **Cross-region DR**, the switchover can be performed only in the region where the standby cluster is located.

----End

Performing a DR Switchback

Scenario

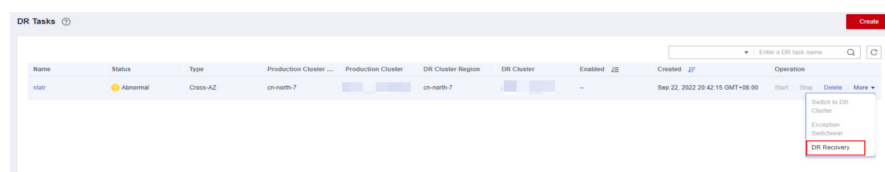
After abnormal switchover, if you have confirmed that the original production cluster was recovered, you can perform a switchback.

Procedure

Step 1 Log in to the GaussDB(DWS) console.

Step 2 In the navigation pane on the left, choose **DR Tasks**.

Step 3 Click **DR Recovery** in the **Operation** column of a DR task.



Step 4 In the displayed dialog box, set **Synchronization Mode** to **Incremental** or **Full**.

NOTE

You are advised to set **Synchronization Mode** to **Incremental** when updating a DR creation task.

Step 5 Click **OK**.

The **Status** will change to **Recovering**.

After the DR recovery is successful, the **Status** will change to **Running**.

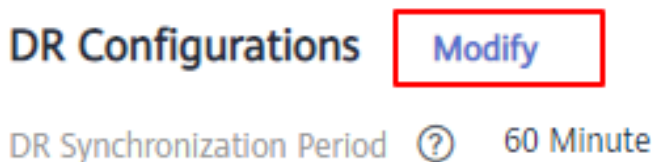
NOTE

- DR is supported only by clusters of 8.1.2 or later.
- During DR recovery, data in the DR cluster will be deleted, and the DR relationship will be re-established with the new production cluster.
- If the DR type is **Cross-region DR**, the recovery can be performed only in the region where the standby cluster is located.

----End

Updating DR Configurations

- Step 1** Log in to the GaussDB(DWS) console.
- Step 2** In the navigation pane on the left, choose **DR Tasks**.
- Step 3** In the DR list, click the DR name to go to the DR information page.
- Step 4** In the **DR Configurations** area, click **Modify**.

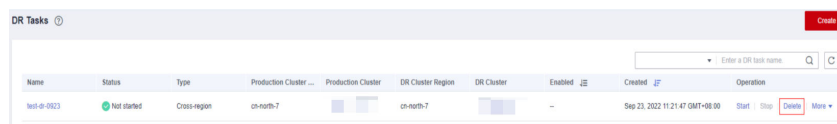
**NOTE**

- Only DR tasks in the **Not started** or **Stopped** state can be modified.
- The new configuration takes effect after DR is restarted.

----End

Deleting DR Tasks

- Step 1** Log in to the GaussDB(DWS) console.
- Step 2** In the navigation pane on the left, choose **DR Tasks**.
- Step 3** Click **Delete** in the **Operation** column of the target DR task.



- Step 4** In the dialog box that is displayed, click **OK**.

The DR status will change to **Deleting**.

NOTE

- You can delete a DR task when **DR Status** is **Creation failed**, **Not started**, **Startup failed**, **Stopped**, **Stop failed**, or **Abnormal**.
- Data cannot be synchronized after a DR task is deleted, and the deleted task cannot be restored.

----End

9.2.5 Mutually Exclusive DR Cases

Case 1: How Do I Scale out a Cluster in the DR State?

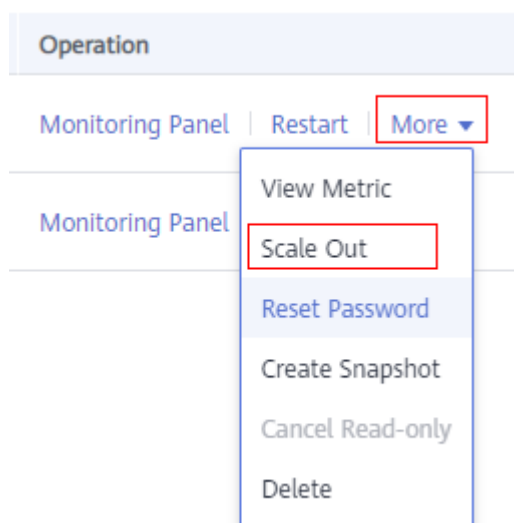
Step 1 Log in to the GaussDB(DWS) console.

Step 2 In the navigation pane on the left, choose **Clusters** > **Dedicated Clusters**.

Step 3 In the cluster list, if **Task Information** of the cluster you want to scale out is **DR not started**, perform [Step 5](#) and [Step 7](#).

Step 4 If the **Task Information** is other than **DR not started**, delete the DR task. For details, see [Deleting DR Tasks](#).

Step 5 In the **Operation** column of the production and DR clusters, choose **More** > **Scale Out**.



Step 6 Create a DR task. For details, see [Creating a DR Task](#).

Step 7 Start the DR task. For details, see [Starting a DR Task](#).

NOTE

After scale-out, the number of DNs in the production cluster must be the same as that in the DR cluster.

----End

10 Intelligent O&M

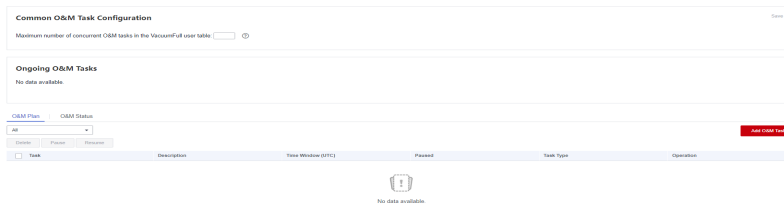
10.1 Overview

GaussDB(DWS) provides the intelligent O&M feature to help users quickly and efficiently execute O&M tasks. Intelligent O&M selects a proper time window and concurrency to complete specified tasks based on the cluster load. During O&M tasks, intelligent O&M monitors user service changes and promptly adapts task execution policies to minimize the impact on user services. Periodic tasks and one-off tasks are supported, and you can configure the time window as required.

Intelligent O&M ensures high availability. When the cluster is abnormal, failed O&M tasks will be retried. If some steps of an O&M task cannot be completed due to an abnormal cluster, the failed steps will be skipped for cost saving.

The intelligent O&M page consists of the following parts:

- Common configuration of O&M tasks: Currently, you can only configure **Maximum number of concurrent O&M tasks in the VacuumFull user table**. This configuration takes effect on all the VACUUM FULL tasks of user tables.
- Information about ongoing O&M tasks. (Currently, only VACUUM tasks are displayed. If disk space is insufficient because of table bloating, you can vacuum tables.)
 - Frequent table creation and deletion can lead to table bloating. To free up space, you can run the **VACUUM** command on system catalogs.
 - Frequently update and delete operations can lead to table bloating. To free up space, you can run the **VACUUM** or **VACUUM FULL** command on system catalogs.
- O&M details: **O&M Plan** and **O&M Status**. **O&M Plan** displays the basic information about all O&M tasks, and **O&M Status** displays the running status.



NOTE

- This feature is supported only in 8.1.3 or later.
- The intelligent O&M function is not supported in hybrid data warehouses (standalone mode).
- Only cluster 8.1.3 and later versions support the common configuration module for O&M tasks. For earlier versions, contact technical support to upgrade them.

10.2 O&M Plans

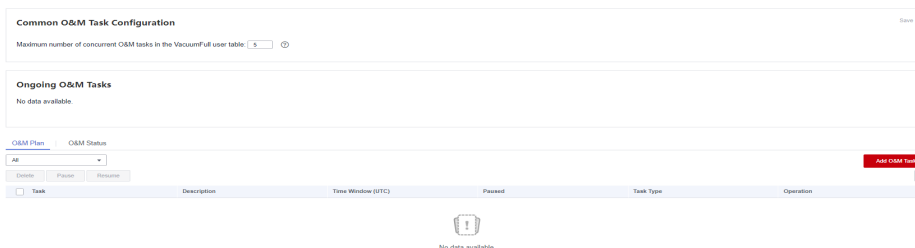
Setting the Common Configurations of O&M Tasks

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 Click the name of the target cluster.

Step 3 In the navigation pane, choose **Intelligent O&M**.

Step 4 In the **Common O&M Task Configuration** area, configure **Maximum number of concurrent O&M tasks in the VacuumFull user table**.



NOTE

- This configuration takes effect for the VACUUM FULL O&M tasks of all user tables.
- The concurrency value range is 1 to 24. Configure it based on the remaining disk space and I/O load. You are advised to set it to 5.

----End

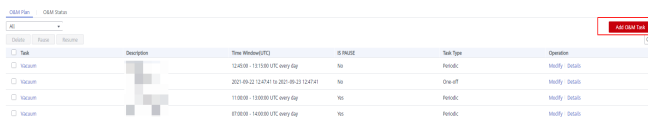
Adding an O&M Plan

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 Click the name of the target cluster.

Step 3 In the navigation pane, choose **Intelligent O&M**.

Step 4 Click the **O&M Plan** tab. Click **Add O&M Task**.



Step 5 In the displayed **Add O&M Task** dialog box, configure basic information about the O&M task.

Table 10-1 Basic configuration items of an O&M task

Configurat ion Item	Description	Example
O&M Task	Vacuum (Currently, only Vacuum O&M tasks are supported.)	Vacuum
Description	Brief description of the intelligent O&M task.	This intelligent O&M task helps users periodically invoke Vacuum commands to reclaim space.
Remarks	Supplementary information.	-
Scheduling Mode	<p>The supports the following scheduling modes:</p> <ul style="list-style-type: none"> • Auto: Intelligent O&M scans the database in a specified time window, and automatically delivers table-level vacuum tasks by service load and reclaimable space of user tables. • Specify: You need to specify a vacuum target. Intelligent O&M will automatically deliver a table-level vacuum task in a specified time window. • Priority: You can specify the preferential vacuum targets. During the remaining time window (if any), Intelligent O&M will automatically scan other tables that can be vacuumed and deliver table-level vacuum tasks. <p>NOTE You are advised to select Specify for VACUUM and VACUUM FULL operations. Do not perform VACUUM FULL on wide column-store tables. Otherwise, memory bloat may occur.</p>	Specify

Configuration Item	Description	Example
Autovacuum	<p>Supported: system catalog Vacuum or user table VacuumFull.</p> <ul style="list-style-type: none"> • A system catalog VACUUM transaction holds a level-5 lock (share update exclusive lock), which does not affect user services. Only the transactions on the DDL process of the system catalog are blocked. • A user table VACUUM FULL transaction holds a level-8 lock (access exclusive lock). All the other transactions on the table are blocked until VACUUM FULL is complete. To avoid affecting services, you are advised to perform VACUUM FULL during off-peak hours. <p>CAUTION During VACUUM FULL, the space usage will first increase and then decrease, because this operation requires the same space as the table to be vacuumed. (Actual table size = Total table size x (1 - dirty page rate). Ensure you have sufficient space before doing VACUUM FULL.</p>	User table (VACUUM FULL)
Vacuum First	You can configure the preferred Vacuum target. Each row corresponds to a table. Each table is represented by the database name, schema name, and table name, which are separated by spaces.	-

Configuration Item	Description	Example
Advanced Settings	<p>Automatic Vacuum is triggered based on either the default or customized conditions. These conditions include the bloat rate and the reclaimable space of the target table. When either of these conditions is met, the Automatic Vacuum process is initiated.</p> <ul style="list-style-type: none">• Default: The default bloat rate that triggers Vacuum is 80%, and the default reclaimable space of is 100 GB.• Custom: You can set the conditions for triggering automatic Vacuum based on the site requirements. <p>NOTE VACUUM bloat rate: After frequent UPDATE and DELETE operations are performed in a database, the deleted or updated rows are logically deleted from the database, but actually still exist in tables. Before VACUUM is complete, such data is still stored in disks, causing table bloat. If the bloat rate reaches the percentage threshold set in an O&M task, VACUUM will be automatically triggered.</p>	Default configuration (table bloat rate 80% or reclaimable space 100 GB.)

Step 6 Click **Next > Schedule** to configure scheduling for O&M tasks.

Select an O&M type.

- **One-off:** Set the start time and end time of the task.
- **Periodic:** Select a time window type, which includes **Daily**, **Weekly**, and **Monthly**, and select a time segment. Intelligent O&M will automatically analyze the time window and deliver O&M tasks accordingly.

CAUTION

- Do not choose peak hours when configuring the time window for autovacuum O&M tasks. Otherwise, automatic Vacuum may cause a deadlock on user services.
- The number of concurrent O&M tasks (vacuum/vacuum full) ranges from 0 to 24 for user tables, and from 0 to 1 for system catalogs. The concurrency value cannot be customized, but can be automatically adjusted based on system **io_util**.
 - Two intervals for 0% to 60%
 - 0% to 30%: The concurrency value increases by 2 each time the value of **io_util** decreases by 15%.
 - 30% to 60%: The concurrency value is incremented by 1 each time the value of **io_util** decreases by 15%.
 - 60% to 70%: The concurrency value remains unchanged.
 - Above 70%: The concurrency value decreases by 1 until it reaches 0.
- The scheduler scans the expansion of column-store compression units (CUs) within the time window. If the average number of CU records in a column-store table is less than 1000, the scheduler scans the table first. The scanning of column-store CUs is not limited by table bloat or table reclaimable space.
- A maximum of 100 tables can be added to the priority list.
- The scheduler autovacuum function depends on the statistics. If the statistics are inaccurate, the execution sequence and results may be affected.
- The scheduler does not support names containing spaces or single quotation marks, including database names, schema names, and table names. Otherwise, the tables will be skipped. Priority tables whose name contains spaces or single quotation marks will also be skipped automatically.

Step 7 Click **Next: Finish**. After you confirm the information, click **Finish** to submit the request.

----End

Modifying an O&M Plan

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 Click the name of the target cluster.

Step 3 In the navigation pane, choose **Intelligent O&M**.

Step 4 In the **O&M Plan** area, click **Modify** in the **Operation** column of the target task.

Task	Description	Time Window(UTC)	IS PAUSE	Task Type	Operation
<input type="checkbox"/> Vacuum		12:45:00 - 13:15:00 UTC every day	No	Periodic	Modify Details
<input type="checkbox"/> Vacuum		2021-09-22 12:47:41 to 2021-09-23 12:47:41	No	One-off	Modify Details
<input type="checkbox"/> Vacuum		11:00:00 - 12:00:00 UTC every day	Yes	Periodic	Modify Details
<input type="checkbox"/> Vacuum		07:00:00 - 14:00:00 UTC every day	Yes	Periodic	Modify Details

Step 5 The **Modify O&M Task** panel is displayed. The configurations are similar to adding an O&M task (see [Adding an O&M Plan](#)).

Step 6 Confirm the modification and click **OK**.

----End

Viewing O&M Task Details

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 Click the name of the target cluster.

Step 3 In the navigation pane, choose **Intelligent O&M**.

Step 4 In the **O&M Plan** area, click **Details** in the **Operation** column of the target task.

Task	Description	Time Window(UTC)	IS PAUSE	Task Type	Operation
<input type="checkbox"/> Vacuum		12:45:00 - 13:15:00 UTC every day	No	Periodic	Modify Details
<input type="checkbox"/> Vacuum		2021-09-22 12:47:41 to 2021-09-23 12:47:41	No	One-off	Modify Details
<input type="checkbox"/> Vacuum		11:00:00 - 13:00:00 UTC every day	Yes	Periodic	Modify Details
<input type="checkbox"/> Vacuum		07:00:00 - 14:00:00 UTC every day	Yes	Periodic	Modify Details

Step 5 The **O&M Task Details** panel is displayed for you to check the information.

----End

10.3 O&M Status

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 Click the name of the target cluster.

Step 3 In the navigation pane, choose **Intelligent O&M**.

Step 4 Switch to the **O&M Status** area.

Task	Status	Progress	Remaining Time Window	Time Window(UTC)
Vacuum	Waiting	100.00%	0h 30m 0s	2021-09-24 12:45:00 to 2021-09-24 13:15:00
Vacuum	Waiting	100.00%	2h 0m 0s	2021-09-24 11:00:00 to 2021-09-24 13:00:00
Vacuum	Waiting	100.00%	7h 0m 0s	2021-09-24 07:00:00 to 2021-09-24 14:00:00
Vacuum	Complete	100.00%	0h 0m 0s	2021-09-23 12:45:00 to 2021-09-23 13:15:00
Vacuum	Waiting	100.00%	0h 0m 0s	2021-09-23 11:00:00 to 2021-09-23 13:00:00

Step 5 Click the name of a specified O&M task to view the status details.

- **O&M Task: Vacuum**
- **Status: Waiting, Running, Completed, or Failed.**
- **Progress**
- **Remaining Time Window**
- **Time Window (UTC)**
- **Tables Being Vacuumed**
- **Tables to Be Vacuumed**

- **Vacuumed Tables**
- **Failed Tables**

 **NOTE**

- A maximum of 100 tables can be displayed for each category of the tables above.
- If the cluster is read-only, the INSERT statement cannot be executed for intelligent O&M tasks. There may be tasks remaining in the **Running** status. The **Running** status in this case is a historical status, and it indicates that the task is not completed within the specified time. If you manually pause the task and the task is not scheduled, the task may remain in the **Waiting** status. In this case, cancel the cluster read-only state and contact technical support to update the task status.

----End

11 Cluster Management

11.1 Modifying Database Parameters

After a cluster is created, you can modify the cluster's database parameters as required. On the GaussDB(DWS) management console, you can configure common database parameters. For details, see [Modifying Parameters](#). You can also view the parameter modification history. For details, see [Viewing Parameter Change History](#). You can run SQL commands to view or set other database parameters. For details, see [Setting Configuration Parameters](#) in the *Data Warehouse Service Database Development Guide*.

Prerequisites

You can modify parameters only when no task is running in the cluster.

Modifying Parameters

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane on the left, choose **Clusters > Dedicated Clusters**.
- Step 3** In the cluster list, find the target cluster and click the cluster name. The **Cluster Information** page is displayed.
- Step 4** Click the **Parameters** tab and modify the parameter values. Then click **Save**.

Parameter Name	CN Value	DN Value	Unit	Value Range	Restart Cluster	Description
fencedUDFMemoryLimit	4411000	44111	KB	0 - 2,147,483,647	No	Controls the virtual memory used by each fenced udf worker process. Default: 0.
UDFWorkerMemHardLimit	1048570000	104857	KB	0 - 2,147,483,647	Yes	Specifies the maximum value of fencedUDFMemoryLimit. Unit: KB. Default: 104...
sql_redistribute_enhancement	off	off	-	-	No	When the aggregate operation is performed, which contains multiple group by col...
alarm_report_interval	123110	100	Second	0 - 2,147,483,647	No	Specifies the interval at which an alarm is reported. Default: 10.
allocate_mem_cost	1.00011e+06	0	-	0 - 1.79769e+308	No	allow_concurrent_hupk_update. Default: 0.
allow_concurrent_hupk_update	on	on	-	-	No	Specifies whether to allow concurrent update. Default: on.
analysis_options	ALL,on,LLVM_COMPILE,off,HA	ALL,on() off,LLVM_COMPILE,H/	-	-	No	Specifies whether to enable function options in the corresponding options to use ...
archive_command		(disabled)	-	-	No	Specifies the command used to archive WALs set by the administrator. You are a...
archive_mode	on	off	-	-	No	Specifies whether to archive WALs. Default: off.
archive_timeout	0	12345611	Second	0 - 1,073,741,823	No	Specifies the archiving period. Default: 0.

Step 5 In the **Modification Preview** dialog box, confirm the modifications and click **Save**.

Step 6 You can determine whether you need to restart the cluster after parameter modification based on the **Restart Cluster** column.

Name	Value	Value Range	Restart Cluster	Description
password_encryption_type	1	0-2	No	Specifies the encryption type of user passwords. 0 indicates that passwords are encrypted in MD5 mode. 1 indic...
timezone	UTC	--	No	Time zone that will be displayed in the timestamps Default: UTC.
log_timezone	UTC	--	No	Time zone for timestamps in the server log Default: UTC.

NOTE

- If cluster restart is not required for a parameter, the parameter modification takes effect immediately.
- If cluster restart is required for parameter modifications to take effect, the new parameter values will be displayed on the page after the modification, but will not take effect until the cluster is restarted. Before a restart, the cluster status is **To be restarted**, and some O&M operations are disabled.

----End

Viewing Parameter Change History

Perform the following steps to view the parameter modification history and check whether the modifications have taken effect:

Procedure

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 In the navigation pane on the left, choose **Clusters > Dedicated Clusters**.

Step 3 In the cluster list, find the target cluster and click the cluster name. The **Cluster Information** page is displayed.

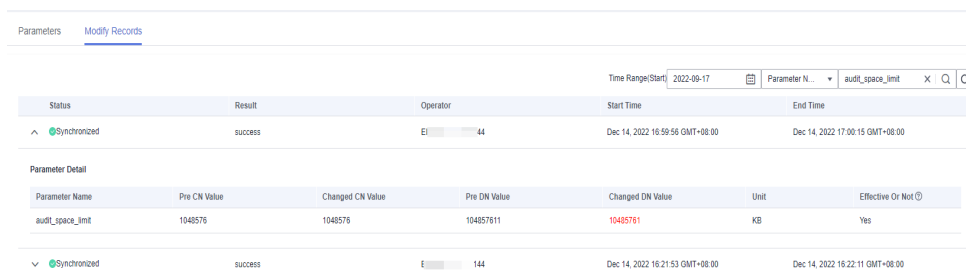
Step 4 Click the **Modify Records** tab.

Status	Result	Operator	Start Time	End Time		
To be restarted	success	E	Dec 16, 2022 10:54:55 GMT+08:00	Dec 16, 2022 10:55:16 GMT+08:00		
Parameter Detail						
Parameter Name	Pre CN Value	Changed CN Value	Pre DN Value	Changed DN Value	Unit	Effective Or Not
alarm_report_interval	12311	123110	100	100	Second	Yes
To be restarted	success	E	Dec 16, 2022 09:56:40 GMT+08:00	Dec 16, 2022 09:56:58 GMT+08:00		
Synchronized	success	E	Dec 15, 2022 11:13:23 GMT+08:00	Dec 15, 2022 11:13:43 GMT+08:00		
Applying failed	async job failed. ("result": "failed", "detail": "errorCod...	E	Dec 15, 2022 10:32:21 GMT+08:00	Dec 15, 2022 10:32:29 GMT+08:00		

NOTE

- If a parameter can take effect immediately after modification, its status will change to **Synchronized** after you modify it.
- If a parameter can take effect only after a cluster restart, its status will change to **To be restarted** after you modify it. You can click the expansion icon on the left to view the parameters that have not taken effect. After the cluster is restarted, the status of the record will change to **Synchronized**.

Step 5 By default, only the change history within a specified period is displayed. To check the entire change history of a parameter, search for it in the search box in the upper right corner.



Status	Result	Operator	Start Time	End Time
^ Synchronized	success	ELI 44	Dec 14, 2022 16:59:56 GMT+08:00	Dec 14, 2022 17:00:15 GMT+08:00

Parameter Name	Pre CN Value	Changed CN Value	Pre DN Value	Changed DN Value	Unit	Effective Or Not
audit_space_limit	1048576	1048576	104857611	10485761	KB	Yes

Status	Result	Operator	Start Time	End Time
v Synchronized	success	E 144	Dec 14, 2022 16:21:53 GMT+08:00	Dec 14, 2022 16:22:11 GMT+08:00

----End

Parameter Description

The following table describes part of the database parameters. You can search for and check more parameters by following the instructions in [Modifying Parameters](#).

NOTE


The default values of the following parameters are for reference only. For more information, see "Setting GUC Parameters".


11.2 Checking the Cluster Status

On the **Clusters > Dedicated Clusters** page of the GaussDB(DWS) management console, you can view the general information about a cluster in the cluster list, such as the cluster status, task information, recent events, and node flavor.

Querying General Information of a Cluster

Log in to the GaussDB(DWS) management console. In the navigation tree on the left, click **Clusters > Dedicated Clusters**. The cluster list displays all clusters. If there are a large number of clusters, you can turn pages to view the clusters in any status.

Enter the cluster name in the search box, and click  to search for a cluster. Alternatively, in the **All projects** drop-down list above the cluster list, select the

target project. Click  to refresh the cluster list. You can also click **Search by Tag** to search for clusters based on the criteria. For details, see [Searching for Clusters Based on Tags](#).

Clusters are listed in chronological order by default, with the most recent clusters displayed at the top. [Table 11-1](#) describes the cluster list parameters.

Table 11-1 Cluster list parameters

Parameter	Description
Cluster Name	Cluster name specified when a cluster is created. NOTE After a cluster is created, its name cannot be changed.
Cluster Status	Cluster running status. For details, see Cluster Status .
Task Information	Cluster task status. For details, see Cluster Task Information .
Node Flavor	Node flavors of clusters.
Recent Events	Number of recent events in a cluster. You can click the number to view event details.
Operation	<ul style="list-style-type: none">• More<ul style="list-style-type: none">– View Metric: For details, see Monitoring Clusters Using Cloud Eye.– Restart: Click Restart to restart a cluster. For details, see Cluster Restart.– Scale Out: For details, see Scaling Out a Cluster.– Change all specifications: For details, see Changing All Specifications.– Scale In: For details, see Scaling In a Cluster.– Redistribute: For details, see Redistributing Data.– View Scaling Details: For details, see Viewing Redistribution Details.– Expand Disk Capacity: For details, see Disk Capacity Expansion of an EVS Cluster.– Reset Password: For details, see Resetting a Password.– Create Snapshot: For details, see Manual Snapshots.– Cancel Readonly: For details, see Removing the Read-only Status.– Delete: Click Delete to delete a cluster. For details, see Deleting a Cluster.– Change node flavor: For details, see Changing the Node Flavor.– Manage CN: For details, see Managing CNs.

Cluster Status

Table 11-2 Cluster status description

Status	Description
Available	Indicates that the cluster runs properly.
Read-only	<p>A cluster goes into this state when the disk usage of the cluster or a single node in the cluster is greater than 90%. The cluster can still work in this state but supports only query operations. Write operations are not supported. When the cluster status becomes read-only, contact technical support engineers.</p> <p>After the read-only status is canceled for the cluster, you are advised to perform the following operations:</p> <ul style="list-style-type: none">• Use the SQL client tool to connect to the database as the administrator and run the following command to periodically clear and reclaim the storage space: <code>VACUUM FULL;</code> After you delete data stored in GaussDB(DWS) data warehouses, dirty data may be generated possibly because the disk space is not released. This results in disk space waste. It is recommended that the storage space be cleared periodically.• You are advised to check the disk capacity and analyze whether the existing cluster specifications meet service requirements. If not, expand the cluster capacity. For details, see Scaling Out a Cluster.
Unbalanced	If the role of a GTM or DN in the cluster is different from the initial role, the cluster is in the Unbalanced state. In the Unbalanced state, the number of primary instances on some nodes increases. As a result, the load pressure is high. In this case, the cluster is normal, but the overall performance is not as good as that in a balanced state. You are advised to switch a cluster to the Available state during off-peak hours.
Redistributing	A cluster goes into this state when it detects that the service data on the original nodes is significantly larger than that on the new node after a new node is added to the cluster. In this case, the system automatically redistributes data on all nodes. The cluster can still work in this state.
Redistribution failed	A cluster goes into this state when data redistribution fails, but no data loss occurs. The cluster can still work in this state. You are advised to contact technical support.
Degraded	A cluster goes into this state when some nodes in the cluster are faulty, but the whole cluster runs properly. You are advised to contact technical support.
Unavailable	A cluster goes into this state when it cannot provide database services. You are advised to contact technical support.
Creating	A cluster goes into this state when it is being created.

Status	Description
Creation failed	A cluster goes into this state when it fails to be created.
Creating, restoring	Indicates that a cluster is being restored from a snapshot. A snapshot will be restored to a new cluster. During the process, the new cluster goes into this status.

Cluster Task Information

Table 11-3 Task information description

Status	Description
Creating snapshot	Indicates that a snapshot is being created in the cluster.
Snapshot creation failed	Indicates that a snapshot fails to be created.
Observing	Indicates that the cluster is to be submitted after the automatic upgrade.
Configuring	Indicates that the system is storing modifications of cluster parameters.
Restarting	Indicates that a cluster is being restarted.
Restart failed	Indicates that a cluster fails to be restarted.
Scaling out	Indicates that a cluster is being scaled out.
Scale-out failed	Indicates that a cluster fails to be scaled out.
Changing all specifications	All the specifications of the cluster being changed.
All specifications change failed	Specifications change failed because of insufficient quotas or permissions, or abnormal cluster status.
Maintaining	A maintenance change operation, such as cluster upgrade or plugin upgrade, is being performed on the cluster.
Maintain_failure	A cluster fails to be restarted.

11.3 Viewing Cluster Details

Log in to the GaussDB(DWS) management console. In the navigation tree on the left, click **Clusters > Dedicated Clusters**. In the cluster list, locate the required cluster and click its name. The **Cluster Information** page is displayed.

On the **Basic Information** page, you can view the following information:

- **Basic Information:** [Table 11-4](#) lists the related parameters.
- **Connection:** [Table 11-5](#) describes the parameters.
- **Network:** [Table 11-6](#) lists the related parameters.
- **Storage/Backup Capacity:** [Table 11-7](#) describes the parameters.

Table 11-4 Basic information

Parameter	Description
Cluster Name	Cluster name specified when a cluster is created.
Cluster Status	Cluster running status. For details, see Cluster Status .
Parameter Configuration Status	Parameter configuration status of a cluster.
Task Information	Cluster task status. For details, see Cluster Task Information .
Current Specifications	Current node specifications.
Nodes	Number of nodes in the cluster.
Cluster ID	ID of the cluster.
Cluster Version	Cluster version information.
Created	Time when the cluster was created.
Node Flavor	Node flavor of the cluster.
Maintenance Window	Maintenance window of the cluster.

Table 11-5 Connection

Parameter	Description
Private Network Domain Name	<p>Domain name for accessing the cluster database through the internal network. The domain name corresponds to all CN IP addresses. The private network domain address is automatically generated when a cluster is created.</p> <p>NOTE</p> <ul style="list-style-type: none">• If the cluster name does not comply with the domain name standards, the prefix of the default access domain name will be adjusted accordingly.• Load balancing is not supported. <p>You can click Modify to change the private network domain name. The access domain name contains 4 to 63 characters, which consists of letters, digits, and hyphens (-), and must start with a letter.</p>
Private Network IP Address	<p>IP address for accessing the database in the cluster over the private network.</p> <p>NOTE</p> <ul style="list-style-type: none">• A private IP address is automatically generated when you create a cluster. The IP address is fixed.• The number of private IP addresses equals the number of CNs. You can log in to any node to connect to the cluster.• If you access a fixed IP address over the internal network, all the resource pools will run on a single CN.
Public Network Domain Name	<p>Name of the domain for accessing the database in the cluster over the public network.</p> <p>NOTE</p> <p>Load balancing is not supported.</p>
Public Network IP Address	<p>IP address for accessing the database in the cluster over the public network.</p> <p>NOTE</p> <ul style="list-style-type: none">• If no EIP is assigned during cluster creation and Public Network IP Address is empty, click Edit to bind an EIP to the cluster.• If an EIP is bound during cluster creation, click Edit to unbind the EIP.
Initial Administrator	<p>Database administrator specified during cluster creation. When you connect to the cluster for the first time, you need to use the initial database administrator and password to connect to the default database.</p>
Port	<p>Port number for accessing the cluster database through the public network or private network. The port number is specified when the cluster is created.</p>
Default Database	<p>Database name specified when the cluster is created. When you connect to the cluster for the first time, connect to the default database.</p>

Parameter	Description
ELB Address	To achieve high availability and avoid single-CN failures, a new cluster needs to be bound to ELB. You are advised to use the ELB address to connect to the cluster.

Table 11-6 Network

Parameter	Description
Region	Current working zone of the cluster.
AZ	AZ selected during cluster creation.
VPC	VPC selected during cluster creation. A VPC is a secure, isolated, and logical network environment. After a data warehouse cluster is created, its VPC cannot be changed. However, you can edit and modify the current VPC. You can click the VPC name to go to the VPC details page to configure it. For details about VPC operations, see VPC and Subnet in the <i>Virtual Private Cloud User Guide</i> .
Subnet	Subnet selected during cluster creation. A subnet provides dedicated network resources that are isolated from other networks, improving network security. After a data warehouse cluster is created, its subnet cannot be changed. However, you can edit and modify the current subnet. You can click the subnet name to go to the subnet details page to configure it. For details about subnet operations, see VPC and Subnet > Modifying a Subnet in the <i>Virtual Private Cloud User Guide</i> .
Security Group	Security group selected during cluster creation. After a data warehouse cluster is created, its security group cannot be changed. However, you can edit and modify the current security group, and add, delete, or modify rules in it. You can click the security group name to go to the security group details page to configure it. For details about security group operations, see Security > Security Group in the <i>Virtual Private Cloud User Guide</i> .

Table 11-7 Storage/Backup capacity

Parameter	Description
Storage	The storage class Ultra-high I/O and the storage space usage are displayed. NOTE <ul style="list-style-type: none">The used storage capacity does not include data on OBS foreign tables. It includes only GaussDB(DWS) data, including files, logs, snapshots, and indexes.The available storage space is half of the actual disk capacity.
Backup	The space in use, free space, and charged space of the cluster are displayed.
Cold Data Used Capacity	OBS capacity used by cold data. NOTE OBS capacity usage. It is synchronized every hour.

11.4 Managing Access Domain Names

Overview

A domain name is a string of characters separated by dots to identify the location of a computer or a computer group on the Internet, for example, www.example.com. You can enter a domain name in the address box of the web browser to access a website or web application.

On GaussDB(DWS), you can access clusters using the private network domain name or the public network domain name.

Private network domain name: Name of the domain for accessing the database in the cluster through the private network. The private network domain name is automatically generated when you create a cluster.

Public network domain name: Name of the domain for accessing the database in the cluster through the public network. If a cluster is not bound to an EIP, it cannot be accessed using the public network domain name. If you bind an EIP during cluster creation, the public network domain name is automatically generated.

NOTE

Neither public nor private domain names support load balancing. To use load balancing, see [Configuring JDBC to Connect to a Cluster \(Load Balancing Mode\)](#).

After a cluster is created, you can set private and public domain names for accessing the cluster as required. The operations are as follows:

- [Modifying a Private Network Domain Name](#)
- [Creating a Public Network Domain Name](#)
- [Modifying a Public Network Domain Name](#)

- [Releasing a Public Network Domain Name](#)


Modifying a Private Network Domain Name

The private network domain name is automatically generated during cluster creation. After the cluster is created, you can modify the default domain name based on site requirements.

To modify the private network domain name, perform the following steps:

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane on the left, choose **Clusters > Dedicated Clusters**.
- Step 3** In the cluster list, find the target cluster and click the cluster name. The **Cluster Information** page is displayed.
- Step 4** In the **Connection** area, click **Modify** next to the automatically generated **Private Network Domain Name**.
- Step 5** In the **Modify Private Network Domain Name** dialog box, enter the target domain name and click **OK**.

The private network domain name contains 4 to 63 characters, which consists of letters, digits, and hyphens (-) and must start with a letter.

After the domain name is modified, click copy button  next to the private network domain name to copy it.

----End

Creating a Public Network Domain Name

A cluster is not bound to an EIP by default during cluster creation. That is, cluster access using the public network is disabled. After a cluster is created, if you want to access it over the public network, bind an EIP to the cluster and create a public network domain name.

NOTE

By default, only accounts or users with Security Administrator permissions can query and create agencies. By default, the IAM users in those accounts cannot query or create agencies. When the users use the EIP, the system makes the binding function unavailable. Contact a user with the **DWS Administrator** permissions to authorize the agency on the current page.

To create a public network domain name, perform the following steps:

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane on the left, choose **Clusters > Dedicated Clusters**.
- Step 3** In the cluster list, find the target cluster and click the cluster name. The **Cluster Information** page is displayed.
- Step 4** In the **Connection** area, **Public Network Domain Name** and **Public Network IP Address** are empty. Click **Edit** to bind the cluster with an EIP.

Step 5 In the **Edit Elastic IP** dialog box, select an EIP from the drop-down list to bind it to a specified CN.

If no available EIPs are displayed, click **View EIP** to go to the **Elastic IP** page and create an EIP that satisfies your needs. After the new EIP is created, click the refresh icon next to the drop-down list. The newly created EIP will be displayed in the **EIP** drop-down list.


After the EIP is bound successfully, the specific public network IP address is displayed in the **Connection** area.



Step 6 In the **Connection** area, click **Create** next to **Public Network Domain Name** to create a public network domain name for the cluster.

Step 7 In the **Apply for Public Network Domain Name** dialog box, enter the target domain name and click **OK**.

The public network domain name contains 4 to 63 characters, which consists of letters, digits, and hyphens (-) and must start with a letter.

The specific public network domain name is displayed in the **Connection** area after being created. Click copy button  to copy the public network domain name.

----End

Modifying a Public Network Domain Name

If you bind an EIP during cluster creation, the public network domain name is automatically generated. After a cluster is created, you can modify the public network domain name as required.

To modify the public network domain name, perform the following steps:

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 In the navigation pane on the left, choose **Clusters > Dedicated Clusters**.

Step 3 In the cluster list, find the target cluster and click the cluster name. The **Cluster Information** page is displayed.

- Step 4** Click **Modify** next to the **Public Network Domain Name** in the **Connection** area.
- Step 5** In the **Modify Public Network Domain Name** dialog box, enter the target domain name and click **OK**.
- End

Releasing a Public Network Domain Name

After a cluster is created, you can release unnecessary public network domain names.

To do so, perform the following steps:

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane on the left, choose **Clusters > Dedicated Clusters**.
- Step 3** In the cluster list, find the target cluster and click the cluster name. The **Cluster Information** page is displayed.
- Step 4** Click **Release** next to the **Public Network Domain Name** in the **Connection** area.
- Step 5** In the **Release Domain Name** dialog box, click **Yes**.
- End

11.5 Cluster Topology

Overview

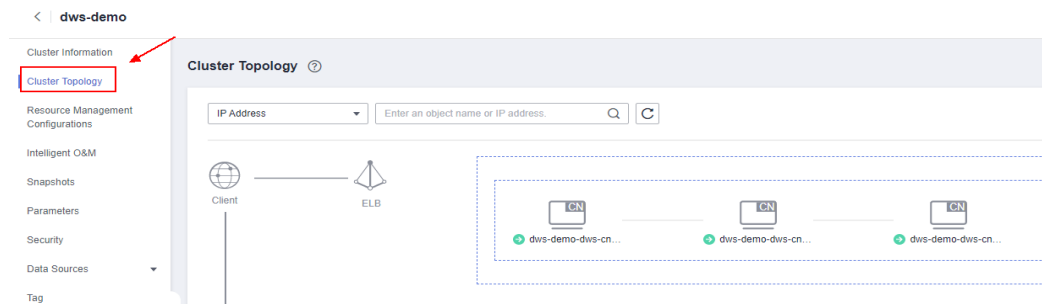
A topology shows all the nodes in a cluster. You can check the node statuses, processes, and IP addresses.

NOTE

- You can check the topology structure and node processes.
- Only cluster versions 8.0.0 and later can display the topology structure. Only cluster versions 8.2.0 and later can display node processes.

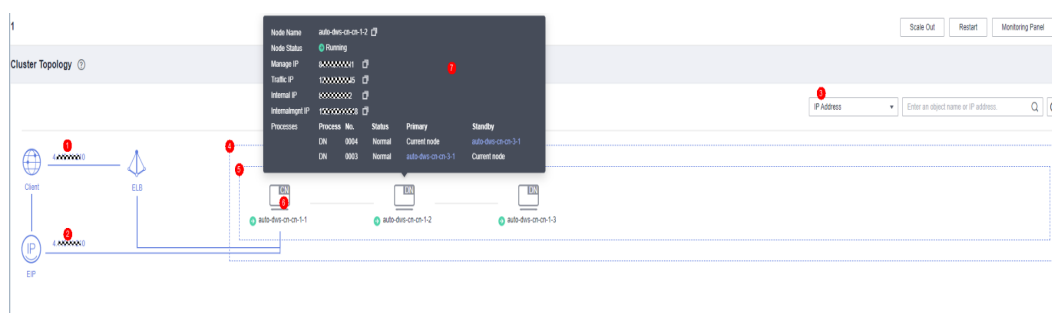
Viewing the Cluster Topology

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the cluster list, click the name of a cluster.
- Step 3** On the **Cluster Details** page, click the **Cluster Topology** tab.
- Step 4** In the upper part of the page, you can select **IP Address** or **Node Name**. After entering the IP address or node name in the search box, you can view the location of the IP address or node name in the cluster topology.



----End

Topology Overview



This figure shows a topology. The elements marked in the figure are as follows:

1. Public IP address of the ELB bound to the cluster. If no public IP addresses are bound to the ELB, the service address is displayed.
2. EIP bound to the cluster.
3. Search category. You can perform exact search by IP address or node name.
4. Rings in the cluster.
5. A ring. Each ring occupies a line. An icon in a ring indicates a node.
6. A node. The type of the node is displayed in the upper right corner of the icon. Currently, the type can only be CN or DN. If there is a CN process on the node, **CN** is displayed. If there are no CN processes on the node, **DN** is displayed.
7. Node details, including the node name, status, IP addresses, and task process. Node details are displayed when you hover your cursor over a node icon.

Terms in the Topology View

Table 11-8 Cluster structure description

Name	Description	Usage
ELB	Elastic Load Balance (ELB) automatically distributes incoming traffic across multiple backend servers based on listening rules you configure.	If the private IP address or EIP of a CN is used to connect to a GaussDB(DWS) cluster, the failure of this CN will lead to a cluster connection failure. If a private or public domain name is used for connection, the DNS service randomly selects a private IP address or EIP for each client. This cannot balance loads or avoid single-CN failures. ELB is used to solve these problems. For details, see Associating and Disassociating ELB .
EIP	The Elastic IP (EIP) service provides static public IP addresses and scalable bandwidths that enable your cloud resources to communicate with the Internet.	EIPs can be bound to or unbound from ECSs, BMSs, virtual IP addresses, load balancers, and NAT gateways.
Ring	A security ring is used for isolating faulty servers. A fault in a ring does not affect servers outside the ring.	Data on a DN has multiple copies in a ring, and will not be lost even if the DN server is faulty. For example, if Server1 in a ring is faulty, the standby DN1 on Server2, the standby DN2 on Server3, and the standby DN3 on Server3 are still running. The loads of servers in a ring are still balanced. A cluster can run properly as long as the number of faulty servers does not exceed the number of rings. NOTE The ring is the minimum unit for a scale-out. When you scale out a cluster, the added nodes must be a multiple of the ring quantity.

Table 11-9 Node IP addresses

Name	Description	Usage
Manage IP	IP address used by a data warehouse node to communicate with the management plane	It is used by the management plane to deliver commands, and used by the node to report node status and monitoring information.
Traffic IP	IP address of a data warehouse node for external access.	This IP address can be bound to an EIP or ELB, or directly connect to a VPC.
Internal IP	IP address used for communication inside a data warehouse cluster.	-
Internalmngt IP	IP address used by nodes to send internal management commands in a data warehouse cluster.	-

Table 11-10 Node processes

Name	Description	Usage
CMS	<p>A Cluster Manager (CM) manages and monitors the running status of functional units and physical resources in the distributed system, ensuring system stability. CM Server (CMS) is a module of CM.</p>	<p>A CM consists of CM Agent, OM Monitor, and CM Server.</p> <ul style="list-style-type: none"> • CM Agent monitors the running status of primary and standby GTMs, CNs, and primary and standby DNAs on the host, and reports the status to CM Server. In addition, it executes the arbitration instruction delivered by CM Server. A CM Agent process runs on each server. • OM Monitor monitors scheduled tasks of CM Agent and restarts CM Agent when CM Agent stops. If CM Agent cannot be restarted, the server will be unavailable. In this case, you need to manually rectify this fault. <p>NOTE A CM Agent restart fails probably because of lack of system resources, which rarely happens.</p> <ul style="list-style-type: none"> • CM Server checks whether the current system is normal according to the instance status reported by CM Agent. In the case of exceptions, CM Server delivers recovery commands to CM Agent. <p>GaussDB(DWS) deploys CM Server in primary/standby mode to ensure system HA. CM Agent</p>

Name	Description	Usage
		connects to the primary CM Server. If the primary CM Server is faulty, the standby CM Server is promoted to primary to prevent single-CM faults.
GTM	A Global Transaction Manager (GTM) generates and maintains the globally unique information, such as the transaction ID, transaction snapshot, and timestamp.	A cluster includes only one pair of GTMs: one primary and one standby GTM.
CN	A Coordinator (CN) receives access requests from applications, and returns execution results to the client; splits tasks and allocates task fragments to different DNs for parallel processing.	<p>CNs in a cluster have equivalent roles and return the same result for the same DML statement. Load balancers can be added between CNs and applications to ensure that CNs are transparent to applications. If a CN is faulty, the load balancer connects its applications to another CN.</p> <p>CNs need to connect to each other in the distributed transaction architecture. To reduce heavy load caused by excessive threads on GTMs, no more than 10 CNs should be configured in a cluster.</p>

Name	Description	Usage
CCN	Central Coordinator (CCN)	GaussDB(DWS) handles the global resource load in a cluster using the Central Coordinator (CCN) for adaptive dynamic load management. When the cluster is started for the first time, the CM selects the CN with the smallest ID as the CCN. If the CCN is faulty, CM replaces it with a new one.
DN	A Data Node (DN) stores data in row-store, column-store, or hybrid mode, executes data query tasks, and returns execution results to CNs.	There are multiple DNs in the cluster. Each DN stores part of data. If DNs are not deployed in primary/standby mode and a DN is faulty, data on the DN will be inaccessible.

11.6 Managing Tags

11.6.1 Overview

A tag is a key-value pair customized by users and used to identify cloud resources. It helps users to classify and search for cloud resources.

Tags are composed of key-value pairs.

- A key in a tag can have multiple values.
- A cloud resource must have a unique key.

On GaussDB(DWS), after creating a cluster, you can add identifiers to items such as the project name, service type, and background information using tags. If you use tags in other cloud services, you are advised to create the same tag key-value pairs for cloud resources used by the same business to keep consistency.

GaussDB(DWS) supports the following two types of tags:

- Resource tags
Non-global tags created on GaussDB(DWS)
- Predefined tags

Predefined tags created on Tag Management Service (TMS). Predefined tags are global tags.

For details about predefined tags, see the *Tag Management Service User Guide*.

On GaussDB(DWS), tags can be added to the following resources:

- Cluster

Tags can be added to a cluster when the cluster is being created or after it is successfully created. You can search for the cluster in the cluster list using tags.

Each cluster can have a maximum of 20 tags.

After you add tags to a cluster and then create a snapshot for the cluster, the tags cannot be restored if you use the snapshot to restore the cluster. Instead, you need to add tags again.

When a cluster is deleted, non-predefined tags associated with the cluster are also deleted. Predefined tags need to be deleted on TMS.

11.6.2 Tag Management

This section describes how to search for clusters based on tags and how to add, modify, and delete tags.

Adding a Tag to a Cluster

Step 1 On the **Clusters > Dedicated Clusters** page, click the name of the cluster to which a tag is to be added, and choose **Tag**.

Step 2 Click **Add Tag**.

Step 3 Configure tag information in the **Add Tag** dialog box. The value of a key cannot be left blank.

Table 11-11 Tag parameters

Parameter	Description	Example Value
Tag key	<p>You can:</p> <ul style="list-style-type: none">Select a predefined tag key or an existing resource tag key from the drop-down list of the text box. <p>NOTE To add a predefined tag, you need to create one on TMS and select it from the drop-down list of Tag key. You can click View predefined tags to enter the Predefined Tags page of TMS. Then, click Create Tag to create a predefined tag.</p> <ul style="list-style-type: none">Enter a tag key in the text box. The tag key can contain a maximum of 128 characters and cannot be an empty string. It cannot start with _sys_. Only letters, digits, spaces, and the following characters are allowed: _ . : = + - @ <p>NOTE A key must be unique in a given cluster.</p>	key01
Tag value	<p>You can:</p> <ul style="list-style-type: none">Select a predefined tag value or resource tag value from the drop-down list of the text box.Enter a tag value in the text box. The tag key can contain a maximum of 255 characters and cannot be an empty string. Only letters, digits, spaces, and the following characters are allowed: : = + - @.	value01

Step 4 Click **OK**.

----End

Searching for Clusters Based on Tags

You can quickly locate a tagged cluster using tags.


Step 1 Log in to the GaussDB(DWS) management console.



Step 2 Choose **Clusters > Dedicated Cluster**.

Step 3 Click **Search by Tag** on the upper right of the cluster list to expand the tab page.

Step 4 In the **Search by Tag** area, click the **Tag Key** text box to select a tag key from the drop-down list and then click the **Tag Value** text box to select the corresponding tag value.

You can only enter a tag key or value that exists in the drop-down list. If no tag key or value is available, create a tag for the cluster. For details, see [Adding a Tag to a Cluster](#).

Step 5 Click  to add the selected tag to the area under the text boxes.

- Select another tag in the text boxes and click  to generate a tag combination for cluster search. You can add a maximum of 10 tags to search for data warehouse clusters. If you specify more than one tag, clusters containing all the specified tags will be displayed.
- To delete an existing tag, click  next to the tag.
- You can click **Reset** to clear all added tags.

Step 6 Click **Search**. The target cluster will be displayed in the cluster list.

----End

Modifying a Tag

Step 1 On the **Clusters > Dedicated Clusters** page, click the name of the cluster to which a tag is to be added, and choose **Tag**.

Step 2 Locate the row that contains the tag to be modified, and click **Edit** in the **Operation** column. The **Edit Tag** dialog box is displayed.

Step 3 Enter the new key value in the **Value** text box.

Step 4 Click **OK**.

----End

Deleting a Tag

Step 1 On the **Clusters > Dedicated Clusters** page, click the name of the cluster from which a tag is to be deleted, and click the **Tags** tab.

Step 2 Locate the row that contains the tag to be deleted, click **Delete** in the **Operation** column. The **Delete Tag** dialog box is displayed.

Step 3 Click **Yes** to delete the tag.

----End

11.7 Managing Enterprise Projects

An enterprise project is a cloud resource management mode. Enterprise Management provides users with comprehensive management in cloud-based . The Enterprise Management console differs from typical management consoles as it focuses on resource management rather than independent control and configuration of cloud products. It assists enterprises in managing within the hierarchy of companies, departments, and projects.

Binding an Enterprise Project

You can select an enterprise project during cluster creation to associate it with the cluster. For details, see [Creating a Cluster](#). The **Enterprise Project** drop-down list displays the projects you created. In addition, the system has a built-in enterprise project (**default**). If you do not select an enterprise project for the cluster, the default project is used.

During cluster creation, if the cluster is successfully bound to an enterprise project, the cluster will be successfully created. If the binding fails, the system sends an alarm and the cluster fails to be created.

Snapshots of a cluster retain the association between the cluster and its enterprise project. When the cluster is restored, the association is also restored.

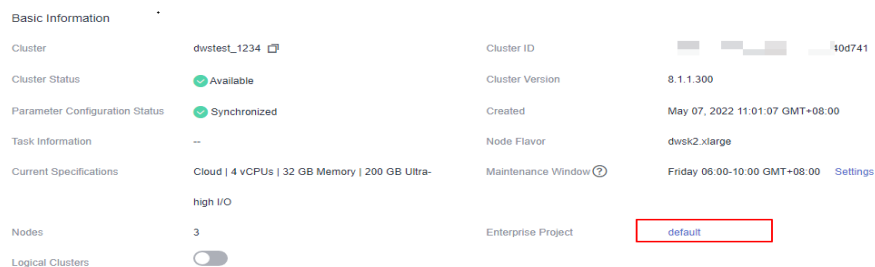
When you delete a cluster, the association between the cluster and its enterprise project is automatically deleted.

Viewing Enterprise Projects

After a cluster is created, you can view the associated enterprise project in the cluster list and **Cluster Information** page. You can query only the cluster resources of the project on which you have the access permission.

- In the cluster list on the **Clusters** page, view the enterprise project to which the cluster belongs.
- In the cluster list, find the target cluster and click the cluster name. The **Cluster Information** page is displayed, on which you can view the enterprise project associated with the cluster. Click the enterprise project name to view and edit it on the Enterprise Management console.

Figure 11-1 Viewing the enterprise project



Basic Information	
Cluster	divstest_1234
Cluster Status	Available
Parameter Configuration Status	Synchronized
Task Information	--
Current Specifications	Cloud 4 vCPUs 32 GB Memory 200 GB Ultra-high I/O
Nodes	3
Logical Clusters	<input type="checkbox"/>
Cluster ID	10d741
Cluster Version	8.1.1.300
Created	May 07, 2022 11:01:07 GMT+08:00
Node Flavor	dwsk2.xlarge
Maintenance Window	Friday 06:00-10:00 GMT+08:00 Settings
Enterprise Project	default

- When querying the resource list of a specified project on the Enterprise Management console, you can also query the GaussDB(DWS) resources.

Searching for Clusters by Enterprise Project

Log in to the GaussDB(DWS) management console, choose **Clusters > Dedicated Clusters**, click **All projects** above the cluster list, and select the required project name from the drop-down list to view all clusters associated with the project.

Migrating a Cluster to or Out of an Enterprise Project

A GaussDB(DWS) cluster can be associated with only one enterprise project. After a cluster is created, you can migrate it from its current enterprise project to another one on the Enterprise Management console, or migrate the cluster from another enterprise project to a specified enterprise project. After the migration, the cluster is associated with the new enterprise project. The association between the cluster and the original enterprise project is automatically released.

11.8 Managing Clusters That Fail to Be Created

If a cluster fails to be created, you can go to the **Clusters > Dedicated Clusters** page of the GaussDB(DWS) management console to view the cluster status and the cause of failure.

Checking the Cause of a Creation Failure

Step 1 Log in to the GaussDB(DWS) management console. In the navigation pane on the left, choose **Clusters > Dedicated Cluster**. The **Dedicated Clusters** page is displayed.

Step 2 In the cluster list, locate the cluster whose **Cluster Status** is **Creation failed**.

Step 3 Click  in the **Cluster Status** column to view the cause of the creation failure.

For details about the error code and how to handle it, see *Error Code Reference*. If the fault persists, contact technical support.

----End

Deleting a Cluster That Fails to Be Created

You can delete a cluster that fails to be created if you do not need it. Before deletion, check the cause of creation failure.

Step 1 Log in to the GaussDB(DWS) management console. In the navigation pane on the left, choose **Clusters > Dedicated Cluster**. The **Dedicated Clusters** page is displayed.

Step 2 In the cluster list, locate the row containing the failed cluster to be deleted, and choose **More > Delete**.

Step 3 (Optional) If the cluster is bound with an EIP during creation, click **Release the EIP bound with the cluster** to release the EIP.

