

# Data Warehouse Service

## User Guide

**Issue** 01  
**Date** 2023-03-08



**Copyright © Huawei Technologies Co., Ltd. 2023. All rights reserved.**

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

## **Trademarks and Permissions**



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

## **Notice**

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

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

---

# Contents

---

|  |           |
|--|-----------|
| <b>1 Service Overview</b>                                  | <b>1</b>  |
| 1.1 What Is GaussDB(DWS)?                                  | 1         |
| 1.2 Advantages   | 4         |
| 1.3 Application Scenarios                                  | 6         |
| 1.4 Functions  | 8         |
| 1.5 Concepts   | 13        |
| 1.5.1 GaussDB(DWS) Management Concepts                     | 13        |
| 1.5.2 GaussDB(DWS) Database Concepts                       | 14        |
| 1.6 Related Services                                       | 14        |
| 1.7 GaussDB(DWS) Permissions Management                    | 16        |
| 1.8 GaussDB(DWS) Access                                    | 20        |
| 1.9 Billing  | 23        |
| 1.10 Restrictions  | 24        |
| 1.11 Technical Specifications                              | 24        |
| <b>2 Getting Started</b>                                   | <b>26</b> |
| 2.1 Step 1: Starting Preparations                          | 26        |
| 2.2 Step 2: Creating a Cluster                             | 26        |
| 2.3 Step 3: Connecting to a Cluster                        | 29        |
| 2.4 Step 4: Viewing Other Documents and Clearing Resources | 33        |
| <b>3 Process for Using GaussDB(DWS)</b>                    | <b>35</b> |
| <b>4 Preparations</b>                                      | <b>39</b> |
| <b>5 Cluster Configuration</b>                             | <b>40</b> |
| 5.1 Accessing the GaussDB(DWS) Management Console          | 40        |
| 5.2 Creating a Cluster                                     | 40        |
| <b>6 Cluster Connection</b>                                | <b>51</b> |
| 6.1 Methods of Connecting to a Cluster                     | 51        |
| 6.2 Obtaining the Cluster Connection Address               | 51        |
| 6.3 Using the gsql CLI Client to Connect to a Cluster      | 54        |
| 6.3.1 Downloading the Client                               | 54        |
| 6.3.2 Preparing an ECS as the gsql Client Host             | 56        |
| 6.3.3 Using the gsql Client to Connect to a Cluster        | 57        |

|   |            |
|---|------------|
| 6.3.4 Establishing Secure TCP/IP Connections in SSL Mode.....                                 | 60         |
| 6.3.5 (Optional) Configuring SSL Connection.....  | 64         |
| 6.3.6 (Optional) Downloading the SSL Certificate.....   | 67         |
| 6.4 Using the Data Studio GUI Client to Connect to a Cluster.....                             | 68         |
| 6.5 Using the JDBC and ODBC Drivers to Connect to a Cluster.....                              | 72         |
| 6.5.1 Development Specifications.....   | 72         |
| 6.5.2 Downloading the JDBC or ODBC Driver.....  | 72         |
| 6.5.3 Using a JDBC Driver to Connect to a Database.....                                       | 73         |
| 6.5.4 Using an ODBC Driver to Connect to a Database.....                                      | 84         |
| 6.5.5 Connecting to a Cluster Using IAM Authentication.....                                   | 90         |
| 6.5.5.1 Overview.....   | 90         |
| 6.5.5.2 Granting an IAM Account the DWS Database Access Permission.....                       | 90         |
| 6.5.5.3 Creating an IAM User Credential.....  | 91         |
| 6.5.5.4 Configuring the JDBC Connection to Connect to a Cluster Using IAM Authentication..... | 91         |
| 6.6 Using the Python Library psycopg2 to Connect to a Cluster.....                            | 95         |
| 6.7 Using the Python Library PyGreSQL to Connect to a Cluster.....                            | 105        |
| 6.8 Managing Database Connections.....  | 121        |
| <b>7 Clusters.....</b>  | <b>124</b> |
| 7.1 Checking the Cluster Status.....  | 124        |
| 7.2 Viewing Basic Cluster Information.....  | 127        |
| 7.3 Managing Access Domain Names.....   | 131        |
| 7.4 Cluster Scale-out.....  | 133        |
| 7.5 Performing a Primary/Standby Switchback.....  | 135        |
| 7.6 Cluster Upgrade.....  | 136        |
| 7.7 Password Reset.....   | 137        |
| 7.8 Cluster Restart.....  | 138        |
| 7.9 Modifying Database Parameters.....  | 139        |
| 7.10 MRS Data Sources.....  | 152        |
| 7.10.1 Importing Data from MRS to GaussDB(DWS).....   | 152        |
| 7.10.2 Creating an MRS Data Source Connection.....  | 153        |
| 7.10.3 Updating the MRS Data Source Configuration.....  | 155        |
| 7.11 Managing Logical Clusters.....   | 156        |
| 7.11.1 Overview.....  | 156        |
| 7.11.2 Adding Logical Clusters.....   | 159        |
| 7.11.3 Editing Logical Clusters.....  | 159        |
| 7.11.4 Deleting Logical Clusters.....   | 160        |
| 7.11.5 Restarting Logical Clusters.....   | 161        |
| 7.11.6 Scaling Out Logical Clusters.....  | 161        |
| 7.12 Managing Tags.....   | 162        |
| 7.12.1 Overview.....  | 162        |
| 7.12.2 Tag Management.....  | 163        |
| 7.13 Deleting Clusters.....   | 165        |

|   |            |
|---|------------|
| 7.14 Managing Parameter Templates.....                | 166        |
| 7.15 Managing Clusters That Fail to Be Created.....   | 169        |
| <b>8 Cluster HA.....</b>                              | <b>171</b> |
| 8.1 Snapshots.....                                    | 171        |
| 8.1.1 Overview.....                                   | 171        |
| 8.1.2 Manual Snapshots.....                           | 171        |
| 8.1.2.1 Manually Creating a Snapshot.....             | 172        |
| 8.1.2.2 Deleting Manual Snapshots.....                | 173        |
| 8.1.3 Automated Snapshots.....                        | 174        |
| 8.1.3.1 Automated Snapshot Overview.....              | 174        |
| 8.1.3.2 Configuring an Automated Snapshot Policy..... | 175        |
| 8.1.3.3 Copying Automated Snapshots.....              | 180        |
| 8.1.3.4 Deleting an Automated Snapshot.....           | 181        |
| 8.1.4 Viewing Snapshot Information.....               | 182        |
| 8.1.5 Restoration Using a Snapshot.....               | 183        |
| 8.1.5.1 Restoring a Snapshot to a New Cluster.....    | 183        |
| 8.2 Cluster DR.....                                   | 185        |
| 8.2.1 DR Overview.....                                | 185        |
| 8.2.2 Creating a DR Task.....                         | 186        |
| 8.2.3 Viewing DR Information.....                     | 187        |
| 8.2.4 DR Management.....                              | 188        |
| 8.2.5 Mutually Exclusive DR Cases.....                | 190        |
| 8.3 Associating and Disassociating ELB.....           | 191        |
| 8.4 CNs.....  | 193        |
| <b>9 Monitoring and Alarms.....</b>                   | <b>196</b> |
| 9.1 Monitoring Clusters Using Cloud Eye.....          | 196        |
| 9.2 Databases Monitoring.....                         | 203        |
| 9.2.1 Database Monitoring Overview.....               | 204        |
| 9.2.2 Monitoring Metrics.....                         | 204        |
| 9.2.3 Cluster Overview.....                           | 218        |
| 9.2.4 Monitoring.....                                 | 220        |
| 9.2.4.1 Node Monitoring.....                          | 220        |
| 9.2.4.2 Performance Monitoring.....                   | 223        |
| 9.2.4.3 Database Monitoring.....                      | 225        |
| 9.2.4.4 Session Monitoring.....                       | 226        |
| 9.2.4.5 Query Monitoring.....                         | 227        |
| 9.2.4.6 Instance Monitoring.....                      | 230        |
| 9.2.4.7 Load Monitoring.....                          | 231        |
| 9.2.5 Utilities.....                                  | 234        |
| 9.2.5.1 SQL Diagnosis.....                            | 234        |
| 9.2.6 Settings.....                                   | 237        |
| 9.2.7 Typical Scenarios.....                          | 238        |