Step 4 In the dialog box that is displayed, click **Yes** to delete the cluster.

If the cluster to be deleted uses an automatically created security group that is not used by other clusters, the security group is automatically deleted when the cluster is deleted.

----End

11.9 Removing the Read-only Status

A cluster in read-only status does not allow write operations. You can remove this status on the management console. A cluster becomes read-only probably because of high disk usage. For details about how to solve this problem, see "High Disk Usage and Read Only Status" in *Data Warehouse Service (DWS) Troubleshooting Guide*.

NOTE

- The read-only status can be canceled for version 1.7.2 or later.
- In 8.2.0 and later versions, you can free up disk space by using **DROP/TRUNCATE TABLE** in a read-only cluster.

Impact on the System

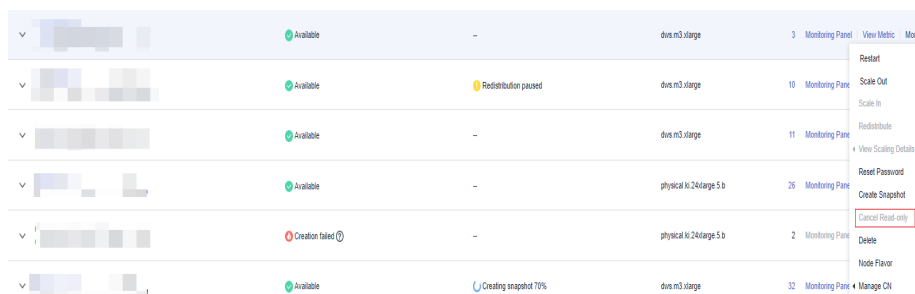
- You can cancel the read-only status only when a cluster is read-only.
- When a cluster is in read-only status, stop the write tasks to prevent data loss caused by used up disk space.
- After the read-only status is canceled, clear the data as soon as possible to prevent the cluster from entering the read-only status again after a period of time.

Removing Read-only Status

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 Choose **Clusters > Dedicated Cluster**. All clusters are displayed by default.

Step 3 In the **Operation** column of the target cluster, choose **More > Cancel Read-only**.



Cluster Name	Status	Message	Version	Nodes	Operation	More	
[blurred]	Available	-	dws.m3.storge	3	Monitoring Panel	View Metric	More
[blurred]	Available	Redistribution paused	dws.m3.storge	10	Monitoring Panel	Restart	Scale Out
[blurred]	Available	-	dws.m3.storge	11	Monitoring Panel	Scale In	Redistribute
[blurred]	Available	-	physical.k1.24large.5.b	26	Monitoring Panel	Reset Password	Create Snapshot
[blurred]	Creation failed	-	physical.k1.24large.5.b	2	Monitoring Panel	Cancel Read-only	Delete
[blurred]	Available	Creating snapshot 70%	dws.m3.storge	32	Monitoring Panel	Node Flavor	Manage OMI

Step 4 In the dialog box that is displayed, click **OK** to confirm and remove the read-only status for the cluster.

----End

11.10 Performing a Primary/Standby Switchback

Context

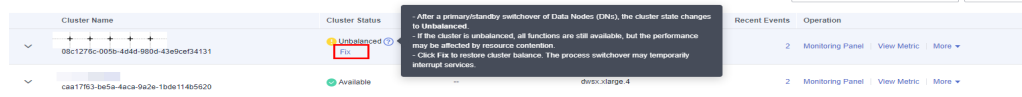
In the **Unbalanced** state, the number of primary instances on some nodes increases. As a result, the load pressure is high. In this case, the cluster is normal, but the overall performance is not as good as that in a balanced state. Restore the primary-standby relationship to recover the cluster to the available state.

NOTE

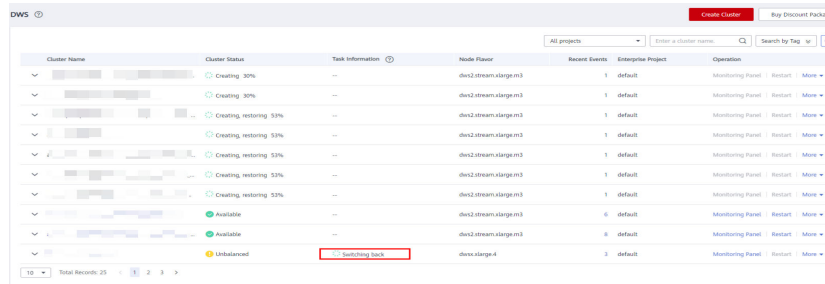
- Only 8.1.1.202 and later versions support primary/standby cluster restoration.
- Cluster restoration interrupts services for a short period of time. The interruption duration depends on the service volume. You are advised to perform this operation during off-peak hours.

Procedure

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** On the **Clusters > Dedicated Clusters** page, locate the cluster in unbalanced state.
- Step 3** In the **Cluster Status** column of the cluster, click **Fix** under **Unbalanced**.



- Step 4** In the dialog box that is displayed, confirm that the service is in off-peak hours, and click **Yes**. A message will be displayed in the upper right corner, indicating that the switchover request is being processed.
- Step 5** Check the cluster status. During the switchover, the cluster status is **Switching back**. After the switchover, the cluster status will change to **Available**.



----End

11.11 Cluster Restart

If a cluster is in the **Unbalanced** state or cannot work properly, you may need to restart it for restoration. After modifying a cluster's configurations, such as security settings and parameters, manually restart the cluster to make the configurations take effect.

Impact on the System

- A cluster cannot provide services during the restart. Therefore, before the restart, ensure that no task is running and all data is saved.

If the cluster is processing service data, such as importing data, querying data, creating snapshots, or restoring snapshots, cluster restarting will cause file damage or restart failure. You are advised to stop all cluster tasks before restarting the cluster.

View the **Session Count** and **Active SQL Count** metrics to check whether the cluster has active events. For details, see [Monitoring Clusters Using Cloud Eye](#).

- The time required for restarting a cluster depends on the cluster scale and services. Generally, it takes about 3 minutes to restart a cluster. The duration does not exceed 20 minutes.

- If the restart fails, the cluster may be unavailable. Try again later or contact technical support.

Procedure

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 Choose **Clusters > Dedicated Cluster**.

Step 3 In the **Operation** column of the cluster to be restarted, click **Restart**.

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

Task Information changes to **Restarting**. When **Cluster Status** changes to **Available** again, the cluster is successfully restarted.

----End

11.12 Resetting a Password

GaussDB(DWS) allows you to reset the password of the database administrator. If a database administrator forgets their password or the account is locked because the number of consecutive incorrect password attempts reaches the upper limit, the database administrator can reset the password on the **Clusters > Dedicated Clusters** page. After the password is reset, the account can be automatically unlocked. You can set the maximum number of incorrect password attempts (10 by default) by configuring the [failed_login_attempts](#) parameter on the **Parameter** page of the cluster. For details, see [Modifying Database Parameters](#).

Resetting a Password

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 Choose **Clusters > Dedicated Cluster**.

Step 3 In the **Operation** column of the target cluster, choose **More > Reset Password**.

Step 4 On the displayed **Reset Password** page, set a new password, confirm the password, and then click **OK**.

The password complexity requirements are as follows:

- Contains 12 to 32 characters.
- Cannot be the username or the username spelled backwards.
- Must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters (~!?,.,:;_){}[]/<>@#%^&*+|\=-)
- Passes the weak password check.
- The new password must be different from the old password, or old password spelled backwards.
- Previous passwords cannot be used repeatedly.

NOTE

If the default database administrator account of the cluster is deleted or renamed, password resetting fails.

----End

11.13 Cluster Upgrade

By default, you do not need to manually upgrade a GaussDB(DWS) cluster. To upgrade a cluster on the console, see [Delivering a Cluster Upgrade Task on the Console](#).

GaussDB(DWS) will notify you of any cluster O&M operation by sending SMS messages. Exercise caution when performing operations on the cluster during the O&M period.

If the upgrade affects the current query requests or service running, contact technical support for emergency handling.

Upgrading a Cluster

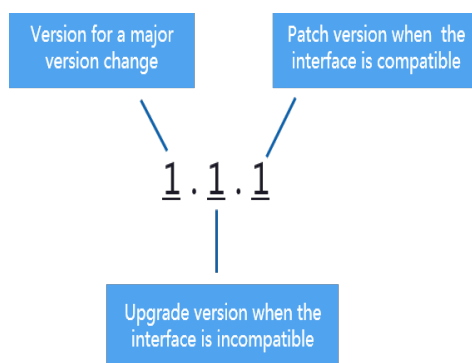
By default, you do not need to care about GaussDB(DWS) cluster patching or upgrading because GaussDB(DWS) will handle version upgrade automatically. After GaussDB(DWS) is upgraded, it will automatically upgrade the cluster to the latest version. During the upgrade, the cluster will be restarted and cannot provide services for a short period of time.

NOTE

- After a cluster is upgraded to 8.1.3 or later, it enters the observation period. During this period, you can check service status and roll back to the earlier version if necessary.
- Upgrading the cluster does not affect the original cluster data or specifications.

The following figure shows the cluster version.

Figure 11-2 Version description



- **Service patch upgrade:** The last digit of cluster version *X.X.X* is changed. For example, the cluster is upgraded from 1.1.0 to 1.1.1.
 - Duration: The whole process will take less than 10 minutes.

- Impact on services: During this period, if the source version is upgraded to 8.1.3 or later, online patching is supported. During the patch upgrade, you do not have to stop services, but the services will be intermittently interrupted for seconds. If the destination version is earlier than 8.1.3, services will be interrupted for 1 to 3 minutes. Therefore, you are advised to perform this operation during off-peak hours.
- **Service upgrade:** The first two digits of cluster version *X.X.X* are changed. For example, the cluster is upgraded from 1.1.0 to 1.2.0.
 - Duration: The whole process will take less than 30 minutes.
 - Impact on services: Online upgrade is supported for update to 8.1.1 or later. During the upgrade, you are not required to stop services, but services are intermittently interrupted for seconds. You are advised to perform the upgrade during off-peak hours.

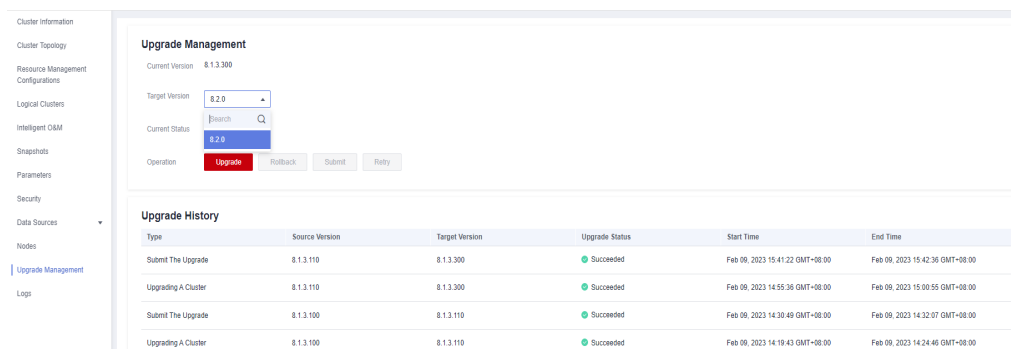
Delivering a Cluster Upgrade Task on the Console

Prerequisites

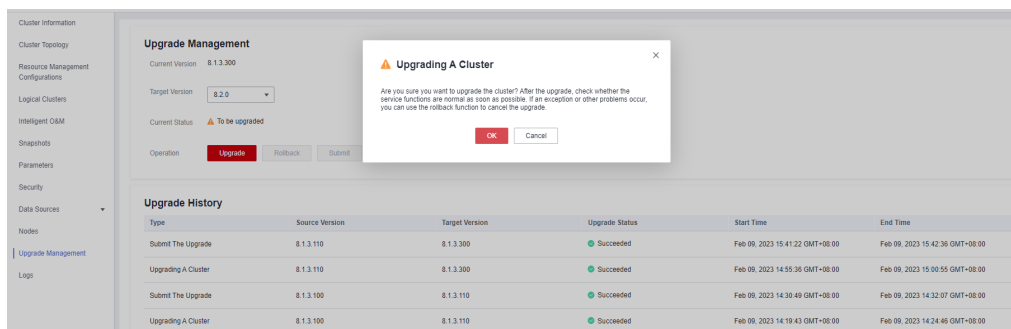
For clusters 8.1.1 or later, you need to deliver cluster upgrade operations on the console.

Procedure

- Step 1** Log in to the GaussDB(DWS) console.
- Step 2** In the cluster list, click the name of a cluster.
- Step 3** In the navigation pane, choose **Upgrade Management**.
- Step 4** On the **Upgrade Management** page, select a version from the **Target Version** drop-down list.



- Step 5** Click **Upgrade**. Click **OK** in the displayed dialog box.

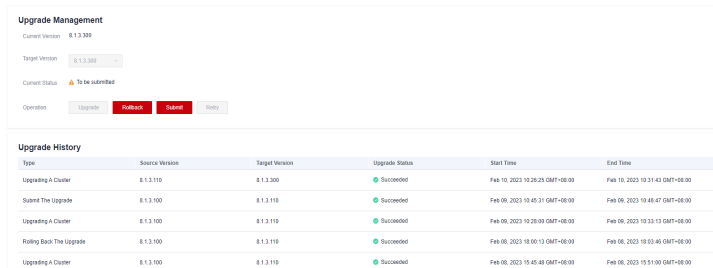


Step 6 Check whether the cluster is successfully upgraded.

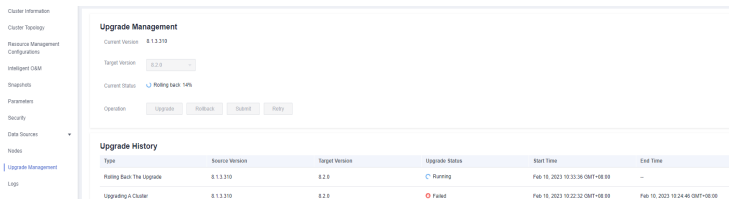
- If the cluster version is 8.1.3 or later, the cluster enters the service observation period after the upgrade is complete. If you have verified your services, click **Submit** on the **Upgrade Management** page to complete the cluster upgrade. If you find your cluster performance affected by the upgrade, you can click **Rollback** to roll back the upgrade.

NOTE

- In versions earlier than 8.1.3, there is neither rollback nor submission button after the upgrade is complete.
- If you do not submit the upgraded version, there will be a **wlm** thread which occupies the system storage space and affects the performance.



- If the cluster upgrade fails, click **Rollback** to roll back to the original cluster version, or click **Retry** to deliver the upgrade again.



----End

11.14 Associating and Disassociating ELB

Overview

If the private IP address or EIP of a CN is used to connect to a cluster, the failure of this CN will lead to cluster connection failure. If a private or public domain name is used for connection, the DNS service randomly selects a private IP address or EIP for each client. This cannot balance loads or avoid single-CN failures. ELB is used to solve these problems.

An ELB distributes access traffic to multiple ECSs for traffic control based on forwarding policies. It improves the fault tolerance capability of application programs. For details, see .

With ELB health checks, CN requests of a cluster can be quickly forwarded to normal CNs. If a CN is faulty, the workload can be immediately shifted to a healthy node, minimizing cluster access faults.

 NOTE

- This feature is supported only in cluster version 8.1.1.200 or later.
- For load balancing and high availability purposes, and to prevent single CN failures, a cluster must be bound to ELB.
- ELB does not support cross-database access.

Constraints and Limitations

- To bind an ELB to a GaussDB(DWS) cluster, the ELB must be in the same region, VPC, and enterprise project as the cluster.
- Only dedicated load balancers can be bound to GaussDB(DWS).

NOTICE

Load balancing is not supported in regions where the dedicated load balancer is not available. You can check whether dedicated load balancers are supported on the ELB console.

-
- The ELB to be associated must use TCP and has a private IP address.
 - When creating an ELB instance, determine its specifications based on your service access traffic. You are advised to select the maximum specifications. On the GaussDB(DWS) console, you can bind to an ELB instance but cannot change its specifications.
 - You only need to create a load balancer if you want to use ELB. GaussDB(DWS) automatically creates the required ELB listeners and backend server groups.
 - When creating a load balancer, ensure that the listeners do not use the same port as the database. Otherwise, ELB cannot be associated.
 - When you associate ELB, the **ROUND_ROBIN** policy is set by default. In addition, the health check interval is set to 10 seconds, the timeout duration is set to 50 seconds, and the number of maximum retries is set to 3. Exercise caution when you modify these ELB parameters.
 - When you disassociate ELB from a cluster, related cluster information is cleared on GaussDB(DWS) but the load balancer is not deleted.
 - If you need to access the ELB cluster using a public IP address or domain name, bind an EIP or domain name on the ELB management console.

Associating ELB

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 Choose **Clusters > Dedicated Cluster**. All clusters are displayed by default.

Step 3 In the cluster list, click the name of the target cluster. The **Cluster Information** page is displayed.

Step 4 On the **Basic Information** page that is displayed, click **Associate ELB** and select the ELB name. If no load balancer exists, create one on the ELB management console. Then refresh the GaussDB(DWS) page and associate ELB with the cluster.

Step 5 After the request is delivered, go back to the **Clusters** page. Task information **Associating ELB** of the cluster is displayed. The process takes some time.

Cluster Name	Cluster Status	Task Information	Node Flavor	Recent Events	Enterprise Project	Operation
...	Available	Associating ELB	dws2.2.large	7 default	default	Monitoring Panel Restart More

Step 6 Log in to the ELB management console, choose **Elastic Load Balance > Load Balancers**, click the name of the bound load balancer, switch to the **Backend Server Groups** tab, and check whether the cluster CNs are associated with the load balancer.

NameID	Status	Private IP Address	Health Check Result	Weight	Backend Port
ip4f26e-8d71-4c30-9c48-28579aa7c4d2	Healthy	192.168.200.211 Elasticsearch INC	Healthy View	1	8000
ip604801-1c28-8d9f-b9d7-89073ca1ba71	Healthy	192.168.147.18 Elasticsearch INC	Healthy View	1	8000
ip7411aa-9d71-43ab-8c76-4376484c7437	Healthy	192.168.129.239 Elasticsearch INC	Healthy View	1	8000

Step 7 In the **Basic Information** area of the **Cluster Information** page, check the **ELB Address**, which is used for connecting to the cluster.

----End

Disassociating ELB

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 Choose **Clusters > Dedicated Cluster**. All clusters are displayed by default.

Step 3 In the cluster list, click the name of the target cluster. The **Cluster Information** page is displayed.

Step 4 On the **Basic Information** page that is displayed, click **Disassociate ELB**.

Step 5 After the request is delivered, go back to the **Clusters** page. Task information **Disassociating ELB** of the cluster is displayed. The process takes some time.

Cluster Name	Cluster Status	Task Information	Node Flavor	Recent Events	Enterprise Project	Operation
...	Available	Disassociating ELB	dws2.2.large	8 default	default	Monitoring Panel Restart More

Step 6 Log in to the ELB management console, click the name of the dissociated ELB, switch to the **Backend Server Groups** tab, and check whether the cluster CNs are deleted.

Backend Server Groups

No data available.

----End

11.15 Managing CNs

Purpose

After a cluster is created, the number of required CNs varies with service requirements. The CN management function enables you to adjust the number of CNs in the cluster. The operations are as follows:

- [Adding CNs](#)
- [Deleting CNs](#)

NOTE

- This feature is supported only in cluster version 8.1.1 or later.
- Only cluster versions 8.1.3.300 and later (excluding 8.2.0) support online CN addition, deletion, and concurrent addition of multiple CNs.

Constraints and Limitations

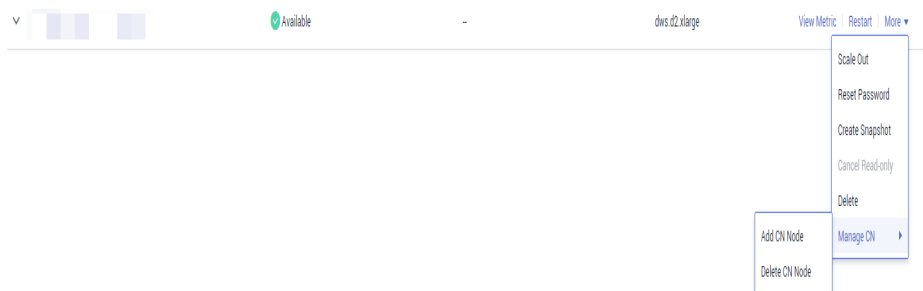
- During resource provisioning, the default number of CNs is 3. You can adjust the number of CNs based on the number of provisioned nodes. The number of CNs ranges from 2 to 20.
- Do not perform other O&M operations when adding or deleting a CN.
- Adding CNs consumes lots of CPU and I/O resources, which will greatly impact job performance. You are advised to perform this operation during off-peak hours or after services are stopped.
- If a fault occurs when you add a CN node and the rollback fails, try adding the CN again. The deletion of a CN node cannot be rolled back.
- For a CN that fails to be added, you can only retry the addition. For a CN that fails to be deleted, you can only retry the deletion. Other O&M operations are not allowed for such CNs.
- If DDL operations, such as schema and function creation, are performed during CN deletion, an error may be reported because the deleted CN cannot be found. In this case, try again.
- If one of your CNs is abnormal, you can only delete this abnormal CN. If two or more CNs are abnormal, you can delete CNs only after the CNs are recovered from faults.

Adding CNs

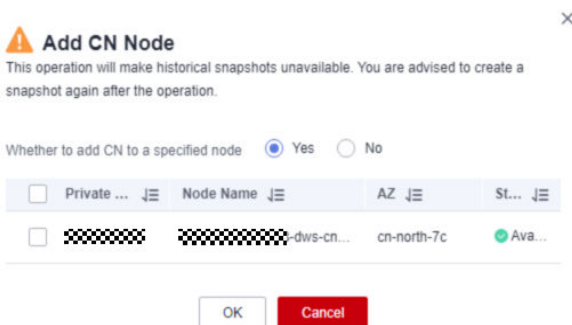
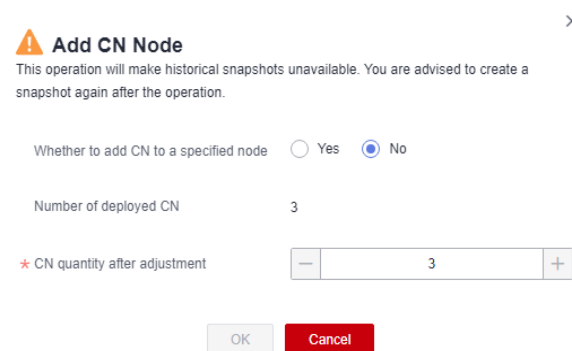
Step 1 Log in to the GaussDB(DWS) management console.

Step 2 On the **Clusters > Dedicated Clusters** page, locate the cluster to which you want to add CNs.

Step 3 In the **Operation** column of the specified cluster, choose **More > Manage CN > Add CN Node**.



Step 4 In the displayed dialog box, determine whether to add CNs to a specified node. If you select **No**, set the number of CNs after adjustment and click **OK**. If you select **Yes**, select a node and click **OK**.



NOTICE

- Before adding a CN, ensure that the cluster is in the **Available** or **Unbalanced** state.
- The number of CNs cannot exceed the total number of nodes after adjustment.
- You cannot add more CNs than the number of CNs that have already been deployed.

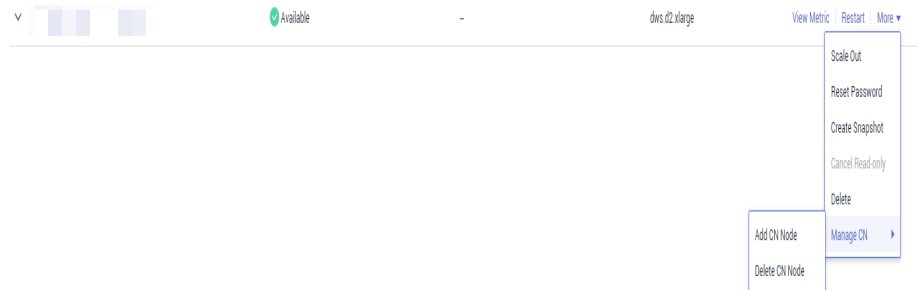
----End

Deleting CNs

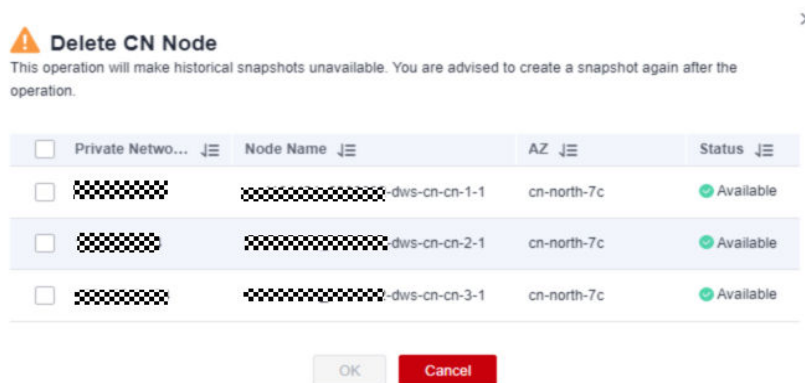
Step 1 Log in to the GaussDB(DWS) management console.

Step 2 On the **Clusters > Dedicated Clusters** page, locate the cluster from which you want to delete CNs.

Step 3 In the **Operation** column of the specified cluster, choose **More > Manage CN > Delete CN Node**.



Step 4 On the displayed page, select the CN to be deleted and click **OK**.



NOTICE

- At least two CN must be retained.
- When you delete a CN, the cluster must be in the **Available**, **Degraded**, or **Unbalanced** state.
- If an elastic IP address has been bound to a CN, the CN cannot be deleted.
- If abnormal nodes exist, only the abnormal CNs can be deleted.
 - If one CN is faulty, only this CN can be deleted.
 - If two or more CNs are faulty, no CN can be deleted.

----End

12 Cluster Log Management

Overview

Cluster logs are collected and sent to Log Tank Service (LTS). You can check or dump the collected cluster logs on LTS.

Currently, the following log types are supported:

- CN logs
- DN logs
- OS messages logs
- Audit logs
- CMS logs
- GTM logs
- Roach client logs
- Roach server logs
- Upgrade logs
- Scaling logs

NOTE

- Cluster log management depends on LTS.
- Only 8.1.1.300 and later versions support cluster log management.
- Only 8.3.0 and later versions support CMS logs, GTM logs, Roach client logs, Roach server logs, scaling logs, and upgrade logs.

Enabling LTS

Step 1 Log in to the GaussDB(DWS) management console.

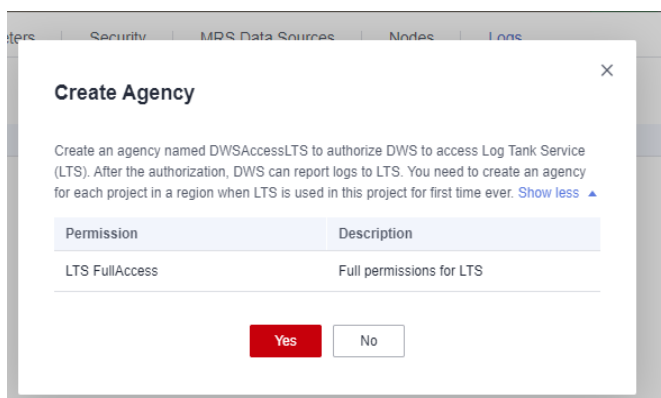
Step 2 Choose **Clusters** > **Dedicated Cluster**. All clusters are displayed by default.

Step 3 Click the name of the target cluster. Choose **Logs**.

Enable LTS

Log Type	Description	Operation
messages	operating system messages log	View Log
expand	dms-expand log	View Log
roach-controller	dms-roach-controller log	View Log
audit	audit Log	View Log
gms	dms-gms log	View Log
roach-agent	dms-roach-agent log	View Log
oms	dms-oms log	View Log
DN	dms-DN node log	View Log
upgrade	dms-upgrade log	View Log
DN	dms-DN node log	View Log

Step 4 On the **Logs** tab, enable LTS. If LTS is enabled for the first time, the following dialog box will be displayed. Confirm the information and click **Yes**.



NOTE

- If LTS has been enabled and authorized to create an agency, no authorization is required when LTS is enabled again.
- By default, only accounts or users with Security Administrator permissions can query and create agencies. IAM users under an account do not have the permission to query or create agencies by default. Contact a user with that permission and complete the authorization on the current page.
- When interconnecting with LTS, you need to grant LTS-related permission policies (**LTS Admin**, **LTS Administrator**, **LTS FullAccess**, and **LTS ReadOnlyAccess**) to users.

Step 5 Check the LTS status.

Enable LTS

Log Type	Description	Operation
messages	operating system messages log	View Log
expand	dms-expand log	View Log
roach-controller	dms-roach-controller log	View Log
audit	audit Log	View Log
gms	dms-gms log	View Log
roach-agent	dms-roach-agent log	View Log
oms	dms-oms log	View Log
DN	dms-DN node log	View Log
upgrade	dms-upgrade log	View Log
DN	dms-DN node log	View Log

----End

Checking Cluster Logs

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 Choose **Clusters > Dedicated Cluster**. All clusters are displayed by default.

Step 3 Click the name of the target cluster. Choose **Logs**.

Step 4 On the **Logs** tab, click **View Log** in the **Operation** column of a log type to go to the Log Tank Service (LTS) page and view logs.

Log Type	Description	Operation
messages	operating system messages log	View Log
expand	divs-expand log	View Log
roach-controller	divs-roach-controller log	View Log
audit	audit Log	View Log
gfm	divs-gfm log	View Log
roach-agent	divs-roach-agent log	View Log
cms	divs-cms log	View Log
CH	divs-CH node log	View Log
upgrade	divs-upgrade log	View Log
DH	divs-DH node log	View Log

----End

Disabling LTS

Step 1 Log in to the GaussDB(DWS) management console.

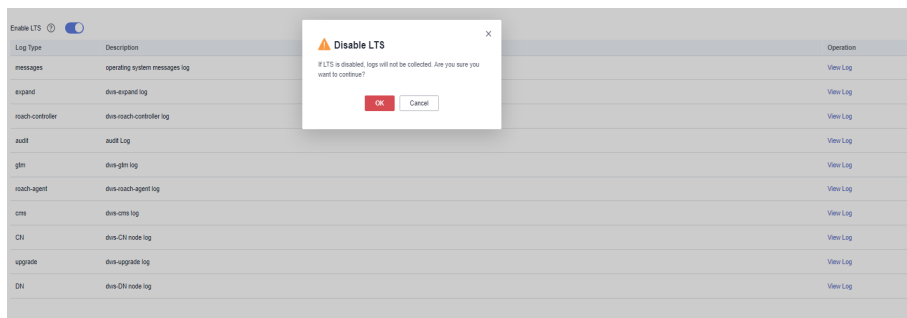
Step 2 Choose **Clusters > Dedicated Cluster**. All clusters are displayed by default.

Step 3 Click the name of the target cluster. Choose **Logs**.

Step 4 Toggle off the LTS switch.

Log Type	Description	Operation
messages	operating system messages log	View Log
expand	divs-expand log	View Log
roach-controller	divs-roach-controller log	View Log
audit	audit Log	View Log
gfm	divs-gfm log	View Log
roach-agent	divs-roach-agent log	View Log
cms	divs-cms log	View Log
CH	divs-CH node log	View Log
upgrade	divs-upgrade log	View Log
DH	divs-DH node log	View Log

Step 5 Click **OK** in the dialog box.



----End

13 Database User Management

13.1 Managing Users

GaussDB(DWS) allows you to manage database users on the console. You can create, delete, and update database users and manage their permissions on the console.

NOTE

- If the current version does not support this feature, contact technical support to upgrade the version.
- After a cluster is created, the users or roles created with it cannot be modified.
- Before using this function, ensure that the cluster is available.

Creating a User

- Step 1** Log in to the GaussDB(DWS) console. In the navigation pane, choose **Clusters > Dedicated Clusters**.
- Step 2** In the cluster list, click the name of the target cluster. The **Cluster Information** page is displayed.
- Step 3** In the navigation pane, choose **User Management**.
- Step 4** On the **Users** tab, click **Create User**. The user creation page is displayed.



No data available.

Step 5 Complete the user information as required, confirm the information, and click **Next**.

- **Username:** A username must start with a letter and can contain letters, numbers, and underscores (_). The length cannot exceed 63 characters.
- **Password:** A password must start with a letter and can contain letters, numbers, and underscores (_). The length cannot exceed 63 characters.

 **NOTE**

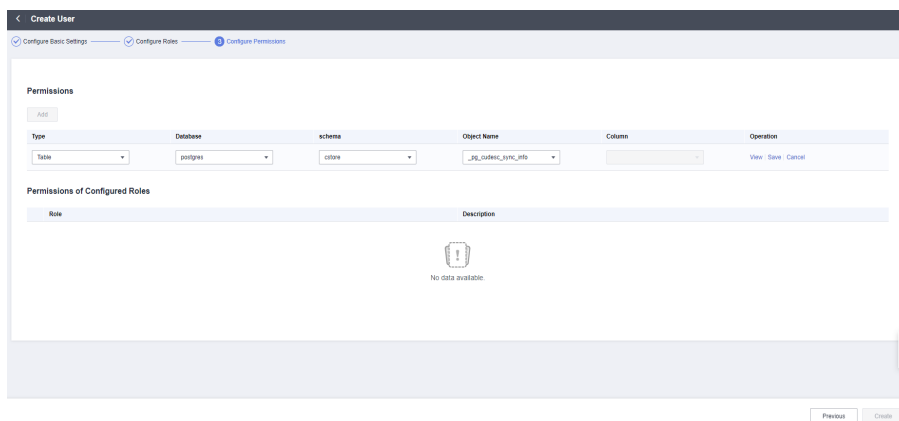
Must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters (~!?,.,:;_){}[]/<>@#%^&*+|\=-)

- **Maximum Connections:** The maximum number of database connections that a user can set up. The value **-1** indicates that the number of connections is not limited.
- **Expires:** Set the expiration time of the user permission.
- **System Administrator:** Whether the user has the system administrator rights.
- **Create Database:** Whether the user has the permission to create databases.
- **Create Role:** Whether the user has the permission to create users and roles.
- **Inherited Permissions:** Whether the user inherits role permissions from its group. **This function is enabled by default. You are advised to retain this setting.**

Step 6 Select the role to be granted to the user and click **Next**.

Step 7 Configure permissions not included in the roles of the user.

Click **Add** to add a permission configuration. Select the database object type and corresponding database object, and select the permission to complete assignment. For details about permission definitions, see "DCL Syntax" > "GRANT" in *SQL Syntax Reference*.



Step 8 After the authorization is complete, click **Create**.

----End

Modifying a User

Step 1 Log in to the GaussDB(DWS) console. In the navigation pane, choose **Clusters** > **Dedicated Clusters**.

- Step 2** In the cluster list, click the name of the target cluster. The **Cluster Information** page is displayed.
- Step 3** In the navigation pane, choose **User Management**.
- Step 4** In the user list, select a user and click **Modify**. The page for modifying user details is displayed.
- Step 5** Modify the user information. For details, see [User Information](#). After confirming that the information is correct, click **Next**.

The screenshot shows the 'Create User' interface with a progress bar at the top indicating three steps: 1. Configure Basic Settings (active), 2. Configure Roles, and 3. Configure Permissions. The 'Basic Settings' section includes the following fields and controls:

- Username:** A text input field with a red asterisk indicating it is required.
- Password:** A text input field with a red asterisk and a visibility icon (eye).
- Confirm Password:** A text input field with a red asterisk.
- Maximum Connections:** A text input field containing '-1' and a help icon (?).
- Expires:** A date and time selector with a calendar icon.
- Create Database:** A toggle switch, currently turned off.
- Create Role:** A toggle switch, currently turned off.
- Inherit Permissions:** A toggle switch, currently turned on.
- Description:** A text area with a character count of '0/500'.

Step 6 Select the role to be granted to the user and click **Next**.

Step 7 Add or remove permissions as required.

The screenshot shows the 'Configure Permissions' step of the user creation process. It features a progress bar at the top with three steps: 1. Configure Basic Settings, 2. Configure Roles, and 3. Configure Permissions (active). The main content area is titled 'Permissions' and includes an 'Add' button. Below this is a table for selecting permissions:

Type	Database	schema	Object Name	Column	Operation
Database	guestdb				View Save Cancel

Below the table is a section titled 'Permissions of Configured Roles' with a table header:

Role	Description
 No data available.	

Step 8 Confirm the permissions. Click **Save**.

----End

Deleting a User

- Step 1** Log in to the GaussDB(DWS) console. In the navigation pane, choose **Clusters > Dedicated Clusters**.
- Step 2** In the cluster list, click the name of the target cluster. The **Cluster Information** page is displayed.
- Step 3** In the navigation pane, choose **User Management**.
- Step 4** Select a user from the user list and click **Delete**. A confirmation dialog box is displayed.
- Step 5** Click **OK**.

NOTE

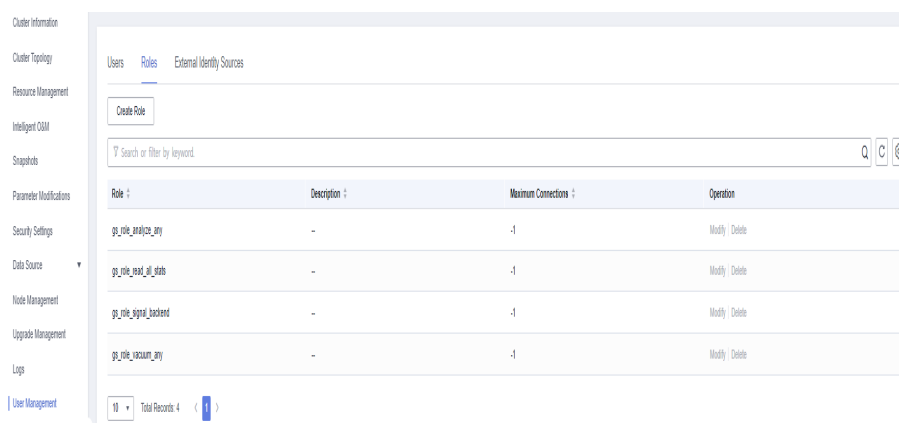
If a user has dependencies on database objects, such as tables, that have not been deleted, the user will fail to be deleted.

----End

13.2 Managing Roles

Creating a Role

- Step 1** Log in to the GaussDB(DWS) console. In the navigation pane, choose **Clusters > Dedicated Clusters**.
- Step 2** In the cluster list, click the name of the target cluster. The **Cluster Information** page is displayed.
- Step 3** In the navigation pane, choose **User Management**.
- Step 4** Click the **Roles** tab and click **Create Role**. The role creation page is displayed.

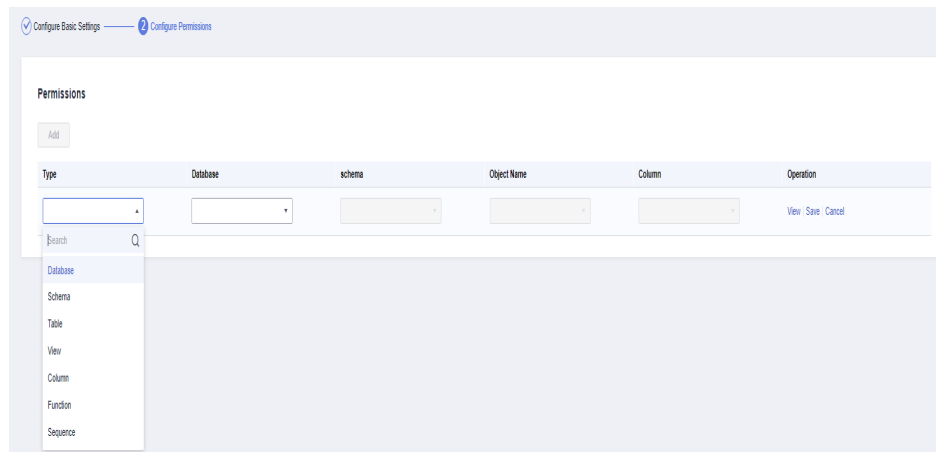


- Step 5** Complete the role information, confirm the information, and click **Next**.
 - **Role Name:** A username must start with a letter and can contain letters, numbers, and underscores (_). The length cannot exceed 63 characters.
 - **Expires:** Set the expiration time of the role permission.
 - **System Administrator:** Whether the role has the system administrator rights.

- **Create Database:** Whether the role has the permission to create databases.
- **Create Role:** Whether the role has the permission to create users and roles.
- **Inherit Permissions:** Whether a role inherits permissions from its group. This function is enabled by default. You are advised to retain this setting.

Step 6 Configure the permissions of the role.

Click **Add** to add a permission configuration. Select the database object type and the corresponding objects. Then, select permissions. For details about permission definitions, see "DCL Syntax" > "GRANT" in *SQL Syntax Reference*.



Step 7 After the authorization is complete, click **Create**. The role is created.

----End

Modifying a Role

Step 1 Log in to the GaussDB(DWS) console. In the navigation pane, choose **Clusters > Dedicated Clusters**.

Step 2 In the cluster list, click the name of the target cluster. The **Cluster Information** page is displayed.

Step 3 In the navigation pane, choose **User Management**.

Step 4 In the role list, select a user and click **Modify**. The page for modifying role details is displayed.

Step 5 Modify the role information, confirm the information, and click **Next**.

- **Role Name:** A username must start with a letter and can contain letters, numbers, and underscores (_). The length cannot exceed 63 characters.
- **Expires:** Set the expiration time of the role permission.
- **System Administrator:** Whether the role has the system administrator rights.
- **Create Database:** Whether the role has the permission to create databases.
- **Create Role:** Whether the role has the permission to create users and roles.
- **Inherit Permissions:** Whether a role inherits the permissions from its group. This function is enabled by default. You are advised to retain this setting.

Step 6 Add or remove permissions as required.



Step 7 Confirm the permissions. Click **Save**.

----End

Deleting a Role

Step 1 Log in to the GaussDB(DWS) console. In the navigation pane, choose **Clusters > Dedicated Clusters**.

Step 2 In the cluster list, click the name of the target cluster. The **Cluster Information** page is displayed.

Step 3 In the navigation pane, choose **User Management**.

Step 4 Select a role from the role list and click **Delete**. A confirmation dialog box is displayed.

Step 5 Click **OK** to delete the role.

NOTE

If the role has dependencies, such as database objects, that have not been deleted, the role will fail to be deleted.

----End

14 Audit Logs

14.1 Audit Log Overview

GaussDB(DWS) provides management console audit logs and database audit logs for users to query service logs, analyze problems, and learn product security and performance status.

Management Console Audit Logs

GaussDB(DWS) uses Cloud Trace Service (CTS) to record mission-critical operations performed on the GaussDB (DWS) management console, such as cluster creation, snapshot creation, cluster scale-out, and cluster restart. The logs can be used in purposes such as security analysis, compliance audit, resource tracing, and fault locating.

For details about how to enable and view management console audit logs, see [Management Console Audit Logs](#).

Database Audit Logs

If the **Security** function is enabled, GaussDB(DWS) records any DML and DDL operations performed by the database. You can locate and analyze faults based on the database audit logs, and perform behavior analysis and security auditing on historical database operations to improve GaussDB (DWS) security.

For details about how to enable and view database audit logs, see [Configuring the Database Audit Logs](#) and [Viewing Database Audit Logs](#).

14.2 Management Console Audit Logs

Enabling CTS

A tracker will be automatically created after CTS is enabled. All traces recorded by CTS are associated with a tracker. Currently, only one tracker can be created for each account.

Step 1 Log in to the management console, choose **Service List > Management & Governance > Cloud Trace Service**. The CTS management console is displayed.

Step 2 Enable CTS.

If you are a first-time CTS user and do not have any trackers in the tracker list, enable CTS first. For details, see "Getting Started > Enabling CTS" in the *Cloud Trace Service User Guide*.

If you have enabled CTS, the system has automatically created a management tracker. Only one management tracker can be created and it cannot be deleted. You can also manually create a data tracker. For details, see **Managing Trackers > Creating a Tracker** in the *Cloud Trace Service User Guide*.

----End

Disabling the Audit Log Function

If you want to disable the audit log function, disable the tracker in CTS.

Step 1 Log in to the management console, choose **Service List > Management & Governance > Cloud Trace Service**. The CTS management console is displayed.

Step 2 Disable the audit log function by disabling the tracker. To enable the audit log function again, you only need to enable the tracker.

For details about how to enable or disable a tracker, see "Tracker Management > Disabling or Enabling a Tracker" in the *Cloud Trace Service User Guide*.

----End

Key Operations

With CTS, you can record operations associated with GaussDB(DWS) for later query, audit, and backtrack operations.

NOTE

The creation and deletion of automatic snapshots are not performed by users, therefore not recorded in audit logs.

Table 14-1 GaussDB(DWS) operations that can be recorded by CTS

Operation	Resource	Event Name
Creating/Restoring a cluster	cluster	createCluster
Deleting a cluster	cluster	deleteCluster
Scaling out a cluster	cluster	resizeCluster
Restarting a cluster	cluster	restartCluster

Operation	Resource	Event Name
Creating a snapshot	backup	createBackup
Deleting a snapshot	backup	deleteBackup
Setting security parameters	configurations	updateConfigurations

Viewing Traces

Step 1 Log in to the management console, choose **Service List > Management & Governance > Cloud Trace Service**. The CTS management console is displayed.

Step 2 In the navigation pane on the left, choose **Trace List**.

Step 3 In the upper right corner of the trace list, click **Filter** to set the search criteria.

The following filters are available:

- **Trace Source, Resource Type, and Search By**
 - **Trace Source:** Select **GaussDB(DWS)**.
 - **Resource Type:** Select **All resource types** or specify a resource type.
 - **Search By:** Select **All filters** or any of the following options:
 - **Trace name:** If you select this option, you also need to select a specific trace name.
 - **Resource ID:** If you select this option, you also need to select or enter a specific resource ID.
 - **Resource name:** If you select this option, you also need to select or enter a specific resource name.
- **Operator:** Select a specific operator (at user level rather than tenant level).
- **Trace Status:** Available options include **All trace statuses, normal, warning, and incident**. You can only select one of them.
- **Start Date and End Date:** You can specify the time period to query traces.

Step 4 Click **Query**.

Step 5 Click  on the left of the trace to be queried to extend its details.

Step 6 Locate the row containing the target trace and click **View Trace** in the **Operation** column.

For details about the key fields in the CTS trace structure, see "Trace References > Trace Structure" and "Trace References > Example Traces" in the *Cloud Trace Service User Guide*.

----End

14.3 Database Audit Logs

14.3.1 Configuring the Database Audit Logs

Prerequisites

Database audit logs are configured on the **Security Settings** page. You can change security settings only when the cluster status is **Available** and **Unbalanced**, and **Task Information** cannot be **Creating snapshot**, **Scaling out**, **Configuring**, or **Restarting**.

Procedure

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 Choose **Clusters** > **Dedicated Cluster**.

Step 3 In the cluster list, click the name of a cluster. Choose **Security**.

By default, **Configuration Status** is **Synchronized**, which indicates that the latest database results are displayed.

Step 4 In the **Audit Settings** area, set the audit items:

NOTE

The default audit log retention policy is space-first, which means audit logs will be automatically deleted when the size of audit logs on a single node exceeds 1 GB. This function prevents node faults or low performance caused by high disk space occupied by audit logs.

[Table 14-2](#) describes the detailed information about the audit items.

Table 14-2 Audit items

Audit Item	Description
Unauthorized access	Specifies whether to record unauthorized operations. This parameter is disabled by default.
DML operations	Specifies whether to record INSERT , UPDATE , and DELETE operations on tables. This parameter is disabled by default.
DDL operations	Specifies whether to record the CREATE , DROP , and ALTER operations of specified database objects. DATABASE , SCHEMA , and USER are selected by default.

Except the audit items listed in [Table 14-2](#), key audit items in [Table 14-3](#) are enabled by default on GaussDB(DWS).

Table 14-3 Key audit items

Parameter	Description
Key audit items	Records successful and failed logins and logout.
	Records database startup, stop, recovery, and switchover.
	Records user locking and unlocking.
	Records the grants and reclaims of user permissions.
	Records the audit function of the SET operation.

Step 5 Enable or disable audit log dumps.

Step 6 Click **Apply**.

Click . The configuration status **Applying** indicates that the configurations are being saved.

When the status changes to **Synchronized**, the configurations are saved and take effect.

----End

14.3.2 Dumping the Database Audit Logs

GaussDB(DWS) records information (audit logs) about connections and user activities in your database. The audit logs help you monitor the database to ensure security, rectify faults, and locate historical operation records.

GaussDB(DWS) audit logs are stored in the database by default. You can dump the audit logs to OBS so that users who monitor database activities can view the logs.

NOTE

- This function cannot be used if OBS is not available.
- Data may during cluster specifications change, CN addition, or CN deletion. You are advised to disable audit log dump during these operations.
- After audit log dumping is enabled, audit logs will be dumped if the size of saved audit logs exceeds 1 GB. This may cause abnormal query results. Exercise caution when performing this operation.
- If a CN node is faulty, data on the CN node may be lost.
- Version support for the audit log dump directory partition is as follows:
 - For 8.1.3.x clusters, only 8.1.3.322 and later versions support this feature. For 8.2.0.x clusters, only 8.2.0.106 and later versions support this feature. By default, the audit log dump directory partition is enabled and cannot be disabled.
 - To use this feature in earlier versions, contact technical support to upgrade your cluster first. Manually enable this feature after the upgrade.

Prerequisites

After a GaussDB(DWS) cluster is created, you can enable log dump for it to dump audit logs to OBS. **Before enabling audit log dump, ensure the following conditions are met:**

Enabling Audit Log Dumps

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 In the navigation pane on the left, choose **Clusters > Dedicated Clusters**.

Step 3 In the cluster list, click the name of the cluster for which you want to enable audit log dump. In the navigation pane, choose **Security Settings**.

Step 4 In the **Audit Settings** area, enable **Audit Log Dump**.

indicates that the function is enabled.

When you enable audit log dump for a project in a region for the first time, the system prompts you to create an agency named **DWSAccessOBS**. After the agency is created, GaussDB(DWS) can dump audit logs to OBS.

By default, only accounts or users with Security Administrator permissions can query and create agencies. IAM users under an account do not have the permission to query or create agencies by default. Contact a user with that permission and complete the authorization on the current page.

- **OBS Foreign Table:** Audit logs can be read using OBS foreign tables during dumping. Audit logs are stored in CSV format and compressed in GZ format.
- **OBS Bucket:** Name of the OBS bucket used to store the audit data. If no OBS bucket is available, click **View OBS Bucket** to access the OBS console and create one. For details, see **Console Operation Guide > Managing Buckets > Creating a Bucket** in the *Object Storage Service User Guide*.
- **OBS Path:** User-defined directory on OBS for storing audit files. Different directory levels are separated by forward slashes (/). The value is a string containing 1 to 50 characters, which cannot start with a forward slash (/). If the entered OBS path does not exist, the system creates one and dumps data to it.
- **Dump Interval (Minute):** Interval based on which GaussDB(DWS) periodically dumps data to OBS. The value range is 5 to 43200. The unit is minute.

Step 5 Click **Apply**.

If **Configuration Status** is **Applying**, the system is saving the settings.

When the status changes to **Synchronized**, the configurations are saved and take effect.

----End

Modifying Audit Log Dump Configurations

After audit log dump is enabled, you can modify the dump configuration. For example, you can modify the OBS bucket and path for storing logs and the dump period.

The procedure is as follows:

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane on the left, choose **Clusters > Dedicated Clusters**.
- Step 3** In the cluster list, click the name of the cluster for which you want to modify the audit log dump configurations. In the navigation pane, choose **Security**.
- Step 4** In the **Audit Settings** area, modify the **Audit Log Dump** configurations.
- Step 5** Click **Apply**.

If **Configuration Status** is **Applying**, the system is saving the settings.

When the status changes to **Synchronized**, the configurations are saved and take effect.

----End

Viewing Audit Log Dumps

After audit log dump is enabled, you can view the dumped audit logs on OBS.

The procedure is as follows:

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane on the left, choose **Clusters > Dedicated Clusters**.
- Step 3** In the cluster list, click the name of the target cluster for which you want to view the log dump history. In the navigation pane, choose **Security**.
- Step 4** In the **Audit Settings** area, click **View Dump Record**.
- Step 5** In the **Audit Log Dump Records** dialog box, click **View OBS Bucket**. The OBS console page is displayed.
- Step 6** Select the OBS bucket and folder where the logs are stored to view the log files.

You can download and decompress the files to view. The fields of audit log files are described as follows:

Table 14-4 Log file fields

Field	Type	Description
begintime	timestamp with time zone	Operation start time
endtime	timestamp with time zone	Operation end time
operation_type	text	Operation type. For details, see Table 14-5 .
audit_type	text	Audit type. For details, see Table 14-6 .
result	text	Operation result

Field	Type	Description
username	text	Name of the user who performs the operation
database	text	Database name
client_conninfo	text	Client connection information, that is, gsql, JDBC, or ODBC.
object_name	text	Object name
command_text	text	Command used to perform the operation
detail_info	text	Operation details
transaction_xid	text	Transaction ID
query_id	text	Query ID
node_name	text	Node name
thread_id	text	Thread ID
local_port	text	Local port
remote_port	text	Remote port

Table 14-5 Operation types

Operation Type	Description
audit_switch	Indicates that the operations of enabling and disabling the audit log function are audited.
login_logout	Indicates that user login and log-out operations are audited.
system	Indicates that the system startup, shutdown, and instance switchover operations are audited.
sql_parse	Indicates that SQL statement parsing operations are audited.
user_lock	Indicates that user locking and unlocking operations are audited.
grant_revoke	Indicates that user permission granting and revoking operations are audited.
violation	Indicates that user's access violation operations are audited.

Operation Type	Description
ddl	Indicates that DDL operations are audited. DDL operations are controlled at a fine granularity based on operation objects. Therefore, audit_system_object is used to control the objects whose DDL operations are to be audited. (The audit function takes effect as long as audit_system_object is configured, no matter whether ddl is set.)
dml	Indicates that the DML operations are audited.
select	Indicates that the SELECT operations are audited.
internal_event	Indicates that internal incident operations are audited.
user_func	Indicates that operations related to user-defined functions, stored procedures, and anonymous blocks are audited.
special_func	Indicates that special function invoking operations are audited. Special functions include pg_terminate_backend and pg_cancel_backend .
copy	Indicates that the COPY operations are audited.
set	Indicates that the SET operations are audited.
transaction	Indicates that transaction operations are audited.
vacuum	Indicates that the VACUUM operations are audited.
analyze	Indicates that the ANALYZE operations are audited.
cursor	Indicates that cursor operations are audited.
anonymous_block	Indicates that the anonymous block operations are audited.
explain	Indicates that the EXPLAIN operations are audited.
show	Indicates that the SHOW operations are audited.
lock_table	Indicates that table lock operations are audited.
comment	Indicates that the COMMENT operations are audited.
preparestmt	Indicates that the PREPARE , EXECUTE , and DEALLOCATE operations are audited.
cluster	Indicates that the CLUSTER operations are audited.
constraints	Indicates that the CONSTRAINTS operations are audited.
checkpoint	Indicates that the CHECKPOINT operations are audited.
barrier	Indicates that the BARRIER operations are audited.