|   |            |
|---|------------|
| 9.2.7.1 SQL Diagnosis.....  | 238        |
| 9.2.7.2 Top Time-Consuming SQL Statements Viewing.....  | 239        |
| 9.3 Event Notifications.....  | 239        |
| 9.3.1 Event Notifications Overview.....   | 239        |
| 9.3.2 Subscribing to Event Notifications.....   | 242        |
| 9.3.3 Viewing Events.....   | 244        |
| 9.4 Alarms.....   | 244        |
| 9.4.1 Alarm Management.....   | 245        |
| 9.4.2 Alarm Rules.....  | 249        |
| 9.4.3 Alarm Subscriptions.....  | 252        |
| <b>10 Cluster Security Management.....</b>  | <b>256</b> |
| 10.1 Configuring Separation of Permissions.....   | 256        |
| 10.2 Encrypting Databases.....  | 259        |
| 10.2.1 Overview.....  | 259        |
| 10.2.2 Rotating Encryption Keys.....  | 261        |
| 10.3 Permissions.....   | 261        |
| 10.3.1 Syntax of Fine-Grained Permissions Policies.....   | 262        |
| 10.3.2 RBAC Syntax of RBAC Policies.....  | 266        |
| 10.4 Fine-Grained Permissions Policies.....   | 268        |
| <b>11 Audit Logs.....</b>   | <b>274</b> |
| 11.1 Overview.....  | 274        |
| 11.2 Viewing Audit Logs of Key Operations on the Management Console.....                        | 275        |
| 11.3 Configuring the Database Audit Logs.....   | 277        |
| 11.4 Dumping the Database Audit Logs.....   | 279        |
| <b>12 FAQs.....</b>   | <b>283</b> |
| 12.1 General Problems.....  | 283        |
| 12.1.1 Why Are Data Warehouses Necessary?.....  | 283        |
| 12.1.2 What Are the Differences Between a Data Warehouse and the Hadoop Big Data Platform?..... | 284        |
| 12.1.3 Why Should I Use GaussDB(DWS)?.....  | 285        |
| 12.1.4 When Should I Use GaussDB(DWS) and MRS?.....   | 285        |
| 12.1.5 What Are the Differences Between GaussDB(DWS) and Hive in Functions?.....                | 285        |
| 12.1.6 What Is the User Quota?.....   | 286        |
| 12.1.7 What Are the Differences Between Users and Roles?.....                                   | 286        |
| 12.1.8 How Do I Check the Creation Time of a Database User?.....                                | 287        |
| 12.1.9 Regions and AZs.....   | 288        |
| 12.1.10 Is My Data Secure in GaussDB(DWS)?.....   | 289        |
| 12.1.11 How Is GaussDB(DWS) Secured?.....   | 290        |
| 12.1.12 Can I Modify the Security Group of a GaussDB(DWS) Cluster?.....                         | 290        |
| 12.1.13 What Is a Database/Data Warehouse/Data Lake/Lakehouse?.....                             | 291        |
| 12.2 Cluster Management.....  | 294        |
| 12.2.1 What Do I Do If Creating a GaussDB(DWS) Cluster Failed?.....                             | 294        |

|   |     |
|---|-----|
| 12.2.2 How Can I Clear and Reclaim the Storage Space?.....  | 294 |
| 12.2.3 Why Did the Used Storage Shrink After Scale-out?.....  | 295 |
| 12.2.4 How Do I View Node Metrics (CPU, Memory, and Disk Usage)?.....   | 295 |
| 12.2.5 Does GaussDB(DWS) Support a Single Node for a Learning Environment?.....                                   | 295 |
| 12.2.6 How Is the Disk Space or Capacity of GaussDB(DWS) Calculated?.....   | 296 |
| 12.2.7 What Are the gaussdb and postgres Databases of GaussDB(DWS)?.....  | 296 |
| 12.2.8 How Do I Set the Maximum Number of Sessions When Adding an Alarm Rule on Cloud Eye?....                    | 296 |
| 12.2.9 When Should I Add CNs or Scale out a cluster?.....   | 297 |
| 12.2.10 What Are the Differences Between Hot Data Storage and Cold Data Storage?.....                             | 298 |
| 12.3 Database Connections.....  | 299 |
| 12.3.1 How Applications Communicate with GaussDB(DWS)?.....   | 299 |
| 12.3.2 Does GaussDB(DWS) Support Third-Party Clients and JDBC and ODBC Drivers?.....                              | 302 |
| 12.3.3 Can I Connect to GaussDB(DWS) Cluster Nodes Using SSH?.....  | 302 |
| 12.3.4 What Should I Do If I Cannot Connect to a Data Warehouse Cluster?.....                                     | 302 |
| 12.3.5 Why Was I Not Notified of Failure Unbinding the EIP When GaussDB(DWS) Is Connected Over the Internet?..... | 303 |
| 12.3.6 How Do I Configure a Whitelist to Protect Clusters Available Through a Public IP Address?.....             | 304 |
| 12.4 Data Import and Export.....  | 304 |
| 12.4.1 What Are the Differences Between Data Formats Supported by OBS and GDS Foreign Tables?....                 | 304 |
| 12.4.2 How Do I Import Incremental Data Using an OBS Foreign Table?.....  | 304 |
| 12.4.3 How Can I Import Data to GaussDB(DWS)?.....  | 304 |
| 12.4.4 How Much Service Data Can a Data Warehouse Store?.....   | 305 |
| 12.4.5 How Do I Use \Copy to Import and Export Data?.....   | 305 |
| 12.4.6 Can I Export GaussDB(DWS) Data from/to OBS Across Regions?.....  | 305 |
| 12.4.7 How Do I Import GaussDB(DWS)/Oracle/MySQL/SQL Server Data to GaussDB(DWS) (Whole Database Migration)?..... | 306 |
| 12.4.8 Can I Import Data over the Public/External Network Using GDS?.....   | 306 |
| 12.4.9 Which Are the Factors That Affect GaussDB(DWS) Import Performance?.....                                    | 306 |
| 12.5 Account, Password, and Permissions.....  | 306 |
| 12.5.1 How Does GaussDB(DWS) Implement Workload Isolation?.....   | 306 |
| 12.5.2 How Do I Change the Password of a Database Account When the Password Expires?.....                         | 310 |
| 12.5.3 How Do I Grant Table Permissions to a User?.....   | 311 |
| 12.5.4 How Do I Grant Schema Permissions to a User?.....  | 315 |
| 12.5.5 How Do I Create a Database Read-only User?.....  | 318 |
| 12.5.6 How Do I Create Private Database Users and Tables?.....  | 318 |
| 12.5.7 How Do I Revoke the CONNECT ON DATABASE Permission from a User?.....                                       | 320 |
| 12.5.8 How Do I View the Table Permissions of a User?.....  | 321 |
| 12.5.9 Who Is User Ruby?.....   | 324 |
| 12.6 Database Usage.....  | 324 |
| 12.6.1 How Do I Change Distribution Columns?.....   | 324 |
| 12.6.2 How Do I View and Set the Database Character Encoding?.....  | 326 |
| 12.6.3 What Do I Do If Date Type Is Automatically Converted to the Timestamp Type During Table Creation?.....     | 327 |