Operation Type	Description
cleanconn	Indicates that the CLEAN CONNECTION operations are audited.
seclabel	Indicates that security label operations are audited.
notify	Indicates that the notification operations are audited.
load	Indicates that the loading operations are audited.

Table 14-6 audit_type parameters

Parameter	Description
audit_open/audit_close	Indicates that the audit type is operations enabling or disabling audit logs.
user_login/user_logout	Indicates that the audit type is operations and users with successful login/logout.
system_start/system_stop/ system_recover/system_switch	Indicates that the audit type is system startup, shutdown, and instance switchover.
sql_wait/sql_parse	Indicates that the audit type is SQL statement parsing.
lock_user/unlock_user	Indicates that the audit type is successful user locking and unlocking.
grant_role/revoke__role	Indicates that the audit type is user permission granting and revoking.
user_violation	Indicates that the audit type is unauthorized user access operations.
ddl_database_object	Indicates that successful DDL operations are audited. DDL operations are controlled at a fine granularity based on operation objects. So, audit_system_object is used to control the objects whose DDL operations are to be audited. (The audit function takes effect as long as audit_system_object is configured, no matter whether ddl is set.) For example, ddl_sequence indicates that the audit type is sequence-related operations.
dml_action_insert/ dml_action_delete/ dml_action_update/ dml_action_merge/ dml_action_select	Indicates that the audit type is DML operations such as INSERT , DELETE , UPDATE , and MERGE .
internal_event	Indicates that the audit type is internal events.

Parameter	Description
user_func	Indicates that the audit type is user-defined functions, stored procedures, or anonymous block operations.
special_func	Indicates that the audit type is special function invocation. Special functions include pg_terminate_backend and pg_cancel_backend .
copy_to/copy_from	Indicates that the audit type is COPY operations.
set_parameter	Indicates that the audit type is SET operations.
trans_begin/trans_commit/ trans_prepare/ trans_rollback_to/ trans_release/trans_savepoint/ trans_commit_prepare/ trans_rollback_prepare/ trans_rollback	Indicates that the audit type is transaction-related operations.
vacuum/vacuum_full/ vacuum_merge	Indicates that the audit type is VACUUM operations.
analyze/analyze_verify	Indicates that the audit type is ANALYZE operations.
cursor_declare/cursor_move/ cursor_fetch/cursor_close	Indicates that the audit type is cursor-related operations.
codeblock_execute	Indicates that the audit type is anonymous blocks.
explain	Indicates that the audit type is EXPLAIN operations.
show	Indicates that the audit type is SHOW operations.
lock_table	Indicates that the audit type is table locking operations.
comment	Indicates that the audit type is COMMENT operations.
prepare/execute/deallocate	Indicates that the audit type is PREPARE , EXECUTE , or DEALLOCATE operations.
cluster	Indicates that the audit type is CLUSTER operations.
constraints	Indicates that the audit type is CONSTRAINTS operations.

Parameter	Description
checkpoint	Indicates that the audit type is CHECKPOINT operations.
barrier	Indicates that the audit type is BARRIER operations.
cleanconn	Indicates that the audit type is CLEAN CONNECTION operations.
seclabel	Indicates that the audit type is security label operations.
notify	Indicates that the audit type is notification operations.
load	Indicates that the audit type is loading operations.

----End

Disabling Audit Log Dumps

You can disable the audit log dump function if you do not want to dump audit logs to OBS.

The procedure is as follows:

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane on the left, choose **Clusters > Dedicated Clusters**.
- Step 3** In the cluster list, click the name of the cluster for which you want to disable audit log dump. In the navigation pane, choose **Security Settings**.
- Step 4** In the **Audit Settings** area, disable audit log dump.
- Step 5** Click **Apply**.

If **Configuration Status** is **Applying**, the system is saving the settings.

When the status changes to **Synchronized**, the configurations are saved and take effect.

----End

14.3.3 Viewing Database Audit Logs

Prerequisites

- The audit function has been enabled by setting **audit_enabled**. The default value of **audit_enabled** is **ON**. To disable audit, set it to **OFF** by referring to [Modifying Database Parameters](#).
- The audit items have been configured. For details about how to enable audit items, see [Configuring the Database Audit Logs](#).

- The database is running properly and a series of addition, modification, deletion, and query operations have been executed in the database. Otherwise, no audit result is generated.
- The audit logs of each database node are recorded separately.
- Only users with the **AUDITADMIN** permission can view audit records.

Viewing Database Audit Logs

Method 1: Audit logs will occupy disk space. To prevent excessive disk usage, GaussDB(DWS) supports audit log dumping. You can enable the **Log Dump** function to dump audit logs to OBS (you need to create an OBS bucket for storing audit logs first). For details about how to view the dumped logs, see [Viewing Audit Log Dumps](#).

Method 2: Use the **Log** function of LTS to view or download the collected database audit logs. For details, see [Checking Cluster Logs](#).

Method 3: Database audit logs are stored in the database by default. After connecting to the cluster, you can use the **pg_query_audit** function to view the logs. For details, see [Using Functions to View Database Audit Logs](#).

Using Functions to View Database Audit Logs

Step 1 Use the SQL client tool to connect to the database cluster. For details, see [Cluster Connection](#).

Step 2 Use the **pg_query_audit** function to query the audit logs of the current CN. The syntax is as follows:

```
pg_query_audit(timestampz starttime,timestampz endtime,audit_log)
```

starttime and **endtime** indicate the start time and end time of the audit record, respectively. **audit_log** indicates the physical file path of the queried audit logs. If **audit_log** is not specified, the audit log information of the current instance is queried.

For example, view the audit records of the current CN node in a specified period.
SELECT * FROM pg_query_audit('2021-02-23 21:49:00','2021-02-23 21:50:00');

The query result is as follows:

```

begintime      |      endtime      | operation_type | audit_type | result | username | database |
client_conninfo | object_name | command_text |          detail_info |
transaction_xid | query_id | node_name |          session_id | local_port | remote_port
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
2021-02-23 21:49:57.76+08 | 2021-02-23 21:49:57.82+08 | login_logout | user_login | ok | dbadmin | gaussdb |
gaussdb | gsql@[local] | gaussdb | login db | login db(gaussdb) successfully, the current user is:
dbadmin | 0 | 0 | coordinator1 | 140324035360512.667403397820909.coordinator1 | 27777
|

```

This record indicates that user **dbadmin** logged in to the **gaussdb** database at 2021-02-23 21:49:57.82 (GMT+08:00). After the host specified by **log_hostname** is started and a client is connected to its IP address, the host name found by reverse DNS resolution is displayed following the at sign (@) in the value of **client_conninfo**.

Step 3 Use the **pgxc_query_audit** function to query audit logs of all CNs. The syntax is as follows:

```
pgxc_query_audit(timestampz starttime,timestampz endtime)
```

For example, view the audit records of all CN nodes in a specified period.

```
SELECT * FROM pgxc_query_audit('2021-02-23 22:05:00','2021-02-23 22:07:00') where audit_type = 'user_login' and username = 'user1';
```

The query result is as follows:

begintime	endtime	operation_type	audit_type	result	username	database	client_conninfo	object_name	command_text	detail_info	transaction_xid
query_id	node_name	session_id	local_port	remote_port							
2021-02-23 22:06:22.219+08	2021-02-23 22:06:22.271+08	login_lgout	user_login	ok	user1	gaussdb	gsq@[local]	gaussdb	login db	login db(gaussdb) successfully, the current user is: user1	0
2021-02-23 22:05:51.697+08	2021-02-23 22:05:51.749+08	login_lgout	user_login	ok	user1	gaussdb	gsq@[local]	gaussdb	login db	login db(gaussdb) successfully, the current user is: user1	0

The query result shows the successful login records of **user1** in to CN1 and CN2.

Step 4 Query the audit records of multiple objects.

```
SET audit_object_name_format TO 'all';
SELECT object_name,result,operation_type,command_text FROM pgxc_query_audit('2022-08-26 8:00:00','2022-08-26 22:55:00') where command_text like '%student%';
```

The query result is as follows:

object_name	result	operation_type	command_text
student	ok	ddl	CREATE TABLE student(stuNo int, stuName TEXT);
studentscore	ok	ddl	CREATE TABLE studentscore(stuNo int, stuscore int);
["public.student_view01","public.studentscore","public.student"]	ok	ddl	CREATE OR REPLACE VIEW student_view01 AS SELECT * FROM student t1 where t1.stuNo in (select stuNo from studentscore t2 where t1.stuNo = t2.stuNo);
["public.student_view01","public.student","public.studentscore"]	ok	dml	SELECT * FROM student_view01;

In the **object_name** column, the table, view, and base table associated with the view are displayed.

----End

15 Cluster Security Management

15.1 Configuring Separation of Permissions

Scenario

By default, the administrator specified when you create a GaussDB(DWS) cluster is the database system administrator. The administrator can create other users and view the audit logs of the database. That is, separation of permissions is disabled.

GaussDB(DWS) supports role-based separation of permissions. In this way, different roles have different permissions and cluster data can be better protected.

For details about the default permissions mode and the separation of permissions mode, see "Database Security Management > Managing Users and Their Permissions > Separation of Permissions" in the *Data Warehouse Service (DWS) Developer Guide*.

Impact on the System

- After you modified the security parameters and the modifications take effect, the cluster may be restarted, which makes the cluster unavailable temporarily.

Prerequisites

To modify the cluster's security configuration, ensure that the following conditions are met:

- The cluster status is **Available**, **To be restarted**, or **Unbalanced**.
- The **Task Information** cannot be **Creating snapshot**, **Scaling out**, **Configuring**, or **Restarting**.

Procedure

Step 1 Log in to the GaussDB(DWS) management console.

Step 2 In the navigation pane on the left, choose **Clusters > Dedicated Clusters**.

Step 3 In the cluster list, click the name of a cluster. On the page that is displayed, click **Security Settings**.

By default, **Configuration Status** is **Synchronized**, which indicates that the latest database result is displayed.

Step 4 On the **Security Settings** page, configure separation of permissions.

indicates that the function is enabled. When separation of permissions is enabled, configure the username and password for **Security Administrator** and **Audit Administrator**. Then the system automatically creates these two users. You can use these two users to connect to the database and perform database-related operations.

. **Rights Separation** is disabled by default.

Table 15-1 Security parameters

Parameter	Description	Example Value
Security Administrator	The username must meet the following requirements: <ul style="list-style-type: none"> Consists of lowercase letters, digits, or underscores. Starts with a lowercase letter or an underscore. Contains 6 to 64 characters. Cannot be a keyword of the GaussDB(DWS) database. For details about the keywords of the GaussDB(DWS) database, see "SQL Syntax Reference > Keyword" in the <i>Data Warehouse Service (DWS) Developer Guide</i>. 	security_admin
Password	The password complexity requirements are as follows: <ul style="list-style-type: none"> Contain 12 to 32 characters. Cannot be the username or the username spelled backwards. Must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters (~!?,.,;_){}[]/<>@#%^&*+ \=-) Passes the weak password check. 	-
Confirm Password	Enter the password of the security administrator again.	-

Parameter	Description	Example Value
Audit Administrator	The username must meet the following requirements: <ul style="list-style-type: none">• Consists of lowercase letters, digits, or underscores.• Starts with a lowercase letter or an underscore.• Contains 6 to 64 characters.• Cannot be a keyword of the GaussDB(DWS) database. For details about the keywords of the GaussDB(DWS) database, see "SQL Syntax Reference > Keyword" in the <i>Data Warehouse Service (DWS) Developer Guide</i>.	audit_admin
Password	The password complexity requirements are as follows: <ul style="list-style-type: none">• Contain 12 to 32 characters.• Cannot be the username or the username spelled backwards.• Must contain at least 3 of the following character types: uppercase letters, lowercase letters, digits, and special characters ~!@#%^&*()-_+ [{]};,.<.>/?• Passes the weak password check.	-
Confirm Password	Enter the password of the audit administrator again.	-

Step 5 Click **Apply**.

Step 6 In the displayed **Save Configuration** dialog box, select or deselect **Restart the cluster** and click **Yes**.

- If you select **Restart the cluster**, the system saves the settings on the **Security Settings** page and restarts the cluster immediately. After the cluster is restarted, the security settings take effect immediately.
- If you do not select **Restart the cluster**, the system only saves the settings on the **Security Settings** page. Later, you need to manually restart the cluster for the security settings to take effect.

After the security settings are complete, **Configuration Status** can be one of the following on the **Security Settings** page:

- **Applying**: The system is saving the settings.
- **Synchronized**: The settings have been saved and taken effect.
- **Take effect after restart**: The settings have been saved but have not taken effect. Restart the cluster for the settings to take effect.

----End

15.2 Encrypting Databases

15.2.1 Overview

Encrypting GaussDB(DWS) Databases

In GaussDB(DWS), you can enable database encryption for a cluster to protect static data. After you enable encryption, data of the cluster and its snapshots is encrypted. Encryption is an optional and immutable setting that can be configured during cluster creation. To encrypt an unencrypted cluster (or in reverse), you need to export all data from the unencrypted cluster and import it to a new cluster that has enabled database encryption. Database encryption is performed when data is written to GaussDB(DWS). That is, GaussDB(DWS) encrypts data when the data is written to GaussDB(DWS). If you want to query the data, GaussDB(DWS) automatically decrypts it and returns the result to you.

If encryption is required, enable it during cluster creation. Although encryption is an optional setting of GaussDB(DWS), you are advised to enable this setting for clusters to protect data.

NOTICE

- The database encryption function can be enabled or disabled only when a cluster is created. It cannot be enabled after a cluster is created. Once enabled, it cannot be disabled. For details, see [Encrypting the Database](#).
 - After **Encrypt DataStore** is enabled, the key cannot be disabled, deleted, or frozen when being used. Otherwise, the cluster becomes abnormal and the database becomes unavailable.
 - Snapshots created after the database encryption function is enabled cannot be restored using open APIs.
-

Viewing Database Encryption Information

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane on the left, choose **Cluster > Dedicated Cluster**.
- Step 3** On the **Dedicated Clusters** page, click the name of a cluster. The **Cluster Information** page is displayed.
- Step 4** In the **Data Encryption Information** area on the cluster information page, view the database encryption information, as shown in [Data encryption information](#).

Table 15-2 Data encryption information

Parameter	Description
Key Name	Indicates the database encryption key of the cluster when Encrypt DataStore is enabled.
Last Key Rotation Time	Indicates the time when the last encryption key is rotated when Encrypt DataStore is enabled.

NOTE

If database encryption is disabled by default during cluster creation, the encryption module is not displayed on the cluster details page.

----End

Encrypting GaussDB(DWS) Databases Using KMS

When you choose KMS to manage GaussDB(DWS) keys, a three-layer key management structure is adopted, including the cluster master key (CMK), cluster encryption key (CEK), and database encryption key (DEK).

- The CMK is used to encrypt the CEK and is stored in KMS.
- The CEK is used to encrypt the DEK. The CEK plaintext is stored in the data warehouse cluster's memory, and the ciphertext is stored in GaussDB(DWS).
- The DEK is used to encrypt database data. The DEK plaintext is stored in the data warehouse cluster's memory, and the ciphertext is stored in GaussDB(DWS).

The procedure of using the keys is as follows:

1. You choose a CMK.
2. GaussDB(DWS) randomly generates the CEK and DEK plaintext.
3. KMS uses the CMK you choose to encrypt the CEK plaintext and imports the encrypted CEK ciphertext to GaussDB(DWS).
4. GaussDB(DWS) uses the CEK plaintext to encrypt the DEK plaintext and saves the encrypted DEK ciphertext.
5. GaussDB(DWS) transfers the DEK plaintext to the cluster and loads it to the cluster's memory.

When the cluster is restarted, it automatically requests the DEK plaintext from GaussDB(DWS) through an API. GaussDB(DWS) loads the CEK and DEK ciphertext to the cluster's memory, invokes KMS to decrypt the CEK using the CMK, loads the CEK to the memory, decrypts the DEK using the CEK plaintext, loads the DEK to the memory, and returns it to the cluster.

Rotating Encryption Keys

Encryption key rotation is used to update the ciphertext stored on GaussDB(DWS). On GaussDB(DWS), you can rotate the encrypted CEK of an encrypted cluster.

The procedure of rotating the keys is as follows:

1. The GaussDB(DWS) cluster starts key rotation.
2. GaussDB(DWS) decrypts the CEK ciphertext stored on GaussDB(DWS) based on the CMK to obtain the CEK plaintext.
3. Use the obtained CEK plaintext to decrypt the DEK ciphertext in GaussDB(DWS) to obtain the DEK plaintext.
4. GaussDB(DWS) randomly generates new CEK plaintext.
5. GaussDB(DWS) uses the new CEK plaintext to encrypt the DEK and saves the encrypted DEK ciphertext.
6. Use the CMK to encrypt the new CEK plaintext and import the encrypted CEK ciphertext to GaussDB(DWS).

You can plan the key rotation interval based on service requirements and data types. To improve data security, you are advised to periodically rotate the keys to prevent the keys from being cracked. Once you find that your keys may have been disclosed, rotate the keys in time.

NOTE

- When GaussDB(DWS) rotates the cluster's CEK, snapshots of the cluster do not need CEK rotation, because the CEK is not stored in snapshots. The CEK plaintext is stored in the GaussDB(DWS) cluster memory, and the ciphertext is stored in GaussDB(DWS).
- The DEK is not updated during key rotation, so data encryption and decryption are not affected.

15.2.2 Rotating Encryption Keys

If you have enabled the **Encrypt DataStore** function in **Advanced Settings** during cluster creation, you can rotate the encryption keys for the cluster after the cluster is created successfully. Each key rotation will update the CEK once. During the key rotation, the cluster is still in **Available** status.

Rotating Encryption Keys for Data Warehouse Clusters

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation tree on the left, choose **Cluster > Dedicated Cluster**.
- Step 3** In the cluster list, find the target cluster and click the cluster name. The **Cluster Information** page is displayed.
- Step 4** In the **Data Encryption Information** area, click **Key Rotation**.
- Step 5** In the dialog box that is displayed, click **Yes**.

----End

15.3 Permissions

15.3.1 Syntax of Fine-Grained Permissions Policies

In actual services, you may need to grant different operation permissions on resources to users of different roles. The IAM service provides fine-grained access

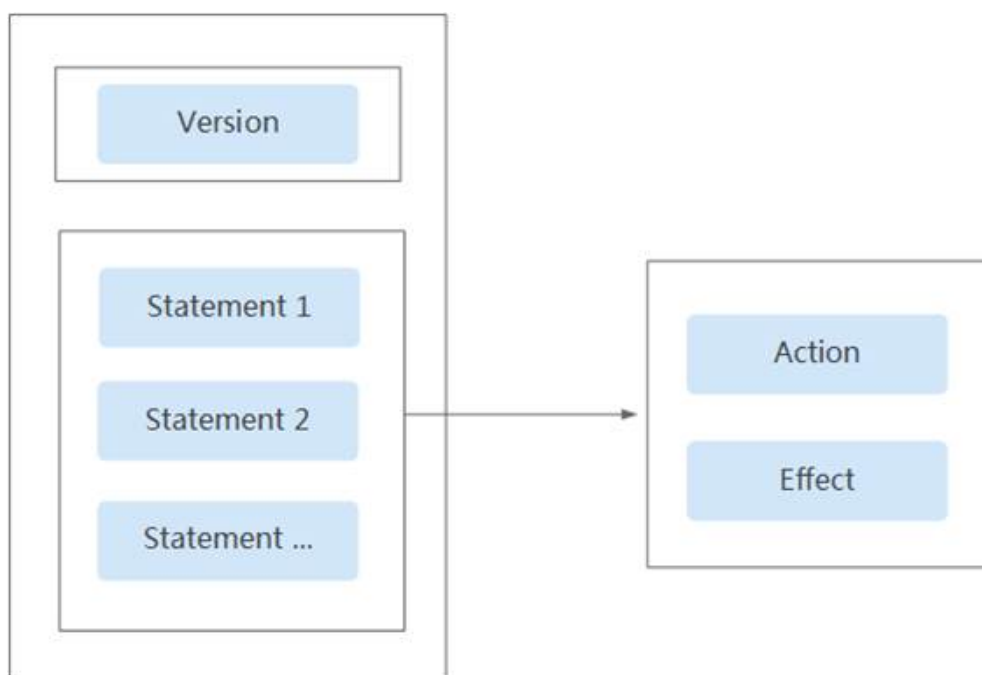
control. An IAM administrator (a user in the **admin** group) can create a custom policy containing required permissions. After a policy is granted to a user group, users in the group can obtain all permissions defined by the policy. In this way, IAM implements fine-grained permission management.

To control the GaussDB(DWS) operations on resources more precisely, you can use the user management function of IAM to grant different operation permissions to users of different roles for fine-grained permission control.

Policy Structure

A fine-grained policy consists of a Version and a Statement. Each policy can have multiple statements.

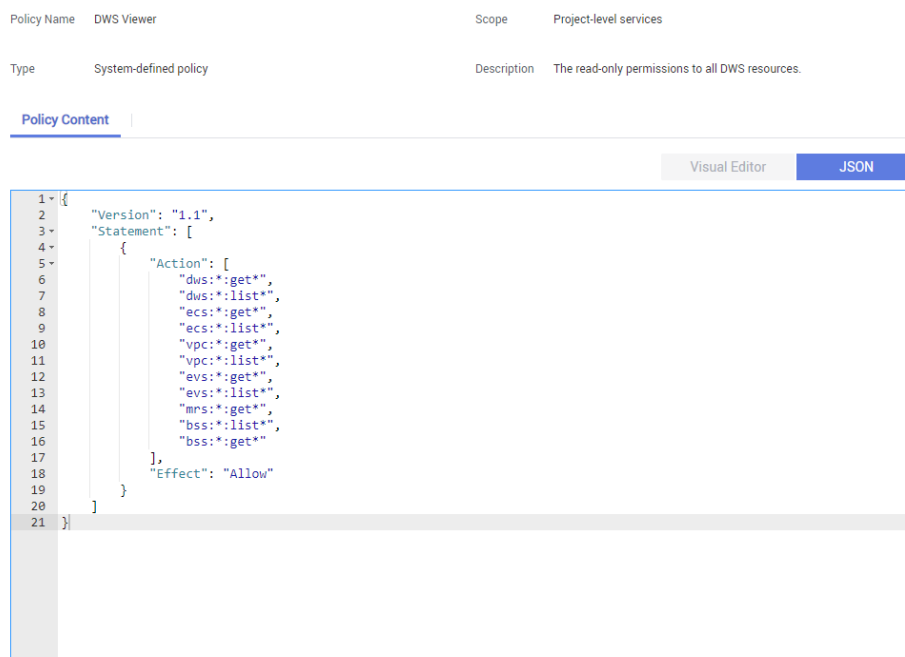
Figure 15-1 Policy structure



Policy Syntax

In the navigation pane on the IAM console, click **Policies** and then click the name of a policy to view its details. The **DWS ReadOnlyAccess** policy is used as an example to describe the syntax of fine-grained policies.

Figure 15-2 Setting the policy



```

{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dws:*:get*",
        "dws:*:list*",
        "ecs:*:get*",
        "ecs:*:list*",
        "vpc:*:get*",
        "vpc:*:list*",
        "evs:*:get*",
        "evs:*:list*",
        "bss:*:list*",
        "bss:*:get*"
      ]
    }
  ]
}

```

- **Version:** Distinguishes between role-based access control (RBAC) and fine-grained policies.
 - **1.0:** RBAC policies. An RBAC policy consists of permissions for an entire service. Users in a group with such a policy assigned are granted all of the permissions required for that service.
 - **1.1:** Fine-grained policies. A fine-grained policy consists of API-based permissions for operations on specific resource types. Fine-grained policies, as the name suggests, allow for more fine-grained control than RBAC policies. Users granted permissions of such a policy can only perform specific operations on the corresponding service. Fine-grained policies include system and custom policies.
- **Statement:** Permissions defined by a policy, including Effect and Action.

- Effect
The valid values for Effect are Allow and Deny. System policies contain only Allow statements. For custom policies containing both Allow and Deny statements, the Deny statements take precedence.
- Action
Permissions in the format of *Service name:Resource type:Operation*. A policy can contain one or more permissions. The wildcard (*) is allowed to indicate all of the services, resource types, or operations depending on its location in the action.
Example: **dws:cluster:create**, permissions for create data warehouse clusters.

List of Supported Actions

When creating a custom policy on IAM, you can add the operations on GaussDB(DWS) resources or the permissions corresponding to RESTful APIs to the action list of the policy authorization statement so that the policy contains the operation permissions. The following table lists the GaussDB(DWS) permissions.

- **REST API**

For details about RESTful API actions supported by GaussDB(DWS), see "Permissions Policies and Supported Actions in " in *Data Warehouse Service API Reference*.

- **Management console operations**

Table 15-3 describes the GaussDB(DWS) operations on resources and corresponding permissions.

 **NOTE**

Some GaussDB(DWS) permissions depend on the actions of ECS, VPC, EVS, ELB, MRS, and OBS. Grant GaussDB(DWS) the required service admin permissions.

Table 15-3 GaussDB(DWS) permissions

Operation	Permission	Dependent Permission	Scope
Creating a cluster	"dws:cluster:create"	"dws:*.get*", "dws:*.list*", "ecs:*.get*", "ecs:*.list*", "ecs:*.create*", "vpc:*.get*", "vpc:*.list*", "vpc:*.create*", "vpc:securityGroupRules:delete", "vpc:ports:update", "evs:*.get*", "evs:*.list*", "evs:*.create*",	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Obtaining the cluster list	"dws:cluster:list"	--	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Obtaining the details of a cluster	"dws:cluster:getDetail"	"dws:*.get*", "dws:*.list*", "vpc:vpcs:list", "vpc:securityGroups:get"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Setting automated snapshot policy	"dws:cluster:setAutomatedSnapshot"	"dws:backupPolicy:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Setting security parameters/parameter groups	"dws:cluster:setSecuritySettings"	"dws:*.get*", "dws:*.list*",	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Restarting a Cluster	"dws:cluster:restart"	"dws:*.get*", "dws:*.list*",	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project

Operation	Permission	Dependent Permission	Scope
Scaling out clusters	"dws:cluster:scaleOut"	"dws:*.get*", "dws:*.list*", "dws:cluster:scaleOutOrOpenAPIResize", "ecs:*.get*", "ecs:*.list*", "ecs:*.create*", "vpc:*.get*", "vpc:*.list*", "vpc:*.create*", "vpc:*.update*", "evs:*.get*", "evs:*.list*", "evs:*.create*",	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Scaling out or resizing a cluster via API	"dws:cluster:scaleOutOrOpenAPIResize"	"dws:*.get*", "dws:*.list*", "vpc:vpcs:list", "vpc:ports:create", "vpc:ports:get", "vpc:ports:update", "vpc:subnets:get", "vpc:subnets:update", "vpc:subnets:create", "vpc:routers:get", "vpc:routers:update", "vpc:networks:create", "vpc:networks:get", "vpc:networks:update", "ecs:serverInterfaces:use", "ecs:serverInterfaces:get", "ecs:cloudServerFlavors:get"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project - Enterprise project

Operation	Permission	Dependent Permission	Scope
Resetting Your Password	"dws:cluster:resetPassword"	"dws:*.get*", "dws:*.list*",	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Deleting a cluster	"dws:cluster:delete"	"dws:*.get*", "dws:*.list*", "ecs:*.get*", "ecs:*.list*", "ecs:*.delete*", "vpc:*.get*", "vpc:*.list*", "vpc:*.delete*", "evs:*.get*", "evs:*.list*", "evs:*.delete*",	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Configuring maintenance windows	"dws:cluster:setMaintenanceWindow"	"dws:*.get*", "dws:*.list*",	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Binding EIPs	"dws:eip:operate"	"dws:*.get*", "dws:*.list*", "eip:*.get*", "eip:*.list*"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Unbinding EIPs	"dws:eip:operate"	"dws:*.get*", "dws:*.list*", "eip:*.get*", "eip:*.list*"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
MRS data source list	"dws:MRSSource:list"	"mrs:cluster:list", "mrs:tag:listResource", "mrs:tag:list", "dws:*.get*", "dws:*.list*"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Adding/Deleting tags	"dws:tag:addAndDelete"	"dws:*.get*", "dws:*.list*", "dws:openAPITag:update", "dws:openAPITag:getResourceTag",	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project

Operation	Permission	Dependent Permission	Scope
Editing tags	"dws:tag:edit"	"dws:*.get*", "dws:*.list*", "dws:openAPITag:update", "dws:openAPITag:getResourceTag",	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Creating a snapshot	"dws:snapshot:create"	"dws:*.get*", "dws:*.list*",	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Obtaining the snapshot list	"dws:snapshot:list"	--	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Viewing the snapshot list of a cluster	"dws:clusterSnapshot:list"	"dws:cluster:list", "dws:openAPICluster:getDetail"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Deleting snapshots	"dws:snapshot:delete"	"dws:snapshot:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Copying snapshots	"dws:snapshot:copy"	"dws:snapshot:list", "dws:snapshot:create"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Restoring data to a new cluster	"dws:cluster:restore"	"dws:*.get*", "dws:*.list*", "ecs:*.get*", "ecs:*.list*", "ecs:*.create*", "vpc:*.get*", "vpc:*.list*", "vpc:*.create*", "evs:*.get*", "evs:*.list*", "evs:*.create*"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project

Operation	Permission	Dependent Permission	Scope
Resizing a cluster	"dws:cluster:resize"	"dws:*.get*", "dws:*.list*", "ecs:*.get*", "ecs:*.list*", "ecs:*.create*", "ecs:*.delete*", "vpc:*.get*", "vpc:*.list*", "vpc:*.create*", "vpc:*.delete*", "evs:*.get*", "evs:*.list*", "evs:*.create*", "evs:*.delete"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Switchback	"dws:cluster:switchover"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Querying the ELB list	"dws:elb:list"	"dws:*.get*", "dws:*.list*", "elb:*.get*", "elb:*.list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Associating ELB	"dws:elb:bind"	"dws:*.get*", "dws:*.list*", "ecs:*.get*", "ecs:*.list*", "vpc:*.get*", "vpc:*.list*", "evs:*.get*", "evs:*.list*", "elb:*.get*", "elb:*.list*", "elb:*.delete*", "elb:*.create"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project

Operation	Permission	Dependent Permission	Scope
Disassociating ELB	"dws:elb:unbind"	"dws*:get*", "dws*:list*", "ecs*:get*", "ecs*:list*", "vpc*:get*", "vpc*:list*", "evs*:get*", "evs*:list*", "elb*:get*", "elb*:list*", "elb*:delete*"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Querying snapshot configurations	"dws:snapshotConfig:list"	"dws*:get*", "dws*:list*"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Updating a snapshot policy	"dws:backupPolicyDetail:update"	"dws*:get*", "dws*:list*"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Deleting a snapshot policy	"dws:backupPolicy:delete"	"dws*:get*", "dws*:list*"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Querying a snapshot policy	"dws:backupPolicy:list"	"dws:cluster:list"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Querying cluster encryption information	"dws:clusterEncryptInfo:list"	"dws*:get*", "dws*:list*", "KMS Administrator"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Creating an agent	"dws:createAgency:create"	"dws*:get*", "dws*:list*", "security administrator"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Querying OBS bucket information	"dws:queryBuckets:list"	"dws*:get*", "dws*:list*"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project

Operation	Permission	Dependent Permission	Scope
Adding a node	"dws:expandWithExisten dNodes:update"	"dws:*.get*", "dws:*.list*", "ecs:*.get*", "ecs:*.list*", "ecs:*.create*", "vpc:*.get*", "vpc:*.list*", "vpc:*.create*", "vpc:*.update*", "evs:*.get*", "evs:*.list*", "evs:*.create*",	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Deleting a DR backup	"dws:disasterRecovery:d elete"	"dws:*.get*", "dws:*.list*", "ecs:*.get*", "ecs:*.list*", "ecs:*.delete*", "vpc:*.get*", "vpc:*.list*", "vpc:*.delete*", "evs:*.get*", "evs:*.list*", "evs:*.delete"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Creating a DR backup	"dws:disasterRecovery:c reate"	"dws:*.get*", "dws:*.list*", "ecs:*.get*", "ecs:*.list*", "ecs:*.create*", "vpc:*.get*", "vpc:*.list*", "vpc:*.create*", "evs:*.get*", "evs:*.list*", "evs:*.create*",	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project

Operation	Permission	Dependent Permission	Scope
Other DR and backup operations	"dws:disasterRecovery:otherOperate"	"dws:*.get*", "dws:*.list*", "ecs:*.get*", "ecs:*.list*", "ecs:*.create*", "vpc:*.get*", "vpc:*.list*", "vpc:*.create*", "evs:*.get*", "evs:*.list*", "evs:*.create*"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Querying DR and backup operations	"dws:disasterRecovery:get"	"dws:*.get*", "dws:*.list*", "ecs:*.get*", "ecs:*.list*", "vpc:*.get*", "vpc:*.list*", "evs:*.get*", "evs:*.list*"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Adding a CN	"dws:module:install"	"dws:*.get*", "dws:*.list*",	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Deleting a CN	"dws:module:uninstall"	"dws:*.get*", "dws:*.list*",	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Removing nodes	"dws:clusterNodes:operate"	"dws:*.get*", "dws:*.list*"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Updating the node alias	dws:instanceAliasName:update	dws:cluster:list	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Redistributing data	"dws:redistribution:operate"	"dws:*.get*", "dws:*.list*",	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Querying redistribution	"dws:redistributionInfo:list"	"dws:*.get*", "dws:*.list*",	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Stopping redistribution	"dws:redistribution:suspend"	"dws:*.get*", "dws:*.list*",	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Resuming redistribution	"dws:redistribution:recover"	"dws:*.get*", "dws:*.list*",	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project

Operation	Permission	Dependent Permission	Scope
Querying product specifications	"dws:specProduct:list"	"dws:*.get*", "dws:*.list*", "ecs:*.get*", "ecs:*.list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Binding the management plane IP address	"dws:bindManagement:operate"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Obtaining user authorization	"dws:checkAuthorize:operate"	"dws:*.get*", "dws:*.list*", "dws:checkSupport:operate"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Authorizing a user	"dws:authorize:operate"	"dws:*.get*", "dws:*.list*", "dws:checkSupport:operate"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Querying user databases	"dws:userDatabase:list"	"dws:*.get*", "dws:*.list*", "dws:checkSupport:operate"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Querying user schemas	"dws:schemas:list"	"dws:*.get*", "dws:*.list*", "dws:checkSupport:operate"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Querying user tables	"dws:tables:list"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Restoring tables	"dws:tableRestore:operate"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Checking the name of the table to be restored	"dws:tableRestoreCheck:operate"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Checking whether a cluster supports fine-grained backup	"dws:checkSupport:operate"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project

Operation	Permission	Dependent Permission	Scope
Querying the list of flavors that can be changed	"dws:supportFlavors:list"	"dws:*.get*", "dws:*.list*",	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Changing the node flavor	"dws:specResize:operate"	"dws:*.get*", "dws:*.list*", "ecs:*.get*", "ecs:*.list*", "ecs:*.create**"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Stopping snapshot creation	"dws:snapshot:stop"	"dws:snapshot:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Terminating a session	"dws:dmsSession:terminate"	"dws:dmsGrpcOuter:operation"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Workload report operations	"dws:dmsWorkloadDiagnosisReport:create"	"dws:dmsGrpcOuter:operation"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Modifying an alarm rule	"dws:dmsAlarmRule:update"	"dws:dmsQuery:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Enabling an alarm rule	"dws:dmsAlarmRule:enable"	"dws:dmsQuery:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Enabling a cluster alarm	"dws:dmsClusterAlarm:enable"	"dws:dmsQuery:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Disabling a cluster alarm	"dws:dmsClusterAlarm:disable"	"dws:dmsQuery:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
gRPC external service	"dws:dmsGrpcOuter:operation"	"dws:dmsQuery:list", "dws:cluster:setSecuritySettings", "obs:bucket:ListAllMyBuckets"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Adding a SQL probe	"dws:dmsProbe:add"	"dws:dmsGrpcOuter:operation"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Modifying a SQL probe	"dws:dmsProbe:update"	"dws:dmsGrpcOuter:operation"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Deleting a SQL probe	"dws:dmsProbe:delete"	"dws:dmsGrpcOuter:operation"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project

Operation	Permission	Dependent Permission	Scope
Enabling or disabling a SQL probe	"dws:dmsProbe:enable"	"dws:dmsGrpcOuter:operation"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Creating a User panel	"dws:dmsUserBoard:create"	"dws:dmsQuery:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Modifying a user panel	"dws:dmsUserBoard:update"	"dws:dmsQuery:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Deleting a user panel	"dws:dmsUserBoard:delete"	"dws:dmsQuery:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Terminating a query	"dws:dmsQuery:terminate"	"dws:dmsGrpcOuter:operation"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Enabling or disabling DMS	"dws:dmsService:enableOrDisable"	"dws:dmsQuery:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Modifying DMS storage configurations	"dws:dmsStorageConfig:modify"	"dws:dmsQuery:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Obtaining, or creating a DDL review	"dws:dmsDdlExamine:getOrCreate"	"dws:dmsGrpcOuter:operation"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Workload snapshot operations	"dws:dmsWorkloadDiagnosisSnapshot:create"	"dws:dmsGrpcOuter:operation"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Creating an alarm rule	"dws:dmsAlarmRule:add"	"dws:dmsQuery:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Deleting an alarm rule	"dws:dmsAlarmRule:delete"	"dws:dmsQuery:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Executing a SQL probe	"dws:dmsProbe:execute"	"dws:dmsGrpcOuter:operation"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Deleting a monitoring item	"dws:dmsPerformanceMonitor:delete"	"dws:dmsQuery:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Enabling or disabling DMS monitoring metrics	"dws:dmsCollectItem:enableOrDisable"	"dws:dmsGrpcOuter:operation"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project
Modifying DMS monitoring configurations	"dws:dmsCollectConfig:modify"	"dws:dmsGrpcOuter:operation"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> - Project

Operation	Permission	Dependent Permission	Scope
OpenAPI Conditional Query	"dws:dmsOpenapiQuery:list"	"dws:cluster:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Disabling an alarm rule	"dws:dmsAlarmRule:disable"	"dws:dmsQuery:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Deleting an alarm record	"dws:dmsAlarmRecord:delete"	"dws:dmsQuery:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Checking SQL probes	"dws:dmsProbe:check"	"dws:dmsGrpcOuter:operation"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Adding a monitoring item	"dws:dmsPerformanceMonitor:add"	"dws:dmsQuery:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Modifying monitoring metrics	"dws:dmsPerformanceMonitor:update"	"dws:dmsQuery:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Downloading historical monitoring trend	"dws:dmsTrendHistory:down"	"dws:dmsQuery:list"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Obtaining cluster ring information	"dws:ring:list"	"dws:*.get*", "dws:*.list*"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Obtaining the cluster process topology	"dws:processTopo:list"	"dws:*.get*", "dws:*.list*"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Querying intelligent O&M information	"dws:operationalTask:get"	"dws:*.get*", "dws:*.list*"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Intelligent O&M Operations	"dws:operationalTask:operate"	"dws:*.get*", "dws:*.list*"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Creating an endpoint service	"dws:vpcEndpointService:create"	"dws:*.get*", "dws:*.list*"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Querying the resource management list	"dws:workLoadManager:get"	"dws:*.get*", "dws:*.list*"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project
Resource management operations	"dws:workLoadManager:operate"	"dws:*.get*", "dws:*.list*"	<ul style="list-style-type: none"> ● Scope: <ul style="list-style-type: none"> – Project

Operation	Permission	Dependent Permission	Scope
LTS operations	"dws:ltsAccess:operate"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Querying LTS Information	"dws:ltsAccess:get"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Querying events	"dws:event:list"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Querying event specifications	"dws:event:list"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Querying event subscriptions	"dws:eventSub:list"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Creating an event subscription	"dws:eventSub:create"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Updating an event subscription	"dws:eventSub:update"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Deleting an event subscription	"dws:eventSub:delete"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Querying alarm statistics	"dws:alarmStatistic:list"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Querying alarm details	"dws:alarmDetail:list"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Querying alarm configurations	"dws:alarmConfig:list"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Querying alarm subscriptions	"dws:alarmSub:list"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Creating an alarm subscription	"dws:alarmSub:create"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Updating an alarm subscription	"dws:alarmSub:update"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project
Deleting an alarm subscription	"dws:alarmSub:delete"	"dws:*.get*", "dws:*.list"	<ul style="list-style-type: none"> Scope: <ul style="list-style-type: none"> – Project

Operation	Permission	Dependent Permission	Scope
Delivering cluster upgrade operations (upgrade, rollback, submission, and retry)	"dws:cluster:doUpdate"	"dws:*.get*", "dws:*.list*"	● Scope: – Project
Querying the available upgrade paths of a cluster	"dws:cluster:getUpgradePaths"	"dws:*.get*", "dws:*.list*"	● Scope: – Project
Querying cluster upgrade records	"dws:cluster:getUpgradeRecords"	"dws:*.get*", "dws:*.list*"	● Scope: – Project

Authorization Using the Fine-Grained Permission Policy

Step 1 Log in to the IAM console and create a custom policy.

For details, see "Fine-Grained Policy Management > Creating a Custom Policy" in the *Identity and Access Management User Guide*.

Refer to the following to create the policy:

- Use the IAM administrator account, that is, the user in the **admin** user group, because only the IAM administrator has the permissions to create users and user groups and modify user group permissions.
- GaussDB(DWS) is a project-level service, so its **Scope** must be set to **Project-level services**. If this policy is required to take effect for multiple projects, authorization is required to each project.
- Two GaussDB(DWS) policy templates are preconfigured on IAM. When creating a custom policy, you can select either of the following templates and modify the policy authorization statement based on the template:
 - **DWS Admin**: has all execution permissions on GaussDB(DWS).
 - **DWS Viewer**: has the read-only permission on GaussDB(DWS).
- You can add permissions corresponding to GaussDB(DWS) operations or RESTful APIs listed in [List of Supported Actions](#) to the action list in the policy authorization statement, so that the policy can obtain the permissions.

For example, if **dws:cluster:create** is added to the action list of a policy statement, the policy has the permission to create or restore clusters.
- If you want to use other services, grant related operation permissions on these services. For details, see the help documents of related services.

For example, when creating a data warehouse cluster, you need to configure the VPC to which the cluster belongs. To obtain the VPC list, add permission **vpc:*.get*** to the policy statement.

Step 2 Create a user group.

For details, see "User and User Group Management > Viewing or Modifying User Group Information > Creating a User Group" in the *Identity and Access Management User Guide*.

Step 3 Add users to the user group and grant the new custom policy to the user group so that users in it can obtain the permissions defined by the policy.

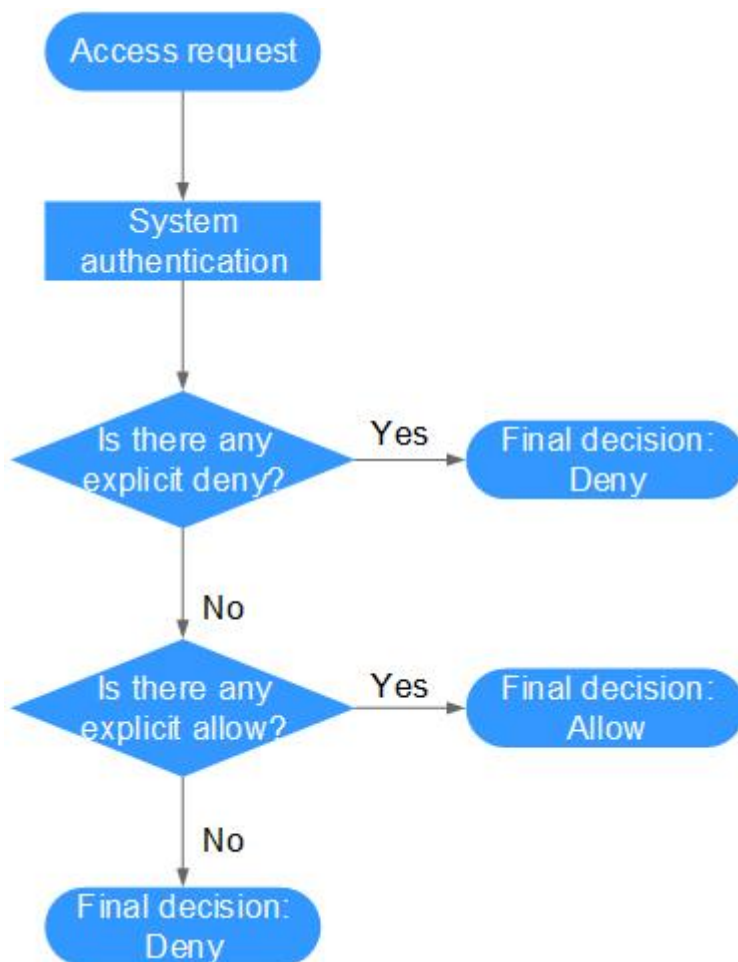
For details, see "User and User Group Management > Viewing or Modifying User Group Information" in the *Identity and Access Management User Guide*.

----End

Authentication Logic

If a user is granted permissions of multiple policies or of only one policy containing both Allow and Deny statements, then authentication starts from the Deny statements. The following figure shows the authentication logic for resource access.

Figure 15-3 Authentication logic



 **NOTE**

The actions in each policy bear the OR relationship.

1. A user accesses the system and makes an operation request.
2. The system evaluates all the permissions policies assigned to the user.
3. In these policies, the system looks for explicit deny permissions. If the system finds an explicit deny that applies, it returns a decision of Deny, and the authentication ends.
4. If no explicit deny is found, the system looks for allow permissions that would apply to the request. If the system finds an explicit allow permission that applies, it returns a decision of Allow, and the authentication ends.
5. If no explicit allow permission is found, IAM returns a decision of Deny, and the authentication ends.

16 Resource Management

16.1 Overview

The system resources (CPU, memory, I/O, and storage resources) of a database are limited. When multiple types of services (such as data loading, batch analysis, and real-time query) are running at the same time, they may compete for resources and hinder operations. As a result, the throughput decreases and the overall query performance deteriorates. To avoid this problem, resources must be properly allocated.

GaussDB(DWS) provides the resource management function. You can put resources into different resource pools, which are isolated from each other. Then, you can associate database users with these resource pools. When a user starts a SQL query, the query will be transferred to the resource pool associated with the user. You can specify the number of queries that can be concurrently executed in a resource pool, the upper limit of memory used for a single query, and the memory and CPU resources that can be used by a resource pool. In this way, you can limit and isolate the resources occupied by different workloads, properly utilizing resources to process hybrid database loads and achieve high query performance.

NOTICE

- This feature is supported only in 8.0 or later.
 - Resources cannot be managed during offline scale-out. If a resource management plan is enabled, stop it before performing offline scale-out.
-

Resource Management Functions

The resource management functions of GaussDB(DWS) can be classified into the following types based on managed resources:

- Computing resource management. It is implemented using resource pools. Computing resources are isolated and controlled to prevent cluster-level issues caused by abnormal SQL queries. Computing resource management includes concurrency management, memory management, CPU management, and exception rules. For details, see [Resource Pool](#).

- Storage space management: Storage is managed at user and schema level to prevent disk exhaustion, which makes the database read only. For details, see [Workspace Management](#).
- Resource management plan: Resources are managed automatically based on a preconfigured plan, which can flexibly cope with complex scenarios. For details, see [Importing or Exporting a Resource Management Plan](#).

The resource management functions of GaussDB(DWS) can be classified into the following types based on when they are implemented:

- Management before a query
The service checks whether there are sufficient resources for a query. If there are, the query can be executed. If there are not, the query waits in a queue, and can be executed only after resources are released by other queries. Concurrency and memory are managed in this phase.
- Management during a query
During query execution, resources used by the query are managed and controlled to prevent cluster exceptions caused by time-consuming SQL statements. Memory, CPU, storage space, and exception rules are managed in this phase.

Simple and Complex Queries

GaussDB(DWS) supports fine-grained resource management. Before workload management is implemented, queries are classified into complex queries (with long execution time and high resource consumption) and simple queries (with short execution time and low resource consumption). Simple and complex queries also differ in their estimated memory usage.

- The estimated memory usage of a simple query is less than 32 MB.
- The estimated memory usage of a complex query is 32 MB or higher.

In a hybrid load database, complex queries often occupy a large number of resources for a long time. A simple query queued after a complex query is time consuming, because it has to wait for the complex query to complete and resources to be freed up. To improve execution efficiency and system throughput, GaussDB(DWS) provides the short query acceleration function, managing simple queries separately.

- If short query acceleration is enabled, simple queries and complex queries are managed separately. Simple queries do not need to compete with complex queries for resources.
- If short query acceleration is disabled, simple and complex queries are under the same resource management rules.

To prevent a large number of simple queries from consuming too many resources during acceleration, concurrency management is performed on the queries. Resource management is not performed, because it may affect query performance and system throughput.

 NOTE

Queries are categorized based on estimated memory usage, but the estimation does not equal the actual usage, nor does it reflect the query duration or CPU usage. In resource pools that are insensitive to performance and only run specific services, you can disable short query acceleration to manage resources and handle exceptions for simple queries.

16.2 Resource Pool

16.2.1 Feature Description

GaussDB(DWS) resource pools provide concurrency management, memory management, CPU management, and exception rules.

Concurrency Management

Concurrency represents the maximum number of concurrent queries in a resource pool. Concurrency management can limit the number of concurrent queries to reduce resource contention and improve resource utilization.

The concurrency management rules are as follows:

- If short query acceleration is enabled, complex queries are under resource pool concurrency control, and simple queries are under short query concurrency control.
- If short query acceleration is disabled, complex and simple queries are both under resource pool concurrency control. Short query concurrency control is invalid.

Memory Management

Each resource pool occupies a certain percentage of memory.

Memory management aims to prevent out of memory (OOM) in a database, isolate the memory of different resource pools, and to control memory usage. Memory is managed from the following aspects:

- Global memory management
 - To prevent OOM, set the global memory upper limit (**max_process_memory**) to a proper value. Global memory management before a query controls memory usage to prevent OOM management. Global memory management during a query prevents errors during query execution.
 - Management before a query
 - The service checks the estimated memory usage of a query in the slow queue, and compares it with the actual usage. The estimation will be adjusted if it is smaller than the actual usage. Before a query is executed, the service checks whether the available memory is sufficient for the query. If yes, the query can be executed directly. If no, the query needs to be queued and executed after other queries release resources.
 - Management during a query

During a query, the service checks whether the requested memory exceeds a certain limit. If yes, an error will be reported, and memory occupied by the query will be released.

- Resource pool memory management

Resource pool memory management puts a limit on dedicated quotas. A workload queue can only use the memory allocated to it, and cannot use idle memory in other resource pools.

The resource pool memory is allocated in percentage. The value range is 0 to 100. The value **0** indicates that the resource pool does not perform memory management. The value **100** indicates that the resource pool performs memory management and can use all the global memory.

The sum of memory percentages allocated to all resource pools cannot exceed 100. Resource pool memory management is performed only before a query in the slow queue starts. It works in a way similar to the global memory management before a query. Before a query in the slow queue in a resource pool is executed, its memory usage is estimated. If the estimation is greater than the resource pool memory, the query needs to be queued and can be executed only after earlier queries in the pool are complete and resources released.

CPU Management

CPU share and CPU limit can be managed.

- CPU share: If the system is heavily loaded, CPU resources are allocated to resource pools based on the specific CPU shares. If the system not busy, this configuration does not take effect.
- CPU limit: It specifies the maximum number of CPU cores used by a resource pool. The resource usage of jobs in the resource pool cannot exceed this limit no matter whether the system is busy or not.

Choose either of the preceding management methods as needed. In CPU share management, CPUs can be shared and fully utilized, but resource pools are not isolated and may affect the query performance of each other. In CPU limit management, the CPUs of different resource pools are isolated, but this may result in the waste of idle resources.

 **NOTE**

The CPU usage limit is supported only by clusters of version 8.1.3 or later.

Exception Rules

To avoid query blocking or performance deterioration, you can configure exception rules to let the service automatically identify and handle abnormal queries, preventing slow SQL statements from occupying too many resources for a long time.

Exception Rule		Edit	
Blocking Time	Not limited	Execution Time	Not limited
Total CPU Time on All DNs	Not limited	Interval for Checking CPU Skew Rate	Not limited
Total CPU Time Skew Rate on All DNs	Not limited	Data Spilled to Disk Per DN	122820 MB
Average CPU Usage Per DN	50 %		

The following table describes exception rules.

Table 16-1 Exception rule parameters

Parameter	Description	Value Range (0 Means No Limit)	Operation
Blocking Time	Job blocking time. It refers to the total time spent in global and local concurrent queuing. The unit is second. For example, if the blocking time is set to 300s, a job executed by a user in the resource pool will be terminated after being blocked for 300 seconds.	An integer in the range 1 to 2,147,483,647 . The value 0 indicates no limit.	Terminated or Not limited
Execution Time	Time that has been spent in executing the job, in seconds. For example, if Time required for execution is set to 100s, a job executed by a user in the resource pool will be terminated after being executed for more than 100 seconds.	An integer in the range 1 to 2,147,483,647 . The value 0 indicates no limit.	Terminated or Not limited
Total CPU time on all DNs.	Total CPU time spent in executing a job on all DNs, in seconds.	An integer in the range 1 to 2,147,483,647 . The value 0 indicates no limit.	Terminated or Not limited
Interval for Checking CPU Skew Rate	Interval for checking the CPU skew, in seconds. This parameter must be set together with Total CPU Time on All DNs .	An integer in the range 1 to 2,147,483,647 . The value 0 indicates no limit.	Terminated or Not limited
Total CPU Time Skew Rate on All DNs	CPU time skew rate of a job executed on DNs. The value depends on the setting of Interval for Checking CPU Skew Rate .	An integer in the range 1 to 100. The value 0 indicates no limit.	Terminated or Not limited

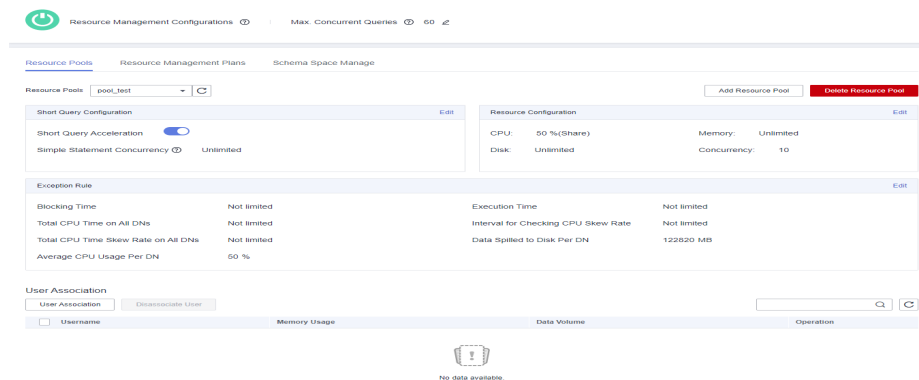
Parameter	Description	Value Range (0 Means No Limit)	Operation
Data Spilled to Disk Per DN	Allowed maximum job data spilled to disks on a DN. The unit is MB. NOTE This rule is supported only by clusters of version 8.2.0 or later.	An integer in the range 1 to 2,147,483,647. The value 0 indicates no limit.	Terminated or Not limited
Average CPU Usage Per DN	Average CPU usage of a job on each DN. If Interval for Checking CPU Skew Rate is configured, the interval takes effect for this parameter. If the interval is not configured, the check interval is 30 seconds by default. NOTE This rule is supported only by clusters of version 8.2.0 or later.	An integer in the range 1 to 100. The value 0 indicates no limit.	Terminated or Not limited

16.2.2 Page Overview

Overview

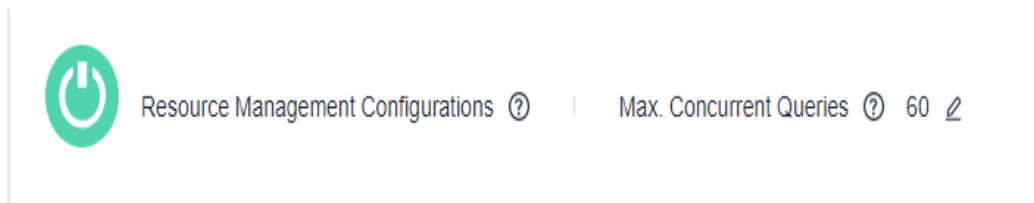
On the resource management page, you can modify global resource management configurations; add, create, and modify resource queues; add database users to a resource pool; and remove a database user from a resource pool. The page consists of the following modules:

- [Enabling or Disabling Resource Management](#)
- [Short Query Configuration](#)
- [Resource Configuration](#)
- [Exception Rule](#)
- [User Association](#)



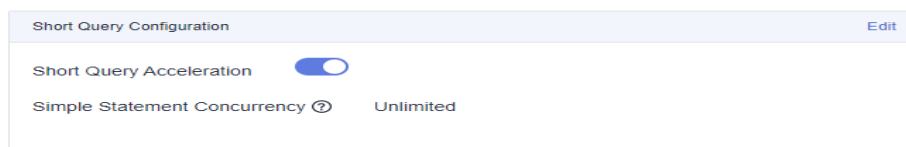
Enabling or Disabling Resource Management

You can enable or disable resource management, and configure the maximum global concurrency. **Max. Concurrent Queries** refers to the maximum concurrent queries on a single CN. If you disable **Resource Management**, all resource management functions will be unavailable.



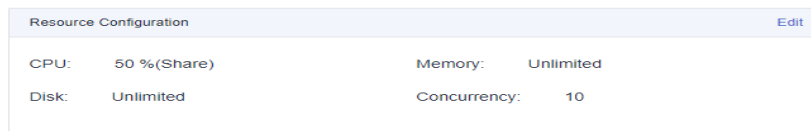
Short Query Configuration

In the **Short Query Configuration** area, you can enable or disable the short query acceleration function. To change the number of simple statements (**-1** by default. **0** or **-1** indicates that the concurrent short queries are not controlled), you can enable short query acceleration.



Resource Configuration

In the **Resource Configuration** area, you can view the resource configuration of the current workload queue.



Exception Rule

In the **Exception Rule** area, you can view the exception rule settings of the current resource pool. You can configure how job exceptions in the resource pool are handled. For more information, see [Exception parameters](#).

Exception Rule			
Blocking Time	Not limited	Execution Time	Not limited
Total CPU Time on All DNS	Not limited	Interval for Checking CPU Skew Rate	Not limited
Total CPU Time Skew Rate on All DNS	Not limited	Data Spilled to Disk Per DN	122820 MB
Average CPU Usage Per DN	50 %		

User Association

In the **User Association** list, you can view the associated users of the current resource pool, and the memory and disk usage of each user at the current time, as shown in the following figure.

User Association

User Association Disassociate User

Username	Memory Usage	Data Volume	Operation
<input type="checkbox"/> testuser0	0 MB	0 MB	Disassociate User
<input type="checkbox"/> testuser1	0 MB	0 MB	Disassociate User
<input type="checkbox"/> testuser11	0 MB	0 MB	Disassociate User
<input type="checkbox"/> testuser13	0 MB	0 MB	Disassociate User
<input type="checkbox"/> testuser14	0 MB	0 MB	Disassociate User
<input type="checkbox"/> testuser2	0 MB	0 MB	Disassociate User

NOTE

If no resource pools are associated with a user, the user will be associated with **default_pool** by default, and its resource usage will be restricted by **default_pool**. The **default_pool** will be automatically created after resource management is enabled.

16.2.3 Creating a Resource Pool

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** Choose **Clusters**. Click the name of a cluster.
- Step 3** Choose **Resource Management Configurations**.
- Step 4** Click **Add Resource Pool**.



NOTE

Up to 63 resource pools can be created.

- Step 5** Refer to [Table 16-2](#) to configure the resource pool.

Add Resource Pool

Name

CPU Resource (%) Share Limit

Memory Resource (%)

Storage Resource (MB) ?

Query Concurrency

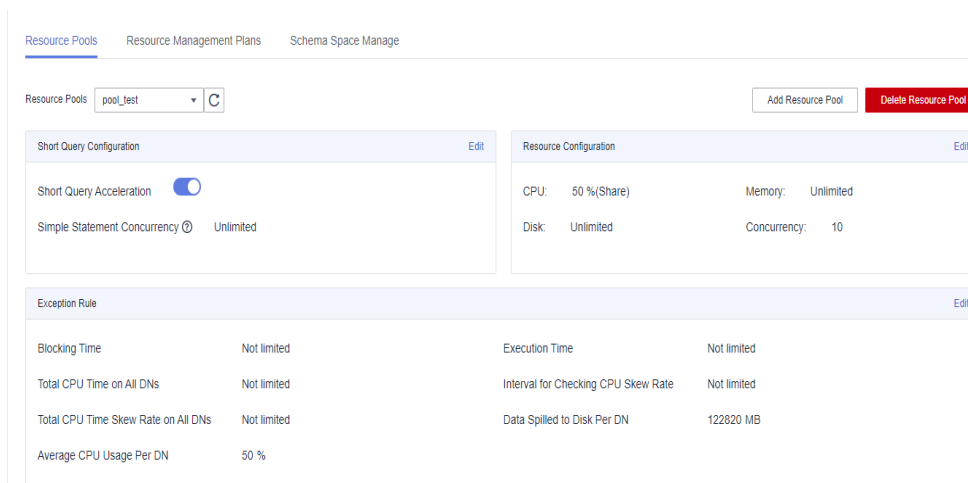
Table 16-2 Resource pool parameters

Parameter	Description	Mandatory	Default Value
Name	Resource pool name	Yes	-
CPU Resource (%)	<ul style="list-style-type: none"> • CPU share: Percentage of CPU time that can be used by users associated with the current resource pool to execute jobs. The value is an integer ranging from 1 to 99. • CPU limit: Maximum percentage of CPU cores used by a database user in a resource pool. The value is an integer ranging from 0 to 100. 0 indicates no limit. <p>NOTE</p> <ul style="list-style-type: none"> • The sum of the parameter values of all the resource pools cannot exceed 99%. If there is only one resource pool, the CPU share parameter does not take effect. • The CPU share parameter takes effect only when CPU contention occurs. For example, resource pools A and B are bound to CPU 1. If A and B are both running, the parameter takes effect. If there is only A running, the parameter does not take effect. • The sum of the CPU limits of all the resource pools cannot exceed 100%. The default value is 0. 	Yes	-
Memory Resource (%)	<p>Percentage of the memory that can be used by a resource pool.</p> <p>CAUTION You can manage memory and query concurrency separately or jointly. Under joint management, jobs can be delivered only when both the memory and concurrency conditions are met.</p>	Yes	0 (not limited)
Storage Resource (MB)	<p>Size of the available space for permanent tables.</p> <p>CAUTION This parameter indicates the total tablespace of all DNs in a resource pool. Available space of a single DN = Configured value/Number of DNs.</p>	Yes	-1 (not limited)
Query Concurrency	<p>Maximum number of concurrent queries in a resource pool.</p> <p>CAUTION You can manage memory and query concurrency separately or jointly. Under joint management, jobs can be delivered only when both the memory and concurrency conditions are met.</p>	Yes	10

NOTE

The CPU usage limit is supported only by clusters of version 8.1.3 or later.

Step 6 Confirm the information and click **OK**.

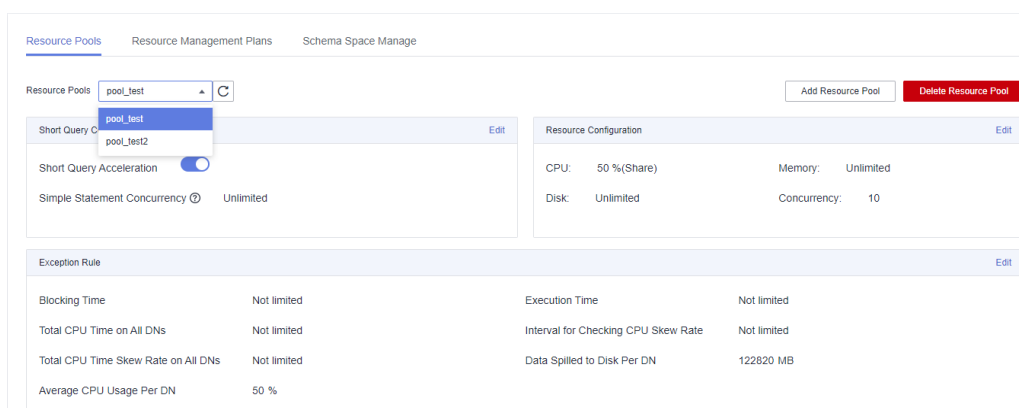


----End

16.2.4 Modifying a Resource Pool

You can modify the parameters of a resource pool on the resource management page.

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** Choose **Clusters**. Click the name of a cluster.
- Step 3** Choose **Resource Management Configurations**.
- Step 4** In the **Resource Pools** drop-down list, click the name of a resource pool. The following configuration areas are displayed, including **Short Query Configuration**, **Resource Configuration**, **Exception Rule**, and **Associated User**.



- Step 5** Modify the short query configuration. Set the parameters as required and click **Save** on the right.

Parameter	Description	Value
Short Query Acceleration	Whether to enable short query acceleration. This function is enabled by default.	Enable
Concurrent Short Queries	A short query is a job whose estimated memory used for execution is less than 32 MB. The default value -1 indicates that the job is not controlled.	10

Step 6 Modify the resource configuration.

1. Click **Edit** on the right and modify the parameters according to [Table 16-3](#).

Table 16-3 Resource pool parameters

Parameter	Description	Mandatory	Default Value
Name	Resource pool name	Yes	-
CPU Resource (%)	<ul style="list-style-type: none">- CPU share: Percentage of CPU time that can be used by users associated with the current resource pool to execute jobs. The value is an integer ranging from 1 to 99.- CPU limit: Maximum percentage of CPU cores used by a database user in a resource pool. The value is an integer ranging from 0 to 100. 0 indicates no limit. NOTE <ul style="list-style-type: none">- The sum of the parameter values of all the resource pools cannot exceed 99%. If there is only one resource pool, the CPU share parameter does not take effect.- The CPU share parameter takes effect only when CPU contention occurs. For example, resource pools A and B are bound to CPU 1. If A and B are both running, the parameter takes effect. If there is only A running, the parameter does not take effect.- The sum of the CPU limits of all the resource pools cannot exceed 100%. The default value is 0.- The CPU limit is supported only by clusters of version 8.1.3 or later.	Yes	-

Parameter	Description	Mandatory	Default Value
Memory Resource (%)	Percentage of the memory that can be used by a resource pool. CAUTION You can manage memory and query concurrency separately or jointly. Under joint management, jobs can be delivered only when both the memory and concurrency conditions are met.	Yes	0 (not limited)
Storage Resource (MB)	Size of the available space for permanent tables. CAUTION This parameter indicates the total tablespace of all DNs in a resource pool. Available space of a single DN = Configured value/Number of DNs.	Yes	-1 (not limited)
Complex Statement Concurrency	Maximum number of concurrent queries in a resource pool. CAUTION You can manage memory and query concurrency separately or jointly. Under joint management, jobs can be delivered only when both the memory and concurrency conditions are met.	Yes	10
Network Bandwidth Weight	Weight for network scheduling. The value is an integer ranging from 1 to 2147483647. The default value is -1. CAUTION Only cluster 8.2.1 and later versions support the network bandwidth weight.	Yes	-1 (not limited)

 **NOTE**

The CPU usage limit is supported only by clusters of version 8.1.3 or later.

2. Click **OK**.

Step 7 Modify the exception rules.

1. Modify rule parameters. See the following table for more information.

Table 16-4 Exception rule parameters

Parameter	Description	Value Range (0 Means No Limit)	Operation
Blocking Time	Job blocking time. It refers to the total time spent in global and local concurrent queuing. The unit is second. For example, if the blocking time is set to 300s, a job executed by a user in the resource pool will be terminated after being blocked for 300 seconds.	An integer in the range 1 to 2,147,483,647. The value 0 indicates no limit.	Terminated, Downgraded, or Not limited
Execution Time	Time that has been spent in executing the job, in seconds. For example, if Time required for execution is set to 100s, a job executed by a user in the resource pool will be terminated after being executed for more than 100 seconds.	An integer in the range 1 to 2,147,483,647. The value 0 indicates no limit.	Terminated, Downgraded, or Not limited
Total CPU time on all DNs.	Total CPU time spent in executing a job on all DNs, in seconds.	An integer in the range 1 to 2,147,483,647. The value 0 indicates no limit.	Terminated, Downgraded, or Not limited
Interval for Checking CPU Skew Rate	Interval for checking the CPU skew, in seconds. This parameter must be set together with Total CPU Time on All DNs .	An integer in the range 1 to 2,147,483,647. The value 0 indicates no limit.	Terminated, Downgraded, or Not limited
Total CPU Time Skew Rate on All DNs	CPU time skew rate of a job executed on DNs. The value depends on the setting of Interval for Checking CPU Skew Rate .	An integer in the range 1 to 100. The value 0 indicates no limit.	Terminated, Downgraded, or Not limited

Parameter	Description	Value Range (0 Means No Limit)	Operation
Data Spilled to Disk Per DN	Allowed maximum job data spilled to disks on a DN. The unit is MB. NOTE This rule is supported only by clusters of version 8.2.0 or later.	An integer in the range 1 to 2,147,483,647. The value 0 indicates no limit.	Terminated, Downgraded, or Not limited
Average CPU Usage Per DN	Average CPU usage of a job on each DN. If Interval for Checking CPU Skew Rate is configured, the interval takes effect for this parameter. If the interval is not configured, the check interval is 30 seconds by default. NOTE This rule is supported only by clusters of version 8.2.0 or later.	An integer in the range 1 to 100. The value 0 indicates no limit.	Terminated, Downgraded, or Not limited
Maximum Bandwidth on a Single DN	Maximum network bandwidth (MB) for a job on a single DN. NOTE This rule is supported only by clusters of version 8.2.1 or later.	An integer in the range 1 to 2,147,483,647. The value 0 indicates no limit.	Terminated, Downgraded, or Not limited

 **NOTE**

Exception rules allow you to control exceptions of jobs executed by users in a resource pool. Currently, you can configure the parameters listed in [Table 16-4](#).

- If you select **Terminate**, you need to set the corresponding time or percentage.
- If you select **No restriction**, the corresponding execution rule does not take effect.

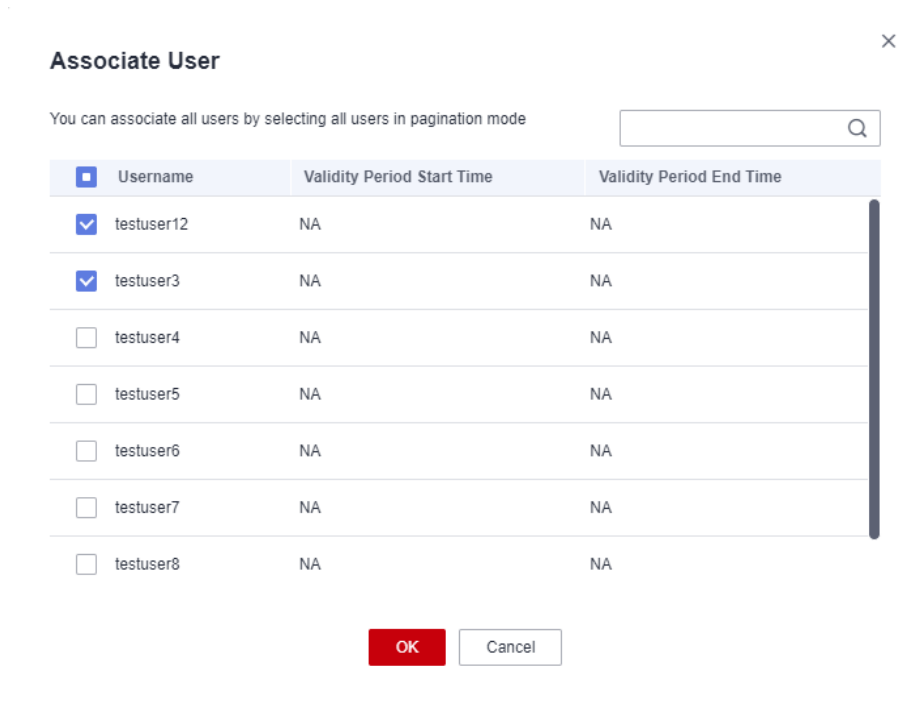
2. Click **Save**.

Step 8 Associate users. **NOTE**

- The resources used by a user to run jobs can be controlled only after the user is added to a resource pool.
- A database user can be added to only one resource pool. Users removed from a resource pool can be added to another pool.
- Database administrators cannot be associated.

1. Click **Add**.

2. Select the users to be added from the current user list. You can select multiple users at a time.



3. Click **OK**.
4. To remove a user, click **Disassociate User** in the **Operation** column of the user.

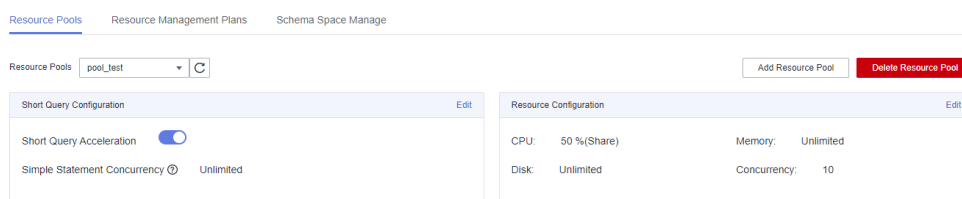
----End

16.2.5 Deleting a Resource Pool

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** Choose **Clusters**. Click the name of a cluster.
- Step 3** Choose **Resource Management Configurations**.
- Step 4** In the **Resource Pools** area on the left, click the name of a resource pool.
- Step 5** Click **Delete Resource Pool**.

NOTE

After a resource pool is deleted, the users (if any) associated with this pool will be associated with the default resource pool instead.



----End

16.3 Resource Management Plan

16.3.1 Managing Resource Management Plans

Overview

The resource management plan is an advanced resource management feature provided by GaussDB(DWS). You can create a resource management plan, add multiple stages to the plan, and configure different queue resource ratios for the stages. After a plan is started, it automatically changes the resource configurations in different stages as scheduled. If you need to run services in different stages with different proportions of resources, you can create a resource management plan to automatically change resource configurations in different stages.

Creating a Resource Management Plan

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** Choose **Clusters**. Click the name of a cluster.
- Step 3** Choose **Resource Management Configurations**.
- Step 4** Click to the **Resource Management Plans** tab and click **Add**.
- Step 5** Enter a plan name and click **OK**.

NOTICE

- Before creating a resource management plan, you must design and create a resource pool. For details, see [Creating a Resource Pool](#).
 - You can create up to 10 resource management plans.
-

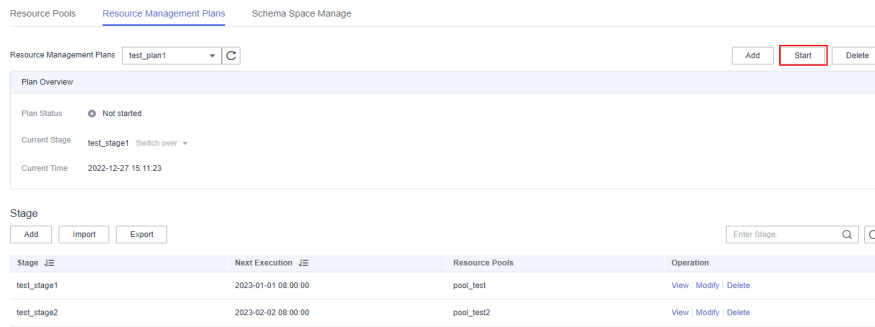
----End

Starting a Resource Management Plan

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** Choose **Clusters**. Click the name of a cluster.
- Step 3** Choose **Resource Management Configurations**.
- Step 4** Enter the plan details page and click **Start** to start a resource management plan.

NOTICE

- Only one plan can be started for each cluster.
 - A plan must have at least two stages before it can be started.
-

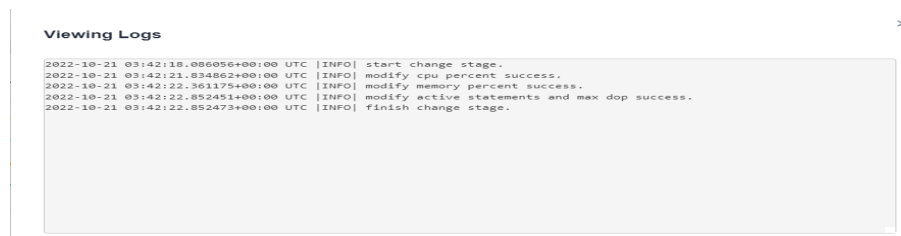


----End

Viewing the Execution Logs of a Resource Management Plan

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** Choose **Clusters**. Click the name of a cluster.
- Step 3** Choose **Resource Management Configurations**.
- Step 4** Go to the plan details page and view the switchover logs in the **Plan Execution Log** area.

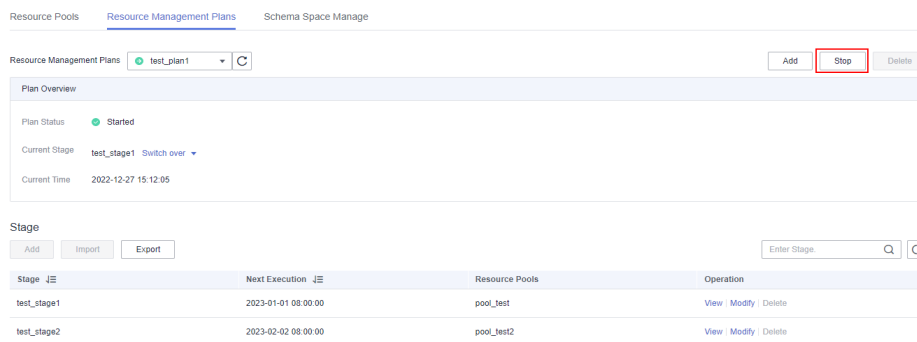
Execution Time	Stage Information	Result	Operation
2022-10-21 11:42:18	stage1	Succeeded	View



----End

Stopping a Resource Management Plan

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** Choose **Clusters**. Click the name of a cluster.
- Step 3** Choose **Resource Management Configurations**.
- Step 4** Enter the plan details page and click **Stop** to stop a resource management plan.



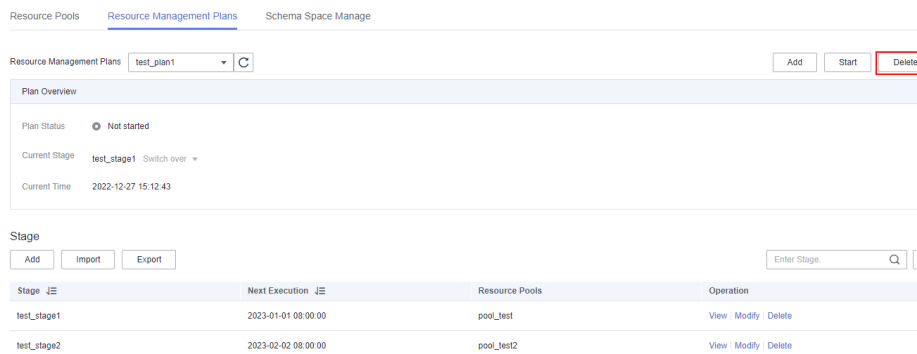
----End

Deleting a Resource Management Plan

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** Choose **Clusters**. Click the name of a cluster.
- Step 3** Choose **Resource Management Configurations**.
- Step 4** Enter the plan details page and click **Delete** to delete a resource management plan.

NOTICE

You cannot delete a running resource management plan.



----End

16.3.2 Managing Resource Management Plan Stages

Prerequisites

The following conditions must be met when you add or modify a resource management plan:

- The total CPU share of all resource pools does not exceed 99%.

- The total CPU limit of all resource pools does not exceed 100%.

 NOTE

- The CPU usage limit can be configured only in 8.1.3 and later versions.

Adding a Resource Management Plan Stage

Step 1 Log in to the GaussDB(DWS) management console.

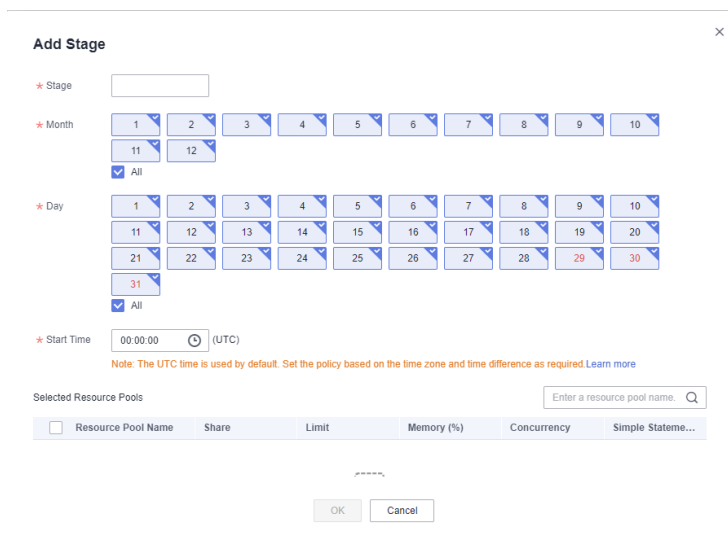
Step 2 Choose **Clusters**. Click the name of a cluster.

Step 3 Choose **Resource Management Configurations**.

Step 4 Go to the plan details page and click **Add** in the **Plan stage** area. On the **Add Stage** page, enter the stage name and configure the resource information. Confirm the configuration and click **OK**.

NOTICE

- Stages cannot be added to a running resource management plan.
- You can add a maximum of 48 stages for each plan.
- The switchover time of all phases in a plan cannot be the same.
- Configure the time, date, and month. Do not set an invalid date, for example, February 30.



Add Stage [Close]

* Stage

* Month

1	2	3	4	5	6	7	8	9	10
11	12								

All

* Day

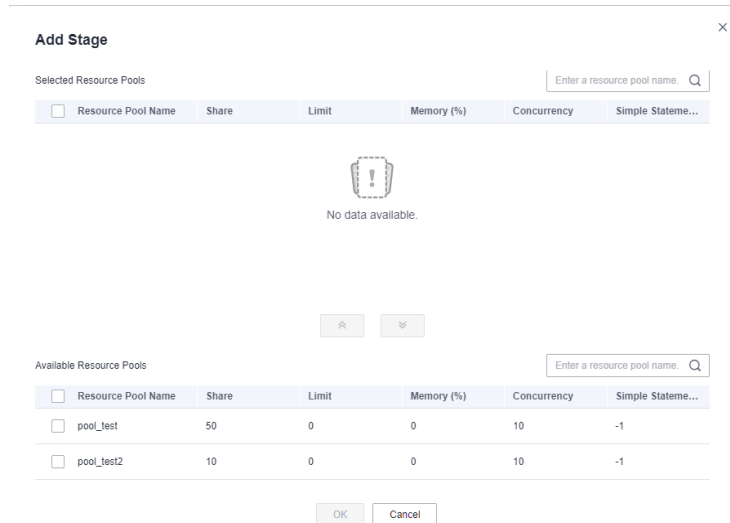
1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31									

All

* Start Time (UTC)
Note: The UTC time is used by default. Set the policy based on the time zone and time difference as required. [Learn more](#)

Selected Resource Pools

<input type="checkbox"/>	Resource Pool Name	Share	Limit	Memory (%)	Concurrency	Simple Stateme...



----End

Modifying a Resource Management Plan Stage

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** Choose **Clusters**. Click the name of a cluster.
- Step 3** Choose **Resource Management Configurations**.
- Step 4** Go to the plan details page and click **Modify** in the **Operation** column of the target plan stage.

Stage

Add Import Export

Enter Stage. Q C

Stage	Next Execution	Resource Pools	Operation
test_stage1	2023-01-01 08:00:00	pool_test	View Modify Delete
test_stage2	2023-02-02 08:00:00	pool_test2	View Modify Delete

- Step 5** Modify parameters, such as the stage changing time and resource configurations.

Modify Stage test_stage1 ×

★ Month: 1 2 3 4 5 6 7 8 9 10
11 12
 All

★ Day: 1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30
31
 All

★ Start Time: (UTC)
Note: The UTC time is used by default. Set the policy based on the time zone and time difference as required [Learn more](#)

Selected Resource Pools

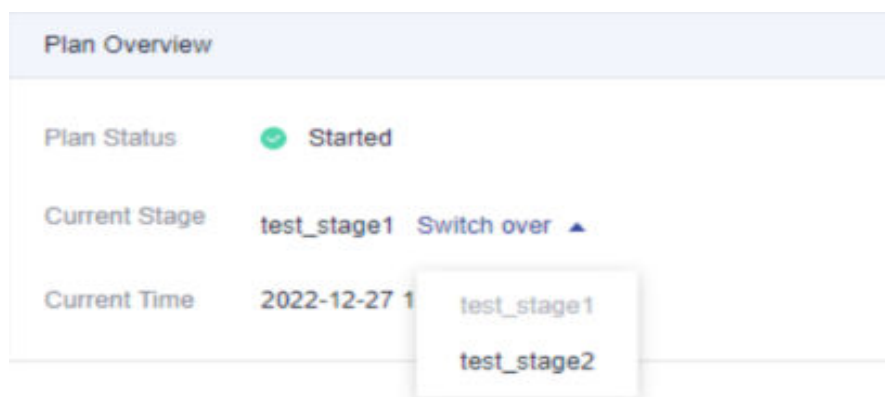
<input type="checkbox"/>	Resource Pool Name	Share	Limit	Memory (%)	Concurrency	Simple Stateme...
<input type="checkbox"/>	pool_test	<input type="text" value="50"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="10"/>	<input type="text" value="-1"/>

----End

Manually Changing the Resource Management Plan Stage

If a running plan needs to be switched to a stage in advance, you can manually do it.

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** Choose **Clusters**. Click the name of a cluster.
- Step 3** Choose **Resource Management Configurations**.
- Step 4** Go to the plan details page, click the **Switch over** button in the plan overview area, and select a stage.



----End

Deleting a Resource Management Plan Stage

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** Choose **Clusters**. Click the name of a cluster.
- Step 3** Choose **Resource Management Configurations**.
- Step 4** Go to the plan details page and click **Delete** in the **Operation** column of the target plan stage.



Stage	Next Execution	Resource Pools	Operation
test_stage1	2023-01-01 08:00:00	pool_test	View Modify Delete
test_stage2	2023-02-02 08:00:00	pool_test2	View Modify Delete

----End

NOTICE

Stages in a running resource management plan cannot be deleted.

16.3.3 Importing or Exporting a Resource Management Plan

Exporting a Resource Management Plan

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** Choose **Clusters**. Click the name of a cluster.
- Step 3** Choose **Resource Management Configurations**.
- Step 4** Enter the plan details page and click **Export** to export a resource management plan.



Stage	Next Execution	Resource Pools	Operation
test_stage1	2023-01-01 08:00:00	pool_test	View Modify Delete
test_stage2	2023-02-02 08:00:00	pool_test2	View Modify Delete

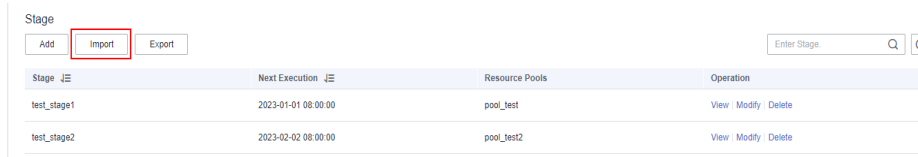
----End

Importing a Resource Management Plan

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** Choose **Clusters**. Click the name of a cluster.
- Step 3** Choose **Resource Management Configurations**.
- Step 4** Enter the plan details page, click **Import**, and select and import a configuration file to the resource management plan.

NOTICE

- Configurations cannot be imported to a running resource management plan.
- Ensure there is a resource pool before import.



The screenshot shows a web interface for managing stages. At the top, there are three buttons: 'Add', 'Import', and 'Export'. The 'Import' button is highlighted with a red rectangular box. To the right of these buttons is a search bar labeled 'Enter Stage' with a magnifying glass icon and a refresh icon. Below the buttons is a table with the following columns: 'Stage', 'Next Execution', 'Resource Pools', and 'Operation'. The table contains two rows of data:

Stage	Next Execution	Resource Pools	Operation
test_stage1	2023-01-01 08:00:00	pool_test	View Modify Delete
test_stage2	2023-02-02 08:00:00	pool_test2	View Modify Delete

----End

16.4 Workspace Management

Overview

Your cluster may run out of space if disk usage is not controlled, resulting in cluster exceptions and service interruption. Once disks are full, it takes long and huge efforts to recover workloads. Setting a database to read-only can reduce disk usage, but it also interrupts services. To solve this problem, GaussDB(DWS) provides multi-dimensional storage management. You can limit the permanent space that can be occupied by a schema; and can limit the usage of permanent space, temporary space, and operator space for a user.

- Schema level: Schema space management allows you to query database and schema space information in a cluster and modify the total schema space.
- User level: User space management allows you to limit users' space usage, preventing task execution from being blocked due to insufficient storage space. When you create a user in GaussDB(DWS), you can specify the space available to the user. The following types of storage space can be managed:
 - Permanent space (**PREM SPACE**)
Space occupied by permanent tables (non-temporary tables) created by users
 - Temporary space (**TEMP SPACE**)
Space occupied by temporary tables created by users
 - Operator spill space (**SPILL SPACE**)
During query execution, if the actual memory usage is greater than estimated, the query may be spilled to disks. The storage space occupied in this case is called operator spill space. You can control a user's operator spill space usage during query execution.

NOTE

- This feature is supported only in cluster version 8.1.1 or later.
- Currently, the GaussDB(DWS) management plane only supports schema space management.

Procedure

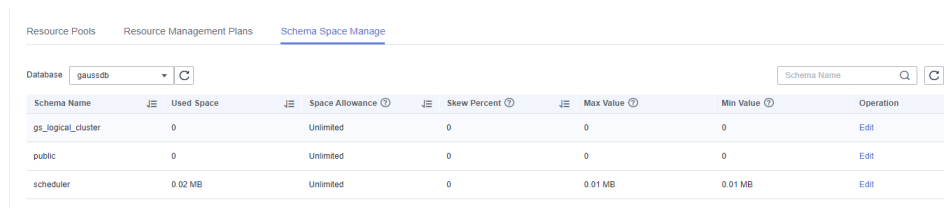
Step 1 Log in to the GaussDB(DWS) management console.

Step 2 Choose **Clusters**. Click the name of a cluster.

Step 3 Choose **Resource Management Configurations**.

Step 4 On the **Schema Space Manage** page, select a database.

Step 5 In the row where the scheme to be edited resides, click **Edit** and modify the space limit.



Schema Name	Used Space	Space Allowance	Skew Percent	Max Value	Min Value	Operation
gs_logical_cluster	0	Unlimited	0	0	0	Edit
public	0	Unlimited	0	0	0	Edit
scheduler	0.02 MB	Unlimited	0	0.01 MB	0.01 MB	Edit

Step 6 Click **OK**.

NOTE

- The space quota limits only common users but not database administrators. Therefore, when the used space is equal to the space limit, the actual used space may exceed the specified value.
- Quota for a single DN = Total quota/Number of DNs. Therefore, the configured value may be slightly different from the displayed value.

-----End

17 Data Source Management

17.1 MRS Data Sources

17.1.1 MRS Data Source Usage Overview

MRS Cluster Overview

MRS is a big data cluster running based on the open-source Hadoop ecosystem. It provides the industry's latest cutting-edge storage and analysis capabilities of massive volumes of data, satisfying your data storage and processing requirements. For details about MRS services, see the *MapReduce Service User Guide*.

You can use Hive/Spark (analysis cluster of MRS) to store massive volumes of service data. Hive/Spark data files are stored in HDFS. On GaussDB(DWS), you can connect a data warehouse cluster to MRS clusters, read data from HDFS files, and write the data to GaussDB(DWS) when the clusters are on the same network.

Operation Process

Perform the following operations to import data from MRS to a data warehouse cluster:

1. Prerequisites
 - a. Create an MRS cluster in a GaussDB(DWS) cluster. For details, see "Buying a Custom Cluster" in *MapReduce User Guide*.
 - b. Create an HDFS foreign table for querying data from the MRS cluster over APIs of a foreign server.

For details, see **Data Import > Importing Data from MRS to a Cluster** in the *Data Warehouse Service Database Development Guide*.

NOTE

- Multiple MRS data sources can exist on the same network, but one GaussDB(DWS) cluster can connect to only one MRS cluster at a time.

2. In the data warehouse cluster, create an MRS data source connection according to [Creating an MRS Data Source Connection](#).
3. Import data from an MRS data source to the cluster. For details, see "Data Import > Importing Data from MRS to a Data Warehouse Cluster".
4. (Optional) When the HDFS configuration of the MRS cluster changes, update the MRS data source configuration on GaussDB(DWS). For details, see [Updating the MRS Data Source Configuration](#).

17.1.2 Creating an MRS Data Source Connection

Scenario

Before GaussDB(DWS) reads data from MRS HDFS, you need to create an MRS data source connection that functions as a channel of transporting data warehouse cluster data and MRS cluster data.

Impact on the System

- You can create only one MRS data source connection in the data warehouse cluster at a time.
- When an MRS data source connection is being created, the system automatically adds inbound and outbound rules to security groups of the data warehouse cluster and MRS cluster. Nodes in the same subnet can be accessed.
- For the MRS cluster with Kerberos authentication enabled, the system automatically adds a **Machine-Machine** user that belongs to user group **supergroup** to the MRS cluster.

Prerequisites

- You have created a data warehouse cluster and recorded the VPC and subnet where the cluster resides.
- An MRS cluster of the analysis type has been created.

Procedure

Step 1 Log in to the management console.

Step 2 Choose **Service List > > MapReduce Service** to enter the MRS management console and create a cluster.

Configure parameters as required. For details, see "Cluster Operation Guide > Custom Creation of a Cluster" in the *MapReduce Service User Guide*.

- The VPC of the MRS cluster must be the same as that of the data warehouse cluster.
- MRS cluster versions 1.9.2, 2.1.0, 3.0.2-LTS, and 3.1.2-LTS are recommended.

NOTE

- For clusters of version 8.1.1.300 and later, MRS clusters support versions 1.6.*; 1.7.*; 1.8.*; 1.9.*; 2.0.*; 3.0.*; 3.1.*; and later (*indicates a number).
- For clusters earlier than version 8.1.1.300, MRS clusters support versions 1.6.*; 1.7.*; 1.8.*; 1.9.*; and 2.0.* (*indicates a number).

- Select the Hadoop component.

If you already have a qualified MRS cluster, skip this step.

Step 3 Choose **Service List > GaussDB(DWS)**.

Step 4 On the GaussDB(DWS) management console, choose **Clusters > Dedicated Clusters**.

Step 5 In the cluster list, click the name of a cluster. The **Cluster Information** page is displayed.

Step 6 In the navigation tree on the left, choose **Data Sources > MRS Data Sources**.

Step 7 Click **Create MRS Cluster Connection** and configure parameters.

Table 17-1 MRS common connection parameters

Parameter	Description
Data Source	GaussDB(DWS) database server name. It can contain 3 to 63 characters, including lowercase letters, numbers, and underscores (_), and must start with a lowercase letter.
Configuration Mode	The way in which the system obtains files. The options are as follows: <ul style="list-style-type: none">• MRS Account: Configure the username and password of the Manager of the MRS cluster. The system will log in to the Manager and automatically download configuration and verification files. For more information, see Table 17-2.• File upload: Download the configuration file from the Manager of the MRS cluster and manually upload it. You can use this method for Kerberos authentication. For more information, see Table 17-3. NOTE If you select File upload , ensure that MRS can communicate with the GaussDB(DWS) cluster.
Database	Database where the data source is located.
Description	Description of the connection.

Table 17-2 Parameters of the MRS Account mode

Parameter	Description
MRS Data Source	<p>Select an MRS cluster that can be connected to GaussDB(DWS) from the drop-down list box. By default, the custom, hybrid, and analytical MRS clusters that are in the same VPC and subnet as the current GaussDB(DWS) cluster and available to the current user are displayed.</p> <p>After you select an MRS cluster, the system automatically displays whether Kerberos authentication is enabled for the selected cluster. Click View MRS Cluster to view its detailed information.</p> <p>If the MRS Data Source drop-down list is empty, click Create MRS Cluster to create an MRS cluster.</p>
MRS Account	Account used when a GaussDB(DWS) cluster connects to an MRS cluster.
Password	<p>Password of the connection user. If you change the password, you need to create a connection again.</p> <p>NOTICE</p> <p>Ensure the account has been used for logging in to MRS Manager. If you use a new account, you will be asked to change your password when you first log in. In this case, the MRS data source will fail to be configured.</p>
Use a Machine-Machine Account	<p>Creates a machine-machine account named dws in MRS and uses it for interaction with MRS. This account is in the supergroup group and has all permissions. If the switch is toggled off, the configured man-machine account will be used. Ensure this account has the permission to access data, or a message will be displayed during data source access, indicating the required file does not exist.</p>

Table 17-3 Parameters of the File upload mode

Parameter	Description
Authentication Credential	<p>Keytab file of a user A credential file downloaded from Manager of the MRS cluster. File name format: Username_Timestamp_keytab.tar</p> <ul style="list-style-type: none">• For MRS 2.x or earlier, choose System > Manage User. In the Operation column of a user, choose More > Download authentication credential.• For MRS 3.x or later, choose System > Permission > User. In the Operation column of a user, choose More > Download Authentication Credential.

Parameter	Description
Client Profile	Client configuration files of HDFS, Hive, and hosts. When downloading the client, set Select Client Type to Configuration Files Only . <ul style="list-style-type: none">• For MRS 2.x or earlier, choose Services and click Download Client.• For MRS 3.x or later, choose Homepage. Click the More icon and choose Download Client.

Step 8 Click **OK** to save the connection.

Configuration Status turns to **Creating**. You can view the connection that is successfully created in the MRS data source list and the connection status is **Available**.

 **NOTE**

- In the **Operation** column, you can click **Update Configurations** to update **MRS Cluster Status** and **Configuration Status**. During configuration update, you cannot create a connection. The system checks whether the security group rule is correct. If the rule is incorrect, the system rectifies the fault. For details, see [Updating the MRS Data Source Configuration](#).
- In the **Operation** column, you can click **Delete** to delete the unnecessary connection. When deleting a connection, you need to manually delete the security group rule.
- If the security group rules are not deleted, nodes in the data warehouse cluster can still communicate with nodes in the MRS cluster. If you have strict requirements on network security, manually delete the rules.

----End

17.1.3 Updating the MRS Data Source Configuration

Scenario

For MRS, if the following parameter configurations of the HDFS cluster change, data may fail to be imported to the data warehouse cluster from the HDFS cluster. Before importing data using the HDFS cluster, you must update the MRS data source configuration.

Parameter	Description
dfs.client.read.shortcircuit	Specifies whether to enable the local read function.
dfs.client.read.shortcircuit.skip.hecksum	Specifies whether to skip data verification during the local read.
dfs.client.block.write.replace-datanode-on-failure.enable	Specifies whether to replace the location storing copies with the new node when data blocks fail to be written to HDFS.

Parameter	Description
dfs.encrypt.data.transfer	<p>Specifies whether to enable data encryption. The value true indicates that the channels are encrypted. The channels are not encrypted by default.</p> <p>NOTE</p> <ul style="list-style-type: none">This parameter is available only for clusters with Kerberos authentication enabled.This parameter is valid only when hadoop.rpc.protection is set to privacy.
dfs.encrypt.data.transfer.algorithm	<p>Specifies the encryption and decryption algorithm for key transmission.</p> <p>This parameter is valid only when dfs.encrypt.data.transfer is set to true.</p> <p>The default value is 3des, indicating that the 3DES algorithm is used for encryption.</p>
dfs.encrypt.data.transfer.cipher.suites	<p>Specifies the encryption and decryption algorithm for the transmission of actually stored data.</p> <p>If this parameter is not specified, the cryptographic algorithm specified by dfs.encrypt.data.transfer.algorithm is used for data encryption. The default value is AES/CTR/NoPadding.</p>
dfs.replication	<p>Specifies the default number of data copies.</p>
dfs.blocksize	<p>Specifies the default size of a data block.</p>
hadoop.security.authentication	<p>Specifies the security authentication mode.</p>
hadoop.rpc.protection	<p>Specifies the RPC communication protection mode.</p> <p>Default value:</p> <ul style="list-style-type: none">Security mode (Kerberos authentication enabled): privacyCommon mode (Kerberos authentication disabled): authentication <p>NOTE</p> <ul style="list-style-type: none">authentication: indicates that only authentication is required.integrity: indicates that authentication and consistency check need to be performed.privacy: indicates that authentication, consistency check, and encryption need to be performed.
dfs.domain.socket.path	<p>Specifies the locally used Domain socket path.</p>

Prerequisites

You have created an MRS data source connection for the data warehouse cluster.

Impact on the System

When you are updating an MRS data source connection, the data warehouse cluster will automatically restart and cannot provide services.

Procedure

- Step 1** On the GaussDB(DWS) management console, choose **Clusters > Dedicated Clusters**.
- Step 2** In the cluster list, click the name of a cluster. On the page that is displayed, click **MRS Data Sources**.
- Step 3** In the MRS data source list, select the MRS data source that you want to update. In the **Operation** column, click **Update Configurations**.

MRS Cluster Status and **Configuration Status** of the current connection will be updated. During configuration update, you cannot create a connection. The system checks whether the security group rule is correct. If the rule is incorrect, the system rectifies the fault.

----End

18 Managing Logical Clusters

18.1 Logical Cluster Overview

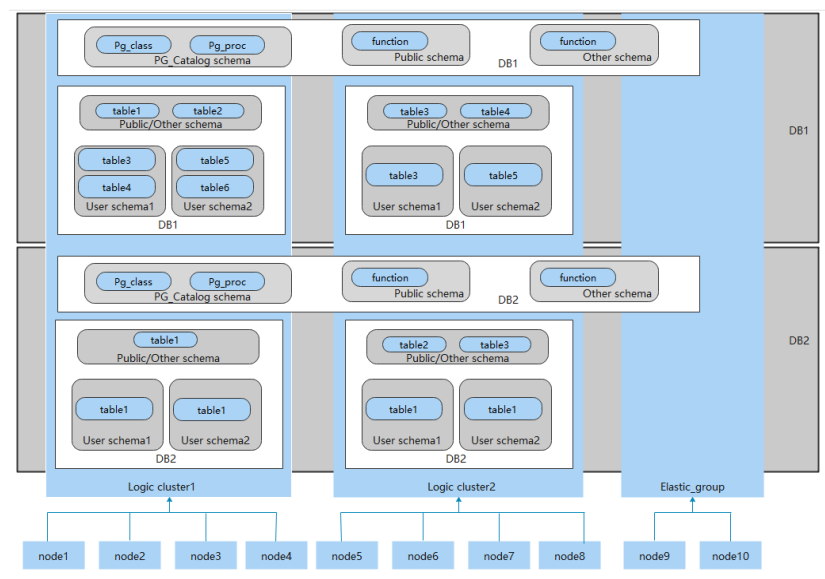
Concepts

A physical cluster can be divided into Node Groups, which are logical clusters. All physical nodes in a physical cluster are divided into multiple logical clusters. A logical cluster is essentially a node group that contains one or more physical nodes. Each physical node belongs to only one logical cluster, and user data tables can only be distributed within the same logical cluster. The data of each logical cluster is isolated from the others. The physical resources allocated to a logical cluster are mainly used for operations on its own data tables, but also for interactive queries with other logical clusters. An enterprise can deploy services on different logical clusters to implement unified service management, and meanwhile isolate the data and resources of services.

Logical clusters are created by dividing nodes of a physical cluster. Tables in a database can be allocated to different physical nodes by logical cluster. A logical cluster can contain tables from multiple databases. [Figure 18-1](#) shows the relationships between logical clusters, databases, and tables.

An elastic cluster is a cluster that always exists in logical cluster mode and consists of nodes that are not part of any logical cluster. It is a special node group that can have multiple or zero DNs. An elastic cluster cannot be manually created. When the first logical cluster is created in a physical cluster, an elastic cluster is also automatically created and all physical nodes not belonging to the logical cluster are automatically added to the elastic cluster. DNs in the elastic cluster will be used for logical clusters created later. To create a logical cluster, ensure that your logical cluster has DNs. (DNs are not required only when you create the first logical cluster in physical cluster mode.) You can add new physical nodes to the elastic cluster through scale-out.

Figure 18-1 Relationships between logical clusters, databases, and tables



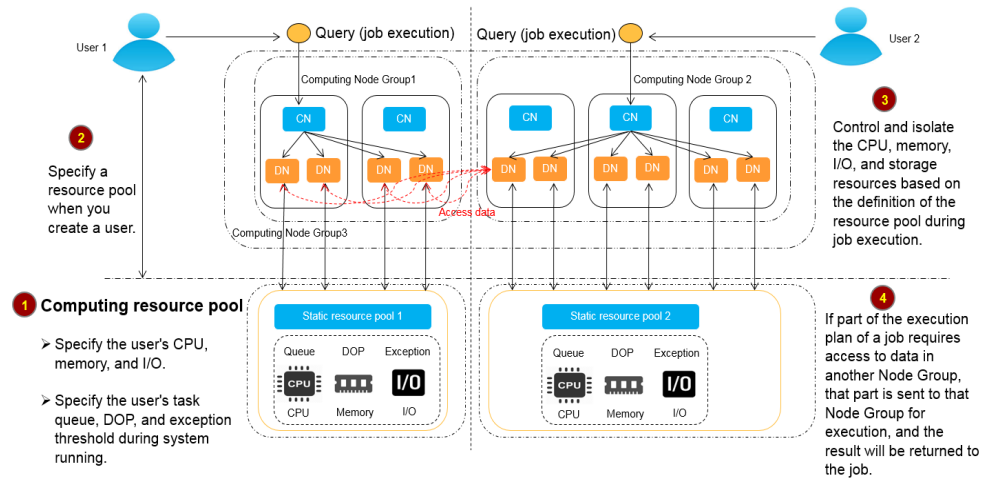
NOTE

- Logical clusters are supported in 8.1.0.100 or later.
- You are advised to allocate tables in a database to the same logical cluster.
- A logical cluster is not an independent sub-cluster. It can isolate data, resource, and permissions, but cannot be independently operated or maintained.
- The **Change all specifications** option does not support logical clusters.

Logical Cluster Architecture

Figure 18-2 shows the architecture of a physical cluster divided into multiple logical clusters. Nodes in the physical cluster are divided into Node Groups. The jobs of users 1 and 2 are executed in different Node Groups. The two users can define resource pools within their own logical cluster to control resources (CPU, memory, and I/O) used for different jobs. If some jobs of user 1 need to access the data of user 2, they can access data across Node Groups after being authorized. For a logical cluster, you can configure resources accessible across logical clusters to ensure its resources are sufficient.

Figure 18-2 Logical Cluster Architecture



A physical cluster is divided into multiple logical ones. You can define a resource pool for each of them based on service requirements. User tables are not distributed across logical clusters. If services do not access data across logical clusters, they will not compete for resources. Resources can be allocated to jobs in the same logical cluster by using resource pools. If necessary, you can let services access data across logical clusters, and control the resources used for such access to reduce resource competition between jobs within and outside a logical cluster.

After creating a physical cluster, you need to decide whether to divide it into logical clusters. You cannot divide it into logical clusters if you have already created user tables before, because these user tables are distributed on all physical nodes. For more information about the limitations, see [Constraints and Limitations](#). If you want to manage an existing cluster (for example, a database cluster built in a version earlier than 8.1.0.100) as a logical cluster, you can upgrade the cluster to 8.1.0.100 or later and then convert all the nodes in the cluster into a single logical cluster. Then, add nodes to the physical cluster and create another logical cluster on the new nodes.

Operations on logical clusters include:

- **Creating a logical cluster:** After converting a physical cluster into a logical cluster, you can group some physical nodes into a logical cluster by specifying the name and the nodes of the logical cluster.
- **Modifying a logical cluster:** You can add or remove nodes from a logical cluster as needed.
- **Resource management (logical cluster mode):** You can manage resources in a specified logical cluster (supported only by and later versions).
- **Scaling out a logical cluster:** This operation increases the number of physical nodes in the logical cluster and redistributes tables in the logical cluster to the new physical nodes.
- **Restarting a logical cluster:** This operation restarts all DNs in the logical cluster. Considering the impact on the entire physical cluster, the DNs in a logical cluster cannot be stopped or started individually.
- **Deleting a logical cluster:** You can delete a logical cluster with a specified name. After the logical cluster is deleted, the released physical nodes will be put in the elastic cluster.