|   |            |
|---|------------|
| 12.6.4 Do I Need to Run VACUUM FULL and ANALYZE on Common Tables Periodically?.....                               | 328        |
| 12.6.5 Do I Need to Set a Distribution Key After Setting a Primary Key?.....                                      | 330        |
| 12.6.6 Is GaussDB(DWS) Compatible with PostgreSQL Stored Procedures?.....   | 330        |
| 12.6.7 What Are Partitioned Tables, Partitions, and Partition Keys?.....  | 330        |
| 12.6.8 How Can I Export the Table Structure?.....   | 331        |
| 12.6.9 How Can I Export All Tables and Views from a Database?.....  | 331        |
| 12.6.10 How Can I Delete Table Data Efficiently?.....   | 331        |
| 12.6.11 How Do I View Foreign Table Information?.....   | 332        |
| 12.6.12 If No Distribution Column Is Specified, How Will Data Be Stored?.....                                     | 333        |
| 12.6.13 How Do I Replace the Null Result with 0?.....   | 334        |
| 12.6.14 How Do I Check Whether a Table Is Row-Stored or Column-Stored?.....                                       | 335        |
| 12.6.15 How Do I Query the Information About GaussDB(DWS) Column-Store Tables?.....                               | 335        |
| 12.6.16 Why Sometimes the GaussDB(DWS) Query Indexes Become Invalid?.....   | 336        |
| 12.6.17 How Do I Use a User-Defined Function to Rewrite the CRC32() Function?.....                                | 344        |
| 12.6.18 What Are the Schemas Starting with <b>pg_toast_temp*</b> or <b>pg_temp*</b> ?.....                        | 345        |
| 12.6.19 Solutions to Inconsistent GaussDB(DWS) Query Results.....   | 345        |
| 12.6.20 Which System Catalogs That the VACUUM FULL Operation Cannot Be Performed on?.....                         | 350        |
| 12.6.21 In Which Scenarios Would a Statement Be "idle in transaction"?.....                                       | 352        |
| 12.7 Database Performance.....  | 354        |
| 12.7.1 Why Is SQL Execution Slow After Long GaussDB(DWS) Usage?.....  | 354        |
| 12.7.2 Why Does GaussDB(DWS) Perform Worse Than a Single-Server Database in Extreme Scenarios?<br>.....           | 354        |
| 12.7.3 How Can I View SQL Execution Records in a Certain Period When Read and Write Requests Are<br>Blocked?..... | 355        |
| 12.8 Snapshot Backup and Restoration.....   | 355        |
| 12.8.1 Why Is Creating an Automated Snapshot So Slow?.....  | 355        |
| 12.8.2 Where Can a Snapshot Be Restored To?.....  | 355        |
| 12.8.3 Does a DWS Snapshot Have the Same Function as an EVS Snapshot?.....  | 355        |
| <b>A Change History.....</b>  | <b>357</b> |



# 1 Service Overview

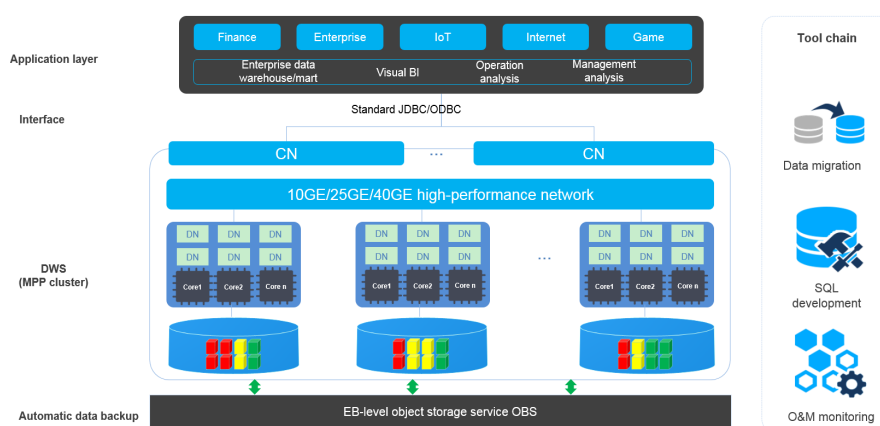
## 1.1 What Is GaussDB(DWS)?

GaussDB(DWS) is an online data processing database that runs on the Huawei Cloud 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.

### 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

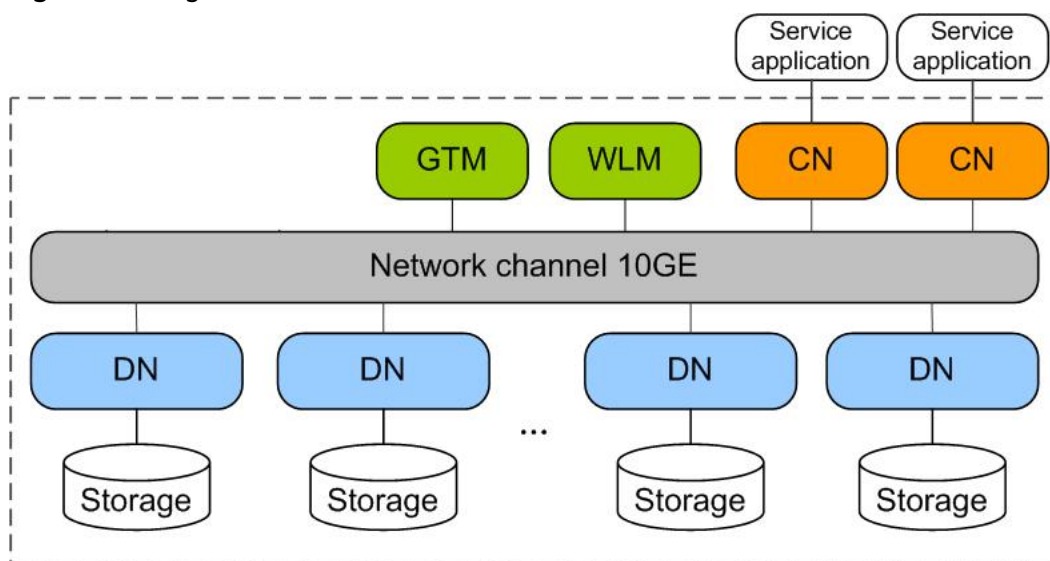


- Application layer**  
 Data loading tools, extract, transform, and load (ETL) tools, business intelligence (BI) tools, as well as data mining and analysis tools, can be integrated with GaussDB(DWS) through standard APIs. GaussDB(DWS) is compatible with the PostgreSQL ecosystem, and the SQL syntax is 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) (MPP cluster)**  
 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 DNs 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

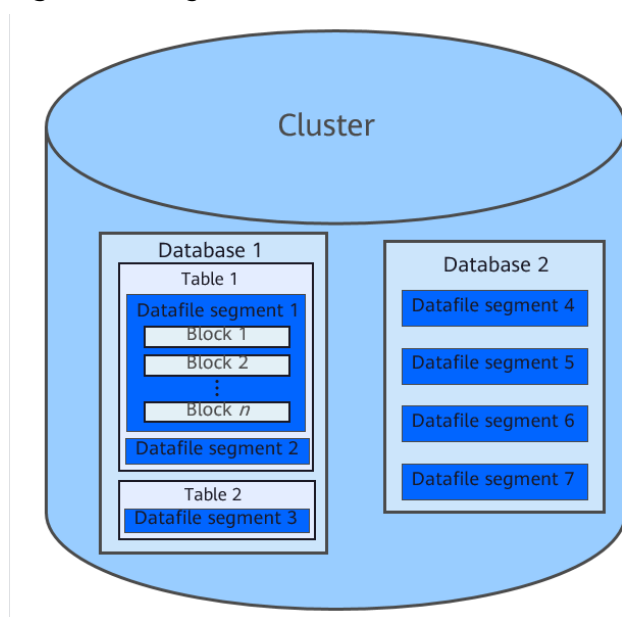
| Name                             | Description   | Remarks  |
|----------------------------------|---|--|
| Global Transaction Manager (GTM) | Generates and maintains the globally unique information, such as the transaction ID, transaction snapshot, and timestamp.   | The cluster includes only one pair of GTMs: one primary GTM and one standby GTM.   |
| Workload Manager (WLM)           | WLM 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.   |
| 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 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> <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> |
| 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.   | A cluster consists of multiple DNs and each DN stores part of data. If no HA solution is available for DNs, data cannot be accessed when a DN is faulty.   |
| 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

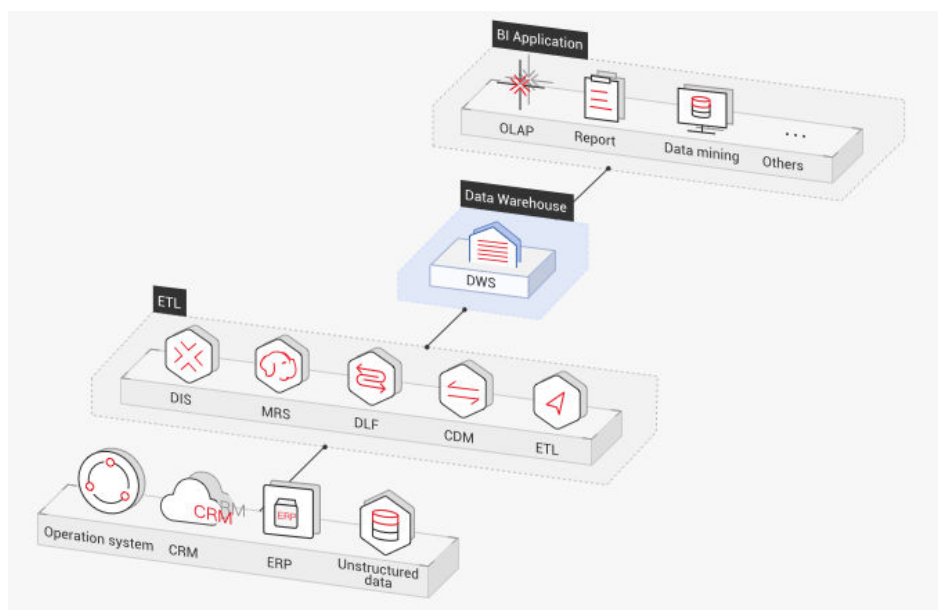
- ACID  
Support for the atomicity, consistency, isolation, and durability (ACID) feature, which ensures strong data consistency for distributed transactions.
- Comprehensive HA design  
All software processes of GaussDB(DWS) are in active/standby mode. Logical components such as the CNs and DNPs of 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

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

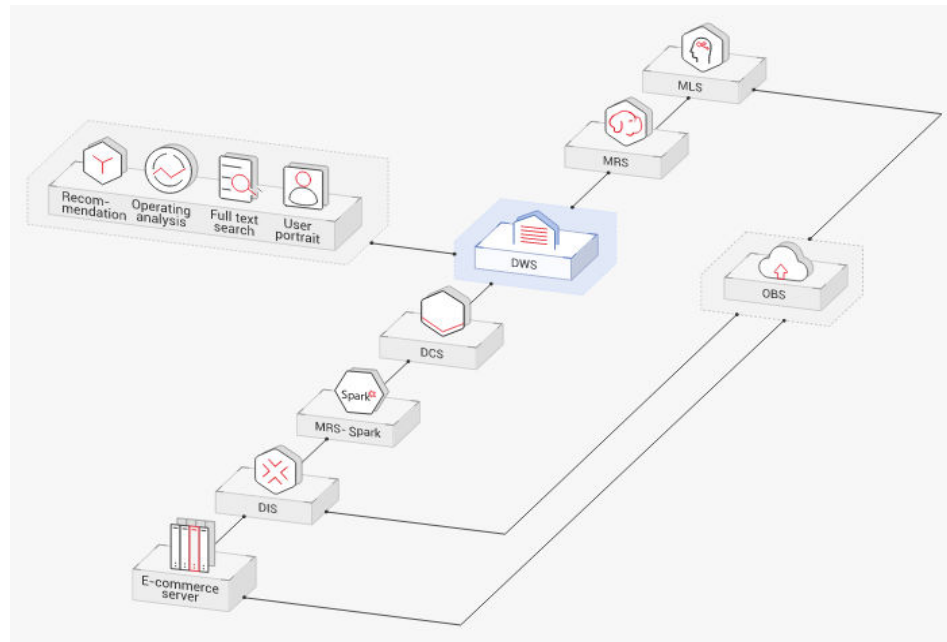


The data warehouse is the pillar of the Business Intelligence (BI) system for collecting, storing, and analyzing massive amounts of data. It provides powerful business analysis support for IoT, mobile Internet, gaming, and Online to Offline (O2O) industries.

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

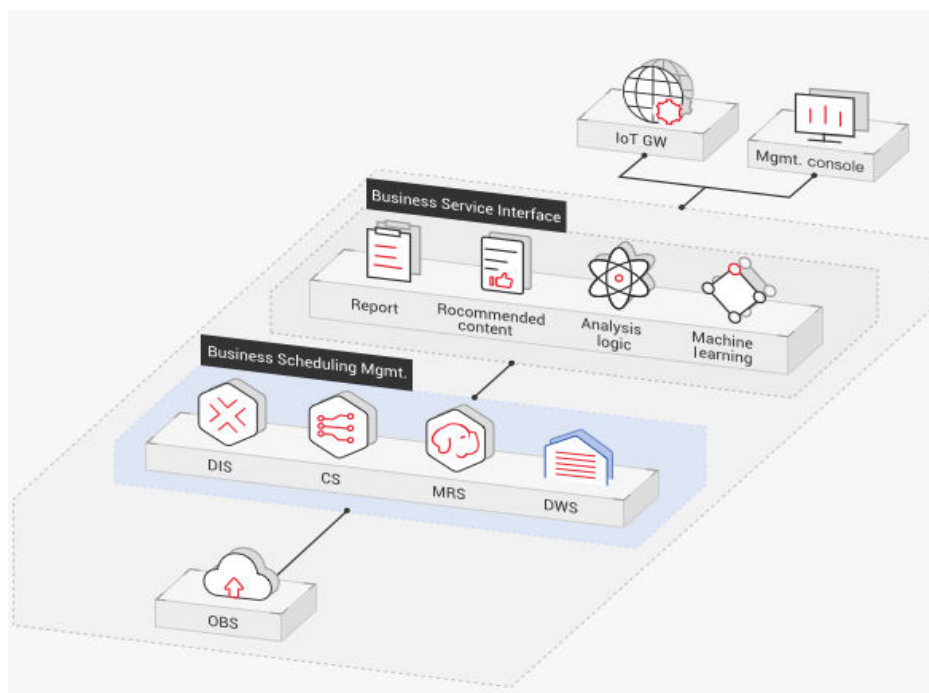


Data of online retailers is mainly used for marketing recommendation, operating and customer analysis, and full text search.

Advantages of GaussDB(DWS) are as follows:

- **Multi-dimensional analysis:** analysis from products, users, operation, and regions
- **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 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:

- **Real-time archiving of stream data:** importing stream data from IoT devices and the gateway to GaussDB(DWS) using DIS
- **Device monitoring and prediction:** device monitoring, control, optimization, supply, self-diagnosis, and self-healing based on data analysis and prediction
- **Information recommendation:** tailed 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
- 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 SQL 92 and SQL 2003 standards, stored procedures, GBK and UTF-8 character sets, and SQL standard functions and OLAP analysis functions.
- Compatible with the PostgreSQL ecosystem and supports interconnection with mainstream database ETL and BI tools provided by third-party vendors.

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 a cluster  
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 creates a new cluster with the same flavor and node quantity as the original one, and imports the snapshot data.

You can delete snapshots that are no longer needed to release the storage space.

- 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 for a scale-out, add them in three batches (60 in 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 a cluster**

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.
- **Managing workloads**

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) workload management function to handle such problems. GaussDB(DWS) workload management uses workload queues as resource bearers. You can create different workload queues for different service types and configure different resource ratios for these queues. Then, add database users to the corresponding queues to restrict their resource usages.
- **Logical cluster**

A physical cluster can be divided into logical clusters that use the node-group mechanism. Tables in a database can be allocated to different physical nodes by logical cluster. A logical cluster can contain tables from multiple databases.
- **Restarting a cluster**

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 a cluster**

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 and snapshots 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".
- **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 Import" in the *Data Warehouse Service (DWS) Developer Guide*.

- Concurrently Importing Data from OBS
- Using GDS to Import Data from a Remote Server
- Running the INSERT Statement to Insert Data
- Running the COPY FROM STDIN Statement to Import Data
- Using a gsql Meta-Command to Import Data
- Importing Data from MRS to a Data Warehouse Cluster
- Using Database Schema Converter (DSC) to Migrate SQL Scripts

In addition, GaussDB(DWS) supports data import using mainstream third-party ETL tools.

## 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 cross-AZ 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 "Cluster Monitoring" in the *Data Warehouse Service (DWS) User Guide*.
- **Monitoring databases**  
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 *Data Warehouse Service User Guide*.
- **Managing 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 "Alarms" in *Data Warehouse Service User Guide*.
- **Event notification**  
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 "Configuring the 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*.

- **gsqL**  
gsqL 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.

## 1.5 Concepts

### 1.5.1 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

Each data warehouse cluster contains at least three nodes, all of which support data storage and analysis.

#### Flavor

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.

## 1.5.2 GaussDB(DWS) Database Concepts

### Database

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 Connection

You can use a client to connect to a GaussDB(DWS) cluster in the Huawei Cloud 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.

## MRS

Data can be migrated from MRS to GaussDB(DWS) clusters for analysis after the data is processed by Hadoop.