Constraints and Limitations

- The smallest unit of the creation, scale-out, and scale-in of a logical cluster is a ring. A ring consists of at least three hosts, where the primary, standby, and secondary DN are deployed.
- During the logical cluster switchover, if the original physical cluster has data, the cluster will be locked. You can run simple DML statements, such as adding, deleting, modifying, and querying data. However, running complex DDL statements, such as operating database objects, will block services and report errors. Exercise caution when performing this operation.
- A logical cluster cannot be independently backed up or restored.
- A logical cluster cannot be independently upgraded.
- A physical cluster cannot be rolled back to a physical cluster after it is converted to a logical cluster.
- In logical cluster mode, only logical clusters can be created, and Node Groups cannot be created. In addition, Node Groups cannot be created in a logical cluster.
- O&M operations (creation, deletion, editing, scale-out, scale-in, and restart) of logical clusters cannot be performed concurrently.
- Public database objects (excluding system catalogs, foreign tables, and views) are distributed on all nodes in a physical cluster. After a node of the logical cluster is restarted, the DDL operations performed by other logical clusters on the objects will be interrupted.
- In logical cluster mode, each DN only contains the tables in the logical cluster that the DN belongs to. User-defined functions need to be created on all DNs. Therefore, **%type** cannot be used to reference table field types in the function body.
- In logical cluster mode, the **WITH RECURSIVE** statement cannot be pushed down.
- In logical cluster mode, partitions can be swapped only in the same logical cluster. Partitioned tables and common tables in different logical clusters cannot be swapped.
- In logical cluster mode, if the function parameters or return values contain table types, these table types must belong to the same logical cluster.
- In logical cluster mode, run the **CREATE TABLE...** command. When creating a foreign table using **LIKE**, ensure that the source table and the foreign table to be created are in the same logical cluster.
- In logical cluster mode, the **CREATE TABLE** statement cannot be used during creation of a schema (**CREATE SCHEMA...**). You need to create a schema first and then create a table in the schema.
- A logical cluster does not support the architecture of one primary node and multiple standby nodes. A logical cluster takes effect only in the architecture of one primary node, one standby node, and one secondary node.
- A logical cluster user cannot access the global temporary tables created by another logical cluster user.

Required permissions on tools

The following describes user permissions for database objects in logical clusters:

- The **CREATE ON NODE GROUP** permission can be granted to any user or role for performing operations such as creating tables in a logical cluster.
 - If the schema specified for a created table is a private schema of a user (that is, the schema has the same name as the user and the owner of the schema is the user), the owner of the created table defaults to the user. You do not need to associate the table with a logical cluster.
 - When a user associated with a logical cluster creates a table, if the **to group** clause is not specified, the table will be created in that logical cluster. The logical cluster associated with the user can be changed.
 - If a user is not associated with any logical cluster, when the user creates a table, the table will be created in the logical cluster specified by **default_storage_nodegroup**. If **default_storage_nodegroup** is set to **installation**, the table will be created in the first logical cluster. In logical cluster mode, the logical cluster with the smallest OID is set as the first logical cluster. If **default_storage_nodegroup** is not set, its value is **installation** by default.
 - The system administrator can run the **ALTER ROLE** command to set **default_storage_nodegroup** for each user. For details about the syntax, see "ALTER ROLE".
- Table creation rules
 - If **to group** is not specified for a user table but **default_storage_nodegroup** is set, tables will be created in the specified logical cluster.
 - If **default_storage_nodegroup** is set to **installation**, tables will be created in the first logical cluster, that is, the logical cluster with the smallest OID.
- The owner of a table can be changed to any user. However, you need to check the schema and node group permissions when performing operations on the table.
- A system administrator can be associated with a logical cluster and can create tables in multiple logical clusters.
 - If the system administrator is associated with a logical cluster and **to group** is not specified when you create a table, the table will be created in the associated logical cluster by default. If **to group** is specified, the table is created in the specified logical cluster.
 - If the system administrator is not associated with a logical cluster and **to group** is not specified, tables are created in the logical cluster of **default_storage_nodegroup**. For details, see the [table creation rules](#).
- System administrator permissions can be granted to a user associated with a logical cluster, but the [table creation rules](#) also apply.
- The logical cluster permission for accessing non-table objects (such as schemas/sequences/functions/triggers) will not be checked.
- A resource pool must be associated with a logical cluster.
 - A logical cluster can be associated with multiple resource pools but a resource pool can be associated with only one logical cluster.
 - Jobs executed by logical cluster users associated with a resource pool can only use resources in the resource pool.

- You do not need to create a workload group to define the number of concurrent jobs in a logical cluster. Therefore, workload groups are not required for logical clusters.
- When a logical cluster is deleted, only the table, foreign table, and resource pool objects are deleted.
 - Objects dependent on the tables (including the partly dependent sequences/functions/triggers) in the logical cluster will also be deleted.
 - Logical cluster associations with its users and parent-child tenants will be removed during the process. As a result, the users will be associated with the default **installation** node group and with the default global resource pool.
- A logical cluster user can create a database if granted the permission.

Replication Table Node Group

A replication table node group is a special node group in logical cluster mode. It can contain one or more logical clusters, but can only create replication tables. One typical scenario is to create public dimension tables. If multiple logical clusters require some common dimension tables, create a replication table node group and add the common dimension tables to it. The logical clusters contained in the replication table node group can access these dimension tables on the local DNs, with no need to access the tables on other DNs. If a logical cluster is scaled in, the replication table node group will be scaled accordingly. If the logical cluster is deleted, the replication table node group will be scaled in. However, if the replication table node group contains only one logical cluster and the logical cluster is deleted, the replication table node group will also be deleted. In this case, create tables in a logical cluster instead.

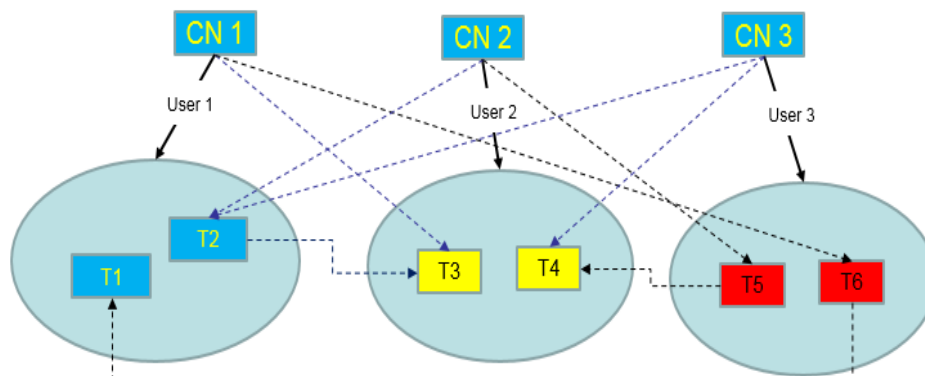
Create a replication table node group using the **CREATE NODE GROUP** SQL statement and delete one using **DROP NODE GROUP**. Before deleting a replication table node group, delete all table objects in the node group.

NOTE

Creation of replication table node groups is supported in 8.1.2 or later.

Application Scenarios

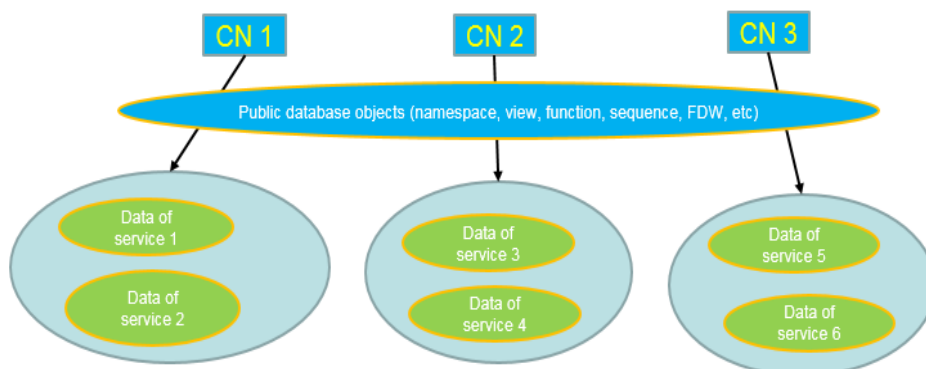
Scenario 1: Isolating data with different resource requirements

Figure 18-3 Logical cluster division based on resource requirements

As shown in the preceding figure, data with different resource requirements is stored in different logical clusters, and different logical clusters also support mutual access. This ensures that functions are not affected while resources are isolated.

- Tables T1 and T2 are used to calculate a large amount of data and generate report data (for example, bank batch processing). This process involves large batch import and big data query, which consume a lot of memory and I/O resources of nodes and take a long time. However, such a query does not require high real-time performance. Therefore, the data of these two tables can be separated into a different logical cluster.
- Tables T3 and T4 contain some computing data and real-time data, which are mainly used for point query and real-time query. These queries need high real-time performance. To prevent the interference of other high-load operations, the data of these two tables can be separated into a different logical cluster.
- Tables T5 and T6 are mainly used for OLTP operations with high concurrency. Data in these tables is frequently updated and sensitive to I/O. To prevent the impact of big data query on I/O, the data of these two tables can be separated into a different logical cluster.

Scenario 2: Isolating data for different services and enhancing the multi-tenancy of a data cluster

Figure 18-4 Logical cluster-based multi-service data and multi-tenant management

A large database cluster often stores data for various services. Each service has its own data tables. To allocate resources for different services, you can create multiple tenants. Specifically, assign different service users to different tenants to minimize resource contention among services. As the service scale grows continuously, the number of services in the cluster system also increases. Creating multiple tenants becomes less effective in controlling resource competition. Since each table is distributed across all DN of a database cluster, every data table operation may involve all DN, which increases network load and system resource consumption. Simply scaling up the cluster is not enough to solve this problem. Therefore, multiple logical clusters can be created to handle the increasing number of services, as shown in the figure above.

You can create a separate logical cluster and assign new services to it. This way, new services have little impact on existing services. Also, if the service scale in existing logical clusters grows, you can scale out the existing logical clusters.


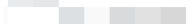

NOTE

A logical cluster is not suitable for managing multiple independent database systems. An independent database system requires independent O&M and needs to be managed, monitored, backed up, and upgraded separately. Moreover, faults must be isolated between clusters. Logical clusters cannot achieve independent O&M and complete fault isolation.

18.2 Adding Logical Clusters


- Step 1** Log in to the GaussDB(DWS) console. In the navigation pane, choose **Clusters > Dedicated Clusters**.
- Step 2** In the cluster list, click the name of the target cluster. The **Cluster Information** page is displayed.
- Step 3** Enable **Logical Clusters**. The **Logical Clusters** menu item will be displayed in the navigation pane on the left.

Basic Information


Cluster	 3	Cluster ID	 99
Cluster Status	✔ Available	Cluster Version	8.2.0
Parameter Configuration Status	✔ Synchronized	Created	Sep 26, 2022 11:17:13 GMT+08:00
Task Information	–	Node Flavor	dwsk.xlarge
Current Specifications	Cloud 4 vCPUs 32 GB Memory 200 GB Ultra-high I/O	Maintenance Window 	Friday 06:00-10:00 GMT+08:00 Settings
Nodes	3	Logical Clusters	<input type="checkbox"/>

Step 4 Go to the **Logical Clusters** tab and click **Add Logical Cluster**.

Logical Cluster Management + Add Logical Cluster

Cluster Name	Status	Task Info	Storage	Nodes	Operation
 No data available.					


Operation History All

Operation	Cluster Name	Start Time	End Time	Result	Action Log
 No data available.					

Step 5 Move the ring you want to add from the right to the left panel, enter the logical cluster name, and click **OK**.

Add Logical Cluster ×

Logical Cluster Name <input type="text"/>	Selected Host Ring <input type="text"/>
---	---

Host Ring	CPU	RAM	Storage	Nodes
 No data available.				

Available Elastic Cluster Host Ring				
Host Ring	CPU	RAM	Storage	Nodes
<input type="checkbox"/> host-172-16-36-196.ho...	12	96	480	3

<< >>

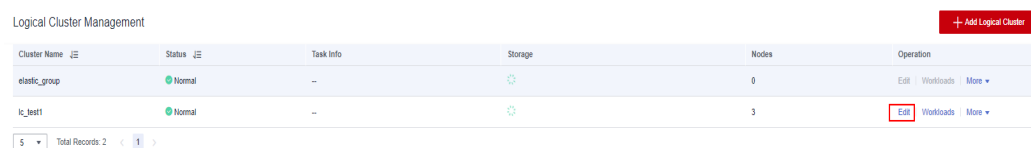
----End

 **CAUTION**

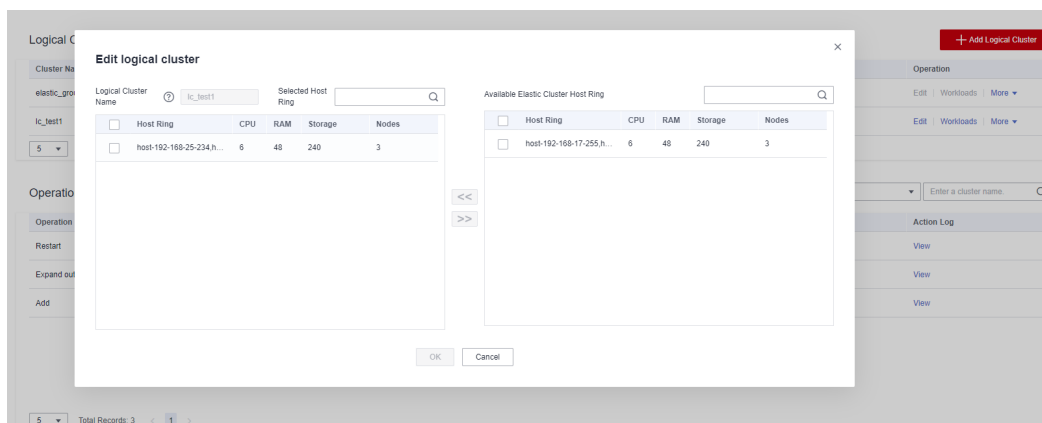
- If you access the **Logical Clusters** page for the first time, the metadata of the logical cluster created at the backend is synchronized to the frontend. After the synchronization is complete, you can view information about the logical clusters at the frontend. The logical cluster name is case sensitive. For example, metadata of **lc1** and **LC1** cannot be synchronized.
- During the conversion from a physical cluster to a logical cluster, the original resource pool configuration will be cleared. The resource pool information configured after the cluster is converted to a logical cluster will be bound to the logical cluster.

18.3 Editing Logical Clusters

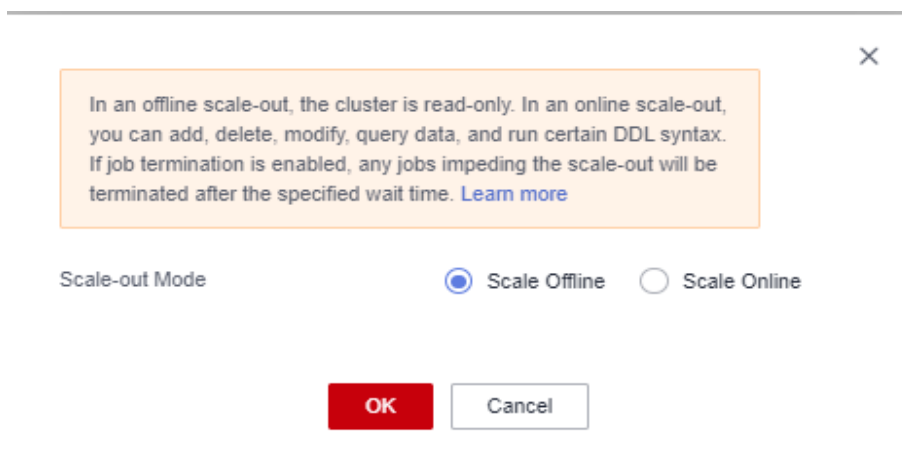
- Step 1** Log in to the GaussDB(DWS) console. In the navigation pane, choose **Clusters > Dedicated Clusters**.
- Step 2** In the cluster list, click the name of the target cluster. The **Cluster Information** page is displayed.
- Step 3** In the navigation pane, choose **Logical Clusters** and click **Edit** in the **Operation** column of the target cluster.



- Step 4** Add a node to the logical cluster by moving the selected ring from the right to the left, or remove a node from the logical cluster by moving the selected ring from the left to the right, and click **OK**.



- Step 5** When adding a node, select online or offline scale-out as needed.



----End

 NOTE

- Nodes are added to or removed from a logical cluster by ring.
- At least one ring must be reserved in a logical cluster.
- The ring removed from the logical cluster will be added to the elastic cluster.
- Logical clusters of version 8.1.3 and later support online scale-out.

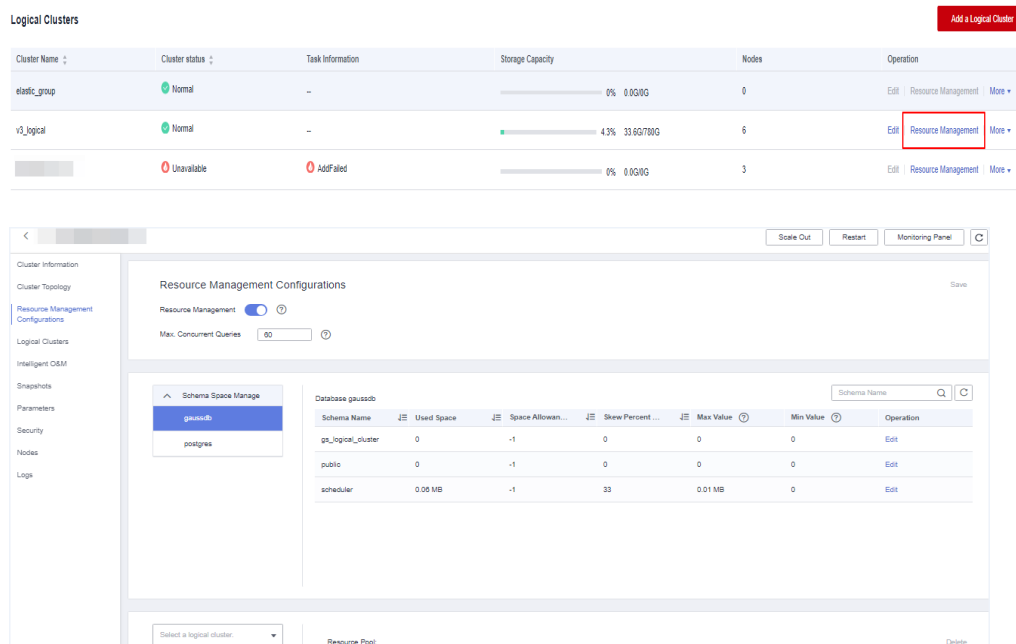
18.4 Managing Resources (in a Logical Cluster)

Precautions

The original resource pool configuration is cleared when the cluster is converted from physical to logical. You have to add the resource pool again if you want to configure it after the conversion.

Procedure

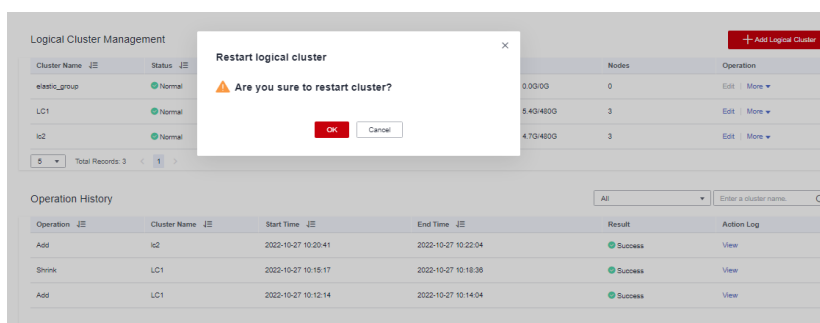
- Step 1** Log in to the GaussDB(DWS) console. In the navigation pane, choose **Clusters > Dedicated Clusters**.
- Step 2** In the cluster list, click the name of the target cluster. The **Cluster Information** page is displayed.
- Step 3** In the navigation pane, choose **Logical Cluster Management**. In the **Operation** column of a logical cluster, click **Resource Management Configurations**. On the displayed page, you can manage resources in a logical cluster. For details, see [Resource Management](#).



----End

18.5 Restarting Logical Clusters

- Step 1** Log in to the GaussDB(DWS) console. In the navigation pane, choose **Clusters > Dedicated Clusters**.
- Step 2** In the cluster list, click the name of the target cluster. The **Cluster Information** page is displayed.
- Step 3** In the navigation pane, choose **Logical Clusters**. Click **Restart** in the **Operation** column of the target cluster, and click **OK** in the displayed dialog box.



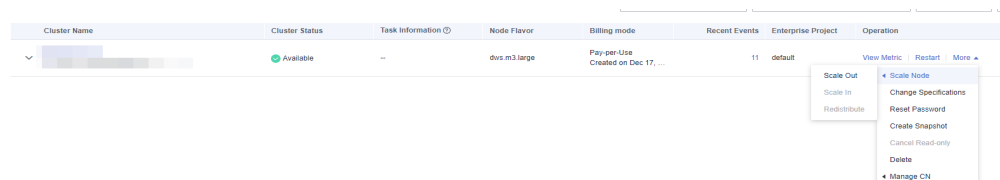
----End

18.6 Scaling Out Logical Clusters

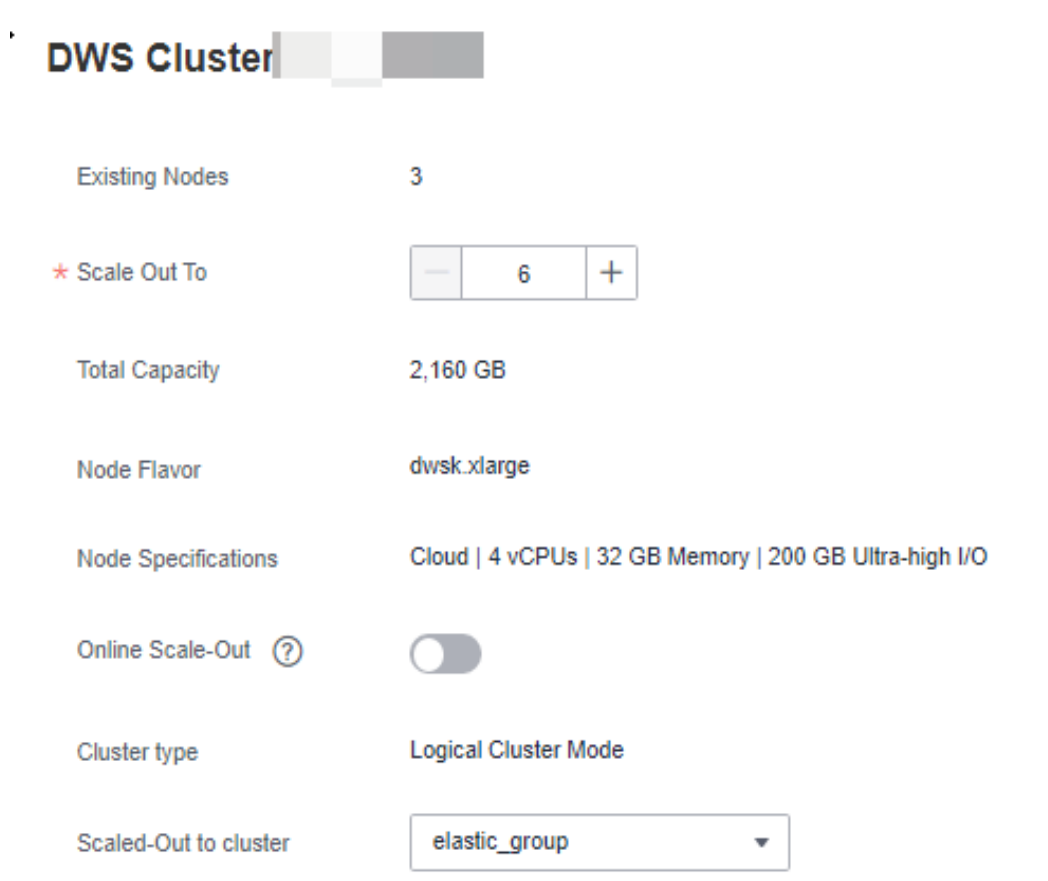
NOTICE

- Logical clusters of version 8.1.3 and later support online scale-out.
- Before a scale-out, you need to enable the logical cluster mode and add a logical cluster.
- After scaling out or scaling in a logical cluster, you need to reconfigure the backup policy for full backup. For details, see [Configuring an Automated Snapshot Policy](#).

- Step 1** Log in to the GaussDB(DWS) console. In the navigation pane, choose **Clusters > Dedicated Clusters**.
- Step 2** On the displayed **Clusters** page, choose **More > Scale Node > Scale Out**.



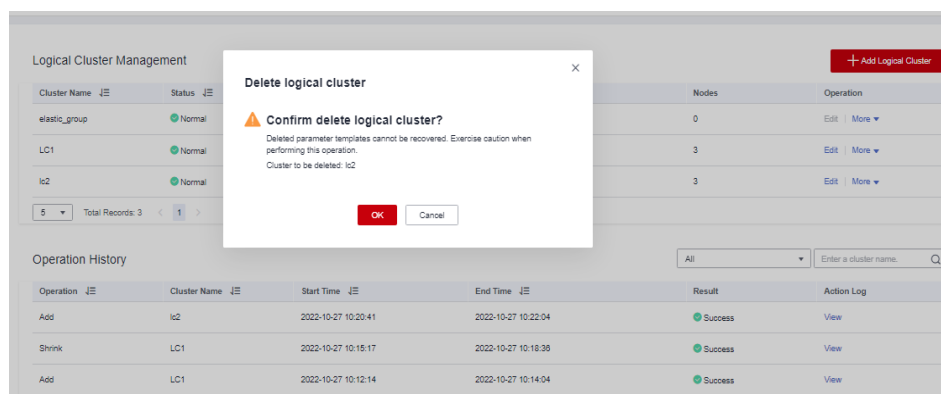
- Step 3** On the scale-out page, select a logical or elastic cluster..



----End

18.7 Deleting Logical Clusters

- Step 1** Log in to the GaussDB(DWS) console. In the navigation pane, choose **Clusters** > **Dedicated Clusters**.
- Step 2** In the cluster list, click the name of the target cluster. The **Cluster Information** page is displayed.
- Step 3** In the navigation pane, choose **Logical Clusters**. Click **Delete** in the **Operation** column of the target cluster, and click **OK** in the displayed dialog box.



----End

NOTICE

- The first added logical cluster cannot be deleted.
- Nodes of the deleted logical cluster are added to the elastic cluster.

18.8 Tutorial: Converting a Physical Cluster That Contains Data into a Logical Cluster

Scenario

A large database cluster usually contains a large amount of data put in different tables. With the feature, you can create resource pools to isolate the resources of different services. Different service users can be allocated to different resource pools to reduce resource (CPU, memory, I/O, and storage) competition between services.

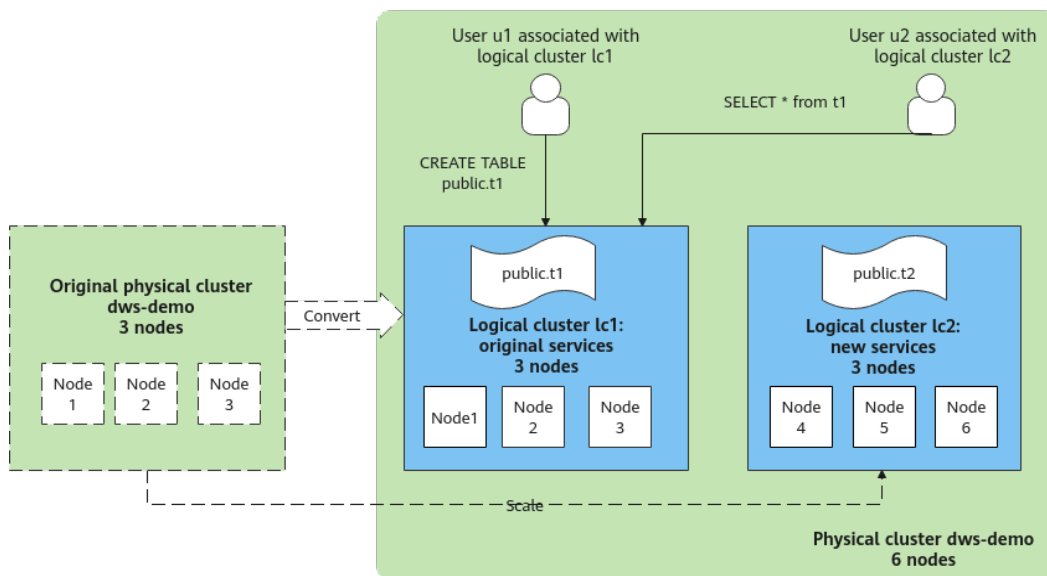
As the service scale grows, the number of services in the cluster system also increases. Creating multiple resource pools becomes less effective in controlling resource competition. GaussDB(DWS) uses the distributed architecture, and its data is distributed on multiple nodes. Each table is distributed across all DNs in the cluster, an operation on a data table may involve all DNs, which increases network loads and system resource consumption. To solve this problem, scale-out is not effective. You are advised to divide a GaussDB(DWS) cluster into multiple logical clusters.

You can create a separate logical cluster and assign new services to it. This way, new services have little impact on existing services. Also, if the service scale in existing logical clusters grows, you can scale out the existing logical clusters.

Figure 18-5 shows an example. The original service data tables of a company are stored in the original physical cluster **dws-demo** (in green). After services are switched over to the logical cluster **lc1** (in blue), a new logical cluster **lc2** is added to the physical cluster through scale-out. The original service data tables are switched to logical cluster **lc1**, and new service data tables are written to logical cluster **lc2**. In this way, the data of old and new services is isolated. User **u2** associated with logical cluster **lc2** can access the tables of logical cluster **lc1** across logical clusters after authorization.

- **Cluster scale:** Scale out the original physical cluster from three nodes to six nodes and split it into two logical clusters.
- **Service isolation:** New and old service data is isolated in different logical clusters.

Figure 18-5 Accessing data across logical clusters



Creating a Cluster and Preparing Table Data

- Step 1** Create a cluster. For details, see [Creating a Cluster](#).
- Step 2** After connecting to the database, create table **t1** as the system administrator **dbadmin** and insert two data records into the table.

```
CREATE TABLE t1 (id int, name varchar(20));
INSERT INTO t1 VALUES (1,'joy'),(2,'lily');
```

----End

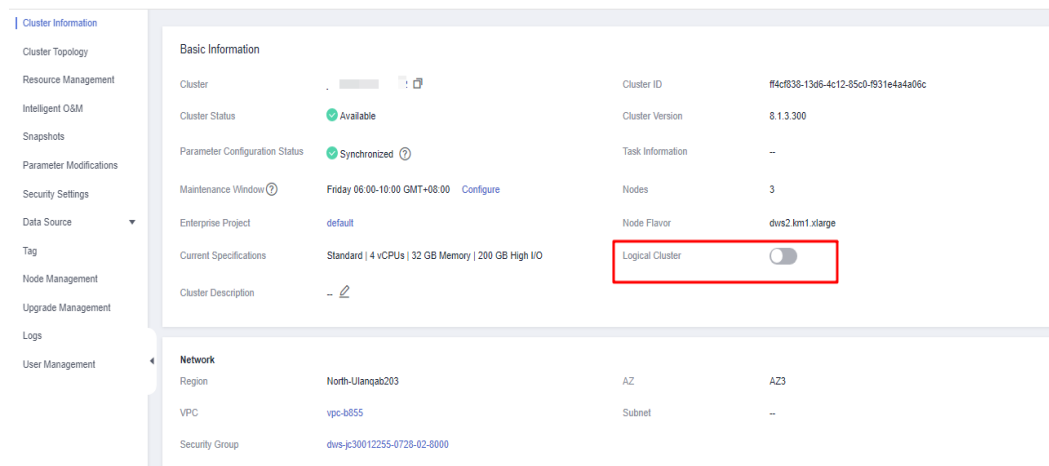
Converting to Logical Cluster lc1

NOTICE

During the conversion, you can run simple DML statements, such as adding, deleting, modifying, and querying data. Complex DDL statements, such as operations on database objects, will block services. You are advised to perform the conversion during off-peak hours.

- Step 1** Log in to the GaussDB(DWS) console. In the navigation pane, choose **Clusters > Dedicated Cluster**. Click the name of a cluster to go to the **Cluster Information** page.
- Step 2** Toggle on the **Logical Cluster** switch.

Figure 18-6 Enabling the logical cluster function



Step 3 In the navigation pane, choose **Logical Clusters**. Click **Add Logical Cluster** in the upper right corner, enter the logical cluster name **lc1**, and click **OK**.

During the switchover, the current cluster is unavailable. Wait for about 2 minutes (the conversion time varies depending on the service data volume). If **lc1** is displayed on the logical cluster page, the conversion is successful.

Figure 18-7 Adding a logical cluster

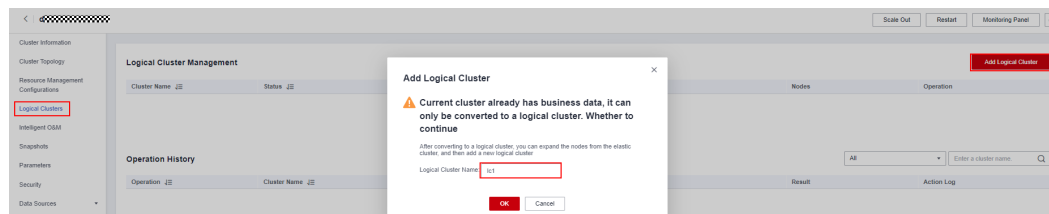
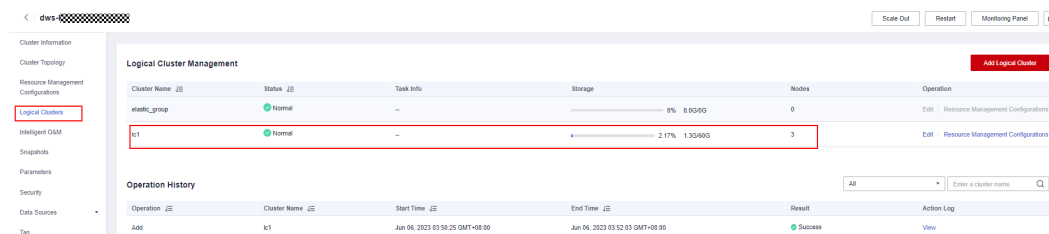


Figure 18-8 Logical cluster conversion succeeded



----End

Adding Nodes to the elastic_group Cluster

Step 1 Return to the **Cluster Management** page. In the **Operation** column of the cluster, choose **More > Scale Node > Scale Out**.

Figure 18-9 Scaling out a cluster

Cluster Name	Cluster Status	Task Information	Node Flavor	Billing Mode	Recent Events	Enterprise Project	Operation
[Cluster Name]	Available	--	dws2.2xlarge.m7	Pay-per-Use	Created on Jul 26, 2023 1...	6	Log In Monitoring Panel More
[Cluster Name]	Available	--	dws2.kn1.xlarge	Pay-per-Use	Created on Jul 26, 2023 1...	9	Log I Change to Yearly/Monthly
[Cluster Name]	Available	--	dws.xlarge.4	Pay-per-Use	Created on Jul 26, 2023 1...	9	Scale Out Scale Nodes View Metric
[Cluster Name]	Available	--	dws2.kn1.xlarge	Pay-per-Use	Created on Jul 26, 2023 1...	8	Restart Change Specifications
[Cluster Name]	Available	--	dws.xlarge	Pay-per-Use	Created on Jul 26, 2023 1...	28	Log I Reset Password Create Snapshot

Step 2 Set **New Nodes** to **3**. Enable **Online Scale-out**. Set **elastic_group** as the target logical cluster. Confirm the settings, select the confirmation check box, and click **Next: Confirm**.

Figure 18-10 Scale-out process

DWS Cluster dws[Cluster Name]

Current Nodes: 3

* New Nodes: You can create 1021 more nodes. [Increase quota](#)

Total Capacity: 600 GB

Node Flavor: dwsk2.xlarge

Node Specifications: Standard | 4 vCPUs | 32 GB Memory | 20 GB Ultra-high I/O

Online Scale-out:

Discount Nodes: -- [Buy Discount Package](#) [View Order](#)

Cluster type: Logical Cluster Mode

Scaled-Out to cluster:

Billing mode: Pay-per-use

Note: 1. If you are using a discount package, you need to unsubscribe from the package and purchase a new discount package after the capacity or number of nodes is adjusted. Otherwise, pay-per-use billing may occur.

I agree

Step 3 Click **Next: Confirm**, and then click **OK**.

Wait for about 10 minutes until the scale-out is successful.

----End

Adding Logical Cluster lc2

Step 1 On the **Cluster Management** page, click the name of a cluster to go to the cluster details page. In the navigation pane, choose **Logical Clusters**.

Step 2 Click **Add Logical Cluster** in the upper right corner, select three nodes from the right pane to add to the left pane, enter the logical cluster name **lc2**, and click **OK**.

After about 2 minutes, the logical cluster is successfully added.

Figure 18-11 Adding a logical cluster

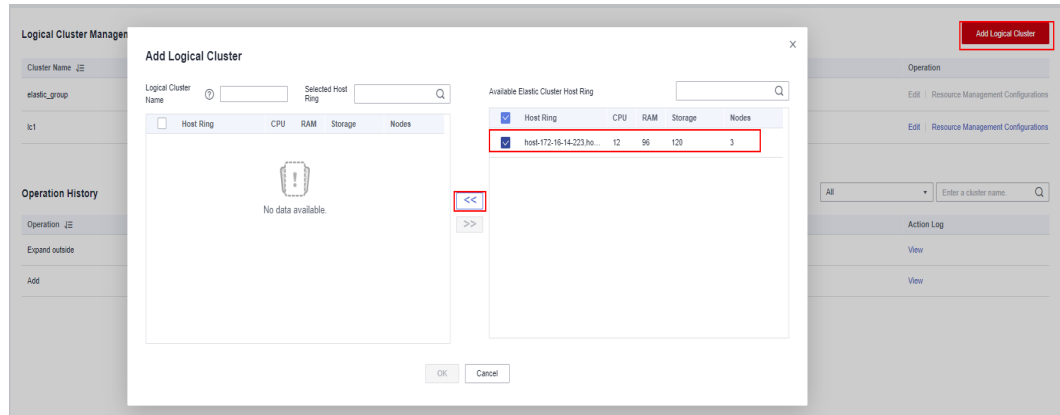


Figure 18-12 Selecting a host ring

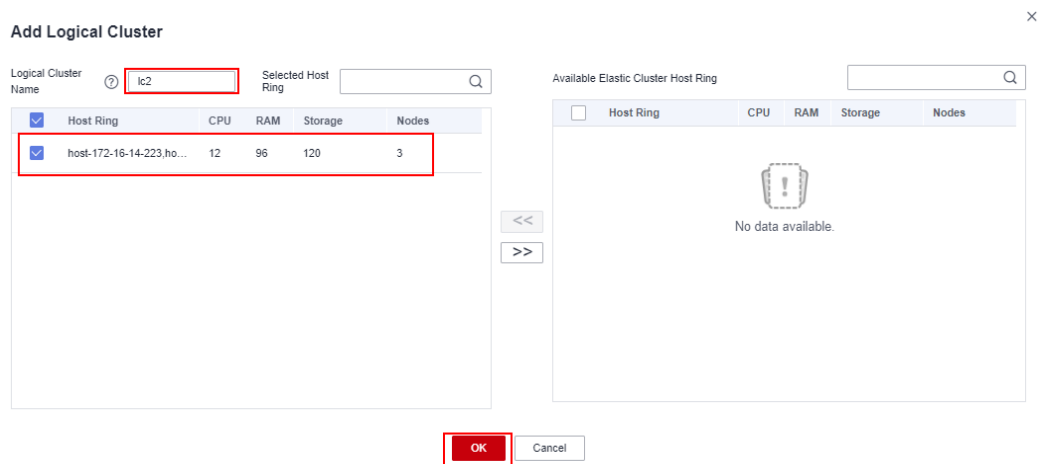


Figure 18-13 Logical cluster added

Cluster Name	Status	Task Info	Storage	Nodes	Operation
elastic_group	Normal	-	0% 0.0G/0G	3	Edit Resource Management Configurations
lc1	Normal	-	2.13% 1.3G/60G	3	Edit Resource Management Configurations
lc2	Normal	-	1.53% 0.9G/60G	3	Edit Resource Management Configurations

----End

Creating Logical Clusters, Associating Them with Users, and Querying Data Across Logical Clusters

Step 1 Connect to the database as the system administrator and run the following SQL statement to query the original service table **t1**:

Verify that service data can be queried after the conversion.

```
SELECT * FROM t1;
```


- Step 2** Run the following statements to associate **u1** with logical cluster **lc1** and **u2** with logical cluster **lc2**, and grant all permissions of the original service table **t1** to user **u1**:

```
CREATE USER u1 NODE GROUP 'lc1' password '{password}';  
CREATE USER u2 NODE GROUP 'lc2' password '{password}';  
GRANT ALL ON TABLE t1 TO u1;
```

- Step 3** Switch to user **u2** and query data in the original service table **t1**. A message is displayed, indicating that you do not have the permission to access logical cluster **lc1**. This indicates data is isolated between logical clusters.

```
SET ROLE u2 PASSWORD '{password}';  
SELECT * FROM t1;
```



- Step 4** Switch back to system administrator **dbadmin** and grant the access permission of logical cluster **lc1** to user **u2**.

```
SET ROLE dbadmin PASSWORD '{password}';  
GRANT USAGE ON NODE GROUP lc1 TO u2;
```

- Step 5** Switch to user **u2** and query the **t1** table. This proves that the user bound to logical cluster **lc2** can query the original service table **t1** across logical clusters. In this way, data is shared between logical clusters.

```
SET ROLE u2 PASSWORD '{password}';  
SELECT * FROM t1;
```

	id	name
1	1	joy
2	2	lily

----End

18.9 Tutorial: Dividing a New Physical Cluster into Logical Clusters

Scenario

This section describes how to divide a new six-node physical cluster (having no service data) into two logical clusters. If your physical cluster already has service data, perform operations by referring to [Tutorial: Converting a Physical Cluster That Contains Data into a Logical Cluster](#).

Prerequisites

Create a six-node cluster. For details, see [Creating a Cluster](#).

Dividing a Cluster into Logical Clusters

- Step 1** On the **Cluster Management** page, click the name of a cluster to go to the cluster details page. In the navigation pane, choose **Logical Clusters**.
- Step 2** Click **Add Logical Cluster** in the upper right corner, select a host ring (three nodes) on the right, add it to the list on the left, enter the logical cluster name **lc1**, and click **OK**.

After about 2 minutes, the logical cluster is added.

- Step 3** Repeat the preceding steps to create the second logical cluster **lc2**.

----End

Creating Logical Clusters, Associating Them with Users, and Querying Data Across Logical Clusters

- Step 1** Connect to the database as system administrator **dbadmin** and run the following SQL statement to check whether the logical cluster is created:

```
SELECT group_name FROM PGXC_GROUP;
```

	group_name
1	group_version1
2	elastic_group
3	lc1
4	lc2

Step 2 Create users **u1** and **u2** and associate them with logical clusters **lc1** and **lc2**, respectively.

```
CREATE USER u1 NODE GROUP "lc1" password '{password}';
CREATE USER u2 NODE GROUP "lc2" password '{password}';
```

Step 3 Switch to user **u1**, create table **t1**, and insert data into the table.

```
SET ROLE u1 PASSWORD '{password}';
CREATE TABLE u1.t1 (id int);
INSERT INTO u1.t1 VALUES (1),(2);
```

Step 4 Switch to user **u2**, create table **t2**, and insert data into the table.

```
SET ROLE u2 PASSWORD '{password}';
CREATE TABLE u2.t2 (id int);
INSERT INTO u2.t2 VALUES (1),(2);
```

Step 5 Query the **u1.t1** table as user **u2**. The command output indicates that the user does not have the permission.

```
SELECT * FROM u1.t1;
```



Step 6 Switch back to the system administrator **dbadmin** and query the **u1.t1** and **u2.t2** tables, which are created in clusters **lc1** and **lc2**, respectively, corresponding to two services. In this way, data is isolated based on logical clusters.

```
SET ROLE dbadmin PASSWORD '{password}';
SELECT p.oid,relname,pgroup,nodeoids FROM pg_class p LEFT JOIN pgxc_class pg ON p.oid = pg.pcrelid
WHERE p.relname = 't1';
SELECT p.oid,relname,pgroup,nodeoids FROM pg_class p LEFT JOIN pgxc_class pg ON p.oid = pg.pcrelid
WHERE p.relname = 't2';
```

oid	relname	pgroup	nodeoids
25374	t1	lc1	16718 16719 16720
oid	relname	pgroup	nodeoids
25377	t2	lc2	16676 16713 16717

Step 7 Grant user **u2** the permissions to access logical cluster **lc1**, schema **u1**, and table **u1.t1**.

```
GRANT usage ON NODE GROUP lc1 TO u2;  
GRANT usage ON SCHEMA u1 TO u2;  
GRANT select ON TABLE u1.t1 TO u2;
```

 **NOTE**

Logical clusters implement permission isolation (by node groups) based on physical clusters. To let a user access data across logical clusters, you need to grant the logical cluster (node-group layer) permissions, schema permissions, and table permissions to the user in sequence. If no logical cluster permissions are granted, the error message "permission denied for node group xx" will be displayed.

Step 8 Switch to user **u2** and query the **u1.t1** table. The query is successful. The logical cluster implements data isolation and allows cross-logical cluster access after user authorization.

```
SET ROLE u2 PASSWORD '{password};  
SELECT * FROM u1.t1;
```

	id
1	2
2	1

----End

19 FAQs

19.1 General Problems

19.1.1 Why Do I Need to Use a Data Warehouse?

Background

Large amounts of data (orders, inventories, materials, and payments) are being generated in enterprises' business operation systems and transactional databases everyday.

Decision makers need to find ways to better utilize and mine such data in order to gain the insights needed to understand the performance of their organizations and make better informed decisions.

Challenges

A data classification and analysis task usually involves simultaneous access to data in multiple database tables, which means multiple tables that are possibly being updated by different transactions need to be locked at the same time. This can be quite difficult for a busy database system.

- Locking multiple tables increases the latency of a complex query.
- Blocking transactions that are updating the database tables causes increased latencies or even interruptions for these transactions.

Solution

Data warehouses excel in data aggregation and association, facilitating large-scale data mining for better decision-making support. Data mining requires complex queries that involve multiple tables.

The ETL process copies data from dedicated business databases to a data warehouse for centralized analysis and computing. Data of multiple systems can be aggregated to one data warehouse for the extraction of more valuable data insights.

Data warehouses are designed differently from standard transaction-oriented databases, such as Oracle, Microsoft SQL Server, and MySQL. Data warehouses are optimized in terms of data aggregation and association, but certain features of standard databases, such as transactional properties and data manipulation operations (add, delete, modify) may be compromised or become unavailable. This is why data warehouses and databases are usually used for different purposes. Transactional databases focus on online transaction processing, while data warehouses are better at complex queries and analysis. You could also say databases are for data updates whereas data warehouses are for data analysis.

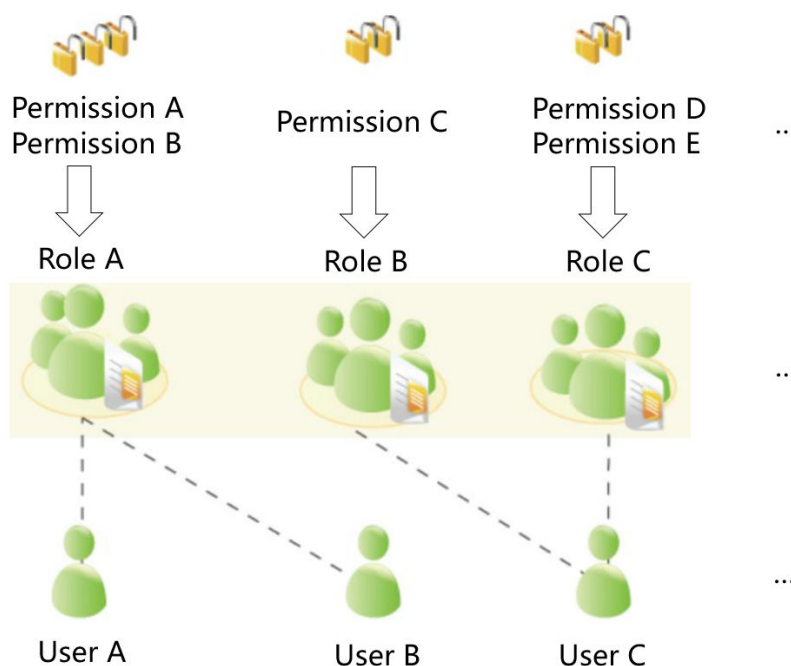
19.1.2 What Are the Differences Between Users and Roles?

Users and roles are shared within the entire cluster, but their data is not shared. That is, a user can connect to any database, but after the connection is successful, any user can access only the database declared in the connection request.

- A role is a set of permissions. Generally, roles are used to sort permissions. Users are used to manage permissions and perform operations.
- A role can inherit permissions from other roles. All users in a user group automatically inherit the operation permissions of the role of the group.
- In a database, the permissions of users come from roles.
- A user group is a group of users who have the same permission.
- A user can be regarded as a role with the login permission.
- A role can be regarded as a user without the login permission.

The permissions provided by Gauss(DWS) include the O&M permissions for components on the management plane. You can assign different permissions to users as needed. The management plane uses roles for better permissions management. You can select specified permissions and assign them to roles in a unified manner. In this way, permissions can be viewed and managed in a centralized manner.

The following figure shows the relationships between permissions, roles, and users in unified permissions management.



GaussDB(DWS) provides various permissions. Select and assign permissions to different users based on service scenarios. A role can be assigned one or more permissions.

After a role is granted to a user through **GRANT**, the user will have all the permissions of the role. It is recommended that roles be used to efficiently grant permissions. A user has permissions only for their own tables, but does not have permissions for other users' tables in their schemas.

- Role A is assigned operation permissions A and B. After role A is allocated to user A and user B, user A and user B can obtain operation permissions A and B.
- Role B is assigned operation permission C. After role B is allocated to user C, user C can obtain operation permissions C.
- Role C is assigned operation permissions D and E. After role C is allocated to user C, user C can obtain operation permissions D and F.

19.1.3 When Should I Use GaussDB(DWS) and MRS?

MRS works better with big data processing frameworks such as Apache Spark, Hadoop, and HBase, to process and analyze ultra-large datasets using custom code. MRS enables you to control cluster configurations and the software installed in the cluster.

GaussDB(DWS) works better with complex queries of large amounts of structured data. GaussDB(DWS) aggregates data from multiple sources, such as inventory, finance, and retail sales systems. To ensure data consistency and accuracy, GaussDB(DWS) stores data in a highly structured manner. This structure builds data consistency rules directly into database tables. Additionally, GaussDB(DWS) is highly compatible with standard SQL statements and syntax used in traditional databases.

GaussDB(DWS) is an ideal choice for performing complex queries on massive collections of structured data, with superb performance.

19.1.4 How Do I Check the Creation Time of a Database User?

Method 1:

When you create a GaussDB(DWS) database user, if the time when the user takes effect (**VALID BEGIN**) is the same as the creation time of the user, and the time when the user takes effect has not been changed, you can check the **valbegin** column in the **PG_USER** view to check the user creation time.

The following is an example:

Create user **jerry** and set its validity start time to its current creation time.

```
CREATE USER jerry PASSWORD 'password' VALID BEGIN '2022-05-19 10:31:56';
```

View users in the **PG_USER** view. The **valbegin** column indicates the time when **jerry** took effect, that is, the time when **jerry** was created.

```
SELECT * FROM PG_USER;
```

username	usesysid	usecreatedb	usesuper	usecatupd	userepl	passwd	valbegin	valuntil
Ruby	10	t	t	t	t	*****		default_pool
dbadmin	16393	f	f	f	f	*****		default_pool
jack	451897	f	f	f	f	*****		default_pool
emma	451910	f	f	f	f	*****		default_pool
jerry	457386	f	f	f	f	*****	2022-05-19 10:31:56+08	default_pool

(5 rows)

Method 2:

Check the **passwordtime** column in the **PG_AUTH_HISTORY** system catalog. This column indicates the time when the user's initial password was created. Only users with system administrator permissions can access the catalog.

```
SELECT roloid, min(passwordtime) as create_time FROM pg_auth_history group by roloid order by roloid;
```

The following is an example:

Query the **PG_USER** view to obtain the OID of user **jerry**, which is **457386**. Query the **passwordtime** column to obtain the creation time of user **jerry**, which is **2022-05-19 10:31:56**.

```
SELECT roloid, min(passwordtime) as create_time FROM pg_auth_history group by roloid order by roloid;
```

roloid	create_time
10	2022-02-25 09:53:38.711785+08
16393	2022-02-25 09:55:17.992932+08
451897	2022-05-18 09:42:26.897855+08
451910	2022-05-18 09:46:33.152354+08
457386	2022-05-19 10:31:56.037706+08

(5 rows)

19.1.5 Regions and AZs

Concepts

A region and availability zone (AZ) identify the location of a data center. You can create resources in regions and AZs.

- A region is a physical data center location. Each region is completely isolated to ensure high fault tolerance and stability. After creating resources in a region, you cannot change the region.
- An AZ is a physical location with independent power supplies and network in a region. A region contains one or more AZs that are physically isolated but interconnected through high-speed internal connections. Faults that occur in one AZ will not affect other AZs. The inter-AZ connections are low-latency and unexpensive.

Selecting a Region

You are advised to select a region close to you or your target users. This reduces network latency and improves speed.

How Do I Select an AZ?

Consider your requirements for DR and network latency when selecting an AZ:

- Deploy resources in different AZs in the same region to improve DR.
- For an application that requires an extremely low latency, deploy all its resources in the same AZ.

Regions and Endpoints

When you use resources with API calls, you must specify the regional endpoint. Obtain the regions and endpoints from the enterprise administrator.

19.1.6 Is My Data Secure in GaussDB(DWS)?

Yes. In the big data era, data has become a core asset. will adhere to the commitment made over the years that we do not touch your applications or data, helping you protect your core assets. This is our commitment to users and the society, laying the foundation for the business success of and their partners.

GaussDB(DWS) is a data warehousing system with telecom-class security to safeguard your data and privacy. Moreover, GaussDB(DWS) delivers carrier-class quality, which can satisfy data security and privacy requirements of governments, financial organizations, and carriers. Therefore, it is widely used by various industries. GaussDB(DWS) won the following security authentication:

- Internal Cyber Security Lab (ICSL) in compliance with cyber security standards issued by the UK authorities.
- Privacy and Security Assessment (PSA) to meet EU requirements of data security and privacy.

Service Data Security

GaussDB(DWS) is built on software infrastructure, including ECS and OBS.

Service data of GaussDB(DWS) users is stored in the ECSs in the cluster. Neither users nor O&M administrators can log in to the ECSs.

The operating system of ECSs is hardened, including kernel hardening, installation of the latest patches, permission control, port management, and protocol and port anti-attack.

GaussDB(DWS) provides comprehensive security measures, such as password policies, authentication, session management, user permissions management, and database auditing.

Snapshot Data Security

GaussDB(DWS) backups are snapshots stored in OBS. OBS supports access permission control, key access, and data encryption features. GaussDB(DWS) snapshots can be used for data backup and restoration only and cannot be accessed by any user. GaussDB(DWS) administrators can view the OBS storage space occupied by snapshots on the GaussDB(DWS) console and through bills.

Network Access Security

The L2 and L3 networks of GaussDB(DWS) can be fully isolated to meet the security requirements of government and financial customers.