## CDM

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

## DIS

You can use Data Ingestion Service (DIS) to ingest 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.

## 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.

## 1.7 GaussDB(DWS) Permissions Management

If you need to assign different permissions to employees in your enterprise to access your Huawei Cloud GaussDB(DWS) resources, 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 Huawei Cloud resources.

With IAM, you can use your Huawei Cloud account to create IAM users for your employees, and assign permissions to the users to control their access to specific resource types. For example, some software developers in your enterprise need to use GaussDB(DWS) resources but must not delete them or perform any high-risk operations. To this end, you can create IAM users for the software developers and grant them only the permissions required for using GaussDB(DWS) resources.

If your Huawei Cloud 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      | Administrator permissions for GaussDB(DWS). Users granted these permissions can perform all operations on GaussDB(DWS).  | System-defined policy | N/A  |
| DWS Administrator   | Administrator permissions for GaussDB(DWS). Users granted these permissions can perform operations on all GaussDB(DWS) resources. <ul style="list-style-type: none"><li>• Users granted permissions of the <b>VPC Administrator</b> policy can create VPCs and subnets.</li><li>• Users granted permissions of the <b>Cloud Eye Administrator</b> policy can view monitoring information of data warehouse clusters.</li></ul> | System-defined role   | Dependent on the <b>Tenant Guest and Server Administrator</b> policies, which must be assigned in the same project as the <b>DWS Administrator</b> 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 <b>DWS Administrator</b> policy, which must be assigned in the same project as the <b>DWS Database Access</b> 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.
- By default, only Huawei Cloud accounts or the users with **Security Administrator** permissions can create agencies. By default, the IAM users in those accounts cannot create agencies. When the users use the EIP, the system displays a message indicating insufficient permissions. Contact a user with the **DWS FullAccess** 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                    | √              | √                  | √                 | √                   |
| Obtaining the details of a cluster            | √              | √                  | √                 | √                   |
| 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 |
|--|----------------|--------------------|-------------------|---------------------|
| Applying parameter templates to clusters | √              | x                  | √                 | x                   |
| Deleting clusters                        | √              | x                  | √                 | x                   |
| Configuring maintenance windows          | √              | x                  | √                 | x                   |
| Binding EIPs                             | √              | x                  | √                 | x                   |
| Unbinding EIPs                           | √              | x                  | √                 | x                   |
| Creating DNS domain names                | √              | x                  | √                 | x                   |
| Releasing DNS domain names               | √              | x                  | √                 | x                   |
| Modifying DNS domain names               | √              | x                  | √                 | x                   |
| Creating MRS connections                 | √              | x                  | √                 | x                   |
| Updating MRS connections                 | √              | x                  | √                 | x                   |
| Deleting MRS connections                 | √              | x                  | √                 | x                   |
| Adding/Deleting tags                     | √              | x                  | √                 | x                   |
| Editing tags                             | √              | x                  | √                 | x                   |
| Creating snapshots                       | √              | x                  | √                 | x                   |

| Operation                     | DWS FullAccess | DWS ReadOnlyAccess | DWS Administrator | DWS Database Access |
|-------------------------------|----------------|--------------------|-------------------|---------------------|
| Obtaining the snapshot list   | √              | √                  | √                 | √                   |
| Deleting snapshots            | √              | x                  | √                 | x                   |
| Copying snapshots             | √              | x                  | √                 | x                   |
| Creating parameter templates  | √              | x                  | √                 | x                   |
| Deleting parameter templates  | √              | x                  | √                 | x                   |
| Modifying parameter templates | √              | x                  | √                 | x                   |

## 1.8 GaussDB(DWS) Access

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

Figure 1-7 Process for using GaussDB(DWS)

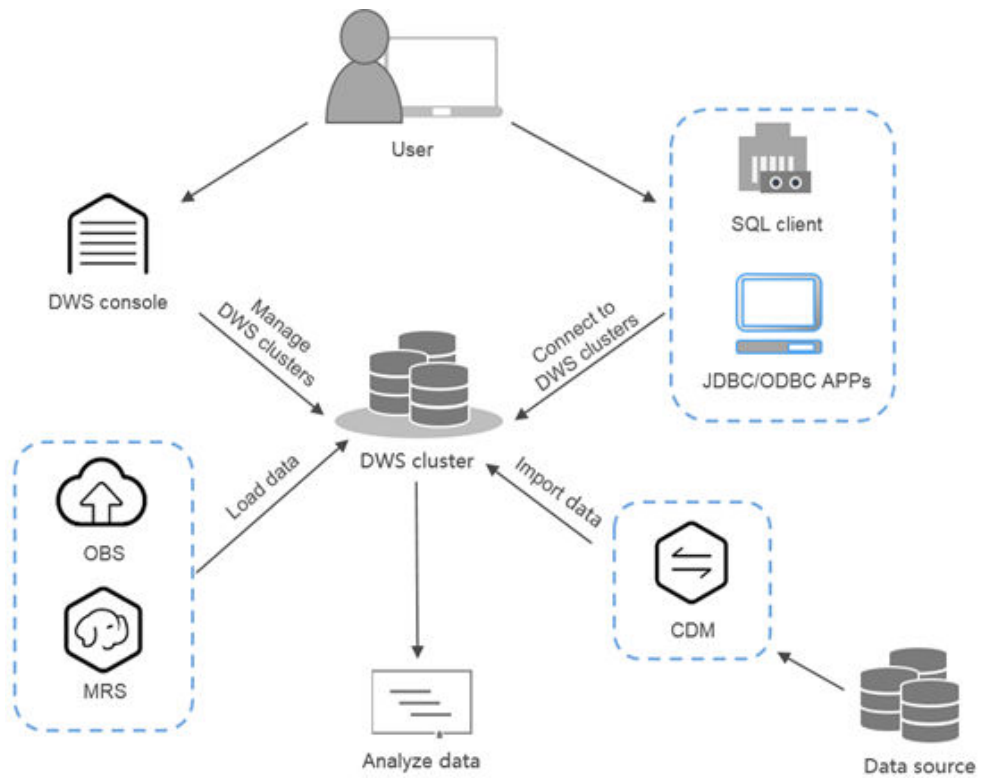
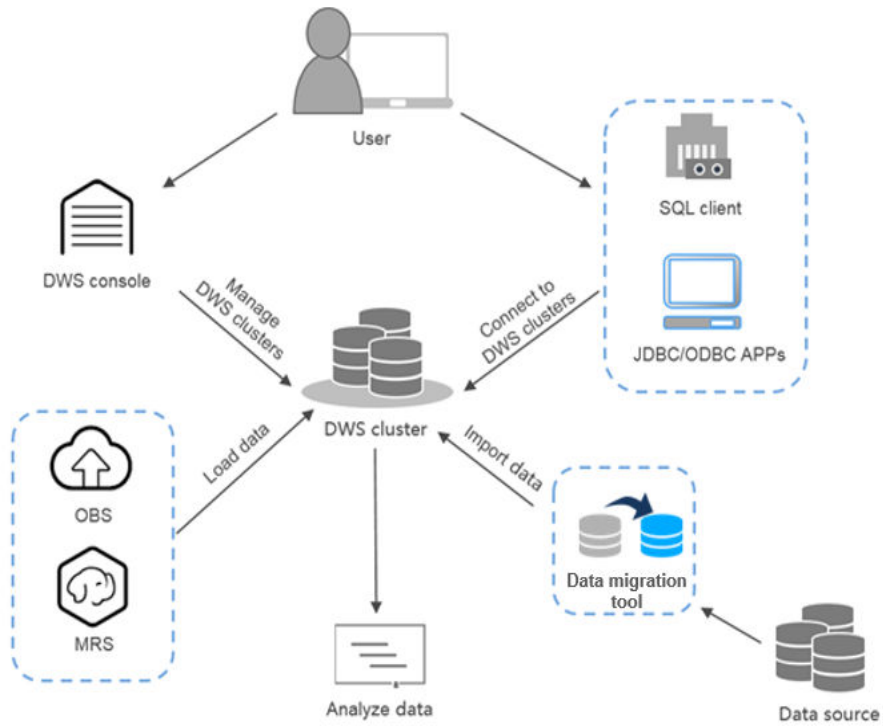


Figure 1-8 Process for using GaussDB(DWS)



## Accessing a Cluster

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

## 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 Clusters" in the *Data Warehouse Service (DWS) User Guide*.

- Open-source PostgreSQL clients

The following lists compatible open-source clients:

- PostgreSQL psql 9.2.4 or later

For more information, see <https://www.postgresql.org/>.