- GaussDB(DWS) is deployed in a dedicated ECS environment, which is not shared with any other tenant. This eliminates the possibility of data leakage caused by computing resource sharing.
- ECSs in a GaussDB(DWS) cluster are isolated through VPCs, preventing the ECSs from being discovered and accessed by other tenants.
- The network is divided into the service plane and management plane. The two planes are physically isolated, ensuring network security.
- The tenants can flexibly customize the security group and access rules.
- External application software access GaussDB(DWS) over SSL.
- Data imported from OBS is encrypted.

19.1.7 How Is GaussDB(DWS) Secured?

GaussDB(DWS) uses IAM and VPC to control user access and isolate cluster network. Cluster access is over SSL and cipher suite. Additionally, GaussDB(DWS) supports two-way digital certificate authentication.

Node OSs in each cluster are hardened to allow valid access to only OS files.

19.1.8 Can I Modify the Security Group of a GaussDB(DWS) Cluster?

After a GaussDB(DWS) cluster is created, you can add, delete, or modify security group rules in the current security group.

Modify an existing security group rule:

1. Log in to the GaussDB(DWS) management console.
2. In the navigation tree on the left, choose **Clusters > Dedicated Clusters**.
3. In the cluster list, find the target cluster and click the cluster name. The **Basic Information** page is displayed.
4. Locate the **Security Group** parameter and click the security group name to switch to the **Security Groups** page on the VPC console, on which you can set the security group.

19.1.9 How Are Dirty Pages Generated in GaussDB(DWS)?

Causes

By using the versioning concurrency control (MVCC) mechanism, GaussDB(DWS) can achieve consistency and concurrency for multiple transactions that access the database simultaneously. This mechanism has the benefit of avoiding read-write conflicts, but the drawback of causing disk bloat and dirty pages.

The scenarios are as follows:

- When the DELETE operation is performed on a table, data is logically deleted but not physically deleted from the disk.
- When the UPDATE operation is performed on a table, GaussDB(DWS) logically marks the data to be updated as delete and inserts new data.

For the DELETE and UPDATE operations in a table, the data marked as deleted is called discarded tuples. The proportion of discarded tuples in the entire table is the dirty page rate. Therefore, when the dirty page rate of a table is high, the proportion of data marked as deleted in the table is high.

Solution:

GaussDB(DWS) provides a system view for querying the dirty page rate. For details, see section "PGXC_STAT_TABLE_DIRTY" in *Data Warehouse Service (DWS) Development Guide*.

To solve the problem of disk space bloat caused by high dirty page rate, GaussDB(DWS) provides the VACUUM function to clear the data logically marked as deleted. For details, see "VACUUM" in *Data Warehouse Service (DWS) SQL Syntax Reference*.

VACUUM does not release the allocated space. To completely reclaim the cleared space, run **VACUUM FULL**.

NOTE

- **VACUUM FULL** clears and releases the space of deleted data, improving database performance and efficiency. However, running **VACUUM FULL** consumes more time and resources, and may cause some tables to be locked. Therefore, run **VACUUM FULL** only when the database load is light.
- To reduce the impact of disk bloat on database performance, you are advised to do **VACUUM FULL** on non-system catalogs whose dirty page rate exceeds 80%. You can determine whether to do **VACUUM FULL** based on service scenarios.

19.2 Database Usage

19.2.1 How Do I Change Distribution Columns?

In a data warehouse database, you need to carefully choose distribution columns for large tables, because they can affect your database and query performance. If an improper distribution key is used, data skew may occur after data is imported. As a result, the usage of some disks will be much higher than that of other disks, and the cluster may even become read-only. If the hash distribution policy is used and data skew occurs, the I/O performance of some DN will be poor, affecting the overall query performance. Proper selection and adjustment of distribution columns are critical to table query performance.

If the hash distribution policy is used, you need to check tables to ensure their data is evenly distributed on each DN. Generally, over 5% difference between the amount of data on different DN is regarded as data skew. If the difference is over 10%, you have to choose another distribution column.

For tables that are not evenly distributed, adjust their distribution columns to reduce data skew and avoid database performance problems.

Choosing an Appropriate Distribution Column

The distribution column in a hash table must meet the following requirements, which are ranked by priority in descending order:

- The values of the distribution key should be discrete so that data can be evenly distributed on each DN. You can select the primary key of the table as the distribution key. For example, for a person information table, choose the ID card number column as the distribution key.
- Do not select the column where a constant filter exists.
- Select the join condition as the distribution column, so that join tasks can be pushed down to DN to execute, reducing the amount of data transferred between the DN.
- Multiple distribution columns can be selected to evenly distribute data.

Procedure

Run the **select version();** statement to query the current database version. Required performance varies according to the version.

```
test_lhy=> select version();
                version
-----
PostgreSQL 9.2.4 (GaussDB 8.1.1 build 7ab61a49) compiled at 2021-06-26 12:05:53 commit 2518 last mr 3356 release
(1 row)
```

- **For 8.0.x and earlier versions, specify the distribution column when rebuilding a table.**

Step 1 Use Data Studio or gsql in Linux to access the database.

Step 2 Create a table.

 NOTE

In the following statements, **table1** is the original table name and **table1_new** is the new table name. **column1** and **column2** are distribution column names.

```
CREATE TABLE IF NOT EXISTS table1_new  
( LIKE table1 INCLUDING ALL EXCLUDING DISTRIBUTION)  
DISTRIBUTE BY  
HASH (column1, column2);
```

Step 3 Migrate data to the new table.

```
START TRANSACTION;  
LOCK TABLE table1 IN ACCESS EXCLUSIVE MODE;  
INSERT INTO table1_new SELECT * FROM table1;  
COMMIT;
```

Step 4 Verify that the table data has been migrated. Delete the original table.

```
SELECT COUNT(*) FROM table1_new;  
DROP TABLE table1;
```

Step 5 Replace the original table.

```
ALTER TABLE table1_new RENAME TO table1;
```

----End

- In 8.1.0 and later versions, you can use the **ALTER TABLE** syntax. For example:

Step 1 Query the table definition. The command output shows that the distribution column of the table is **c_last_name**.

```
SELECT pg_get_tabledef('customer_t1');
```

```
gaussdb=> select pg_get_tabledef ('customer_t1');  
pg_get_tabledef  
-----  
SET search_path = public; +  
CREATE TABLE customer_t1 ( +  
  c_customer_sk integer, +  
  c_customer_id character(5), +  
  c_first_name character(6), +  
  c_last_name character(8) +  
) +  
WITH (orientation=column, compression=middle, colversion=2.0, enable_delta=false)+  
DISTRIBUTE BY HASH(c_last_name) +  
TO GROUP group_version1; +  
(1 row)
```

Step 2 Try updating data in the distribution column. An error message will be displayed.

```
UPDATE customer_t1 SET c_last_name = 'Jimmy' WHERE c_customer_sk = 6885;
```

```
gaussdb=> update customer_t1 set c_last_name = 'Jimmy' where c_customer_sk = 6885;  
ERROR: Distributed key column can't be updated in current version
```

Step 3 Change the distribution column of the table to a column that cannot be updated, for example, **c_customer_sk**.

```
ALTER TABLE customer_t1 DISTRIBUTE BY hash (c_customer_sk);
```

```
gaussdb=> alter table customer_t1 DISTRIBUTE BY hash (c_customer_sk);  
ALTER TABLE
```

Step 4 Update the data in the old distribution column.

```
UPDATE customer_t1 SET c_last_name = 'Jimmy' WHERE c_customer_sk = 6885;
```

```
gaussdb=> update customer_t1 set c_last_name = 'Jimy' where c_customer_sk = 6885;  
UPDATE 1
```

----End

19.2.2 How Do I View and Set the Database Character Encoding?

Viewing the Database Character Encoding

Use the **server_encoding** parameter to check the character set encoding of the current database. For example, the character encoding of database **music** is UTF8.

```
music=> SHOW server_encoding;  
server_encoding  
-----  
UTF8  
(1 row)
```

Setting the Database Character Encoding

NOTE

GaussDB(DWS) does not support the modification of the character encoding format of a created database.

If you need to specify the character encoding format of a database, use **template0** and the **CREATE DATABASE** syntax to create a database. To make your database compatible with most characters, you are advised to use the UTF8 encoding when creating a database.

CREATE DATABASE syntax

```
CREATE DATABASE database_name  
[ [ WITH ] { [ OWNER [=] user_name ] |  
[ TEMPLATE [=] template ] |  
[ ENCODING [=] encoding ] |  
[ LC_COLLATE [=] lc_collate ] |  
[ LC_CTYPE [=] lc_ctype ] |  
[ DBCOMPATIBILITY [=] compatibility_type ] |  
[ CONNECTION LIMIT [=] connlimit ] }[...];
```

- **TEMPLATE [=] template**

Indicates the template name, that is, the name of the template to be used to create the database. GaussDB(DWS) creates a database by copying a database template. GaussDB(DWS) has two initial template databases **template0** and **template1** and a default user database .

Value range: an existing database name. If this is not specified, the system copies **template1** by default. Its value cannot be .

NOTICE

Currently, database templates cannot contain sequences. If sequences exist in the template library, database creation will fail.

- **ENCODING [=] encoding**

Character encoding used by the database. The value can be a character string (for example, **SQL_ASCII**) or an integer number.

If this parameter is not specified, the encoding of the template database is used by default. The encoding of template databases **template0** and **template1** depends on the OS by default. The character encoding of **template1** cannot be changed. To change the encoding, use **template0** to create a database.

Value range: **GBK**, **UTF8**, and **Latin1**

NOTICE

The character set encoding of the new database must be compatible with the local settings (**LC_COLLATE** and **LC_CTYPE**).

Examples

Create database **music** using UTF8 (the local encoding type is also UTF8).

```
CREATE DATABASE music ENCODING 'UTF8' template = template0;
```

19.2.3 What Do I Do If Date Type Is Automatically Converted to the Timestamp Type During Table Creation?

When creating a database, you can set the **DBCMPATIBILITY** parameter to the compatible database type. The value of **DBCMPATIBILITY** can be **ORA**, **TD**, and **MySQL**, indicating Oracle, Teradata, and MySQL databases, respectively. If this parameter is not specified during database creation, the default value **ORA** is used. In **ORA** compatibility mode, the date type is automatically converted to timestamp(0). The date type is only supported in the MySQL compatibility mode.

To solve the problem, you need to change the compatibility mode to MySQL. The compatibility mode of an existing database cannot be changed. It can only be specified during creation of the database. GaussDB(DWS) supports the MySQL compatibility mode in cluster version 8.1.1 and later. To configure this mode, run the following commands:

```
gaussdb=> CREATE DATABASE mydatabase DBCMPATIBILITY='mysql';
CREATE DATABASE
gaussdb=> \c mydatabase
Non-SSL connection (SSL connection is recommended when requiring high-security)
You are now connected to database "mydatabase" as user "dbadmin".
mydatabase=> create table t1(c1 int, c2 date);
NOTICE: The 'DISTRIBUTE BY' clause is not specified. Using round-robin as the distribution mode by default.
HINT: Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
CREATE TABLE
```

If the problem cannot be solved by changing the compatibility, you can try to change the column type. For example, insert data of the date type as strings into a table. Example:

```
gaussdb=> CREATE TABLE mytable (a date,b int);
CREATE TABLE
gaussdb=> INSERT INTO mytable VALUES(date '12-08-2023',01);
INSERT 0 1
```

```
gaussdb=> SELECT * FROM mytable;
  a      | b
-----+--
2023-12-08 00:00:00 | 1
(1 row)
gaussdb=> ALTER TABLE mytable MODIFY a VARCHAR(20);
ALTER TABLE
gaussdb=> INSERT INTO mytable VALUES('2023-12-10',02);
INSERT 0 1
gaussdb=> SELECT * FROM mytable;
  a      | b
-----+--
2023-12-08 00:00:00 | 1
2023-12-10          | 2
(2 rows)
```

19.2.4 Do I Need to Run VACUUM FULL and ANALYZE on Common Tables Periodically?

Yes.

For tables that are frequently added, deleted, or modified, you need to periodically perform **VACUUM FULL** and **ANALYZE** to reclaim the disk space occupied by updated or deleted data, preventing performance deterioration caused by data bloat and inaccurate statistics.

- Generally, you are advised to perform **ANALYZE** after a large number of **adding or modification** operations are performed on a table.
- After a table is deleted, you are advised to run **VACUUM** rather than **VACUUM FULL**. However, you can run **VACUUM FULL** in some particular cases, such as when you want to physically narrow a table to decrease the occupied disk space after deleting most rows of the table. For details about the differences between **VACUUM** and **VACUUM FULL**, see [VACUUM and VACUUM FULL](#).

Syntax

Perform **ANALYZE** on a table.

```
ANALYZE table_name;
```

Perform **ANALYZE** on all tables (non-foreign tables) in the database.

```
ANALYZE;
```

Perform **VACUUM** on a table.

```
VACUUM table_name;
```

Perform **VACUUM FULL** on a table.

```
VACUUM FULL table_name;
```

For details, see .

 NOTE

- If the physical space usage does not decrease after you run the **VACUUM FULL** command, check whether there were other active transactions (started before you delete data transactions and not ended before you run **VACUUM FULL**). If yes, run this command again when the transactions have finished.
- In version 8.1.3 or later, **VACUUM/VACUUM FULL** can be invoked on the management plane. For details, see "Intelligent O&M" in the *Data Warehouse Service (DWS) User Guide*.

VACUUM and VACUUM FULL

In GaussDB(DWS), the **VACUUM** operation is like a vacuum cleaner used to absorb dust. Here, "dust" means old data. If the data is not cleared in a timely manner, more database space will be used to store such data, causing performance downgrade or even a system breakdown.

Purposes of VACUUM:

- Solve space bloat: Clear obsolete tuples and corresponding indexes, which include the tuple (and index) of a committed **DELETE** transaction, the old version (and index) of an **UPDATE** transaction, the inserted tuple (and index) of a rolled back **INSERT** transaction, the new version (and index) of an **UPDATE** transaction, and the tuple (and index) of a **COPY** transaction.
- **VACUUM FREEZE**: Prevents system breakdown caused by transaction ID wraparound. It converts transaction IDs smaller than OldestXmin to freeze xids, update relfrozenxids in a table, and update relfrozenxids and truncate clogs in a database.
- Update statistics: **VACUUM ANALYZE** updates statistics, enabling the optimizer to select a better way to execute SQL statements.

The **VACUUM** statement includes **VACUUM** and **VACUUM FULL**. Currently, **VACUUM** can only work on row-store tables. **VACUUM FULL** can be used to release space of column-store tables. For details, see the following table.

Table 19-1 VACUUM and VACUUM FULL

Item	VACUUM	VACUUM FULL
Clearing space	If the deleted record is at the end of a table, the space occupied by the deleted record is physically released and returned to the operating system. If the data is not at the end of a table, the space occupied by dead tuples in the table or index is set to be available for reuse.	Despite the position of the deleted data, the space occupied by the data is physically released and returned to the operating system. When data is inserted, a new disk page is allocated.
Lock type	Shared lock. The VACUUM operation can be performed in parallel with other operations.	Exclusive lock. All operations based on the table are suspended during execution.

Item	VACUUM	VACUUM FULL
Physical space	Not released	Released
Transaction ID	Not reclaimed	Reclaimed
Execution overhead	The overhead is low and the operation can be executed periodically.	The overhead is high. You are advised to perform it when the disk page space occupied by the database is close to the threshold and the data operations are few.
Effect	It improves the efficiency of operations on the table.	It greatly improves the efficiency of operations on the table.

19.2.5 Do I Need to Set a Distribution Key After Setting a Primary Key?

No, you only need to set the primary key. By default, the first column of the primary key is selected as the distribution key. If both are set, the primary key must contain the distribution key.

19.2.6 Is GaussDB(DWS) Compatible with PostgreSQL Stored Procedures?

Yes.

GaussDB(DWS) is compatible with PostgreSQL stored procedures. For details, see "Stored Procedures" in the *Developer Guide*.

19.2.7 What Are Partitioned Tables, Partitions, and Partition Keys?

Partitioned table: Partitioning refers to splitting what is logically one large table into smaller physical pieces based on specific schemes. The table based on the logic is called a partitioned table, and a physical piece is called a partition. Data is stored on these smaller physical pieces, namely, partitions, instead of the larger logical partitioned table.

Partition: In the GaussDB(DWS) distributed system, data partitioning is to horizontally partition table data within a node based on a specified policy. The table is divided into partitions that do not overlap within a specific range.

Partition key: A partition key is an ordered set of one or more table columns. The values in the table partition keys are used to determine the data partition that a row belongs to.

19.2.8 How Can I Export the Table Structure?

You are advised to use the Data Studio graphical client to export table data. You can export data from:

- A specific table
- All tables in a schema
- All tables in a database

19.2.9 How Do I Delete Table Data Efficiently?

Yes. **TRUNCATE** is more efficient than **DELETE** for deleting massive data.

Function

TRUNCATE quickly removes all rows from a table. It has the same effect as an unqualified **DELETE** but since it does not actually scan the table it is faster. This is most effective on large tables.

Functions

- **TRUNCATE TABLE** works like a **DELETE** statement with no **WHERE** clause, that is, emptying a table.
- **TRUNCATE TABLE** uses less system and transaction log resources.
 - **DELETE** deletes a row each time, and records each deletion in the transaction log.
 - **TRUNCATE TABLE** deletes all rows in a table by releasing the data page, and only records each releasing of the data page in the transaction log.
- **TRUNCATE**, **DELETE**, and **DROP** are different in that:
 - **TRUNCATE TABLE** deletes content, releases space, but does not delete definitions.
 - **DELETE TABLE** deletes content, but does not delete definitions or release space.
 - **DROP TABLE** deletes content and definitions, and releases space.

Examples

- Create a table.

```
CREATE TABLE tpcds.reason_t1 AS TABLE tpcds.reason;
```

Truncate the table.

```
TRUNCATE TABLE tpcds.reason_t1;
```

Delete the table.

```
DROP TABLE tpcds.reason_t1;
```
- Create a partitioned table.

```
CREATE TABLE tpcds.reason_p  
(  
  r_reason_sk integer,  
  r_reason_id character(16),  
  r_reason_desc character(100)  
)PARTITION BY RANGE (r_reason_sk)
```



```
CREATE TABLE warehouse1
(
  W_WAREHOUSE_SK      INTEGER      PRIMARY KEY,
  W_WAREHOUSE_ID      CHAR(16)      NOT NULL,
  W_WAREHOUSE_NAME    VARCHAR(20)
);
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "warehouse1_pkey" for table
"warehouse1"
CREATE TABLE

SELECT getdistributekey('warehouse1');
getdistributekey
-----
w_warehouse_sk
(1 row)
```

- Scenario 2

If the primary key or unique constraint is not included during table creation but there are columns whose data types can be used as distribution columns, hash distribution is selected. The distribution column is the first column whose data type can be used as a distribution column.

```
CREATE TABLE warehouse2
(
  W_WAREHOUSE_SK      INTEGER      ,
  W_WAREHOUSE_ID      CHAR(16)      NOT NULL,
  W_WAREHOUSE_NAME    VARCHAR(20)
);
NOTICE: The 'DISTRIBUTE BY' clause is not specified. Using 'w_warehouse_sk' as the distribution
column by default.
HINT: Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
CREATE TABLE

SELECT getdistributekey('warehouse2');
getdistributekey
-----
w_warehouse_sk
(1 row)
```

- Scenario 3

If the primary key or unique constraint is not included during table creation and no column whose data type can be used as a distribution column exists, round-robin distribution is selected.

```
CREATE TABLE warehouse3
(
  W_WAREHOUSE_ID      CHAR(16)      NOT NULL,
  W_WAREHOUSE_NAME    VARCHAR(20)
);
NOTICE: The 'DISTRIBUTE BY' clause is not specified. Using 'w_warehouse_id' as the distribution
column by default.
HINT: Please use 'DISTRIBUTE BY' clause to specify suitable data distribution column.
CREATE TABLE

SELECT getdistributekey('warehouse3');
getdistributekey
-----
w_warehouse_id
(1 row)
```

19.2.12 How Do I Replace the Null Result with 0?

When OUTER JOIN (LEFT JOIN, RIGHT JOIN, and FULL JOIN) is executed, the match failure in the outer join generates a large number of NULL values. You can replace these null values with 0.

You can use the **COALESCE** function to do that. This function returns the first non-null parameter value in the parameter list. For example:

```
SELECT coalesce(NULL,'hello');
coalesce
-----
hello
(1 row)
```

Use left join to join the tables **course1** and **course2**.

```
SELECT * FROM course1;
stu_id | stu_name | cour_name
-----+-----+-----
20110103 | ALLEN   | Math
20110102 | JACK   | Programming Design
20110101 | MAX    | Science
(3 rows)

SELECT * FROM course2;
cour_id | cour_name | teacher_name
-----+-----+-----
1002 | Programming Design | Mark
1001 | Science | Anne
(2 rows)

SELECT course1.stu_name,course2.cour_id,course2.cour_name,course2.teacher_name FROM course1 LEFT
JOIN course2 ON course1.cour_name = course2.cour_name ORDER BY 1;
stu_name | cour_id | cour_name | teacher_name
-----+-----+-----+-----
ALLEN    |         |          |
JACK     | 1002 | Programming Design | Mark
MAX      | 1001 | Science | Anne
(3 rows)
```

Use the **COALESCE** function to replace null values in the query result with 0 or other non-zero values:

```
SELECT course1.stu_name,
coalesce(course2.cour_id,0) AS cour_id,
coalesce(course2.cour_name,'NA') AS cour_name,
coalesce(course2.teacher_name,'NA') AS teacher_name
FROM course1
LEFT JOIN course2 ON course1.cour_name = course2.cour_name
ORDER BY 1;
stu_name | cour_id | cour_name | teacher_name
-----+-----+-----+-----
ALLEN    | 0 | NA | NA
JACK     | 1002 | Programming Design | Mark
MAX      | 1001 | Science | Anne
(3 rows)
```

19.2.13 How Do I Check Whether a Table Is Row-Stored or Column-Stored?

The storage mode of a table is controlled by the **ORIENTATION** parameter in the table creation statement. **row** indicates row storage, and **column** indicates column storage.

You can use the table definition function **PG_GET_TABLEDEF** to check whether the created table is row-store or column-store.

For example, **orientation=column** indicates a column-store table.

Currently, you cannot run the **ALTER TABLE** statement to modify the parameter **ORIENTATION**.

```
SELECT * FROM PG_GET_TABLEDEF('customer_t1');
           pg_get_tabledef
-----
SET search_path = tpchobs;
CREATE TABLE customer_t1 (
  c_customer_sk integer,
  c_customer_id character(5),
  c_first_name character(6),
  c_last_name character(8)
)
WITH (orientation=column, compression=middle, colversion=2.0, enable_delta=false)+
DISTRIBUTE BY HASH(c_last_name)
TO GROUP group_version1;
(1 row)
```

19.2.14 How Do I Query the Information About GaussDB(DWS) Column-Store Tables?

The following SQL statements are used to query common information about column-store tables:

Create a column-store table named `my_table`, and insert data into the table.

```
CREATE TABLE my_table
(
  product_id INT,
  product_name VARCHAR2(40),
  product_quantity INT
)
WITH (ORIENTATION = COLUMN)
PARTITION BY range(product_quantity)
(
  partition my_table_p1 values less than(600),
  partition my_table_p2 values less than(800),
  partition my_table_p3 values less than(950),
  partition my_table_p4 values less than(1000));

INSERT INTO my_table VALUES(1011, 'tents', 720);
INSERT INTO my_table VALUES(1012, 'hammock', 890);
INSERT INTO my_table VALUES(1013, 'compass', 210);
INSERT INTO my_table VALUES(1014, 'telescope', 490);
INSERT INTO my_table VALUES(1015, 'flashlight', 990);
INSERT INTO my_table VALUES(1016, 'ropes', 890);
```

Run the following command to view the created column-store partitioned table:

```
SELECT * FROM my_table;
product_id | product_name | product_quantity
-----+-----+-----
1013 | compass | 210
1014 | telescope | 490
1011 | tents | 720
1015 | flashlight | 990
1012 | hammock | 890
1016 | ropes | 890
(6 rows)
```

Querying the Boundary of a Partition

```
SELECT relname, partstrategy, boundaries FROM pg_partition where parentid=(select parentid from
pg_partition where relname='my_table');
relname | partstrategy | boundaries
-----+-----+-----
```

```
my_table | r |
my_table_p1 | r | {600}
my_table_p2 | r | {800}
my_table_p3 | r | {950}
my_table_p4 | r | {1000}
(5 rows)
```

Querying the Number of Columns in a Column-Store Table

```
SELECT count(*) FROM ALL_TAB_COLUMNS where table_name='my_table';
count
-----
3
(1 row)
```

Querying Data Distribution on DNs

```
SELECT table_skewness('my_table');
table_skewness
-----
("dn_6007_6008 " ,3,50.000%)
("dn_6009_6010 " ,2,33.333%)
("dn_6003_6004 " ,1,16.667%)
("dn_6001_6002 " ,0,0.000%)
("dn_6005_6006 " ,0,0.000%)
("dn_6011_6012 " ,0,0.000%)
(6 rows)
```

Querying the Names of the Cudesc and Delta Tables in Partition P1 on a DN

```
EXECUTE DIRECT ON (dn_6003_6004) 'select a.relname from pg_class a, pg_partition b where
(a.oid=b.reldeltarelid or a.oid=b.relcudescrelid) and b.relname="my_table_p1";
relname
-----
pg_delta_part_60317
pg_cudesc_part_60317
(2 rows)
```

19.2.15 Why Sometimes the GaussDB(DWS) Query Indexes Become Invalid?

Creating indexes for tables can improve database query performance. However, sometimes indexes cannot be used in a query plan. This section describes several common reasons and optimization methods.

Reason 1: The Returned Result Sets Are Large.

The following uses Seq Scan and Index Scan on a row-store table as an example:

- Seq Scan: searches table records in sequence. All records are retrieved during each scan. This is the simplest and most basic table scanning method, and its cost is high.
- Index Scan: searches the index first, find the target location (pointer) in the index, and then retrieve data on the target page.

Index scan is faster than sequence scan in most cases. However, if the obtained result sets account for a large proportion (more than 70%) of all data, Index Scan needs to scan indexes before reading table data. This makes it slower table scan.

Reason 2: ANALYZE Is Not Performed In a Timely Manner.

ANALYZE is used to update table statistics. If **ANALYZE** is not executed on a table or a large amount of data is added to or deleted from a table after **ANALYZE** is executed, the statistics may be inaccurate, which may cause a query to skip the index.

Optimization method: Run the **ANALYZE** statement on the table to update statistics.

Reason 3: Filtering Conditions Contains Functions or Implicit Data Type Conversion

If calculation, function, or implicit data type conversion is contained in filter criteria, indexes may fail to be selected.

For example, when a table is created, indexes are created in columns **a**, **b**, and **c**.

```
CREATE TABLE test(a int, b text, c date);
```

- Perform calculation on the indexed columns.

The following command output indicates that both **where a = 101** and **where a = 102 - 1** use the index in column **a**, but **where a + 1 = 102** does not use the index.

```
explain verbose select * from test where a = 101;
QUERY PLAN
```

id	operation	E-rows	E-distinct	E-memory	E-width	E-costs
1	-> Streaming (type: GATHER)	1			44	16.27
2	-> Index Scan using index_a on public.test	1		1MB	44	8.27

Predicate Information (identified by plan id)

```
2 --Index Scan using index_a on public.test
Index Cond: (test.a = 101)
```

Targetlist Information (identified by plan id)

```
1 --Streaming (type: GATHER)
Output: a, b, c
Node/s: dn_6005_6006
2 --Index Scan using index_a on public.test
Output: a, b, c
Distribute Key: a
```

==== Query Summary =====

```
System available mem: 3358720KB
Query Max mem: 3358720KB
Query estimated mem: 1024KB
(24 rows)
explain verbose select * from test where a = 102 - 1;
QUERY PLAN
```

id	operation	E-rows	E-distinct	E-memory	E-width	E-costs
1	-> Streaming (type: GATHER)	1			44	16.27
2	-> Index Scan using index_a on public.test	1		1MB	44	8.27

Predicate Information (identified by plan id)

```
2 --Index Scan using index_a on public.test
Index Cond: (test.a = 101)
```

Targetlist Information (identified by plan id)

```
-----
1 --Streaming (type: GATHER)
  Output: a, b, c
  Node/s: dn_6005_6006
2 --Index Scan using index_a on public.test
  Output: a, b, c
  Distribute Key: a
```

=====
Query Summary
=====

```
-----
System available mem: 3358720KB
Query Max mem: 3358720KB
Query estimated mem: 1024KB
(24 rows)
explain verbose select * from test where a + 1 = 102;
QUERY PLAN
```

id	operation	E-rows	E-distinct	E-memory	E-width	E-costs
1	-> Streaming (type: GATHER)	1			44	22.21
2	-> Seq Scan on public.test	1		1MB	44	14.21

Predicate Information (identified by plan id)

```
-----
2 --Seq Scan on public.test
  Filter: ((test.a + 1) = 102)
```

Targetlist Information (identified by plan id)

```
-----
1 --Streaming (type: GATHER)
  Output: a, b, c
  Node/s: All datanodes
2 --Seq Scan on public.test
  Output: a, b, c
  Distribute Key: a
```

=====
Query Summary
=====

```
-----
System available mem: 3358720KB
Query Max mem: 3358720KB
Query estimated mem: 1024KB
(24 rows)
```

Optimization method: Use constants instead of expressions, or put constant calculation on the right of the equal sign (=).

- Use functions on indexed columns.

According to the following execution result, if a function is used on an indexed column, the index fails to be selected.

```
explain verbose select * from test where to_char(c, 'yyyymmdd') =
to_char(CURRENT_DATE,'yyyymmdd');
```

QUERY PLAN

id	operation	E-rows	E-distinct	E-memory	E-width	E-costs
1	-> Streaming (type: GATHER)	1			44	22.28
2	-> Seq Scan on public.test	1		1MB	44	14.28

Predicate Information (identified by plan id)

```
-----
2 --Seq Scan on public.test
  Filter: (to_char(test.c, 'yyyymmdd'::text) = to_char(('2022-11-30'::pg_catalog.date)::timestamp
with time zone, 'yyyymmdd'::text))
```

```

Targetlist Information (identified by plan id)
-----
1 --Streaming (type: GATHER)
  Output: a, b, c
  Node/s: All datanodes
2 --Seq Scan on public.test
  Output: a, b, c
  Distribute Key: a

===== Query Summary =====
-----
System available mem: 3358720KB
Query Max mem: 3358720KB
Query estimated mem: 1024KB
(24 rows)
explain verbose select * from test where c = current_date;
          QUERY PLAN
-----
id |          operation          | E-rows | E-distinct | E-memory | E-width | E-costs
---+-----+-----+-----+-----+-----+-----
 1 | -> Streaming (type: GATHER) |      1 |           |          |    44 | 16.27
 2 | -> Index Scan using index_c on public.test |      1 |           |    1MB  |    44 | 8.27

Predicate Information (identified by plan id)
-----
2 --Index Scan using index_c on public.test
  Index Cond: (test.c = '2022-11-30':pg_catalog.date)

Targetlist Information (identified by plan id)
-----
1 --Streaming (type: GATHER)
  Output: a, b, c
  Node/s: All datanodes
2 --Index Scan using index_c on public.test
  Output: a, b, c
  Distribute Key: a

===== Query Summary =====
-----
System available mem: 3358720KB
Query Max mem: 3358720KB
Query estimated mem: 1024KB
(24 rows)

```

Optimization method: Do not use unnecessary functions on indexed columns.

- Implicit conversion of data types.

This scenario is common. For example, the type of column **b** is Text, and the filtering condition is **where b = 2**. During plan generation, the Text type is implicitly converted to the Bigint type, and the actual filtering condition changes to **where b::bigint = 2**. As a result, the index in column **b** becomes invalid.

```

explain verbose select * from test where b = 2;
          QUERY PLAN
-----
id |          operation          | E-rows | E-distinct | E-memory | E-width | E-costs
---+-----+-----+-----+-----+-----+-----
 1 | -> Streaming (type: GATHER) |      1 |           |          |    44 | 22.21
 2 | -> Seq Scan on public.test |      1 |           |    1MB  |    44 | 14.21

Predicate Information (identified by plan id)
-----
2 --Seq Scan on public.test
  Filter: ((test.b)::bigint = 2)

Targetlist Information (identified by plan id)
-----
1 --Streaming (type: GATHER)

```

```

Output: a, b, c
Node/s: All datanodes
2 --Seq Scan on public.test
Output: a, b, c
Distribute Key: a

===== Query Summary =====
-----
System available mem: 3358720KB
Query Max mem: 3358720KB
Query estimated mem: 1024KB
(24 rows)
explain verbose select * from test where b = '2';
QUERY PLAN
-----
id | operation | E-rows | E-distinct | E-memory | E-width | E-costs
-----+-----+-----+-----+-----+-----+-----
1 | -> Streaming (type: GATHER) | 1 | | | 44 | 16.27
2 | -> Index Scan using index_b on public.test | 1 | | 1MB | 44 | 8.27

Predicate Information (identified by plan id)
-----
2 --Index Scan using index_b on public.test
Index Cond: (test.b = '2'::text)

Targetlist Information (identified by plan id)
-----
1 --Streaming (type: GATHER)
Output: a, b, c
Node/s: All datanodes
2 --Index Scan using index_b on public.test
Output: a, b, c
Distribute Key: a

===== Query Summary =====
-----
System available mem: 3358720KB
Query Max mem: 3358720KB
Query estimated mem: 1024KB
(24 rows)

```

Optimization method: Use constants of the same type as the indexed column to avoid implicit type conversion.

Scenario 4: Hashjoin Is Replaced with Nestloop + Indexscan.

When two tables are joined, the number of rows in the result set filtered by the WHERE condition in one table is small, thus the number of rows in the final result set is also small. In this case, the effect of nestloop+indexscan is better than that of hashjoin. The better execution plan is as follows:

You can see that the Index Cond: (t1.b = t2.b) at layer 5 has pushed the join condition down to the base table scanning.

```

explain verbose select t1.a,t1.b from t1,t2 where t1.b=t2.b and t2.a=4;
id | operation | E-rows | E-distinct | E-memory | E-width | E-costs
-----+-----+-----+-----+-----+-----+-----
1 | -> Streaming (type: GATHER) | 26 | | | 8 | 17.97
2 | -> Nested Loop (3,5) | 26 | | 1MB | 8 | 11.97
3 | -> Streaming(type: BROADCAST) | 2 | | 2MB | 4 | 2.78
4 | -> Seq Scan on public.t2 | 1 | | 1MB | 4 | 2.62
5 | -> Index Scan using t1_b_idx on public.t1 | 26 | | 1MB | 8 | 9.05
(5 rows)

Predicate Information (identified by plan id)
-----
4 --Seq Scan on public.t2

```

```

Filter: (t2.a = 4)
5 --Index Scan using t1_b_idx on public.t1
  Index Cond: (t1.b = t2.b)
(4 rows)

Targetlist Information (identified by plan id)
-----
1 --Streaming (type: GATHER)
  Output: t1.a, t1.b
  Node/s: All datanodes
2 --Nested Loop (3,5)
  Output: t1.a, t1.b
3 --Streaming (type: BROADCAST)
  Output: t2.b
  Spawn on: datanode2
  Consumer Nodes: All datanodes
4 --Seq Scan on public.t2
  Output: t2.b
  Distribute Key: t2.a
5 --Index Scan using t1_b_idx on public.t1
  Output: t1.a, t1.b
  Distribute Key: t1.a
(15 rows)

===== Query Summary =====
-----
System available mem: 9262694KB
Query Max mem: 9471590KB
Query estimated mem: 5144KB
(3 rows)

```

If the optimizer does not select such an execution plan, you can optimize it as follows:

```

set enable_index_nestloop = on;
set enable_hashjoin = off;
set enable_seqscan = off;

```

Reason 5: The Scan Method Is Incorrectly Specified by Hints.

GaussDB(DWS) plan hints can specify three scan method: tablescan, indexscan, and indexonlyscan.

- Table Scan: full table scan, such as Seq Scan of row-store tables and CStore Scan of column-store tables.
- Index Scan: scans indexes and then obtains table records based on the indexes.
- Index-Only Scan: scans indexes, which cover all required results. Compared with the index scan, the index-only scan covers all queried columns. In this way, only indexes are retrieved, and data records do not need to be retrieved.

In Index-Only Scan scenarios, Index Scan specified by a hint will be invalid.

```

explain verbose select /*+ indexscan(test)*/ b from test where b = '1';
WARNING: unused hint: IndexScan(test)
          QUERY PLAN
-----
id | operation | E-rows | E-distinct | E-memory | E-width | E-costs
---+-----+-----+-----+-----+-----+-----
1 | -> Streaming (type: GATHER) | 1 | | | 32 | 16.27
2 | -> Index Only Scan using index_b on public.test | 1 | | 1MB | 32 | 8.27

Predicate Information (identified by plan id)
-----
2 --Index Only Scan using index_b on public.test

```

```

Index Cond: (test.b = '1':text)

Targetlist Information (identified by plan id)
-----
1 --Streaming (type: GATHER)
  Output: b
  Node/s: All datanodes
2 --Index Only Scan using index_b on public.test
  Output: b
  Distribute Key: a

===== Query Summary =====
-----
System available mem: 3358720KB
Query Max mem: 3358720KB
Query estimated mem: 1024KB
(24 rows)
explain verbose select /*+ indexonlyscan(test)*/ b from test where b = '1';
                                QUERY PLAN
-----
id |          operation          | E-rows | E-distinct | E-memory | E-width | E-costs
---+-----+-----+-----+-----+-----+-----
1 | -> Streaming (type: GATHER) |      1 |           |          |      32 | 16.27
2 | -> Index Only Scan using index_b on public.test |      1 |           |      1MB |      32 | 8.27

Predicate Information (identified by plan id)
-----
2 --Index Only Scan using index_b on public.test
  Index Cond: (test.b = '1':text)

Targetlist Information (identified by plan id)
-----
1 --Streaming (type: GATHER)
  Output: b
  Node/s: All datanodes
2 --Index Only Scan using index_b on public.test
  Output: b
  Distribute Key: a

===== Query Summary =====
-----
System available mem: 3358720KB
Query Max mem: 3358720KB
Query estimated mem: 1024KB
(24 rows)

```

Optimization method: Correctly specify Index scan and Index-Only Scan.

Reason 6: Incorrect Use of GIN Index in Full-Text Retrieval

To accelerate text search, you can create a GIN index for full-text search.

```
CREATE INDEX idxb ON test using gin(to_tsvector('english',b));
```

When creating the GIN index, you must use the 2-argument version of to_tsvector. Only when the query also uses the 2-argument version and the arguments are the same as that in the Gin index, the GIN index can be called.

NOTE

The to_tsvector() function accepts one or two arguments. If the one-argument version of the index is used, the system will use the configuration specified by **default_text_search_config** by default. To create an index, the two-argument version must be used, or the index content may be inconsistent.

```
explain verbose select * from test where to_tsvector(b) @@ to_tsquery('cat') order by 1;
                                QUERY PLAN
```

```

-----
id |          operation          | E-rows | E-distinct | E-memory | E-width | E-costs
-----+-----+-----+-----+-----+-----+-----
1 | -> Streaming (type: GATHER) |      2 |           |          |         | 44 | 22.23
2 | -> Sort                    |      2 |           | 16MB    |         | 44 | 14.23
3 | -> Seq Scan on public.test |      1 |           | 1MB     |         | 44 | 14.21
-----
Predicate Information (identified by plan id)
-----
3 --Seq Scan on public.test
  Filter: (to_tsvector(test.b) @@ "'cat'::tsquery)
-----
Targetlist Information (identified by plan id)
-----
1 --Streaming (type: GATHER)
  Output: a, b, c
  Merge Sort Key: test.a
  Node/s: All datanodes
2 --Sort
  Output: a, b, c
  Sort Key: test.a
3 --Seq Scan on public.test
  Output: a, b, c
  Distribute Key: a

===== Query Summary =====
-----
System available mem: 3358720KB
Query Max mem: 3358720KB
Query estimated mem: 1024KB
(29 rows)
explain verbose select * from test where to_tsvector('english',b) @@ to_tsquery('cat') order by 1;
          QUERY PLAN
-----
id |          operation          | E-rows | E-distinct | E-memory | E-width | E-costs
-----+-----+-----+-----+-----+-----+-----
1 | -> Streaming (type: GATHER) |      2 |           |          |         | 44 | 20.03
2 | -> Sort                    |      2 |           | 16MB    |         | 44 | 12.03
3 | -> Bitmap Heap Scan on public.test |      1 |           | 1MB     |         | 44 | 12.02
4 | -> Bitmap Index Scan      |      1 |           | 1MB     |         |  0 | 8.00
-----
Predicate Information (identified by plan id)
-----
3 --Bitmap Heap Scan on public.test
  Recheck Cond: (to_tsvector('english'::regconfig, test.b) @@ "'cat'::tsquery)
4 --Bitmap Index Scan
  Index Cond: (to_tsvector('english'::regconfig, test.b) @@ "'cat'::tsquery)
-----
Targetlist Information (identified by plan id)
-----
1 --Streaming (type: GATHER)
  Output: a, b, c
  Merge Sort Key: test.a
  Node/s: All datanodes
2 --Sort
  Output: a, b, c
  Sort Key: test.a
3 --Bitmap Heap Scan on public.test
  Output: a, b, c
  Distribute Key: a

===== Query Summary =====
-----
System available mem: 3358720KB
Query Max mem: 3358720KB
Query estimated mem: 2048KB
(32 rows)

```

Optimization method: Use the 2-argument version of `to_tsvector` for the query and ensure that the argument values are the same as those in the index.

19.2.16 How Do I Use a User-Defined Function to Rewrite the CRC32() Function?

Currently, GaussDB(DWS) does not have a built-in `CRC32()` function. Instead, you can use the user-defined function of GaussDB(DWS) to rewrite the `CRC32()` function.

- `CRC32(expr)`
- Description: Calculates the cyclic redundancy. The input parameter `expr` is a string. If the parameter is NULL, NULL is returned. Otherwise, a 32-bit unsigned value is returned after redundancy calculation.

Example of rewriting the `CRC32()` function using the user-defined function statement of GaussDB(DWS):

```
CREATE OR REPLACE FUNCTION crc32(text_string text) RETURNS bigint AS $$
DECLARE
    val bigint;
    i int;
    j int;
    byte_length int;
    binary_string bytea;
BEGIN
    IF text_string is null THEN
        RETURN null;
    ELSIF text_string = '' THEN
        RETURN 0;
    END IF;

    i = 0;
    val = 4294967295;
    byte_length = bit_length(text_string) / 8;
    binary_string = decode(replace(text_string, E'\\', E'\\\\\\'), 'escape');
    LOOP
        val = (val # get_byte(binary_string, i)::bigint;
        i = i + 1;
        j = 0;
        LOOP
            val = ((val >> 1) # (3988292384 * (val & 1)))::bigint;
            j = j + 1;
            IF j >= 8 THEN
                EXIT;
            END IF;
        END LOOP;
        IF i >= byte_length THEN
            EXIT;
        END IF;
    END LOOP;
    RETURN (val # 4294967295);
END
$$ IMMUTABLE LANGUAGE plpgsql;
```

Verify the rewriting result.

```
select crc32(null),crc32(''),crc32('1');
crc32 | crc32 |  crc32
-----+-----+-----
| 0 | 2212294583
(1 row)
```

For details about how to use user-defined functions, see section "CREATE FUNCTION" in *SQL Syntax References*.

19.2.17 What Are the Schemas Starting with pg_toast_temp* or pg_temp*?

When you query the schema list, the query result may contain schemas starting with **pg_temp*** or **pg_toast_temp***, as shown in the following figure.

```
SELECT * FROM pg_namespace;
```

nsname	nspowner	nspacl	nspispartition	permspace	usedspace
pg_toast	10	0		-1	0
cstore	10	0		-1	0
gs_logical_cluster	10	0		-1	0
sys	10	0		-1	0
dbes_on	10	0	{Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}	-1	24576
dbes_job	10	0	{Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}	-1	0
pg_catalog	10	0	{Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}	-1	13008896
public	10	0	{Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}	-1	622592
information_schema	10	0	{Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}	-1	352256
utl_file	10	0	{Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}	-1	0
dbes_output	10	0	{Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}	-1	0
dbes_random	10	0	{Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}	-1	0
utl_raw	10	0	{Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}	-1	0
dbes_sql	10	0	{Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}	-1	0
dbes_job	10	0	{Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}	-1	0
scheduler	10	0		-1	8192
u1	24954	0		-1	0
u2	24955	0		-1	0
u3	24962	0		-1	0
u4	24966	0		-1	0
s1	16833	0	{dbadmin=UC/dbadmin,dbadmin=LP/dbadmin,rs1_select=U/dbadmin,rs1_update=U/dbadmin,rs2_select=U/dbadmin,rs2_update=U/dbadmin}	-1	0
s2	16833	0	{dbadmin=UC/dbadmin,dbadmin=LP/dbadmin,rs1_select=U/dbadmin,rs1_update=U/dbadmin,rs2_select=U/dbadmin,rs2_update=U/dbadmin}	-1	0
pg_temp_cn_5893_4_1_281471119284272	10	0		-1	0
pg_toast_temp_cn_5893_4_1_281471119284272	10	0		-1	0
cstore	10	0		-1	0

These schemas are created when temporary tables are created. Each session has an independent schema starting with **pg_temp** to ensure that the temporary tables are visible only to the current session. Therefore, you are not advised to manually delete schemas starting with **pg_temp** or **pg_toast_temp** during routine operations.

Temporary tables are visible only in the current session and are automatically deleted after the session ends. The corresponding schemas are also deleted.

19.2.18 Solutions to Inconsistent GaussDB(DWS) Query Results

In GaussDB(DWS), sometimes a SQL query may get different results. This problem is most likely caused by improper syntax or usage. To avoid this problem, use the syntax correctly. The following are some examples of query results inconsistency along with the solutions.

Window Function Results Are Incompletely Sorted

Scenario:

In the window function **row_number()**, column **c** of table **t3** is queried after sorting. The two query results are different.

```
SELECT * FROM t3 order by 1,2,3;
```

```
a | b | c
---+---
1 | 2 | 1
1 | 2 | 2
1 | 2 | 3
(3 rows)
```

```
SELECT c,rn FROM (select c,row_number() over(order by a,b) as rn from t3) where rn = 1;
```

```
c | rn
---+---
1 | 1
(1 row)
```

```
SELECT c,rn FROM (select c,row_number() over(order by a,b) as rn from t3) where rn = 1;
```

```
c | rn
---+---
3 | 1
(1 row)
```

Analysis:

As shown above, run **select c,rn from (select c,row_number() over(order by a,b) as rn from t3) where rn = 1;** twice, the results are different. That is because duplicate values **1** and **2** exist in the sorting columns **a** and **b** of the window function while their values in column **c** are different. As a result, when the first record is obtained based on the sorting result in columns **a** and **b**, the obtained data in column **c** is random, as a result, the result sets are inconsistent.

Solution:

The values in column **c** need to be added to the sorting.

```
SELECT c,rn FROM (select c,row_number() over(order by a,b,c) as rn from t3) where rn = 1;
c | rn
---+---
1 | 1
(1 row)
```

Using Sorting in Subviews/Subqueries

Scenario

After table **test** and view **v** are created, the query results are inconsistent when sorting is used to query table **test** in a subquery.

```
CREATE TABLE test(a serial ,b int);
INSERT INTO test(b) VALUES(1);
INSERT INTO test(b) SELECT b FROM test;
...
INSERT INTO test(b) SELECT b FROM test;
CREATE VIEW v as SELECT * FROM test ORDER BY a;
```

Problem SQL:

```
SELECT * FROM v limit 1;
a | b
---+---
3 | 1
(1 row)

SELECT * FROM (select * from test order by a) limit 10;
a | b
---+---
14 | 1
(1 row)

SELECT * FROM test order by a limit 10;
a | b
---+---
1 | 1
(1 row)
```

Analysis:

ORDER BY is invalid for subviews and subqueries.

Solution:

You are not advised to use **ORDER BY** in subviews and subqueries. To ensure that the results are in order, use **ORDER BY** in the outermost query.

LIMIT in Subqueries

Scenario: When **LIMIT** is used in a subquery, the two query results are inconsistent.

```
SELECT * FROM (select a from test limit 1 ) order by 1;
a
---
5
(1 row)

SELECT * FROM (select a from test limit 1 ) order by 1;
a
---
1
(1 row)
```

Analysis:

The LIMIT in the subquery causes random results to be obtained.

Solution:

To ensure the stability of the final query result, do not use **LIMIT** in subqueries.

Using String_agg

Scenario: When **string_agg** is used to query the table **employee**, the query results are inconsistent.

```
SELECT * FROM employee;
empno | ename | job | mgr | hiredate | sal | comm | deptno
-----+-----+-----+-----+-----+-----+-----+-----
 7654 | MARTIN | SALEMAN | 7698 | 2022-11-08 00:00:00 | 12000 | 1400 | 30
 7566 | JONES | MANAGER | 7839 | 2022-11-08 00:00:00 | 32000 | 0 | 20
 7499 | ALLEN | SALEMAN | 7698 | 2022-11-08 00:00:00 | 16000 | 300 | 30
(3 rows)
```

```
SELECT count(*) FROM (select deptno, string_agg(ename, ',') from employee group by deptno) t1, (select
deptno, string_agg(ename, ',') from employee group by deptno) t2 where t1.string_agg = t2.string_agg;
count
-----
      2
(1 row)
```

```
SELECT count(*) FROM (select deptno, string_agg(ename, ',') from employee group by deptno) t1, (select
deptno, string_agg(ename, ',') from employee group by deptno) t2 where t1.string_agg = t2.string_agg;
count
-----
      1
(1 row)
```

Analysis:

The **string_agg** function is used to concatenate data in a group into one row. However, if you use **string_agg(ename, ',')**, the order of concatenated results needs to be specified. For example, in the preceding statement, **select deptno, string_agg(ename, ',') from employee group by deptno;**

can output either of the following:

```
30 | ALLEN,MARTIN
```

Or:

```
30 | MARTIN,ALLEN
```

In the preceding scenario, the result of subquery **t1** may be different from that of subquery **t2** when deptno is **30**.

Solution:

Add **ORDER BY** to **String_agg** to ensure that data is concatenated in sequence.

```
SELECT count(*) FROM (select deptno, string_agg(ename, ',' order by ename desc) from employee group by deptno) t1 ,(select deptno, string_agg(ename, ',' order by ename desc) from employee group by deptno) t2 where t1.string_agg = t2.string_agg;
```

Database Compatibility Mode

Scenario: The query results of empty strings in the database are inconsistent.

database1 (TD-compatible):

```
td=# select " is null;
isnull
-----
f
(1 row)
```

database2 (ORA compatible):

```
ora=# select " is null;
isnull
-----
t
(1 row)
```

Analysis:

The empty string query results are different because the syntax of the empty string is different from that of the null string in different database compatibility.

Currently, GaussDB(DWS) supports three types of database compatibility: Oracle, TD, and MySQL. The syntax and behavior vary depending on the compatibility mode. For details about the compatibility differences, see "Syntax Compatibility Differences Among Oracle, Teradata, and MySQL" in *GaussDB(DWS) Developer Guide*.

Databases in different compatibility modes have different compatibility issues. You can run **select datname, datcompatibility from pg_database;** to check the database compatibility.

Solution:

The problem is solved when the compatibility modes of the databases in the two environments are set to the same. The **DBCMPATIBILITY** attribute of a database does not support **ALTER**. You can only specify the same **DBCMPATIBILITY** attribute when creating a database.

The configuration item **behavior_compat_options** for database compatibility behaviors is configured inconsistently.

Scenario: The calculation results of the **add_months** function are inconsistent.

database1:

```
SELECT add_months('2018-02-28',3) from dual;
add_months
-----
2018-05-28 00:00:00
(1 row)
```

database2:

```
SELECT add_months('2018-02-28',3) from dual;
add_months
-----
2018-05-31 00:00:00
(1 row)
```

Analysis:

Some behaviors may vary depending on the settings of the database compatibility configuration item **behavior_compat_options**. For details about the options of this item, see "GUC Parameters > Miscellaneous Parameters > behavior_compat_options" in *GaussDB(DWS) Developer Guide*.

The **end_month_calculate** in **behavior_compat_options** controls the calculation logic of the **add_months** function. If this parameter is specified, and the **Day** of **param1** indicates the last day of a month shorter than **result**, the **Day** in the calculation result will equal that in **result**.

Solution:

The **behavior_compat_options** parameter must be configured consistently. This parameter is of the **USERSET** type and can be set at the session level or modified at the cluster level.

The attributes of the user-defined function are not properly set.

Scenario: When the customized function **get_count()** is invoked, the results are inconsistent.

```
CREATE FUNCTION get_count() returns int
SHIPPABLE
as $$
declare
    result int;
begin
result = (select count(*) from test); --test table is a hash table.
    return result;
end;
$$
language plpgsql;
```

Call this function.

```
SELECT get_count();
get_count
-----
    2106
(1 row)

SELECT get_count() FROM t_src;
get_count
-----
    1032
(1 row)
```

Analysis:

This function specifies the **SHIPPABLE** attribute. When a plan is generated, the function pushes it down to DN for execution. The test table defined in the function is a hash table. Therefore, each DN has only part of the data in the table, the result returned by **select count(*) from test;** is not the result of full data in the test table. The expected result changes after **from** is added.

Solution:

Use either of the following methods (the first method is recommended):

1. Change the function to not push down: **ALTER FUNCTION get_count() not shippable;**
2. Change the table used in the function to a replication table. In this way, the full data of the table is stored on each DN. Even if the plan is pushed down to DNs for execution, the result set will be as expected.

Using the Unlogged Table

Scenario:

After an unlogged table is used and the cluster is restarted, the associated query result set is abnormal, and some data is missing in the unlogged table.

Analysis:

If **max_query_retry_times** is set to **0** and the keyword **UNLOGGED** is specified during table creation, the created table will be an unlogged table. Data written to unlogged tables is not written to the write-ahead log, which makes them considerably faster than ordinary tables. However, an unlogged table is automatically truncated after a crash or unclean shutdown, incurring data loss risks. The contents of an unlogged table are also not replicated to standby servers. Any indexes created on an unlogged table are not automatically logged as well. If the cluster restarts unexpectedly (process restart, node fault, or cluster restart), some data in the memory is not flushed to disks in a timely manner, and some data is lost, causing the result set to be abnormal.

Solution:

The security of unlogged tables cannot be ensured if the cluster goes faulty. In most cases, unlogged tables are only used as temporary tables. If a cluster is faulty, you need to rebuild the unlogged table or back up the data and import it to the database again to ensure that the data is normal.

19.2.19 Which System Catalogs That the VACUUM FULL Operation Cannot Be Performed on?

VACUUM FULL can be performed on all GaussDB(DWS) system catalogs. However, during the process, level 8 locks will be imposed on the system catalogs, and services involving these system catalogs will be blocked.