- pgadmin

For more information, see <https://www.pgadmin.org/>.

- dbeaver

For more information, see <https://dbeaver.jkiss.org/download/>.

- JDBC and ODBC API calling

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

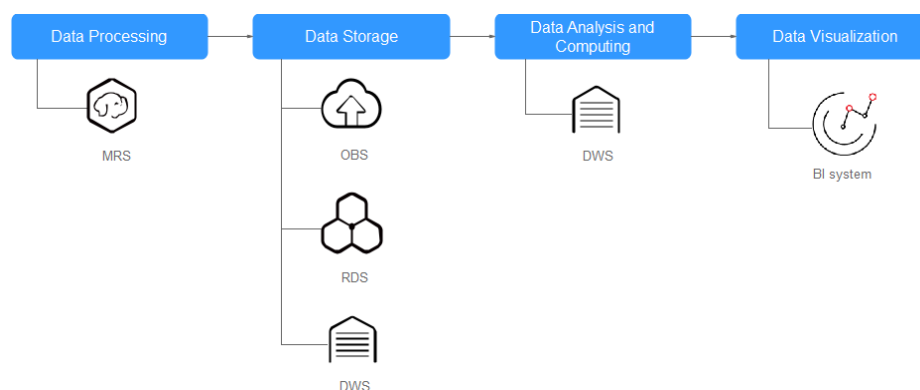
For details, see "Using the JDBC and ODBC Drivers to Connect to a Cluster" in the *Data Warehouse Service (DWS) User Guide*.

## End-to-End Data Analysis Process

GaussDB(DWS) has been seamlessly integrated with other services on the Huawei Cloud, 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 1-9** End-to-end data analysis process



## 1.9 Billing

### Item

To use G42 GaussDB(DWS), you only need to pay for the data warehouse node flavor you choose and corresponding resources you use.

| Item                                     | Description   |
|--|---|
| Data warehouse node                      | You pay for the node flavor you choose. <ul style="list-style-type: none"><li>Pay per use (hourly)</li></ul>  |
| Snapshot storage space                   | GaussDB(DWS) provides some free-of-charge storage space for you to store the snapshot data. However, when you use more storage space than the free-of-charge storage space, the excessive space usage will be billed based on OBS billing standards on a pay-per-use basis.<br><br>The free-of-charge space is the same as the size of the total storage space of your cluster. (Free-of-charge space = Storage space of a single node x Number of nodes) |
| Cloud data warehouse storage             | You can customize the storage type and capacity used by the cluster. The storage resources are charged separately.  |
| (Optional) EIP and bandwidth             | <ul style="list-style-type: none"><li>Bind an EIP to a GaussDB(DWS) cluster. The EIP is billed as per VPC EIP pricing rules.</li><li>You are not billed for the traffic generated by the GaussDB(DWS) cluster on the G42 network.</li></ul>   |
| (Optional) Database encryption key (DEK) | If the <b>Encrypt DataStore</b> function is enabled when you create a GaussDB(DWS) cluster, keys of Key Management Service (KMS) are required to encrypt and decrypt the database. Keys are billed based on the DEW billing standards.  |

### Billing Mode

- Pay per use (hourly):** In this billing mode, you can enable or disable GaussDB(DWS) as you like. You are charged based on your usage duration (accurate to minutes). A bill is generated every natural hour.

### Changing Configurations

GaussDB(DWS) provides multiple node flavors. After a cluster is created, you can add nodes to scale it out as your service volume increases.

If the scale-out configuration change method provided by GaussDB(DWS) does not meet your requirements, you can create a cluster again and migrate data to the cluster to realize cluster configuration change.

## 1.10 Restrictions

- 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.
- You cannot change the flavor of an existing cluster. If you need nodes with a higher flavor, create a new one.
- If you use a client to connect to a cluster, its VPC subnet must be the same as that of the cluster.
- 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.

## 1.11 Technical Specifications

This section describes the technical specifications of GaussDB(DWS) in different versions.

**Table 1-4** GaussDB(DWS) technical specifications

| Technical Specifications               | Maximum Value of 8.0.x | Maximum Value of 8.1.0 | Maximum Value of 8.1.1 |
|--|------------------------|------------------------|------------------------|
| Data capacity                          | 10 PB                  | 10 PB                  | 20 PB                  |
| Number of cluster nodes                | 256                    | 256                    | 2048                   |
| Size of a single table                 | 1 PB                   | 1 PB                   | 1 PB                   |
| Size of data in each row               | 1 GB                   | 1 GB                   | 1 GB                   |
| Size of a single column in each record | 1 GB                   | 1 GB                   | 1 GB                   |
| Number of records in each table        | 2 <sup>55</sup>        | 2 <sup>55</sup>        | 2 <sup>55</sup>        |
| Number of columns in each table        | 1600                   | 1600                   | 1600                   |
| Number of indexes in each table        | Unlimited              | Unlimited              | Unlimited              |



| <b>Technical Specifications</b>                            | <b>Maximum Value of 8.0.x</b>  | <b>Maximum Value of 8.1.0</b>  | <b>Maximum Value of 8.1.1</b>  |
|--|--|--|--|
| Number of columns in the index of each table               | 32   | 32   | 32   |
| Number of constraints in each table                        | Unlimited  | Unlimited  | Unlimited  |
| Number of concurrent connections                           | 60 for analytical long transactions;<br>5,000 for short transactions | 60 for analytical long transactions;<br>5,000 for short transactions | 80 for analytical long transactions;<br>5,000 for short transactions |
| Number of partitions in a partitioned table                | 32,768   | 32768  | 32768  |
| Size of each partition in a partitioned table              | 1 PB   | 1 PB   | 1 PB   |
| Number of records in each partition in a partitioned table | $2^{55}$   | $2^{55}$   | $2^{55}$   |

# 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 together 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** Choose **Clusters** in the navigation pane on the left.
- Step 3** On the **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 account and its password to connect to the cluster's database.
  - **Administrator Password:** Enter the password.
  - **Confirm Password:** Enter the 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.

**Figure 2-1** Configuring the cluster

|                        |   |   |
|------------------------|---|---|
| Cluster Name           | <input type="text"/>                              | ? |
| Cluster Version        | <input type="checkbox"/> <input type="checkbox"/> |   |
| Default Database       | gaussdb   |   |
| Administrator Account  | <input type="text" value="dbadmin"/>              | ? |
| Administrator Password | <input type="password"/>                          |   |
| Confirm Password       | <input type="password"/>                          |   |
| Database Port          | <input type="text" value="8000"/>                 | ? |

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

- **Subnet:** When you create a VPC, a subnet is created by default. You can select the corresponding subnet.
- **Security Group:** Select **Automatic creation**.

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<br><b>NOTE</b><br>Enter the value of <b>Database Port</b> 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 <b>IP address</b> , and enter the IP address and subnet mask of the client host that accesses GaussDB(DWS), for example, <b>192.168.0.10/16</b> .  |

- **EIP:** Select **Automatically assign** to apply for a cluster EIP as the public network IP address of the cluster. In addition, set the EIP bandwidth.

**Step 8** 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.
  - **Parameter Template:** The default database parameter template is associated with the cluster.
  - **Tag:** By default, no tag is added to the cluster.
  - **Encrypt DataStore:** This parameter is disabled by default, indicating that the database is not encrypted.
- **Custom:** Select this option to configure the following advanced parameters: **CNs, Parameter Template, Tag, Encrypt DataStore**

**Step 9** Click **Create Now**. The **Confirm** page is displayed.

**Step 10** Click **Submit**.

After the submission is successful, the creation starts. Click **Back to Cluster List**. The **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 [Methods of Connecting to a Cluster](#).

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 database 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 tree on the left, click **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 **Connections**.