The suggestions are based on database versions:

8.1.3 and Later Versions

- For clusters of version 8.1.3 or later, **AUTO VACUUM** is enabled by default (controlled by the **autovacuum** parameter). After you set the parameter, the system automatically performs **VACUUM FULL** on all system catalogs and row-store tables.
 - If the value of **autovacuum_max_workers** is **0**, neither on the system catalogs nor on ordinary tables will **VACUUM FULL** be automatically performed.
 - If **autovacuum** is set to **off**, **VACUUM FULL** will be automatically performed on ordinary tables, but not system catalogs.

- This applies only to row-store tables. To automatically trigger **VACUUM** for column-store tables, you need to configure intelligent scheduling tasks on the management console.

8.1.1 and Earlier Versions

1. Reforming **VACUUM FULL** on the following system catalogs affects all services. Perform this operation in an idle time window or when services are stopped.
 - **pg_statistic** (Statistics information. You are advised not to clear it because it affects service query performance.)
 - pg_attribute
 - pgxc_class
 - pg_type
 - pg_depend
 - pg_class
 - pg_index
 - pg_proc
 - pg_partition
 - pg_object
 - pg_shdepend
2. The following system catalogs affect resource monitoring and table size query interfaces, but do not affect other services.
 - gs_wlm_user_resource_history
 - gs_wlm_session_info
 - gs_wlm_instance_history
 - gs_respool_resource_history
 - pg_relfilenode_size
3. Other system catalogs do not occupy space and do not need to be cleared.
4. During routine O&M, you are advised to monitor the sizes of these system catalogs, and collect statistics every week. If the space must be reclaimed, clear the space based on the sizes of the system tables.

The statement is as follows:

```
SELECT c.oid,c.relname, c.relkind, pg_relation_size(c.oid) AS size FROM pg_class c WHERE c.relkind IN ('r') AND c.oid <16385 ORDER BY size DESC;
```

19.2.20 In Which Scenarios Would a Statement Be "idle in transaction"?

When user SQL information is queried in the **PGXC_STAT_ACTIVITY** view, the **state** column in the query result sometimes shows **idle in transaction**. **idle in transaction** indicates that the backend is in a transaction, but no statement is being executed. This status indicates that a statement has been executed. Therefore, the value of **query_id** is 0, but the transaction has not been committed or rolled back. Statements in this state have been executed and do not occupy CPU and I/O resources, but they occupy connection resources such as connections and concurrent connections.

If a statement is in the **idle in transaction** state, rectify the fault by referring to the following common scenarios and solutions:

Scenario 1: A Transaction Is Started But Not Committed, and the Statement Is in the "idle in transaction" State

BEGIN/START TRANSACTION is manually executed to start a transaction. After statements are executed, **COMMIT/ROLLBACK** is not executed. View the **PGXC_STAT_ACTIVITY**:

```
SELECT state, query, query_id FROM pgxc_stat_activity;
```

The result shows that the statement is in the **idle in transaction** state.

state	query	query_id
active		0
idle		0
idle		0
active	WLM fetch collect info from data nodes	73464968921613282
active	WLM calculate space info process	0
active	WLM monitor update and verify local info	73464968921613276
active	WLM arbiter sync info by CCN and CNS	0
idle in transaction	select count(1) from t group by a order by 1 desc limit 1;	0
idle		0
active	select state,query,query_id from pgxc_stat_activity;	73464968921613283
active		0
idle		0
idle		0
active	WLM fetch collect info from data nodes	145522562959541153
active	WLM calculate space info process	0
active	WLM monitor update and verify local info	145522562959541123
active	WLM arbiter sync info by CCN and CNS	0
active	SELECT * FROM pg_stat_activity	73464968921613283
idle		0

(19 rows)

Solution: Manually execute **COMMIT/ROLLBACK** on the started transaction.

Scenario 2: After a DDL Statement in a Stored Procedure Is Executed, Other Nodes of the Stored Procedure Is In the "idle in transaction" State

Create a stored procedure:

```
CREATE OR REPLACE FUNCTION public.test_sleep()
RETURNS void
LANGUAGE plpgsql
AS $$
BEGIN
    truncate t1;
    truncate t2;
    EXECUTE IMMEDIATE 'select pg_sleep(6)';
    RETURN;
END$$;
```

View the **PGXC_STAT_ACTIVITY** view:

```
SELECT coorname,pid,query_id,state,query,username FROM pgxc_stat_activity WHERE username='jack';
```

The result shows that **truncate t2** is in the **idle in transaction** state and **coorname** is **coordinator2**. This indicates that the statement has been executed on **cn2** and the stored procedure is executing the next statement.

coorname	pid	query_id	state	query	username
coordinator1	139767124588288	73464968921614213	active	select test_sleep();	jack
coordinator2	140055318353664	0	idle in transaction	truncate t2	jack

(2 rows)

Solution: This problem is caused by slow execution of the stored procedure. Wait until the execution of the stored procedure is complete. You can also optimize the statements that are executed slowly in the stored procedure.

Scenario 3: A Large Number of SAVEPOINT/RELEASE Statements Are in the "idle in transaction" State (Cluster Versions Earlier Than 8.1.0)

View the `PGXC_STAT_ACTIVITY` view:

```
SELECT coorname,pid,query_id,state,query,username FROM pgxc_stat_activity WHERE username='jack';
```

The result shows that the SAVEPOINT/RELEASE statement is in the **idle in transaction** state.

coorname	pid	query_id	state	query	username
coordinator1	140127877723904	77687093572141691	active	select test sleep();	jack
coordinator2	139773127153408	0	idle in transaction	release s1	jack
coordinator3	140193352906496	0	idle in transaction	release s1	jack

(3 rows)

Solution:

SAVEPOINT and **RELEASE** statements are automatically generated by the system when a stored procedure with **EXCEPTION** is executed. In versions later than 8.1.0, **SAVEPOINT** is not delivered to CNs. GaussDB(DWS) stored procedures with **EXCEPTION** are implemented based on subtransactions, the mapping is as follows:

```
begin
  (Savepoint s1)
  DDL/DML
exception
  (Rollback to s1)
  (Release s1)
...
end
```

If there is **EXCEPTION** in a stored procedure when it is started, a subtransaction will be started. If there is an exception during the execution, the current transaction is rolled back and the exception is handled; if there is no exception, the subtransaction is committed.

This problem may occur when there are many such stored procedures and the stored procedures are nested. Similar to scenario 2, you only have to wait after the entire stored procedure is executed. If there are a large number of **RELEASE** messages, the stored procedure triggered multiple exceptions. In this case, you must re-examine the logic of the stored procedure.

19.2.21 How Does GaussDB(DWS) Implement Row-to-Column and Column-to-Row Conversion?

This section describes how to use SQL statements to convert rows to columns and convert columns to rows in GaussDB(DWS).

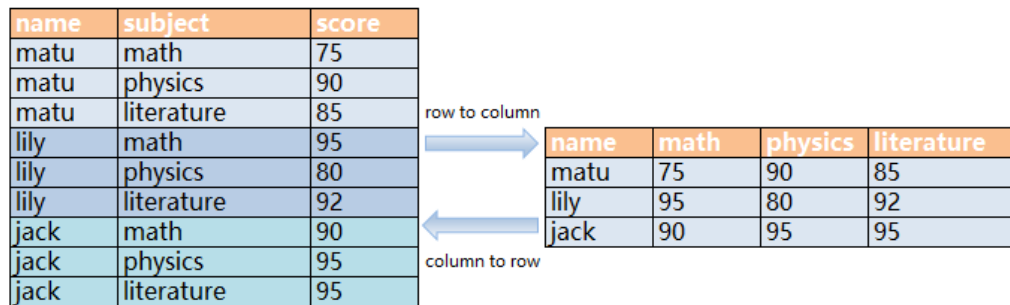
Scenario

Use a student score table as an example:

Teachers record the score of each subject of each student in a table, but students care only about their own scores. A student needs to use row-to-column conversion to view their scores of all subjects. If the teacher of a subject wants to view the scores of all students of that subject, the teacher needs to use the column-to-row conversion.

The following figure shows the row-to-column and column-to-row conversion.

Figure 19-1 Diagram



- Rows-to-column conversion
Convert multiple rows of data into one row, or convert one column of data into multiple columns.
- Column-to-row conversion
Convert a row of data into multiple rows, or convert multiple columns of data into one column.

Example

- Create a row-store table **students_info**, and insert data into the table.

```
CREATE TABLE students_info(name varchar(20),subject varchar(100),score bigint) distribute by hash(name);
INSERT INTO students_info VALUES('lily','math',95);
INSERT INTO students_info VALUES('lily','physics',80);
INSERT INTO students_info VALUES('lily','literature',92);
INSERT INTO students_info VALUES('matu','math',75);
INSERT INTO students_info VALUES('matu','physics',90);
INSERT INTO students_info VALUES('matu','literature',85);
INSERT INTO students_info VALUES('jack','math',90);
INSERT INTO students_info VALUES('jack','physics',95);
INSERT INTO students_info VALUES('jack','literature',95);
```

View information about the **students_info** table.

```
SELECT * FROM students_info;
name | subject | score
-----+-----+-----
matu | math    | 75
matu | physics | 90
matu | literature | 85
lily | math    | 95
lily | physics | 80
lily | literature | 92
jack | math    | 90
jack | physics | 95
jack | literature | 95
```

- Create a column-store table **students_info1**, and insert data into the table.

```
CREATE TABLE students_info1(name varchar(20), math bigint, physics bigint, literature bigint) with
(orientation = column) distribute by hash(name);
INSERT INTO students_info1 VALUES('lily',95,80,92);
INSERT INTO students_info1 VALUES('matu',75,90,85);
INSERT INTO students_info1 VALUES('jack',90,95,95);
```

View information about table **students_info1**.

```
SELECT * FROM students_info1;
name | math | physics | literature
-----+-----+-----+-----
matu | 75 | 90 | 85
lily | 95 | 80 | 92
jack | 90 | 95 | 95
(3 rows)
```

Static row-to-column conversion

Static row-to-column conversion requires you to manually specify the column names using the given values. If no value is given to a column, the default value 0 is assigned to the column.

```
SELECT name,
sum(case when subject='math' then score else 0 end) as math,
sum(case when subject='physics' then score else 0 end) as physics,
sum(case when subject='literature' then score else 0 end) as literature FROM students_info GROUP BY
name;
name | math | physics | literature
-----+-----+-----+-----
matu | 75 | 90 | 85
lily | 95 | 80 | 92
jack | 90 | 95 | 95
(3 rows)
```

Dynamic row-to-column conversion

For clusters of 8.1.2 or later, you can use **GROUP_CONCAT** to generate column-store statements.

```
SELECT group_concat(concat('sum(IF(subject = ', subject, ', score, 0)) AS ', name, ', '))FROM students_info;
group_concat
-----
-----
-----
-----
-----
sum(IF(subject = 'literature', score, 0)) AS "jack",sum(IF(subject = 'literature', score, 0)) AS
"lily",sum(IF(subject = 'literature', score, 0)) AS "matu",sum(IF(subject = 'math', score, 0)) AS "jack",sum(IF
(subject = 'math', score, 0)) AS "lily",sum(IF(subject = 'math', score, 0)) AS "matu",sum(IF(subject =
'physics', score, 0)) AS "jack",sum(IF(subject = 'physics', score, 0)) AS "lily",sum(IF(subject = 'physics
', score, 0)) AS "matu"
(1 row)
```

In 8.1.1 and earlier versions, you can use **LISTAGG** to generate column-store statements.

```
SELECT listagg(concat('sum(case when subject = ', subject, ' then score else 0 end) AS ', subject, ',,')
within GROUP(ORDER BY 1)FROM (select distinct subject from students_info);
listagg
-----
-----
--
sum(case when subject = 'literature' then score else 0 end) AS "literature",sum(case when subject =
'physics' then score else 0 end) AS "physics",sum(case when subject = 'math' then score else 0 end) AS
"math
```

```
"  
(1 row)
```

Dynamically rebuild the view:

```
CREATE OR REPLACE FUNCTION build_view()  
RETURNS VOID  
LANGUAGE plpgsql  
AS $$ DECLARE  
sql text;  
rec record;  
BEGIN  
sql := 'select LISTAGG(  
  CONCAT( "sum(case when subject = ''''', subject, '''' then score else 0 end) AS ""', subject, '''' )  
  ,','' ) within group(order by 1) from (select distinct subject from students_info);'  
EXECUTE sql INTO rec;  
sql := 'drop view if exists get_score';  
EXECUTE sql;  
sql := 'create view get_score as select name, ' || rec.LISTAGG || ' from students_info group by name';  
EXECUTE sql;  
END$$;
```

Rebuild the database:

```
CALL build_view();
```

Query view:

```
SELECT * FROM get_score;  
name | literature | physics | math  
-----+-----+-----+-----  
matu |      85 |      90 |      75  
lily |      92 |      80 |      95  
jack |      95 |      95 |      90  
(3 rows)
```

Column-to-Row Conversion

Use **UNION ALL** to merge subjects (math, physics, and literature) into one column. The following is an example:

```
SELECT * FROM  
(  
  SELECT name, 'math' AS subject, math AS score FROM students_info1  
  union all  
  SELECT name, 'physics' AS subject, physics AS score FROM students_info1  
  union all  
  SELECT name, 'literature' AS subject, literature AS score FROM students_info1  
)  
order by name;  
name | subject | score  
-----+-----+-----  
jack | math    | 90  
jack | physics | 95  
jack | literature | 95  
lily | math    | 95  
lily | physics | 80  
lily | literature | 92  
matu | math    | 75  
matu | physics | 90  
matu | literature | 85  
(9 rows)
```

19.2.22 What Are the Differences Between Unique Constraints and Unique Indexes?

- The concepts of a unique constraint and a unique index are different.
A unique constraint specifies that the values in a column or a group of columns are all unique. If **DISTRIBUTE BY REPLICATION** is not specified, the column table that contains only unique values must contain distribution columns.
A unique index is used to ensure the uniqueness of a field value or the value combination of multiple fields. **CREATE UNIQUE INDEX** creates a unique index.
- The functions of a unique constraint and a unique index are different.
Constraints are used to ensure data integrity, and indexes are used to facilitate query.
- The usages of a unique constraint and a unique index are different.
 - a. Both unique constraints and unique indexes can be used to ensure the uniqueness of column values which can be NULL.
 - b. When a unique constraint is created, a unique index with the same name is automatically created. The index cannot be deleted separately. When the constraint is deleted, the index is automatically deleted. A unique constraint uses a unique index to ensure data uniqueness. GaussDB(DWS) row-store tables support unique constraints, but column-store tables do not.
 - c. A created unique index is independent and can be deleted separately. Currently in GaussDB(DWS), unique indexes can only be created using B-Tree.
 - d. If you want to have both a unique constraint and a unique index on a column, and they can be deleted separately, you can create a unique index and then a unique constraint with the same name.
 - e. If a field in a table is to be used as a foreign key of another table, the field must have a unique constraint (or it is a primary key). If the field has only a unique index, an error is reported.

Example: Create a composite index for two columns, which is not required to be a unique index.

```
CREATE TABLE t (n1 number,n2 number,n3 number,PRIMARY KEY (n3));  
CREATE INDEX t_idx ON t(n1,n2);
```

GaussDB (DWS) supports multiple unique indexes for a table.

```
CREATE UNIQUE INDEX u_index ON t(n3);  
CREATE UNIQUE INDEX u_index1 ON t(n3);
```

You can use the index **t_idx** created in the example above to create a unique constraint **t_uk**, which is unique only on column **n1**. A unique constraint is stricter than a unique index.

```
ALTER TABLE t ADD CONSTRAINT t_uk UNIQUE USING INDEX u_index;
```

19.2.23 What Are the Differences Between Functions and Stored Procedures?

Functions and stored procedures are two common objects in database management systems. They have similarities and differences in implementing specific functions. Understanding their characteristics and application scenarios is important for properly designing the database structure and improving database performance.

Table 19-2 Differences between functions and stored procedures

Function	Stored procedures
Both can be used to implement specific functions. Both functions and stored procedures can encapsulate a series of SQL statements to complete certain specific operations.	
Both can receive input parameters and perform corresponding operations based on the parameters.	
The identifier of a function is FUNCTION .	The identifier of the stored procedure is PROCEDURE .
A function must return a specific value of the specified numeric type.	A stored procedure can have no return value, one return value, or multiple return values. You can use output parameters to return results or directly use the SELECT statement in a stored procedure to return result sets.
Functions are used to return single values, for example, a number calculation result, a string processing result, or a table.	Stored procedures are used for DML operations, for example, inserting, updating, and deleting data in batches.

- **Creating and Invoking a Function**

Create the **emp** table and insert data into the table. The table data is as follows:

```
SELECT * FROM emp;
empno | ename | job | mgr | hiredate | sal | comm | deptno
-----+-----+-----+-----+-----+-----+-----+-----
7369 | SMITH | CLERK | 7902 | 1980-12-17 00:00:00 | 800.00 | | 20
7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 00:00:00 | 1600.00 | 300.00 | 30
7566 | JONES | MANAGER | 7839 | 1981-04-02 00:00:00 | 2975.00 | | 20
7521 | WARD | SALESMAN | 7698 | 1981-02-22 00:00:00 | 1250.00 | 500.00 | 30
(4 rows)
```

Create the **emp_comp** function to accept two numbers as input and return the calculated value.

```
CREATE OR REPLACE FUNCTION emp_comp (
  p_sal NUMBER,
  p_comm NUMBER
) RETURN NUMBER
IS
BEGIN
```

```
RETURN (p_sal + NVL(p_comm, 0)) * 24;
END;
/
```

Run the **SELECT** command to invoke the function:

```
SELECT ename "Name", sal "Salary", comm "Commission", emp_comp(sal, comm) "Total
Compensation" FROM emp;
Name | Salary | Commission | Total Compensation
-----+-----+-----+-----
SMITH | 800.00 | | 19200.00
ALLEN | 1600.00 | 300.00 | 45600.00
JONES | 2975.00 | | 71400.00
WARD | 1250.00 | 500.00 | 42000.00
(4 rows)
```

- **Creating and Invoking a Stored Procedure**

Create the **MATCHES** table and insert data into the table. The table data is as follows:

```
SELECT * FROM MATCHES;
matchno | teamno | playerno | won | lost
-----+-----+-----+-----+-----
1 | 1 | 6 | 3 | 1
7 | 1 | 57 | 3 | 0
8 | 1 | 8 | 0 | 3
9 | 2 | 27 | 3 | 2
11 | 2 | 112 | 2 | 3
(5 rows)
```

Create the stored procedure **delete_matches** to delete all matches that a specified player participates in.

```
CREATE PROCEDURE delete_matches(IN p_playerno INTEGER)
AS
BEGIN
DELETE FROM MATCHES WHERE playerno = p_playerno;
END;
/
```

Invoke the stored procedure **delete_matches**.

```
CALL delete_matches(57);
```

Query the **MATCHES** table again. The returned result indicates that the data of the player whose **playerno** is **57** has been deleted.

```
SELECT * FROM MATCHES;
matchno | teamno | playerno | won | lost
-----+-----+-----+-----+-----
11 | 2 | 112 | 2 | 3
8 | 1 | 8 | 0 | 3
1 | 1 | 6 | 3 | 1
9 | 2 | 27 | 3 | 2
(4 rows)
```

19.2.24 How Do I Delete Duplicate Table Data?

When clearing dirty data in the database, you may retain only one piece of duplicate data. In this scenario, you can use the aggregate function or window function.

Constructing Table Data

Step 1 Create a table **t_customer** and insert data that contains duplicate records into the table.

```
CREATE TABLE t_customer (
id int NOT NULL,
cust_name varchar(32) NOT NULL COMMENT' Name',
gender varchar(10) NOT NULL COMMENT' Gender',
```

```
email varchar(32) NOT NULL COMMENT 'email',
PRIMARY KEY (id)
);

INSERT INTO t_customer VALUES ('1', 'Tom', 'Male', 'high_salary@sample.com');
INSERT INTO t_customer VALUES ('2', 'Jennifer', 'Female', 'good_job@sample.com');
INSERT INTO t_customer VALUES ('3', 'Tom', 'Male', 'high_salary@sample.com');
INSERT INTO t_customer VALUES ('4', 'John', 'Male', 'good_job@sample.com');
INSERT INTO t_customer VALUES ('5', 'Jennifer', 'Female', 'good_job@sample.com');
INSERT INTO t_customer VALUES ('6', 'Tom', 'Male', 'high_salary@sample.com');
```

Step 2 Query the t_customer table.

```
SELECT * FROM t_customer ORDER BY id;
```

id	cust_name	gender	email
1	Tom	Male	high_salary@sample.com
2	Jennifer	Female	good_job@sample.com
3	Tom	Male	high_salary@sample.com
4	John	Male	good_job@sample.com
5	Jennifer	Female	good_job@sample.com
6	Tom	Male	high_salary@sample.com

----End

If the name, gender, and email of a customer are the same, the customer is regarded as a duplicate record. In the t_customer table, data whose IDs are 1, 3, and 6 is duplicate, and data whose IDs are 2 and 5 is also duplicate. Delete redundant data and retain one of them.

Method 1: Use the aggregate function **min(expr)**.

Use aggregate functions to obtain non-duplicate rows with the smallest ID through subqueries, and then use NOT IN to delete duplicate data.

Step 1 Run the following command to query the unique row with the smallest ID:

```
SELECT
  min(id) id,
  cust_name,
  gender,
  COUNT( cust_name ) count
FROM t_customer
GROUP BY cust_name,gender
ORDER BY id;
```

id	cust_name	gender	count
1	Tom	Male	3
2	Jennifer	Female	2
4	John	Male	1

According to the query result, duplicate data rows whose IDs are 3, 5, and 6 are filtered out.

Step 2 Use NOT IN to filter out duplicate data rows and delete them.

```
DELETE from t_customer where id not in (
  SELECT
  min(id) id
  FROM t_customer
  GROUP BY cust_name,gender
);
```

Step 3 Query the t_customer table after duplicate data is deleted.


```
SELECT * FROM t_customer ORDER BY id;
```

id	cust_name	gender	email
1	Tom	Male	high_salary@sample.com
2	Jennifer	Female	good_job@sample.com
4	John	Male	good_job@sample.com

The command output indicates that duplicate data has been deleted.

----End

Method 2: Use the window function row_number().

Use PARTITION BY to partition and sort columns, generate sequence number columns, and delete rows whose sequence numbers are greater than 1.

Step 1 Partition query. Sort columns by partition and generate sequence number columns.

```
SELECT
  id,
  cust_name,
  gender,
  ROW_NUMBER() OVER (PARTITION BY cust_name,gender ORDER BY id) num
FROM t_customer;
```

id	cust_name	gender	num
4	John	Male	1
1	Tom	Male	1
3	Tom	Male	2
6	Tom	Male	3
2	Jennifer	Female	1
5	Jennifer	Female	2

According to the command output, the data in num>1 is duplicate.

Step 2 Delete the data of num>1.

```
DELETE FROM t_customer WHERE id in (
  SELECT id FROM(
    SELECT * FROM (
      SELECT ROW_NUMBER() OVER w AS row_num,id
    FROM t_customer
    WINDOW w AS (PARTITION BY cust_name,gender ORDER BY id) )
    WHERE row_num >1 )
);
```

Step 3 Query the t_customer table after duplicate data is deleted.

```
SELECT * FROM t_customer ORDER BY id;
```

id	cust_name	gender	email
1	Tom	Male	high_salary@sample.com
2	Jennifer	Female	good_job@sample.com
4	John	Male	good_job@sample.com

----End

19.3 Cluster Management

19.3.1 What Do I Do If Creating a GaussDB(DWS) Cluster Failed?

Possible Causes

Check that you have enough quota for creating the cluster.

Contacting Customer Service

If the fault cannot be identified, submit a service ticket to report the problem: Log in to the management console and choose .

19.3.2 How Can I Clear and Reclaim the Storage Space?

After you delete data stored in GaussDB(DWS) data warehouses, dirty data may be generated possibly because the disk space is not released. This results in disk space waste and deteriorates snapshot creation and restoration performance. The following describes the impact on the system and subsequent operation to clear the disk space:

Points worth mentioning during clearing and reclaiming storage space:

- Unnecessary data needs to be deleted to release the storage space.
- Frequent read and write operations may affect proper database use. Therefore, it is good practice to clear and reclaim the storage space when not in peak hours.
- The data clearing time depends on the data stored in the database.

Perform the following steps to clear and reclaim the storage space:

1. Run the following command to clear and reclaim the storage space:

VACUUM FULL;

By default, tables the current user has the permission on are deleted. Other tables are skipped.

The following information is displayed once the space is cleared:

VACUUM

NOTE

- **VACUUM FULL** reclaims all expired row space, however it requires an exclusive lock on each table being processed, and might take a long time to complete on large, distributed database tables. You are advised to do **VACUUM FULL** to specified tables. If you want to do **VACUUM FULL** to the entire database, you are advised to do it during database maintenance.
- The statistical information will be lost if you use the **FULL** parameter. To collect the statistics, add keyword **ANALYZE**, for example, **VACUUM FULL ANALYZE;**

19.3.3 Why Did the Used Storage Shrink After Scale-out?

Cause Analysis

If you do not run **VACUUM** to clear and reclaim the storage space before the scale-out, the data deleted from GaussDB(DWS) may not free up the occupied disk space.

During the scale-out, the system redistributes the data because the service data volume on the original nodes is significantly larger than that on the newly added nodes. When the redistribution starts, the system automatically performs **VACUUM** to free up the storage space. In this way, the used storage is reduced.

Handling Procedure

You are advised to periodically clear and reclaim the storage space by running **VACUUM FULL** to prevent data expansion.

If the used storage space is still large after you run **VACUUM FULL**, analyze whether the existing cluster flavor meets service requirements. If no, scale out the cluster.

19.3.4 How Do I View Node Metrics (CPU, Memory, and Disk Usage)?

You can view the used capacity of a cluster CPU, memory, and disks on the Cloud Eye management console. Perform the following steps to view the information:

Step 1 Log in to the GaussDB(DWS) console and click **Viewing Metric** next to a cluster.

Step 2 Click  to return to the **Cloud Service Monitoring** page, switch to the **Data Warehouse Node** page, and click **View Metric** on the right of the corresponding node to view its disk usage.

----End

19.3.5 How Is the Disk Space or Capacity of GaussDB(DWS) Calculated?

1. Total disk capacity of GaussDB(DWS): For a three-node cluster, if each node is 320 GB, the total capacity is 960 GB. When 1 GB data is stored, GaussDB(DWS) stores 1 GB data on two nodes due to duplication, a security mechanism, thereby occupying a total of 2 GB space. As a result, more than 2 GB space is occupied if metadata and indexes are calculated. Therefore, a three-node cluster with a total capacity of 960 GB can store 480 GB data. This mechanism ensures data security.

When you nodes on the console, you are billed by the available capacity of a node. For example, the actual space of **dws.m3.xlarge** is 320 GB and the available space displayed is 160 GB, the space you will be billed for.

2. Check the disk usage of a single node.

Similarly, if the total capacity is 960 GB and there are three data nodes, the disk capacity of each node is 320 GB.

Log in to the Gauss(DWS) console and choose **Monitoring > Node Monitoring > Overview** to view the usage of disks and other resources on each node.

NOTE

- The disk space displayed on the **Node Management** page is the total capacity of all disks, that is, system disks and data disks, in the data warehouse cluster. The disk space displayed on the **Overview** page is only the available space for storing table data in the cluster. In addition, tables in the data warehouse cluster have backup copies, these copies also occupy the disk storage.
- If the cluster is read-only and an alarm for insufficient disk space is generated, expand the cluster capacity by following the instructions provided in *Scaling Out a Cluster*.

19.3.6 What Are the gaussdb and postgres Databases of GaussDB(DWS)?

The **gaussdb** and **postgres** databases are built-in databases of GaussDB(DWS). You can create schemas and tables in them. However, you are advised to recreate a database and create schemas and tables in the new database.


19.3.7 How Do I Set the Maximum Number of Sessions When Adding an Alarm Rule on Cloud Eye?

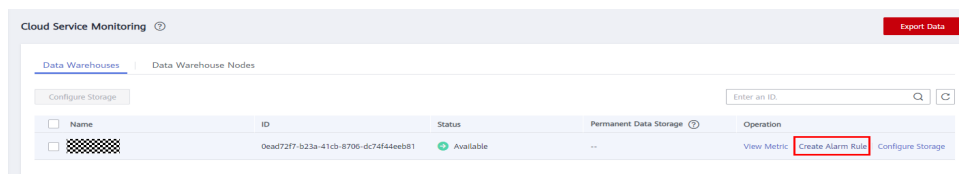
After connecting to a database, run the following SQL statement to check the maximum number of concurrent sessions globally:

```
show max_active_statements;
```

Go to the Cloud Eye console and set the threshold to 70% to 80% of the obtained value. For example, if the value of **max_active_statements** is **80**, set the threshold to **56** (80 x 70%).

Procedure:

1. On the GaussDB(DWS) management console, choose **Clusters > Dedicated Clusters**.
2. Click **View Metric** in the **Operation** column of the target cluster to go to the Cloud Eye console.
3. Click  in the upper left corner on the displayed page and click **Create Alarm Rule** of the target cluster.



4. Set **Method** to **Configure manually**, **Metric Name** to **Session Count**, **Alarm Policy** to **56**, and **Alarm Severity** to **Major**. Then click **Create**.

The screenshot displays the configuration page for an alarm policy. Key elements include:

- Name:** alarm-hjvd
- Description:** (Empty text area, 0/256 characters)
- Enterprise Project:** default (with a link to 'Create Enterprise Project')
- Resource Type:** Data Warehouse Service
- Dimension:** Data Warehouses
- Monitoring Scope:** Specific resources
- Monitored Object:** (Checkered icon)
- Method:** 'Use template' and 'Configure manually' (highlighted with a red box)
- Alarm Policy Table:**

Metric Name	Alarm Policy	Alarm Severity	Operation
Session Count (highlighted with a red box)	Raw d... (highlighted with a red box)	3 consecuti...	>= 56 (highlighted with a red box) One day
- Footer:** Add Alarm Policy You can add 19 more.

19.3.8 When Should I Add CNs or Scale out a cluster?

Introduction to CN Concurrency

Coordinator Node (CN) is an important component in GaussDB(DWS) that is most closely related to users. It provides external application interfaces, optimizes global execution plans, distributes the execution plans to DataNodes, and summarizes and processes execution results. A CN is an interface to external applications. The concurrency capability of the CN determines the service concurrency.

CN concurrency is determined by the following parameters:

- **max_connections:** specifies the maximum number of concurrent connections to the database. This parameter affects the concurrent processing capability of the cluster. The default value depends on the cluster specifications. For details, see "Managing Database Connections".
- **max_active_statements:** specifies the maximum number of concurrent jobs. This parameter applies to all the jobs on one CN. The default value is **60**, which indicates a maximum of 60 jobs can run at the same time. Other jobs will be queued.

Add CNs or Scale out a Cluster?

- **Insufficient connections:** When a cluster is created for the first time, the default number of CNs in the cluster is 3, which can meet the customer's basic connection requirements. If the cluster has a large number of concurrent requests and the number of connections to each CN is large, or the CPU usage of a CN exceeds its capacity, you are advised to add CNs. For details, see "CNs".
- **Insufficient storage capacity and performance:** If your business grows and you have higher requirements on storage capacity and performance, or the CPU of your cluster is insufficient, you are advised to scale out your cluster. For details, see "Scaling Out a Cluster".

With the expansion of cluster nodes, more CNs are needed to meet the distribution requirements of GaussDB(DWS). In short, adding CNs does not necessarily require cluster scale-out. However, after cluster scale-out, CNs may need to be added.

19.3.9 How Should I Select from a Small-Flavor Many-Node Cluster and a Large-Flavor Three-Node Cluster with Same CPU Cores and Memory?

- Small-flavor many-node:
If your data volume is small and you have requirement for node scaling, but you have limited budget, you can select a small-flavor many-node cluster.
For example, a small-flavor cluster (dwsx2.h.2xlarge.4.c6) with 8 cores and 32 GB memory can provide strong computing capabilities. The cluster has a large number of nodes and can process high concurrent requests. In this case, you only need to ensure that the network speed between nodes is normal to avoid cluster performance limitation.
- Large-flavor three-node:
If you have a large amount of data to be processed, have high requirement on computing, and have a high budget, you can select a large-flavor three-node cluster.
For example, a large-flavor cluster (dws2.m6.8xlarge.8) with 32 cores and 256 GB memory has faster CPU processing capability and larger memory, and can process data more quickly. However, the cluster has limited nodes, which may cause low performance in high-concurrency scenarios.

19.3.10 What Are the Differences Between Hot Data Storage and Cold Data Storage?

The biggest difference between hot data storage and cold data storage lies in the storage media.

- Hot data is frequently queried or updated and has high requirements on access response time. It is stored on **DN data disks**.
- Cold data is not updated and is occasionally queried, and does not have high requirements on access response time. It is stored in **OBS**.

Different storage media determine the cost, performance, and application scenarios of the two storage mode, as shown in [Table 19-3](#).

Table 19-3 Differences between hot and cold data storage

Storage	Read and Write	Cost	Capacity	Application Scenario
Hot storage	Fast	High	Fixed and restricted	This mode is applicable to scenarios where the data volume is limited and needs to be frequently read and updated.
Cold storage	Slow	Low	Large and unlimited	This mode is applicable to scenarios such as data archiving. It features low cost and large capacity.

19.3.11 What Do I do if the Scale-in Button Is Dimmed?

Symptom

When a user performs a scale-in operation, the **Scale In** button is unavailable and the user cannot proceed to the next scale-in operation.

Possible Causes

The system verifies the cluster's eligibility for scaling in before each operation. The **Scale In** button is dimmed if the cluster does not qualify.

Solution

Check the cluster configuration information and check whether the scale-in meets the following conditions:

- The cluster consists of rings of four or five hosts each, with primary, standby, and secondary DN's deployed on them. A cluster ring is the smallest unit for scaling in, which requires at least two rings. The system removes nodes from the last ring to the first when scaling in.
- The removed nodes cannot contain the GTM, CM Server, or CN component.
- The cluster status is **Normal**, and no other task information is displayed.
- The cluster tenant account cannot be in the read-only, frozen, or restricted state.
- The cluster is not in logical cluster mode.

19.4 Database Connections

19.4.1 How Applications Communicate with GaussDB(DWS)?

For applications to communicate with GaussDB(DWS), make sure the networks between them are connected. The following table lists common connection scenarios.

Table 19-4 Communication between applications and GaussDB(DWS)

Scenario		Description	Supported Connection Type
Cloud	Service Application and GaussDB(DWS) Are in the Same VPC in the Same Region	Two private IP addresses in the same VPC can directly communicate with each other.	<ul style="list-style-type: none"> • <code>gsq</code> • Data Studio • JDBC/ODBC For more connection modes, see .
	Service Applications and GaussDB(DWS) Are in Different VPCs in the Same Region	After a VPC peering connection is created between two VPCs, the two private IP addresses can directly communicate with each other.	
	Service Applications and GaussDB(DWS) Are in Different Regions	After a cloud connection (CC) is established between two regions, the two regions communicate with each other through private IP addresses.	
On-premises and cloud	Service applications are deployed in on-premise data centers and need to communicate with GaussDB(DWS).	<ul style="list-style-type: none"> • Use the public IP address of GaussDB(DWS) for communication. • Use Direct Connect (DC) is for communication. 	

Service Application and GaussDB(DWS) Are in the Same VPC in the Same Region

To ensure low service latency, you are advised to deploy service applications and GaussDB(DWS) in the same region. For example, if a service application is deployed on an ECS, you are advised to deploy the data warehouse cluster in the same VPC as the ECS. In this way, the application can directly communicate with GaussDB(DWS) through an intranet IP address. In this case, deploy the data warehouse cluster in the same region and VPC where the ECS resides.

For example, if the ECS is deployed in , select for the GaussDB(DWS) cluster and ensure that the GaussDB(DWS) cluster and the ECS are both in **VPC1**. The private IP address of the ECS is **192.168.120.1**, the private IP address of GaussDB(DWS) is **192.168.120.2**. Therefore, they can communicate with each other through private IP addresses.

The key points in communication check are the ECS outbound rule and GaussDB(DWS) inbound rule. The check procedure is as follows:

Step 1 Check the ECS outbound rules:

Ensure that the outbound rule of the ECS security group allows access. If access is not allowed, see the .

Action ?	Protocol & Port ?	Type	Destination ?
Allow	All	IPv4	0.0.0.0/0 ?

Step 2 Check the GaussDB(DWS) inbound rules:

If no security group is configured when GaussDB(DWS) is created, the default inbound rule allows TCP access from all IPv4 addresses and port 8000. To ensure security, you can also allow only one IP address. For details, see

Allow	TCP : 8000	IPv4	0.0.0.0/0 ?
-------	------------	------	-------------

Step 3 Log in to the ECS. If the internal IP address of GaussDB(DWS) can be pinged, the network connection is normal. If the IP address cannot be pinged, check the preceding configuration. If the ECS has a firewall, check the firewall configuration.

----End

Example of using gsql for connection:

```
gsql -d gaussdb -h 192.168.120.2 -p 8000 -U dbadmin -W password -r
```

Service Applications and GaussDB(DWS) Are in Different VPCs in the Same Region

To ensure low service latency, you are advised to deploy service applications and GaussDB(DWS) in the same region. For example, if service applications are deployed on an ECS, you are advised to deploy the data warehouse cluster in the same VPC as the ECS. If a different VPC is selected for the data warehouse cluster, the ECS cannot directly connect to GaussDB(DWS).

For example, both ECS and GaussDB(DWS) are deployed in , but ECS is in VPC1 and GaussDB(DWS) is in VPC2. In this case, you need to create a between VPC1 and VPC2 so that ECS can access GaussDB(DWS) using the private IP address of GaussDB(DWS).

The key points for checking the communication are the ECS outbound rules, GaussDB(DWS) inbound rules, and VPC peering connection. The check procedure is as follows:

Step 1 Check the ECS outbound rules:

Ensure that the outbound rule of the ECS security group allows access. If access is not allowed, see the .

Action ?	Protocol & Port ?	Type	Destination ?
Allow	All	IPv4	0.0.0.0/0 ?

Step 2 Check the GaussDB(DWS) inbound rules:

If no security group is configured when GaussDB(DWS) is created, the default inbound rule allows TCP access from all IPv4 addresses and port 8000. To ensure security, you can also allow only one IP address. For details, see

Allow	TCP : 8000	IPv4	0.0.0.0/0 ?
-------	------------	------	--------------------------

Step 3 Create a between VPC1 where the ECS is and VPC2 where GaussDB(DWS) is.

Step 4 Log in to the ECS. If the internal IP address of GaussDB(DWS) can be pinged, the network connection is normal. If the IP address cannot be pinged, check the preceding configuration. If the ECS has a firewall, check the firewall configuration.

----End

Example of using gsql for connection:

```
gsql -d gaussdb -h 192.168.120.2 -p 8000 -U dbadmin -W password -r
```

Service Applications and GaussDB(DWS) Are in Different Regions

If the service application and GaussDB(DWS) are in different regions, for example, ECS is in and GaussDB(DWS) is in , you need to establish a between the two regions for communication.

Service applications are deployed in on-premise data centers and need to communicate with GaussDB(DWS).

If service applications are not on the cloud but in the local data center, they need to communicate with GaussDB(DWS) on the cloud.

- **Scenario 1:** On-premises service applications communicate with GaussDB(DWS) through GaussDB(DWS) public IP addresses.

Example of using gsql for connection:

```
gsql -d gaussdb -h public_IP_address -p 8000 -U dbadmin -W password -r
```

- **Scenario 2:** On-premises services cannot access the external network. In this case, is required for communication.

19.4.2 Does GaussDB(DWS) Support Third-Party Clients and JDBC and ODBC Drivers?

Yes, but GaussDB(DWS) clients and drivers are recommended. Unlike open-source PostgreSQL clients and drivers, GaussDB(DWS) clients and drivers have two key advantages:

- **Security hardening:** PostgreSQL drivers only support MD5 authentication, but GaussDB(DWS) drivers support SHA256 and MD5.
- **Data type enhancement:** GaussDB(DWS) drivers support new data types `smalldatetime` and `tinyint`.

GaussDB(DWS) supports open-source PostgreSQL clients and JDBC and ODBC drivers.

The compatible client and driver versions are:

- PostgreSQL `psql` 9.2.4 or later
- PostgreSQL JDBC Driver 9.3-1103 or later
- PSQL ODBC 09.01.0200 or later

For details about how to use JDBC/ODBC to connect to GaussDB(DWS), see .

19.4.3 Can I Connect to GaussDB(DWS) Cluster Nodes Using SSH?

No, you can't, VMs at the bottom layer of GaussDB(DWS) serve as the compute nodes for data analysis. However, the GaussDB(DWS) compute nodes cannot be accessed directly. You can only access cluster databases using the private or public network access address.

19.4.4 What Should I Do If I Cannot Connect to a GaussDB(DWS) Cluster?

Possible Causes

Check the following:

- Whether the cluster status is normal.
- Whether the connection command, username, password, IP address, and port number are incorrect.
- Whether the operating system type and version of the client are correct.
- Whether the client is incorrectly installed.

If cluster connection failed on the , check for the following as well:

- The ECSs are not in the same AZ, VPC, subnet, and security group as the cluster.
- Some of the inbound and outbound rules of the security group are incorrect.

If cluster connection failed through the Internet, confirm the following:

- Whether your network is connected to the Internet.
- Whether the firewall blocked the access.
- Whether you need to access the Internet through a proxy.

Contacting Customer Service

If the fault cannot be identified, submit a service ticket to report the problem: Log in to the management console and choose .

19.4.5 Why Was I Not Notified of Failure Unbinding the EIP When GaussDB(DWS) Is Connected Over the Internet?

The network is disconnected when the EIP is unbound. However, the TCP layer does not detect a faulty physical connection in time due to keepalive settings. As a result, the gsql, ODBC, and JDBC clients also cannot identify the network fault in time.

The time for the client to wait for the database to return is related to the setting of the **keepalive** parameter, and may be specifically expressed as: **keepalive_time + keepalive_probes*keepalive_intvl**.

Keepalive values affect network communication stability. Adjust them to service pressure and network conditions.

On Linux, run the **sysctl** command to modify the following parameters:

- net.ipv4.tcp_keepalive_time
- net.ipv4.tcp_keealive_probes
- net.ipv4.tcp_keepalive_intvl

For example, if you want to change the value of **net.ipv4.tcp_keepalive_time**, run the following command to change it to **120**.

```
sysctl net.ipv4.tcp_keepalive_time=120
```

On Windows, modify the following configuration information in registry **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters**:


- KeepAliveTime
- KeepAliveInterval
- TcpMaxDataRetransmissions (equivalent to **tcp_keepalive_probes**)

NOTE

If you cannot find the preceding parameters in registry **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters**, add these parameters. Open **Registry Editor**, right-click the blank area on the right, and choose **Create > DWORD (32-bit) Value** to add these parameters.

19.4.6 How Do I Configure a Whitelist to Protect Clusters Available Through a Public IP Address?

You can also log in to the VPC management console to manually create a security group. Then, go back to the page for creating data warehouse clusters, click the

 button next to the **Security Group** drop-down list to refresh the page, and select the new security group.

To enable the GaussDB(DWS) client to connect to the cluster, you need to add an inbound rule to the new security group to grant the access permission to the database port of the GaussDB(DWS) cluster.

- **Protocol: TCP**

- **Port: 8000** Use the database port set when creating the GaussDB(DWS) cluster. This port is used for receiving client connections to GaussDB(DWS).
- **Source:** Select **IP address** and use the host IP address of the client host, for example, **192.168.0.10/32**.

Figure 19-2 Adding an inbound rule

Add Inbound Rule [Learn more about security group configuration.](#)

Some security group rules will not take effect for ECSs with certain specifications. [Learn more](#)

Security Group [redacted]

You can import multiple rules in a batch.

Priority	Action	Protocol & Port	Type	Source	Description	Operation
1	Allow	Protocols/TCP (Custo... 8000	IPv4	IP address 192.168.0.10/32		Replicate Delete

+ Add Rule

OK Cancel

The whitelist will be added.

19.5 Data Import and Export

19.5.1 What Are the Differences Between Data Formats Supported by OBS and GDS Foreign Tables?

The file formats supported by OBS and GDS foreign tables are as follows:

OBS supports ORC, TEXT, JSON, CSV, CARBONDATA and PARQUET file formats for data import and ORC, CSV, and TEXT file formats for data export. The default format is TEXT.

GDS supports the following file formats: TEXT, CSV, and FIXED. The default format is TEXT.

19.5.2 How Do I Import Incremental Data Using an OBS Foreign Table?

When you use an OBS foreign table to import data, **INSERT** imports the data to a local physical table. When OBS data is updated, you do not need to run the **INSERT** statement again.

19.5.3 How Can I Import Data to GaussDB(DWS)?

GaussDB(DWS) supports efficient data import from multiple data sources. The following lists typical data import modes. For details, see section "Importing Data" in the *Data Warehouse Service (DWS) Developer Guide*.

- Importing data from the OBS
Upload data to OBS and then export it to GaussDB(DWS) clusters. Data formats such as CSV and TEXT are supported.
- Inserting data with **INSERT** statements
Use the gsql client tool provided by GaussDB(DWS) or the JDBC/ODBC driver to write data to GaussDB(DWS) from upper-layer applications. GaussDB(DWS) supports complete database transaction-level CRUD operations. This is the simplest method and is applicable to scenarios with small data volume and low concurrency.
- Importing data with the **COPY FROM STDIN** command
Run the **COPY FROM STDIN** command to write data to a table.

19.5.4 How Much Service Data Can a Data Warehouse Store?

Each node in a data warehouse cluster has a default storage capacity of. A cluster can house 3 to 256 nodes and the total storage capacity of the cluster expands proportionally as the cluster scale grows.

To enhance reliability, each node has a copy, which occupies half of the storage space.

The GaussDB(DWS) system backs up data and generates indexes, temporary cache files, and run logs, which occupy storage space. Therefore, the actual storage space of each node is about half of the total storage capacity.

19.5.5 How Do I Use \Copy to Import and Export Data?

GaussDB(DWS) is a fully managed service on the cloud. Users cannot log in to the background to import or export data by using **COPY**, so the **COPY** syntax is disabled. You are advised to store data files on OBS and use OBS foreign tables to import data. If you want to use **COPY** to import and export data, perform the following operations:

1. Place the data file on the client.
2. Use gsql to connect to the target cluster.
3. Run the following command to import data. Enter the directory name and file name of the data file on the client and specify the import option in **with**. The command is almost the same as the common **COPY** command. You only need to add a backslash (\) before the command. When the data is successfully imported, no notification will be displayed.

```
\copy tb_name from '/directory_name/file_name' with(...);
```
4. Run the following command to export data to a local file. Retain the default settings of parameters.

```
\copy table_name to '/directory_name/file_name';
```
5. Specify the **copy_option** parameter to export data to a CSV file.

```
\copy table_name to '/directory_name/file_name' CSV;
```
6. Use **with** to specify parameters, exporting data as CSV files that use vertical bars (|) as delimiters.

```
\copy table_name to '/directory_name/file_name' with(format 'csv',delimiter '|') ;
```

19.5.6 How Do I Implement Fault Tolerance Import Between Different Encoding Libraries

To import data from database A (UTF8) to database B (GBK), there may be a character set mismatch error which causes the data import to fail.

To import a small amount of data, run the `\COPY` command. The procedure is as follows:

- Step 1** Create databases A and B. The encoding format of database A is UTF8, and that of database B is GBK.

```
postgres=> CREATE DATABASE A ENCODING 'UTF8' template = template0;
postgres=> CREATE DATABASE B ENCODING 'GBK' template = template0;
```

- Step 2** View the database list. You can view the created databases A and B.

```
postgres=> \l
          List of databases
  Name | Owner | Encoding | Collate | Ctype | Access privileges
-----+-----+-----+-----+-----+-----
 a     | dbadmin | UTF8     | C       | C     |
 b     | dbadmin | GBK      | C       | C     |
 gaussdb | Ruby | SQL_ASCII | C       | C     |
 postgres | Ruby | SQL_ASCII | C       | C     |
 template0 | Ruby | SQL_ASCII | C       | C     | =c/Ruby +
        |        |          |        | Ruby=CTc/Ruby
 template1 | Ruby | SQL_ASCII | C       | C     | =c/Ruby +
        |        |          |        | Ruby=CTc/Ruby
 xiaodi  | dbadmin | UTF8     | C       | C     |
(7 rows)
```

- Step 3** Switch to database A and enter the user password. Create a table named **test01** and insert data into the table.

```
postgres=> \c a
Password for user dbadmin:
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_128_GCM_SHA256, bits: 128)
You are now connected to database "a" as user "dbadmin".

a=> CREATE TABLE test01
(
  c_customer_sk      integer,
  c_customer_id      char(5),
  c_first_name       char(6),
  c_last_name        char(8)
)
with (orientation = column,compression=middle)
distribute by hash (c_last_name);
CREATE TABLE
a=> INSERT INTO test01(c_customer_sk, c_customer_id, c_first_name) VALUES (3769, 'hello', 'Grace');
INSERT 0 1
a=> INSERT INTO test01 VALUES (456, 'good');
INSERT 0 1
```

- Step 4** Run the `\COPY` command to export data from the UTF8 library in Unicode format to the **test01.dat** file.

```
\copy test01 to '/opt/test01.dat' with (ENCODING 'Unicode');
```

- Step 5** Switch to database B and create a table with the same name **test01**.

```
a=> \c b
Password for user dbadmin:
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_128_GCM_SHA256, bits: 128)
You are now connected to database "b" as user "dbadmin".

b=> CREATE TABLE test01
(
```

```
c_customer_sk      integer,  
c_customer_id     char(5),  
c_first_name      char(6),  
c_last_name       char(8)  
)  
with (orientation = column,compression=middle)  
distribute by hash (c_last_name);
```

Step 6 Run the `\COPY` command to import the `test01.dat` file to database B.

```
\copy test01 from '/opt/test01.dat' with (ENCODING 'Unicode',COMPATIBLE_ILLEGAL_CHARS 'true');
```

 **NOTE**

- The error tolerance parameter `COMPATIBLE_ILLEGAL_CHARS` specifies that invalid characters are tolerated during data import. Invalid characters are converted and then imported to the database. No error message is displayed. The import is not interrupted.
- The `BINARY` format is not supported. When data of such format is imported, error "cannot specify bulkload compatibility options in `BINARY` mode" will occur.
- The parameter is valid only for data importing using the `COPY FROM` option.

Step 7 View data in the `test01` table in database B.

```
b=> select * from test01;  
c_customer_sk | c_customer_id | c_first_name | c_last_name  
-----+-----+-----+-----  
          3769 | hello        | Grace       |  
          456 | good         |             |  
(2 rows)
```

Step 8 After the preceding operations are performed, data is imported from database A (UTF8) to database B (GBK).

----End

19.5.7 Can I Import and Export Data to and from OBS Across Regions?

No. No, GaussDB(DWS) does not support OBS data import or export across regions. The GaussDB(DWS) cluster and OBS must be in the same region.

19.5.8 Can I Import Data over the Public/External Network Using GDS?

No. The GDS server and GaussDB(DWS) can only communicate with each other on the intranet. Each DN in the GaussDB(DWS) cluster is used to connect to the GDS server in parallel to import a large amount of data. The GDS server and the cluster must be in the same network. If GDS is deployed on an offline server, the firewall needs to be enabled and the cluster needs an EIP. However, one cluster can be bound only to one EIP, and data import with multiple DNs cannot be implemented.

19.5.9 Which Are the Factors That Affect GaussDB(DWS) Import Performance?

The GaussDB(DWS) import performance is affected by the following factors:

1. Cluster specifications: disk I/O, network throughput, memory, and CPU specifications

2. Service planning: type of table fields, compress, and row-store or column-store
3. Data storage: local cluster, OBS
4. Data import mode

19.6 Account, Password, and Permission

19.6.1 How Does GaussDB(DWS) Implement Workload Isolation?

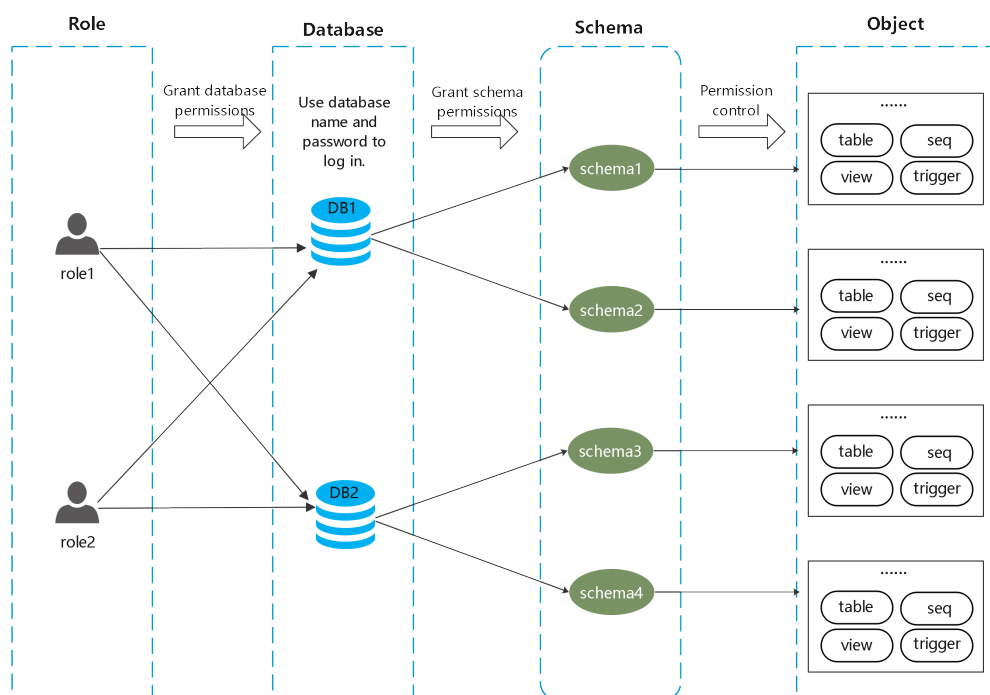
Workload Isolation

In GaussDB(DWS), you can isolate workloads through database and schema configurations. Their differences are as follows:

- Databases cannot communicate with each other and share very few resources. Their connections and permissions can be isolated.
- Schemas share more resources than databases do. User permissions on schemas and subordinate objects can be flexibly configured using the **GRANT** and **REVOKE** syntax.

You are advised to use schemas to isolate services for convenience and resource sharing. It is recommended that system administrators create schemas and databases and then assign required permissions to users.

Figure 19-3 Used for permission control.



DATABASE

A database is a physical collection of database objects. Resources of different databases are completely isolated (except some shared objects). Databases are used to isolate workloads. Objects in different databases cannot access each other. For example, objects in Database B cannot be accessed in Database A. Therefore, when logging in to a cluster, you must connect to the specified database.