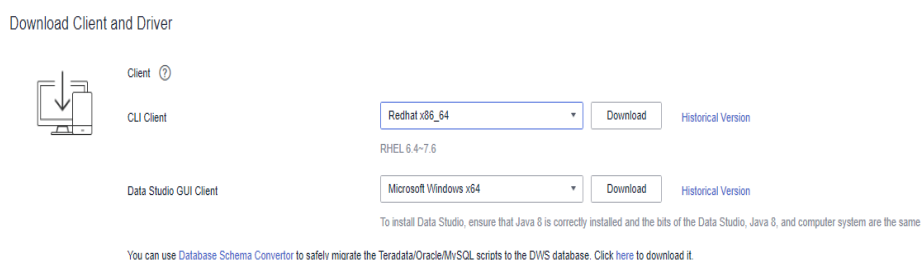
**Step 4** On the **Download Client and Driver** page, download **Data Studio GUI Client**.

- 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.

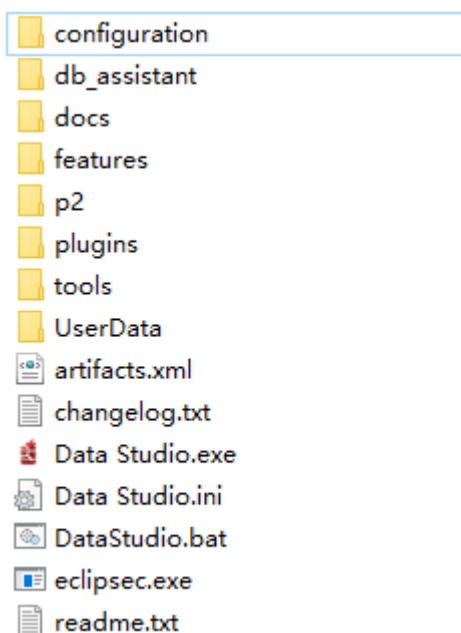
**Figure 2-2** Downloading clients



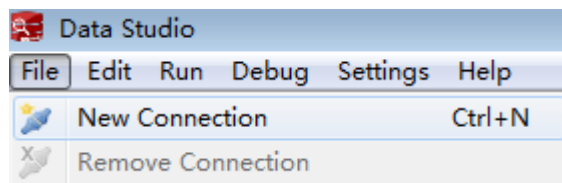
**Step 5** Decompress the downloaded client software package (32-bit or 64-bit) to the installation directory.

**Step 6** Open the installation directory and double-click **Data Studio.exe** to start the Data Studio client. See [Figure 2-3](#).

**Figure 2-3** Starting the client



**Step 7** Choose **File > New Connection** from the main menu. See [Figure 2-4](#).

**Figure 2-4** Creating a connection

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

**Table 2-2** Connection parameters

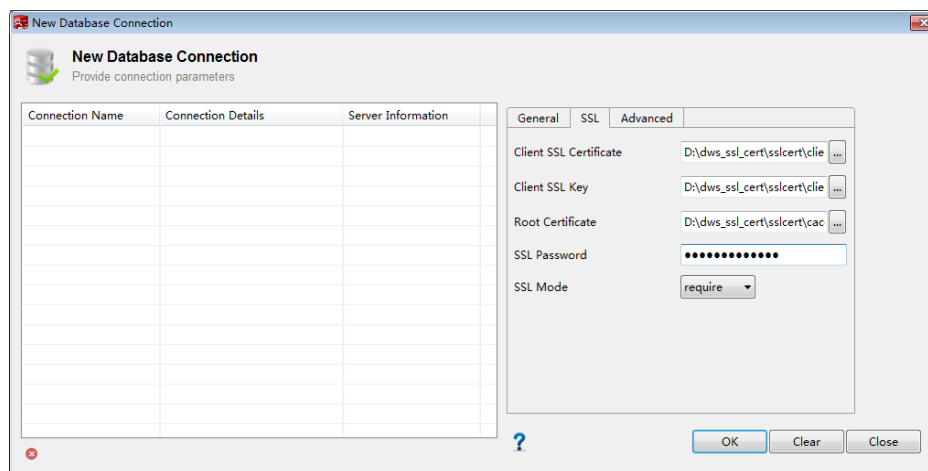
| Parameter       | Description  | Example   |
|-----------------|--|-----------|
| Database Type   | Select <b>GaussDB A</b> .  | GaussDB A |
| Connection Name | Name of a connection   | dws-demo  |
| Host            | IP address (IPv4) or domain name of the cluster to be connected  | -         |
| Host Port       | Database port  | 8000      |
| Database Name   | Database name  | gaussdb   |
| User Name       | 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"><li>● <b>Current Session Only</b>: The password is saved only in the current session.</li><li>● <b>Do Not Save</b>: The password is not saved.</li></ul> | -         |
| Enable SSL      | If <b>Enable SSL</b> 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 and decompress the SSL certificate. For details, see [\(Optional\) Downloading the SSL Certificate](#). Click the **SSL** tab and configure the following parameters:

**Table 2-3** Configuring SSL parameters

| Parameter              | Description   |
|------------------------|---|
| Client SSL Certificate | Select the <b>sslcert\client.crt</b> file in the decompressed SSL certificate directory.  |
| Client SSL Key         | Only the PK8 format is supported. Select the <b>sslcert\client.key.pk8</b> file in the directory where the SSL certificate is decompressed.   |
| Root Certificate       | When <b>SSL Mode</b> is set to <b>verify-ca</b> , the root certificate must be configured. Select the <b>sslcert\cacert.pem</b> 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"> <li>• require</li> <li>• verify-ca</li> </ul> GaussDB(DWS) does not support the <b>verify-full</b> mode.            |

**Figure 2-5** Configuring SSL parameters



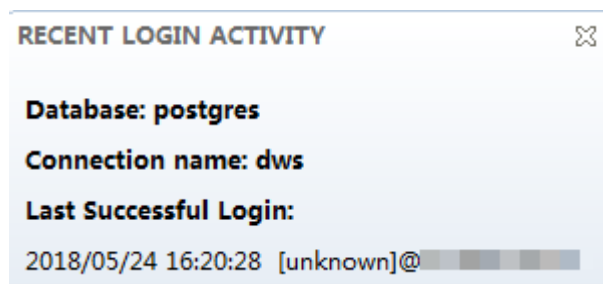
**Step 9** Click **OK** to establish the database connection.

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 2-6 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 Clearing 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 (DWS) User Guide*: This guide provides detailed information about the concepts and operations related to creating, managing, monitoring, and connecting clusters.
- *Data Warehouse Service (DWS) Developer Guide*: This guide provides comprehensive and detailed information on how to build, manage, and query GaussDB(DWS) databases, covering 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 **Clusters**, 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 Huawei Cloud 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)

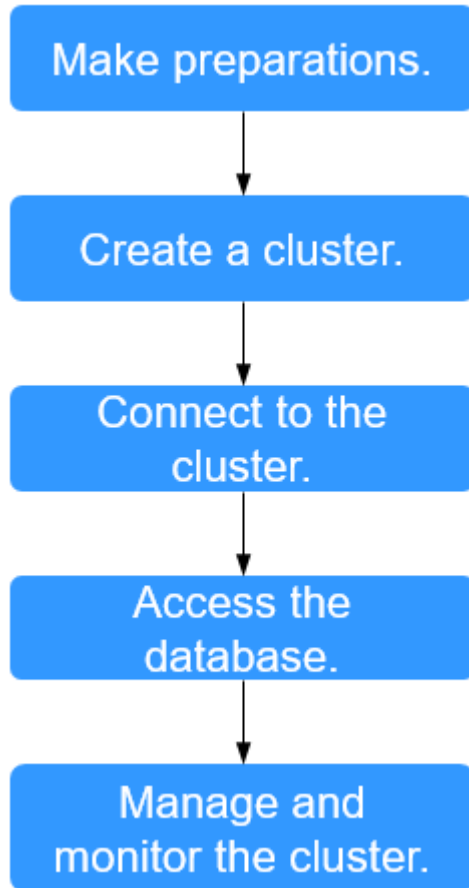


Table 3-1 Process description

| 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. | <b>Preparations</b>   |

| Process                         | Task                 | Description  | Operation Instruction                              |
|---------------------------------|----------------------|--|--|
| 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.   | <a href="#">Creating a Cluster</a>                 |
| 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 <b>Connection Management</b> page of the GaussDB(DWS) management console. | <a href="#">Methods of Connecting to a Cluster</a> |
| 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.  | <a href="#">Managing clusters</a>                  |
|                                 | Manage the snapshot. | Create snapshots to back up and restore the cluster.   | <a href="#">Managing snapshots</a>                 |

| Process | Task                        | Description  | Operation Instruction  |
|---------|-----------------------------|--|--|
|         | 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"> <li>● <a href="#">Monitoring Clusters Using Cloud Eye</a></li> <li>● <a href="#">Event Notifications</a></li> <li>● <a href="#">Audit Logs</a></li> <li>● "Resource Load Management"<br/>For details, see "Resource Load Management" in the <i>Data Warehouse Service (DWS) Developer Guide</i>.</li> </ul> |

# 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 Cluster Configuration

---

## 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** On the management console, choose **Analytics > GaussDB(DWS)**.
  - Step 3** Choose **Analytics > GaussDB(DWS)** to enter the GaussDB(DWS) management console.
- End

## 5.2 Creating a Cluster

To use Huawei Cloud GaussDB(DWS), create a 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. In a GaussDB(DWS) cluster, the number of DNs on each node varies depending on the cluster flavor. Generally, the number ranges from 1 to 4. A DN stores service data by column or row or in hybrid mode, executes data query tasks, and returns execution results.