SCHEMA

In a database, database objects are logically divided and isolated based on schemas.

With permission management, you can access and operate objects in different schemas in the same session. Schemas contain objects that applications may access, such as tables, indexes, data in various types, functions, and operators.

Database objects with the same name cannot exist in the same schema, but object names in different schemas can be the same.

```
gaussdb=> CREATE SCHEMA myschema;  
CREATE SCHEMA  
gaussdb=> CREATE SCHEMA myschema_1;  
CREATE SCHEMA  
  
gaussdb=> CREATE TABLE myschema.t1(a int, b int) DISTRIBUTE BY HASH(b);  
CREATE TABLE  
gaussdb=> CREATE TABLE myschema.t1(a int, b int) DISTRIBUTE BY HASH(b);  
ERROR: relation "t1" already exists  
gaussdb=> CREATE TABLE myschema_1.t1(a int, b int) DISTRIBUTE BY HASH(b);  
CREATE TABLE
```

Schemas logically divide workloads. These workloads are interdependent with the schemas. Therefore, if a schema contains objects, deleting it will cause errors with dependency information displayed.

```
gaussdb=> DROP SCHEMA myschema_1;  
ERROR: cannot drop schema myschema_1 because other objects depend on it  
Detail: table myschema_1.t1 depends on schema myschema_1  
Hint: Use DROP ... CASCADE to drop the dependent objects too.
```

When a schema is deleted, the **CASCADE** option is used to delete the objects that depend on the schema.

```
gaussdb=> DROP SCHEMA myschema_1 CASCADE;  
NOTICE: drop cascades to table myschema_1.t1  
gaussdb=> DROP SCHEMA
```

USER/ROLE

Users and roles are used to implement permission control on the database server (cluster). They are the owners and executors of cluster workloads and manage all object permissions in clusters. A role is not confined in a specific database. However, when it logs in to the cluster, it must explicitly specify a user name to ensure the transparency of the operation. A user's permissions to a database can be specified through permission management.

A user is the subject of permissions. Permission management is actually the process of deciding whether a user is allowed to perform operations on database objects.

Permissions Management

Permission management in GaussDB(DWS) falls into three categories:

- System permission

System permissions are also called user attributes, including **SYSADMIN**, **CREATEDB**, **CREATEROLE**, **AUDITADMIN**, and **LOGIN**.

They can be specified only by the **CREATE ROLE** or **ALTER ROLE** syntax. The **SYSADMIN** permission can be granted and revoked using **GRANT ALL PRIVILEGE** and **REVOKE ALL PRIVILEGE**, respectively. System permissions cannot be inherited by a user from a role, and cannot be granted using **PUBLIC**.

- Permissions

Grant a role's or user's permissions to one or more roles or users. In this case, every role or user can be regarded as a set of one or more database permissions.

If **WITH ADMIN OPTION** is specified, the member can in turn grant permissions in the role to others, and revoke permissions in the role as well. If a role or user granted with certain permissions is changed or revoked, the permissions inherited from the role or user also change.

A database administrator can grant permissions to and revoke them from any role or user. Roles having **CREATEROLE** permission can grant or revoke membership in any role that is not an administrator.

- Object permission

Permissions on a database object (table, view, column, database, function, schema, or tablespace) can be granted to a role or user. The **GRANT** command can be used to grant permissions to a user or role. These permissions granted are added to the existing ones.

Schema Isolation Example

Example 1:

By default, the owner of a schema has all permissions on objects in the schema, including the delete permission. The owner of a database has all permissions on objects in the database, including the delete permission. Therefore, you are advised to strictly control the creation of databases and schemas. Create databases and schemas as an administrator and assign related permissions to users.

- Step 1** Assign the permission to create schemas in the **testdb** database to user **user_1** as user **dbadmin**.

```
testdb=> GRANT CREATE ON DATABASE testdb to user_1;  
GRANT
```

- Step 2** Switch to user **user_1**.

```
testdb=> SET SESSION AUTHORIZATION user_1 PASSWORD '*****';  
SET
```

Create a schema named **myschema_2** in the **testdb** database as **user_1**.

```
testdb=> CREATE SCHEMA myschema_2;  
CREATE SCHEMA
```

Step 3 Switch to the administrator **dbadmin**.

```
testdb=> RESET SESSION AUTHORIZATION;  
RESET
```

Create **table t1** in schema **myschema_2** as the administrator **dbadmin**.

```
testdb=> CREATE TABLE myschema_2.t1(a int, b int) DISTRIBUTE BY HASH(b);  
CREATE TABLE
```

Step 4 Switch to user **user_1**.

```
testdb=> SET SESSION AUTHORIZATION user_1 PASSWORD '*****';  
SET
```

Delete table **t1** created by administrator **dbadmin** in schema **myschema_2** as user **user_1**.

```
testdb=> drop table myschema_2.t1;  
DROP TABLE
```

----End

Example 2:

Due to the logical isolation of schemas, database objects need to be verified at both the schema level and the object level.

Step 1 Grant the permission on the **myschema.t1** table to **user_1**.

```
gaussdb=> GRANT SELECT ON TABLE myschema.t1 TO user_1;  
GRANT
```

Step 2 Switch to user **user_1**.

```
SET SESSION AUTHORIZATION user_1 PASSWORD '*****';  
SET
```

Query the table **myschema.t1**.

```
gaussdb=> SELECT * FROM myschema.t1;  
ERROR: permission denied for schema myschema  
LINE 1: SELECT * FROM myschema.t1;
```

Step 3 Switch to the administrator **dbadmin**.

```
gaussdb=> RESET SESSION AUTHORIZATION;  
RESET
```

Grant the permission on the **myschema.t1** table to user **user_1**.

```
gaussdb=> GRANT USAGE ON SCHEMA myschema TO user_1;  
GRANT
```

Step 4 Switch to user **user_1**.

```
gaussdb=> SET SESSION AUTHORIZATION user_1 PASSWORD '*****';  
SET
```

Query the table **myschema.t1**.

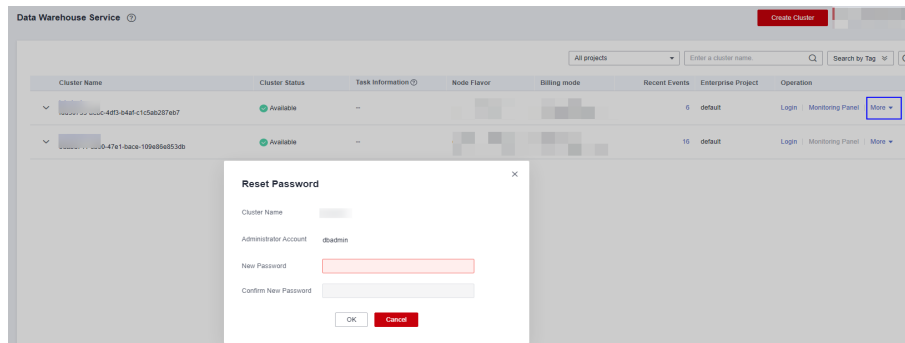
```
gaussdb=> SELECT * FROM myschema.t1;  
a | b  
---+---  
(0 rows)
```

----End

19.6.2 How Do I Change the Password of a Database Account When the Password Expires?

- To change the password of the database administrator **dbadmin**, log in to the console and choose **More > Reset Password** in cluster row.

Figure 19-4 Resetting the password of user dbadmin



For security, the following two parameters manage account passwords. Log in to the console, click the cluster name and switch to the parameter modification page to modify the parameters.

- failed_login_attempts**: maximum number of consecutive incorrect password attempts before the account is locked. Run the following statement as user **dbadmin** to unlock the account:
ALTER USER user_name ACCOUNT UNLOCK;
- password_effect_time**: validity period of the account password, in days. The default value is **90**.
- You can also connect to the database and run the **ALTER USER** command to change the password validity period of a database account (common user and administrator dbadmin).
ALTER USER username PASSWORD EXPIRATION 90;

19.6.3 How Do I Grant Table Permissions to a User?

This section describes how to grant users the SELECT, INSERT, UPDATE, or full permissions of tables to users.

Syntax

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIGGER | ANALYZE |
ANALYZE } [, ...]
      | ALL [ PRIVILEGES ] }
ON { [ TABLE ] table_name [, ...]
     | ALL TABLES IN SCHEMA schema_name [, ...] }
TO { [ GROUP ] role_name | PUBLIC } [, ...]
[ WITH GRANT OPTION ];
```

Scenario

Assume there are users **u1**, **u2**, **u3**, **u4**, and **u5** and five schemas named after these users. Their permission requirements are as follows:

- User **u2** is a read-only user and requires the SELECT permission for the **u1.t1** table.
- User **u3** requires the SELECT permission for the **u1.t1** table.
- User **u3** requires the UPDATE permission for the **u1.t1** table.
- User **u5** requires all permissions of table **u1.t1**.

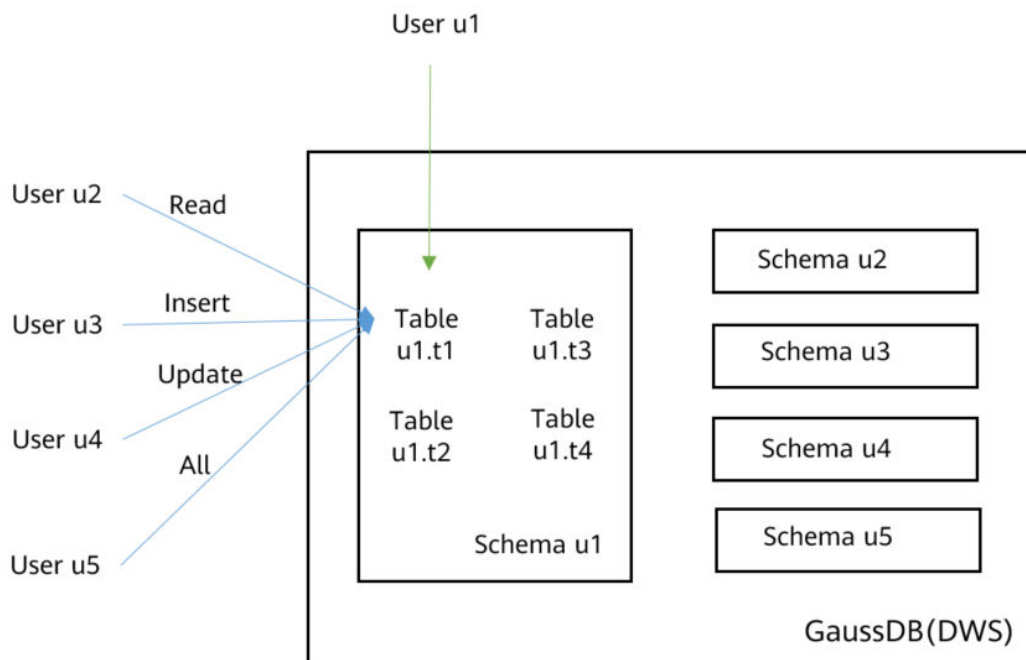


Table 19-5 Permissions of the u1.t1 table

User	Type	GRANT Statement	Query	Insert	Update	Delete
u1	Owner	-	√	√	√	√
u2	Read-only user	GRANT SELECT ON u1.t1 TO u2;	√	x	x	x
u3	INSERT user	GRANT INSERT ON u1.t1 TO u3;	x	√	x	x

Us er	Ty pe	GRANT Statement	Qu ery	Ins ert	Up dat e	Del ete
u4	UP DA TE use r	GRANT SELECT,UPDATE ON u1.t1 TO u4; NOTICE The UPDATE permission must be granted together with the SELECT permission, or information leakage may occur.	√	x	√	x
u5	Us ers wit h all per mis sions	GRANT ALL PRIVILEGES ON u1.t1 TO u5;	√	√	√	√

Procedure

Perform the following steps to grant and verify permissions:

- Step 1** Connect to your database as **dbadmin**. Run the following statements to create users **u1** to **u5**. Five schemas will be created and named after the users by default.

```
CREATE USER u1 PASSWORD '{password}';
CREATE USER u2 PASSWORD '{password}';
CREATE USER u3 PASSWORD '{password}';
CREATE USER u4 PASSWORD '{password}';
CREATE USER u5 PASSWORD '{password}';
```

- Step 2** Create table **u1.t1** in schema **u1**.

```
CREATE TABLE u1.t1 (c1 int, c2 int);
```

- Step 3** Insert two records to the table.

```
INSERT INTO u1.t1 VALUES (1,2);
INSERT INTO u1.t1 VALUES (1,2);
```

- Step 4** Grant schema permissions to users.

```
GRANT USAGE ON SCHEMA u1 TO u2,u3,u4,u5;
```

- Step 5** Grant user **u2** the permission to query the **u1.t1** table.

```
GRANT SELECT ON u1.t1 TO u2;
```

- Step 6** Start a new session and connect to the database as user **u2**. Verify that user **u2** can query the **u1.t1** table but cannot write to or modify the table.

```
SELECT * FROM u1.t1;
INSERT INTO u1.t1 VALUES (1,20);
UPDATE u1.t1 SET c2 = 3 WHERE c1 =1;
```

```
gaussdb=> SELECT * FROM u1.t1;
 c1 | c2
----+----
  1 |  2
  1 |  2
(2 rows)

gaussdb=> INSERT INTO u1.t1 VALUES (1,20);
ERROR: permission denied for relation t1
gaussdb=> UPDATE u1.t1 SET c2 = 3 WHERE c1 =1;
ERROR: permission denied for relation t1
```

Step 7 In the session started by user **dbadmin**, grant permissions to users **u3**, **u4**, and **u5**.

```
GRANT INSERT ON u1.t1 TO u3; -- Allow u3 to insert data.
GRANT SELECT,UPDATE ON u1.t1 TO u4; -- Allow u4 to modify the table.
GRANT ALL PRIVILEGES ON u1.t1 TO u5; -- Allow u5 to query, insert, modify, and delete table data.
```

Step 8 Start a new session and connect to the database as user **u3**. Verify that user **u3** can query the **u1.t1** table but cannot query or modify the table.

```
SELECT * FROM u1.t1;
INSERT INTO u1.t1 VALUES (1,20);
UPDATE u1.t1 SET c2 = 3 WHERE c1 =1;
```

```
gaussdb=> SELECT * FROM u1.t1;
ERROR: permission denied for relation t1
gaussdb=> INSERT INTO u1.t1 VALUES (1,20);
INSERT 0 1
gaussdb=> UPDATE u1.t1 SET c2 = 3 WHERE c1 =1;
ERROR: permission denied for relation t1
```

Step 9 Start a new session and connect to the database as user **u4**. Verify that user **u4** can modify and query the **u1.t1** table, but cannot insert data to the table.

```
SELECT * FROM u1.t1;
INSERT INTO u1.t1 VALUES (1,20);
UPDATE u1.t1 SET c2 = 3 WHERE c1 =1;
```

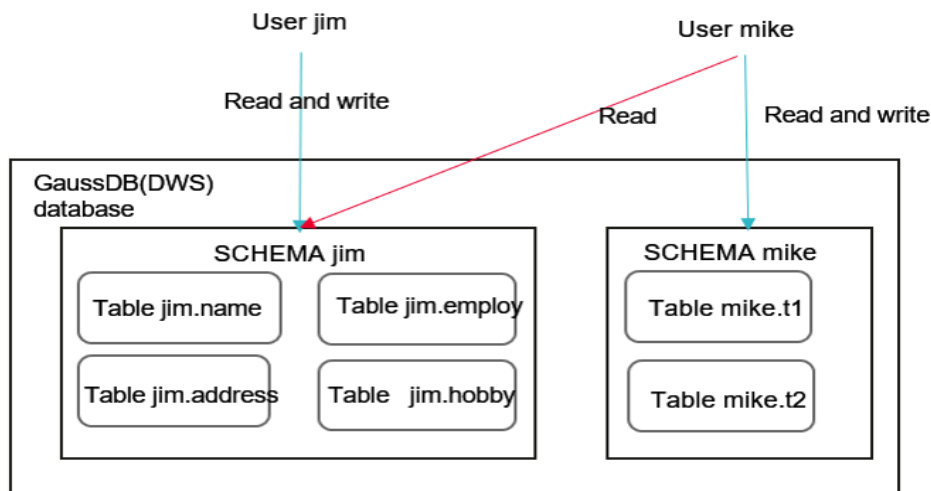
```
gaussdb=> SELECT * FROM u1.t1;
 c1 | c2
----+----
  1 |  2
  1 |  2
  1 | 20
(3 rows)

gaussdb=> INSERT INTO u1.t1 VALUES (1,20);
ERROR: permission denied for relation t1
gaussdb=> UPDATE u1.t1 SET c2 = 3 WHERE c1 =1;
UPDATE 3
```

Step 10 Start a new session and connect to the database as user **u5**. Verify that user **u5** can query, insert, modify, and delete data in the **u1.t1** table.

```
SELECT * FROM u1.t1;
INSERT INTO u1.t1 VALUES (1,20);
UPDATE u1.t1 SET c2 = 3 WHERE c1 =1;
DELETE FROM u1.t1;
```


Figure 19-5 User **mike** accesses a table in SCHEMA **jim**.



Step 1 Connect to your database as **dbadmin**. Run the following statements to create users **jim** and **mike**. Two schemas will be created and named after the users by default.

```
CREATE USER jim PASSWORD '{password}';
CREATE USER mike PASSWORD '{password}';
```

Step 2 Create tables **jim.name** and **jim.address** in schema **jim**.

```
CREATE TABLE jim.name (c1 int, c2 int);
CREATE TABLE jim.address (c1 int, c2 int);
```

Step 3 Grant the access permission of schema **jim** to user **mike**.

```
GRANT USAGE ON SCHEMA jim TO mike;
```

Step 4 Grant user **mike** the permission to query table **jim.name** in schema **jim**.

```
GRANT SELECT ON jim.name TO mike;
```

Step 5 Start a new session and connect to the database as user **mike**. Verify that user **mike** can query the **jim.name** table but not the **jim.address** table.

```
SELECT * FROM jim.name;
SELECT * FROM jim.address;
```

Result Information	SQL Details	Status	Times
[Statistics in some tables or columns(jim.name) are not collected.] □	SELECT * FROM jim.name	Run successfully	268ms
["error_code":"DWS.S0010001","error_msg":"sql error, STATE: 42501, message: permission denied to user 'mike' for relation 'jim.address'"]	ERROR: SELECT SELECT * FROM jim.address	Running failed	31ms

Step 6 In the session started by user **dbadmin**, grant user **mike** the permission to query all the tables in schema **jim**.

```
GRANT SELECT ON ALL TABLES IN SCHEMA jim TO mike;
```

Step 7 In the session started by user **mike**, verify that **mike** can query all tables.

```
SELECT * FROM jim.name;
SELECT * FROM jim.address;
```

Result Information	SQL Details	Status	Times
[Statistics in some tables or columns(jim.name) are not collected.]	SELECT * FROM jim.name	Run successfully	13ms
[Statistics in some tables or columns(jim.address) are not collected.]	SELECT * FROM jim.address	Run successfully	18ms

Step 8 In the session started by user **dbadmin**, create table **jim.employ**.

```
CREATE TABLE jim.employ (c1 int, c2 int);
```

Step 9 In the session started by user **mike**, verify that user **mike** does not have the query permission for **jim.employ**. It indicates that user **mike** has the permission to access all the existing tables in schema **jim**, but not the tables to be created in the future.

```
SELECT * FROM jim.employ;
```

Result Information	SQL Details	Status	Times
["error_code": "DWS.S0010001", "error_msg": "sql error: STATE: 42501, message: permission denied to user 'mike' for relation 'jim.employ'"]	SELECT * FROM jim.employ	Running failed	17ms

Step 10 In the session started by user **dbadmin**, grant user **mike** the permission to query the tables to be created in schema **jim**. Create table **jim.hobby**.

```
ALTER DEFAULT PRIVILEGES FOR ROLE jim IN SCHEMA jim GRANT SELECT ON TABLES TO mike;  
CREATE TABLE jim.hobby (c1 int, c2 int);
```

Step 11 In the session started by user **mike**, verify that user **mike** can access table **jim.hobby**, but does not have the permission to access **jim.employ**. To let the user access table **jim.employ**, you can grant permissions by performing [Step 4](#).

```
SELECT * FROM jim.hobby;
```

Result Information	SQL Details	Status	Times
[Statistics in some tables or columns(jim.hobby) are not collected.]	SELECT * FROM jim.hobby	Run successfully	19ms

----End

19.6.5 How Do I Create a Database Read-only User?

Scenario

In service development, database administrators use schemas to classify data. For example, in the financial industry, liability data belong to schema **s1**, and asset data belong to schema **s2**.

Now you have to create a read-only user **user1** in the database. The user can access all tables (including new tables to be created in the future) in schema **s1** for daily reading, but cannot insert, modify, or delete data.

Principles

DWS provides role-based user management. You need to create a read-only role **role1** and grant the role to **user1**.

Procedure

Step 1 Connect to the DWS database as user **dbadmin**.

Step 2 Run the following SQL statement to create role **role1**:

```
CREATE ROLE role1 PASSWORD disable;
```

Step 3 Run the following SQL statement to grant permissions to **role1**:

```
The GRANT usage ON SCHEMA s1 TO role1; -- grants the access permission to schema s1.
GRANT select ON ALL TABLES IN SCHEMA s1 TO role1; -- grants the query permission on all tables in
schema s1.
ALTER DEFAULT PRIVILEGES FOR USER tom IN SCHEMA s1 GRANT select ON TABLES TO role1; -- grants
schema s1 the permission to create tables. tom is the owner of schema s1.
```

Step 4 Run the following SQL statement to grant the role **role1** to the actual user **user1**:

```
GRANT role1 TO user1;
```

Step 5 Read all table data in schema **s1** as read-only user **user1**.

----End

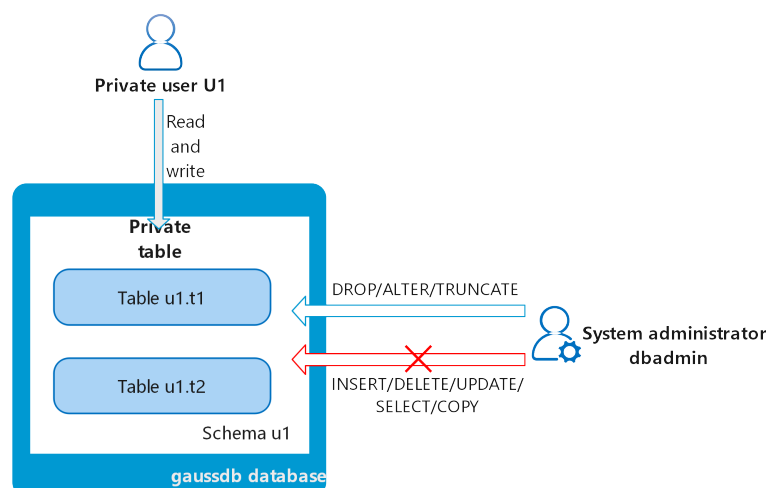
19.6.6 How Do I Create Private Database Users and Tables?

Scenario

The system administrator **dbadmin** has the permission to access tables created by common users by default. When is enabled, the administrator **dbadmin** does not have the permission to access tables of common users or perform control operations (DROP, ALTER, and TRUNCATE).

If a private user and a private table (table created by the private user) need to be created, and the private table can be accessed only by the private user and the system administrator **dbadmin** and other common users do not have the permission to access the table (INSERT, DELETE, UPDATE, SELECT, and COPY). However, the system administrator **dbadmin** sometimes need to perform the DROP, ALTER, or TRUNCATE operations without authorization from the private user. In this case, you can create a user (private user) with the INDEPENDENT attribute.

Figure 19-6 Private users



Principles

This function is implemented by creating a user with the INDEPENDENT attribute.

INDEPENDENT | NOINDEPENDENT defines private and independent roles. For a role with the **INDEPENDENT** attribute, administrators' rights to control and access this role are separated. Specific rules are as follows:

- Administrators have no rights to add, delete, query, modify, copy, or authorize the corresponding table objects without the authorization from the **INDEPENDENT** role.
- Administrators have no rights to modify the inheritance relationship of the **INDEPENDENT** role without the authorization from this role.
- Administrators have no rights to modify the owner of the table objects for the **INDEPENDENT** role.
- Administrators have no rights to change the database password of the **INDEPENDENT** role. The **INDEPENDENT** role must manage its own password, which cannot be reset if lost.
- The **SYSADMIN** attribute of a user cannot be changed to the **INDEPENDENT** attribute.

Procedure

Step 1 Connect to the DWS database as user **dbadmin**.

Step 2 Run the following SQL statement to create private user **u1**:

```
CREATE USER u1 WITH INDEPENDENT IDENTIFIED BY 'password';
```

Step 3 Switch to user **u1**, create the table **test**, and insert data into the table.

```
CREATE TABLE test (id INT, name VARCHAR(20));  
INSERT INTO test VALUES (1, 'joe');  
INSERT INTO test VALUES (2, 'jim');
```

Step 4 Switch to user **dbadmin** and run the following SQL statement to check whether user **dbadmin** can access the private table **test** created by private user **u1**:

```
SELECT * FROM u1.test;
```

The query result indicates that the user **dbadmin** does not have the access permission. This means the private user and private table are created successfully.

```
gaussdb=> SELECT * FROM u1.test;  
ERROR: SELECT permission denied to user "dbadmin" for relation "u1.test"
```

Step 5 Run the **DROP** statement as user **dbadmin** to delete the table **test**.

```
DROP TABLE u1.test;
```

```
gaussdb=> drop table u1.test;  
DROP TABLE
```

----End

19.6.7 How Do I Revoke the CONNECT ON DATABASE Permission from a User?

Scenario

In a service, the permission of user **u1** to connect to a database needs to be revoked. After the **REVOKE CONNECT ON DATABASE gaussdb FROM u1**;

command is executed successfully, user **u1** can still connect to the database. This means the revocation does not take effect.

Cause Analysis

If you run the **REVOKE CONNECT ON DATABASE gaussdb from u1** command to revoke the permissions of user **u1**, the revocation does not take effect because the **CONNECT** permission of the database is granted to **PUBLIC**. Therefore, you need to specify **PUBLIC**.

- GaussDB(DWS) provides an implicitly defined group **PUBLIC** that contains all roles. By default, all new users and roles have the permissions of **PUBLIC**. To revoke permissions of **PUBLIC** from a user or role, or re-grant these permissions to them, add the **PUBLIC** keyword in the **REVOKE** or **GRANT** statement.
- GaussDB(DWS) grants the permissions for objects of certain types to **PUBLIC**. By default, permissions on tables, columns, sequences, foreign data sources, foreign servers, schemas, and tablespaces are not granted to **PUBLIC**, but the following permissions are granted to **PUBLIC**;
 - **CONNECT** permission of a database
 - **CREATE TEMP TABLE** permission of a database
 - **EXECUTE** permission of a function
 - **USAGE** permission for languages and data types (including domains)
- An object owner can revoke the default permissions granted to **PUBLIC** and grant permissions to other users as needed.

Example Operations

Run the following command to revoke the permission for user **u1** to access database **gaussdb**:

Step 1 Connect to the GaussDB(DWS) database **gaussdb**.

```
gsql -d gaussdb -p 8000 -h 192.168.x.xx -U dbadmin -W password -r  
gaussdb=>
```

Step 2 Create user **u1**.

```
gaussdb=> CREATE USER u1 IDENTIFIED BY 'xxxxxxx';
```

Step 3 Verify that user **u1** can access GaussDB.

```
gsql -d gaussdb -p 8000 -h 192.168.x.xx -U u1 -W password -r  
gaussdb=>
```

Step 4 Connect to database **gaussdb** as administrator **dbadmin** and run the **REVOKE** command to revoke the **connect on database** permission of user **public**.

```
gsql -d gaussdb -h 192.168.x.xx -U dbadmin -p 8000 -r  
gaussdb=> REVOKE CONNECT ON DATABASE gaussdb FROM public;  
REVOKE
```

Step 5 Verify the result. Use **u1** to connect to the database. If the following information is displayed, the **connect on database** permission of user **u1** has been revoked successfully:

```
gsql -d gaussdb -p 8000 -h 192.168.x.xx -U u1 -W password -r  
gsql: FATAL: permission denied for database "gaussdb"  
DETAIL: User does not have CONNECT privilege.
```

----End

19.6.8 How Do I View the Table Permissions of a User?

Scenario 1: Run the `information_schema.table_privileges` command to view the table permissions of a user. Example:

```
SELECT * FROM information_schema.table_privileges WHERE GRANTEE='user_name';
```

```
gaussdb-> SELECT * FROM information_schema.table_privileges WHERE GRANTEE='u2';
grantor | grantee | table_catalog | table_schema | table_name | privilege_type | is_grantable | with_hierarchy
-----|-----|-----|-----|-----|-----|-----|-----
u2      | u2      | gaussdb      | u2           | t2         | INSERT        | YES          | NO
u2      | u2      | gaussdb      | u2           | t2         | SELECT        | YES          | YES
u2      | u2      | gaussdb      | u2           | t2         | UPDATE        | YES          | NO
u2      | u2      | gaussdb      | u2           | t2         | DELETE        | YES          | NO
u2      | u2      | gaussdb      | u2           | t2         | TRUNCATE      | YES          | NO
u2      | u2      | gaussdb      | u2           | t2         | REFERENCES    | YES          | NO
u2      | u2      | gaussdb      | u2           | t2         | TRIGGER       | YES          | NO
u2      | u2      | gaussdb      | u2           | t2         | ANALYZE       | YES          | NO
u2      | u2      | gaussdb      | u2           | t2         | VACUUM        | YES          | NO
u2      | u2      | gaussdb      | u2           | t2         | ALTER         | YES          | NO
u2      | u2      | gaussdb      | u2           | t2         | DROP          | YES          | NO
u1      | u2      | gaussdb      | u1           | t1         | SELECT        | NO           | YES
(12 rows)
```

Table 19-6 table_privileges columns

Column	Data Type	Description
grantor	sql_identifier	Permission grantor
grantee	sql_identifier	Permission grantee
table_catalog	sql_identifier	Database where the table is
table_schema	sql_identifier	Schema where the table is
table_name	sql_identifier	Table name
privilege_type	character_data	Type of the granted permission. The value can be SELECT , INSERT , UPDATE , DELETE , TRUNCATE , REFERENCES , ANALYZE , VACUUM , ALTER , DROP , or TRIGGER .
is_grantable	yes_or_no	Indicates if the permission can be granted to other users. YES indicates that the permission can be granted to other users, and NO indicates that the permission cannot be granted to other users.
with_hierarchy	yes_or_no	Indicates if specific operations are allowed to be inherited at the table level. If the specific operation is SELECT , YES is displayed. Otherwise, NO is displayed.

In the preceding figure, user **u2** has all permissions of table **t2** in schema **u2** and the **SELECT** permission of table **t1** in schema **u1**.

`information_schema.table_privileges` can query only the permissions directly granted to the user, the `has_table_privilege()` function can query both directly granted permissions and indirect permissions (obtained by GRANT role to user). For example:

```
CREATE TABLE t1 (c1 int);
CREATE USER u1 password '*****';
```

```
CREATE USER u2 password '*****';
GRANT dbadmin to u2; //Indirectly grant permissions through roles.
GRANT SELECT on t1 to u1; // Directly grant the permission.

SET ROLE u1 password '*****';
SELECT * FROM public.t1; // Directly grant the permission to access the table.
c1
-----
(0 rows)

SET ROLE u2 password '*****';
SELECT * FROM public.t1; // Indirectly grant the permission to access the table.
c1
-----
(0 rows)

RESET role; //Switch back to dbadmin.
SELECT * FROM information_schema.table_privileges WHERE table_name = 't1'; // Can only view direct grants.
 grantor | grantee | table_catalog | table_schema | table_name | privilege_type | is_grantable |
with_hierarchy
-----+-----+-----+-----+-----+-----+-----+-----
dbadmin | u1 | gaussdb | public | t1 | SELECT | NO | YES
(1 rows)

SELECT has_table_privilege('u2', 'public.t1', 'select'); // Can view both direct and indirect grants.
has_table_privilege
-----
t
(1 row)
```

Scenario 2: To check whether a user has permissions on a table, perform the following steps:

Step 1 Query the **pg_class** system catalog.

```
SELECT * FROM pg_class WHERE relname = 'tablename';
```

Check the **relacl** column. The command output is shown in the following figure. For details about the permission parameters, see [Table 19-7](#).

- *rolename=xxxx/yyyy*: indicates that *rolename* has the *xxxx* permission on the table and the permission is obtained from *yyyy*.
- *=xxxx/yyyy*: indicates that **public** has the *xxxx* permission on the table and the permission is obtained from *yyyy*.

Take the following figure as an example:

joe=arwdDxtA: indicates that user **joe** has all permissions (**ALL PRIVILEGES**).

leo=arw/joe: indicates that user **leo** has the read, write, and modify permissions, which are granted by user **joe**.

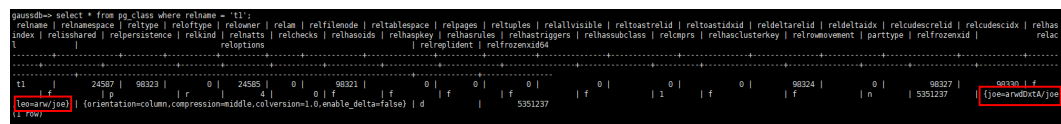


Table 19-7 Permissions parameters

Parameter	Description
r	SELECT (read)

Parameter	Description
w	UPDATE (write)
a	INSERT (insert)
d	DELETE
D	TRUNCATE
x	REFERENCES
t	TRIGGER
X	EXECUTE
U	USAGE
C	CREATE
c	CONNECT
T	TEMPORARY
A	ANALYZE ANALYSE
arwdDxtA	ALL PRIVILEGES (for tables)
*	Actions for preceding permissions

Step 2 You can also use the `has_table_privilege` function to query user permissions on tables.

```
SELECT * FROM has_table_privilege('Username','Table_name','select');
```

For example, query whether user `joe` has the query permission on table `t1`.

```
SELECT * FROM has_table_privilege('joe','t1','select');
```

```
gaussdb=> select * from has_table_privilege('joe','t1','select');
has_table_privilege
-----
 t
(1 row)
```

----End

19.6.9 Who Is User Ruby?

When you run the `SELECT * FROM pg_user` statement to view users in the current system, you may see user **Ruby** many times, who has many permissions.

User **Ruby** is an official O&M account. After a GaussDB(DWS) database is created, user **Ruby** is generated by default. There is no security risk in regard to user **Ruby**.

```
gaussdb>> SELECT * FROM pg_user;
username | usesysid | usecreatedb | useuser | usecatupd | use repl | passwd | valbegin | valuntil | respool | parent | spacelimit | useconfig | nodegroup | temppacelimit | spillpa
celimit
-----
 dbadmin | 16384 | f | f | f | f | ***** | | | default_pool | 0 | | | | |
 Ruby | 18 | t | t | t | t | ***** | | | default_pool | 0 | | | | |
 user_1 | 24584 | f | f | f | f | ***** | | | default_pool | 0 | | | | |
 u1 | 24593 | f | f | f | f | ***** | | | default_pool | 0 | | | | |
 u2 | 24597 | f | f | f | f | ***** | | | default_pool | 0 | | | | |
(5 rows)
```

19.7 Database Performance

19.7.1 Why Is SQL Execution Slow After Long GaussDB(DWS) Usage?

After a database is used for a period of time, the table data increases as services grow, or the table data is frequently added, deleted, or modified. As a result, bloating tables and inaccurate statistics are incurred, deteriorating database performance.

You are advised to periodically run **VACUUM FULL** and **ANALYZE** on tables that are frequently added, deleted, or modified. Perform the following operations:

- Step 1** By default, 100 out of 30,000 records of statistics are collected. When a large amount of data is involved, the SQL execution is unstable, which may be caused by a changed execution plan. In this case, the sampling rate needs to be adjusted for statistics. You can run **set default_statistics_target** to increase the sampling rate, which helps the optimizer generate the optimal plan.

```
gaussdb=> set default_statistics_target=-2;  
SET
```

- Step 2** Run **ANALYZE** again. For details, see .

```
gaussdb=> ANALYZE customer_t1;  
ANALYZE
```

----End

NOTE

To test whether disk fragments affect database performance, use the following function:
`SELECT * FROM pgxc_get_stat_dirty_tables(30,100000);`

19.7.2 Why Does GaussDB(DWS) Perform Worse Than a Single-Server Database in Extreme Scenarios?

Due to the MPP architecture limitation of GaussDB(DWS), a few PostgreSQL methods and functions cannot be pushed to DN for execution. As a result, performance bottlenecks occur on CNs.

Explanation:

- An operation can be executed concurrently only when it is logically a concurrent operation. For example, SUM performed on all DN concurrently must centralize the final summarization on one CN. In this case, most of the summarization work has been completed on DN, so the work on the CN is relatively lightweight.
- In some scenarios, the operation must be executed centrally on one node. For example, assigning a globally unique name to a transaction ID is implemented using the system GTM. Therefore, the GTM is also a globally unique component (active/standby). All globally unique tasks are

implemented through the GTM in GaussDB(DWS), but software code is optimized to reduce this kind of tasks. Therefore, the GTM does not have much of a bottleneck. In some scenarios, GTM-Free and GTM-Lite can be implemented.

- To ensure excellent performance, services need to be slightly modified for adaptation during migration from the application development mode of the traditional single-node database to that of the parallel database, especially for the traditional stored procedure nesting of Oracle.

Solutions:

- Alternatively, contact technical support to modify and optimize services.

19.7.3 How Can I View SQL Execution Records in a Certain Period When Read and Write Requests Are Blocked?

You can use the top SQL feature to view SQL statements executed in a specified period. SQL statements of the current CN or all CNs can be viewed.

Top SQL allows you to view real-time and historical SQL statements.

- For details about real-time SQL statement query, see section .
- For details about how to query historical SQL statements, see section .

19.7.4 What Do I Do If My Cluster Is Unavailable Because of Insufficient Space?

You can use a snapshot to restore your cluster to a new one that has larger storage space, and then delete the old cluster. You can learn cluster storage by checking **Available Storage**. To restore your cluster to a new one, see "Restoring a Snapshot to a New Cluster" in *Data Warehouse Service (DWS) User Guide* To delete a cluster, see "Deleting a Cluster" in *Data Warehouse Service (DWS) User Guide*.

 **NOTE**

This method is only supported by the standard data warehouse.

19.7.5 GaussDB(DWS) CPU Resource Management

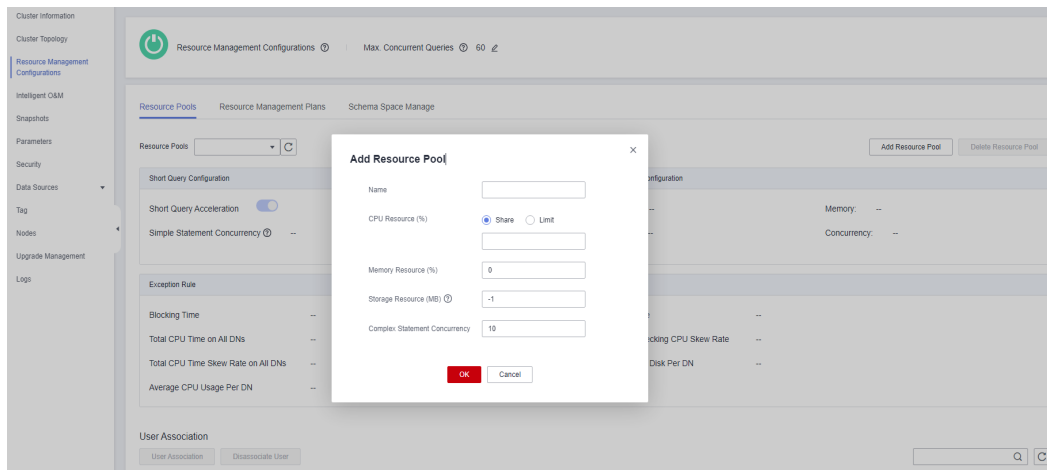
Overview of CPU Resource Management

In different service scenarios, system resources (CPU, memory, I/O, and storage resources) of the database are properly allocated to queries to ensure query performance, and service stability.

GaussDB(DWS) provides the resource management function. You can put resources into different resource pools, which are isolated from each other. Then, you can associate database users with these resource pools. When a user starts a SQL query, the query will be transferred to the resource pool associated with the user. You can specify the number of queries that can be concurrently executed in a resource pool, the upper limit of memory used for a single query, and the memory and CPU resources that can be used by a resource pool. In this way, you can limit and isolate the resources occupied by different workloads.

GaussDB(DWS) uses cgroups to manage and control CPU resources, involving the CPU, cpuacct, and cpuset subsystems. CPU share is implemented based on the CPU subsystem cpu.shares. The advantages of CPU share are as follows: CPU control is not triggered when the OS CPU is not fully occupied. CPU limit is implemented based on cpuset, which is a CPU subsystem used to monitor CPU resource usage.

When adding a resource pool on the GaussDB(DWS) management console, you should choose between **Share** and **Limit**.



CPU Share

CPU Share: Percentage of CPU time that can be used by users associated with the current resource pool to execute jobs.

The share has two meanings:

- **Share:** The CPU is shared by all Cgroups, and other Cgroups can use idle CPU resources.
- **Limit:** When the CPU is fully loaded during peak hours, Cgroups preempt CPU resources based on their limits.

CPU share is implemented based by cpu.shares and takes effect only when the CPU is fully loaded. When the CPU is idle, there is no guarantee that a Cgroup will preempt CPU resources appropriate to its quota. There can still be resource contention when the CPU is idle. Tasks in a Cgroup can use CPU resources without restriction. Although the average CPU usage may not be high, CPU resource contention may still occur at a specific time.

For example, 10 jobs are running on 10 CPUs, and one job is running on each CPU. In this case, any job request for CPU resources will be responded instantly, and there is no contention. If 20 jobs are running on 10 CPUs, the CPU usage may still not be high because the jobs do not always occupy the CPU and may wait for I/O and network resources. The CPU resources seem idle. However, if 2 or more jobs request one CPU at the same time, CPU resource contention occurs, affecting job performance.

CPU Limit

CPU Limit: specifies the percentage of the maximum number of CPU cores that can be used by a database user in the resource pool.

The limit has two meanings:

- **Dedicated:** The CPU is dedicated to a Cgroup. Other Cgroups cannot use idle CPU resources.
- **Quota:** Only the CPU resources in the allocated quota can be used. Idle CPU resources of other Cgroups cannot be preempted.

CPU limit is implemented based on `cpuset.cpu`. You can set a proper quota to implement absolute isolation of CPU resources between Cgroups. In this way, tasks of different Cgroups will not affect each other. However, the absolute CPU isolation will cause idle CPU resources in a Cgroup to be wasted. Therefore, the limit cannot be too large. A larger limit may not bring a better performance.

For example, in one case, 10 jobs are running on 10 CPUs and the average CPU usage is about 5%. In another case, 10 jobs are running on 5 CPUs and the average CPU usage is about 10%. According to the preceding analysis, although the CPU usage is low when 10 jobs run on five CPUs. However, CPU resource contention still exists. Therefore, the performance of running 10 jobs on 10 CPUs is better than that of running 10 jobs on 5 CPUs. However, it is not the more CPUs, the better. If ten jobs run on 20 CPUs, at any time point, at least 10 CPUs are idle. Therefore, theoretically, running 10 jobs on 20 CPUs does not have better performance than running 10 CPUs. For a Cgroup with a concurrency of N , if the number of allocated CPUs is less than N , the job performance is better with more CPUs. However, if the number of allocated CPUs is greater than N , the job performance will not be improved with more CPUs.

Application Scenarios of CPU Resource Management

The CPU limit and CPU share both have their own advantages and disadvantages. CPU share can fully utilize CPU resources. However, resources of different Cgroups are not completely isolated, which may affect the query performance. CPU limit can implement absolute isolation of CPU resources. However, idle CPU resources will be wasted. Compared with CPU limit, CPU share has higher CPU usage and overall job throughput. Compared with CPU share, CPU limit has complete CPU isolation, which can better meet the requirements of performance-sensitive users.

If CPU contention occurs when multiple types of jobs are running in the database system, you can select different CPU resource control modes based on different scenarios.

- **Scenario 1:** Fully utilize CPU resources. Focus on the overall CPU throughput instead of the performance of a single type of jobs.
Suggestion: You are not advised to isolate CPUs between users. No matter which type of CPU control is implemented, the overall CPU usage is affected.
- **Scenario 2:** A certain degree of CPU resource contention and performance loss are allowed. When the CPU is idle, the CPU resources are fully utilized. When the CPU is fully loaded, each service type needs to use the CPU proportionally.
Suggestion: You can use CPU share to improve the overall CPU usage while implementing CPU isolation and control when the CPUs are fully loaded.

- Scenario 3: Some jobs are sensitive to performance and CPU resource waste is allowed.
Suggestion: You can use CPU limit to implement absolute CPU isolation between different types of jobs.

19.7.6 Why the Tasks Executed by an Ordinary User Are Slower Than That Executed by the dbadmin User?

The execution speed of an ordinary user is slower than that of the dbadmin user in the following scenarios:

Scenario 1: Ordinary Users Are Subject to Resource Management.

Ordinary users are queuing: **waiting in queue/waiting in global queue/waiting in ccn queue.**

1. Ordinary users are **waiting in queue/waiting in global queue.**
Ordinary users are queuing because the number of active statements exceeds the value of **max_active_statements**. Administrators do not need to wait in queue because they are not subject to resource management. You can change the value of **max_active_statements** on the management console.
 - a. Log in to the GaussDB(DWS) management console.
 - b. In the navigation tree on the left, choose **Clusters > Dedicated Clusters**.
 - c. In the cluster list, find the target cluster and click the cluster name. The **Basic Information** page is displayed.
 - d. Go to the **Parameter Modifications** page of the cluster, search for the **max_active_statements** parameter, change its value, and click **Save**.
2. It takes a long time for an ordinary user to wait in the ccn queue.
When dynamic resource management is enabled (**enable_dynamic_workload=on**), if the concurrency is high and the available memory is small, ordinary users may wait in the ccn queue. Administrators are not subject to these controls. To mitigate this, one can either terminate certain statements or increase the memory parameter value. If the memory consumption on each DN remain low, consider disabling dynamic resource management (**enable_dynamic_workload=off**).

Scenario 2: The OR condition in the execution plan checks the statements executed by common users one by one. This consumes a lot of time.

The **OR** conditions in the execution plans contain permission-related checks. This scenario usually occurs when the system view is used. For example, in the following SQL statement:

```
SELECT distinct(dtp.table_name),
ta.table_catalog,
ta.table_schema,
ta.table_name,
ta.table_type
from information_schema.tables ta left outer join DBA_TAB_PARTITIONS dtp
on (dtp.schema = ta.table_schema and dtp.table_name = ta.table_name)
where ta.table_schema = 'public';
```

Part of the execution plan is as follows:

```

HashAggregate (cost=286.79..282.15 rows=1418 width=377)
  Group By key: (c.relnamespace::character varying(64), (c.postgres::character varying)::information_schema.sql_identifier, (c.nspname)::information_schema.sql_identifier, (c.relname)::information_schema.sql_identifier, (CASE WHEN (c.oid = pg_my_temp_schema()) THEN 'LOCAL_TEMP_SCHEMA'::text WHEN (c.relnamespace = 'pg_catalog' THEN 'pg_catalog'::text WHEN (c.relnamespace = 'pg_catalog') THEN 'pg_catalog'::text WHEN (c.relnamespace = 'pg_catalog') THEN 'pg_catalog'::text ELSE NULL::text END)::information_schema.character_data)
  Hash Cond: (((c.nspname)::information_schema.sql_identifier)::text = ((c.nspname)::character varying(64)::text) AND (((c.relname)::information_schema.sql_identifier)::text = ((c.relname)::character varying(64)::text))
    Hash Join (cost=138.49..134.73 rows=1418 width=377)
      Hash Cond: (t.oid = c.relofftype)
      Hash Join (cost=138.49..134.73 rows=1418 width=4)
        Hash Cond: (t.typnamespace = nt.oid)
        Seq Scan on pg_class t (cost=0.00..46.38 rows=1418 width=4)
        Hash (cost=132.11..126 rows=26 width=4)
          Seq Scan on pg_namespace nt (cost=0.00..1.26 rows=26 width=4)
      Hash (cost=142.08..142.08 rows=1418 width=137)
        Hash Join (cost=142.08..142.08 rows=1418 width=137)
          Hash Cond: (c.relnamespace = nc.oid)
          Seq Scan on pg_class c (cost=0.00..121.91 rows=661 width=72)
            Filter: ((NOT pg_is_other_temp_schema(relnamespace)) AND (relkind = ANY ('r,v,f')::'char')) AND (pg_has_role(reowner, 'USAGE'::text) OR has_table_privilege(c.oid, 'SELECT, INSERT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER'::text) OR has_any_column_privilege(c.oid, 'SELECT, INSERT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER'::text))
            Hash (cost=146.148 rows=1 width=62)
              Seq Scan on pg_namespace nc (cost=0.00..1.48 rows=1 width=48)
              Filter: ((NOT pg_is_other_temp_schema(oid)) AND ((nspname)::information_schema.sql_identifier)::text = 'public'::text))
          Hash (cost=124.121..121 rows=1 width=118)
            Hash Cond: (oid = b.namespace)
            Nested Loop (cost=0.00..12.35 rows=1 width=48)
              Hash Loop Join (cost=0.00..12.40 rows=1 width=72)
                Seq Scan on pg_partition p (cost=0.00..3.79 rows=1 width=4)
                Filter: (parttype = 'p'::'char')
                Index Scan using pg_class_oid_index on pg_class c (cost=0.00..0.27 rows=1 width=76)
                  Index Cond: (oid = b.namespace)
              Index Only Scan using pg_authid_oid_index on pg_authid a (cost=0.00..0.28 rows=1 width=4)
                Index Cond: (oid = c.relnamespace)
            Index Scan using pg_namespace_oid_index on pg_namespace n (cost=0.00..0.28 rows=1 width=68)
              Index Cond: (oid = c.relnamespace)
            Filter: (((nspname)::character varying(64)::text = 'public'::text)
  (37 rows)

```

In the system view, the **OR** condition is used for permission check.

```
pg_has_role(c.reowner, 'USAGE'::text) OR has_table_privilege(c.oid, 'SELECT, INSERT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER'::text) OR has_any_column_privilege(c.oid, 'SELECT, INSERT, UPDATE, REFERENCES'::text)
```

true is always returned for **pg_has_role** of the **dbadmin** use. Therefore, the conditions after **OR** do not need to be checked.

While the **OR** conditions of an ordinary user need to be checked one by one. If there are a large number of tables in the database, the execution time of the ordinary user is longer than that of the **dbadmin** user.

In this scenario, if the number of output result sets is small, you can set **set enable_hashjoin** and **enable_seqscan** to **off**, to use the index+nestloop plan.

Scenario 3: The resource pools allocated to ordinary users and administrators are different.

Run the following command to check whether the resource pools corresponding to an ordinary user are the same as that of the administrator user. If they are different, check whether the tenant resources allocated to the two users are different.

```
SELECT * FROM pg_user;
```

19.7.7 What are the Factors Related to the Single-Table Query Performance in GaussDB(DWS)?

GaussDB(DWS) uses the shared-nothing architecture, and data is stored in a distributed manner. Therefore, the distribution key, data volume, and number of partitions affect the overall query performance of a single table.

1. Distribution Key Design

By default, GaussDB(DWS) takes the first column of the primary key as the distribution key. When you define both a primary key and a distribution key for a table, the distribution key must be a subset of the primary key. Distribution keys determine data distribution among partitions. If distribution keys are well distributed among partitions, query performance can be improved.

If the distribution key is incorrectly selected, data skew may occur after data is imported. The usage of some disks may be much higher than that of other disks, and the cluster may become read-only in some extreme cases. Proper selection of distribution keys is critical to table query performance. In addition, proper distribution keys enable data indexes to be created and maintained more quickly.

2. Data Volume Stored in a Single Table

The larger the amount of data stored in a single table, the poorer the query performance. If a table contains a large amount of data, you need to store the data in partitions. To convert an ordinary table to a partitioned table, you need to create a partitioned table and import data to it from the ordinary table. When you design tables, plan whether to use partitioned tables based on service requirements.

To partition a table, comply with the following principles:

- Use fields with obvious ranges for partitioning, for example, date or region.
- The partition name must reflect the data characteristics of the partition. For example, its format can be Keyword+Range characteristics.
- Set the upper limit of a partition to **MAXVALUE** to prevent data overflow.

3. Number of Partitions

Tables and indexes can be divided into smaller and easier-to-manage units. This significantly reduces search space and improves access performance.

The number of partitions affects the query performance. If the number of partitions is too small, the query performance may deteriorate.

GaussDB(DWS) supports range partitioning and list partitioning. In range partitioning, records are divided and inserted into multiple partitions of a table. Each partition stores data of a specific range (ranges in different partitions do not overlap). List partitioning is supported only by clusters of 8.1.3 and later versions.

When designing a data warehouse, you need to consider these factors and perform experiments to determine the optimal design scheme.

19.8 Snapshot Backup and Restoration

19.8.1 Why Is Creating an Automated Snapshot So Slow?

This happens when the data to be backed up is large. Automated snapshots are incremental backups, and the lower the frequency you set (for example, one week), the longer it takes. Increase backup frequency to speed up the process.

The following table lists the snapshot backup and restoration rates. (The rates are obtained from the lab test environment with local SSDs as the backup media. The rates are for reference only. The actual rate depends on your disk, network, and bandwidth resources.)

- Backup rate: 200 MB/s/DN
- Restoration rate: 125 MB/s/DN

19.8.2 Does a DWS Snapshot Have the Same Function as an EVS Snapshot?

No.

GaussDB(DWS) snapshots are used to restore all the configurations and service data of a cluster. EVS snapshots are used to restore the service data of a data disk or system disk within a specific time period.

DWS Snapshot

A GaussDB(DWS) snapshot is a full or incremental backup of a GaussDB(DWS) cluster at a specific point in time. It records the current database data and cluster information, including the number of nodes, node specifications, and administrator name. Snapshots can be created manually or automatically.

When a snapshot is used for restoration, GaussDB(DWS) creates a new cluster based on the cluster information recorded in the snapshot and restores data from the snapshot.

For details, see "Managing Snapshots" in *Data Warehouse Service Management Guide*.

EVS snapshot

An EVS snapshot is a complete copy or image of the disk data taken at a specific time point. Snapshot is a major disaster recovery approach, and you can completely restore data of a snapshot to the time when the snapshot was created.

You can create snapshots to rapidly save the disk data at specified time points. In addition, you can use snapshots to create new disks so that the created disks will contain the snapshot data in the beginning.

You can create snapshots to rapidly save the disk data at specified time points to implement data disaster recovery.

- If data loss occurs, you can use a snapshot to completely restore the data to the time point when the snapshot was created.
- You can use snapshots to create new disks so that the created disks will contain the snapshot data.

For details, see section "EVS Snapshot (OBT)" in the *Elastic Volume Service Product Description*.