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 [Cluster Scale-out](#).

- A network access topology has been designed.  
Plan an appropriate AZ and configure the network to isolate the GaussDB(DWS) cluster from other cloud services.
- Ensure that the number of available nodes meets the following conditions. Otherwise, the cluster cannot be created.
  - At least three nodes are required for creating a cluster. You can view the number of available nodes on the **Clusters** page.
  - The number of nodes in the cluster to be created must be less than or equal to the number of available nodes.

## Creating a Cluster

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

**Step 2** Choose **Clusters** in the navigation pane on the left.

**Step 3** On the **Clusters** page, click **Create Cluster**.

**Step 4** Select **Region**.

**Table 5-1** Region parameters

| Parameter | Description  | Example Value |
|-----------|--|---------------|
| Region    | Select the actual region where the cluster nodes run.<br>For more information about regions, visit <a href="#">Regions and Endpoints</a> . | -             |
| AZ        | Select an AZ associated with the cluster region.   | -             |

**Step 5** Configure **Resource**, **CPU Architecture**, and **Node Flavor**.

**Table 5-2** Node configuration parameters

| Parameter           | Description   | Example Value |
|---------------------|---|---------------|
| Cluster Type        | GaussDB(DWS) cluster types include: <ul style="list-style-type: none"><li>• <b>Standard:</b> It is an enterprise-grade data warehouse featuring high performance, high scalability, high availability, and easy O&amp;M. A cluster can have up to 2048 nodes and process 20 PB data. A company can migrate its large-scale data warehouse to the cloud and without performance deterioration.</li></ul> | Standard      |
| CPU Architecture    | The CPU architecture includes: <ul style="list-style-type: none"><li>• <b>x86</b></li></ul> <b>NOTE</b><br>The only difference between the x86 and Kunpeng architectures lies in the underlying architecture, of which the application layer is unaware. The same SQL syntax is used. If x86 servers are sold out when you create a cluster, select the Kunpeng architecture.                           | x86           |
| Node Flavor         | Select the desired node flavor based on service requirements. Each node flavor displays the vCPU, memory, and recommended application scenario.   | dws.m3.xlarge |
| Nodes               | Specify the number of nodes in the cluster.<br><br>The number of nodes ranges from 3 to 256.  | 3             |
| Total Capacity (GB) | Displays the total capacity of a cluster.<br>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 6** Configure cluster parameters.

**Figure 5-1** Cluster parameters

|                        |                                      |   |
|------------------------|--------------------------------------|---|
| Cluster Name           | <input type="text"/>                 | ? |
| Cluster Version        | <input type="text"/>                 |   |
| Default Database       | gaussdb                              |   |
| Administrator Account  | <input type="text" value="dbadmin"/> | ? |
| Administrator Password | <input type="password"/>             |   |
| Confirm Password       | <input type="password"/>             |   |
| Database Port          | <input type="text" value="8000"/>    | ? |


**Table 5-3** Cluster parameters

| Parameter        | Description   | Example Value |
|------------------|---|---------------|
| Cluster Name     | Set the name of the data warehouse cluster.<br>Enter 4 to 64 characters. Only case-insensitive letters, digits, hyphens (-), and underscores (_) are allowed. The value must start with a letter.<br><b>NOTE</b><br>After a cluster is created, its name cannot be changed. | dws-demo      |
| Cluster Version  | Displays the version of the database instance installed in the cluster. The figure is for reference only.   | 8.1.1.202     |
| Default Database | The default database name of the cluster is <b>gaussdb</b> .<br><b>NOTE</b><br>This name cannot be changed.   | gaussdb       |

| Parameter              | Description   | Example Value |
|------------------------|---|---------------|
| Administrator Account  | <p>Set the database administrator name.</p> <p>The administrator username must:</p> <ul style="list-style-type: none"><li>• Consist of lowercase letters, digits, or underscores.</li><li>• Start with a lowercase letter or an underscore.</li><li>• Contain 6 to 64 characters.</li><li>• Cannot be a keyword of the GaussDB(DWS) database. For details about the keywords of the GaussDB(DWS) database, see "SQL Reference &gt; Keyword" in the <i>Data Warehouse Service (DWS) Developer Guide</i>.</li></ul>   | dbadmin       |
| 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"><li>• Contains 8 to 32 characters.</li><li>• Cannot be the username or the username spelled backwards.</li><li>• Must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters (~!`? ,.;-_'"(){}[]/ &lt;&gt;@#% ^&amp;*+ \=)</li><li>• Passes the weak password check.</li></ul> <p><b>NOTE</b><br/>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>  | 8000          |

**Step 7** Configure network parameters.

**Table 5-4** 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 <b>View VPC</b>. 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 &gt; Creating a VPC" in the <i>Virtual Private Cloud User Guide</i>.</p> <p>After selecting a VPC from the drop-down list, click <b>View VPC</b> 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 <b>VPC</b> 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>   | subnet-dws    |

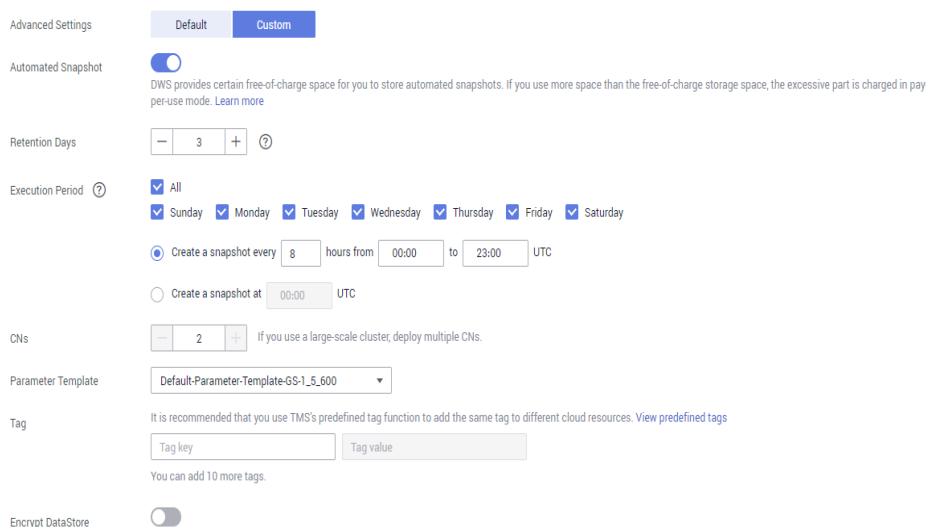
| Parameter      | Description  | Example Value      |
|----------------|--|--------------------|
| Security Group | <p>Specify a VPC security group.</p> <p>A security group restricts access rules to enhance security when GaussDB(DWS) and other services access each other.</p> <ul style="list-style-type: none"> <li> <b>Automatic creation</b><br/>                     If <b>Automatic creation</b> is selected, the system automatically creates a default security group. This option is selected by default.<br/><br/>                     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.<br/><br/>                     The format of the default security group name is <code>dws-<i>Cluster_name</i>-<i>Cluster_database_port</i></code>, for example, <b>dws-dws-demo-8000</b>.                 </li> </ul> <p><b>NOTE</b><br/>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"> <li> <b>Manual creation</b><br/>                     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 <b>Security Group</b> drop-down list to refresh the page, and select the new security group.<br/><br/>                     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 data warehouse cluster. The following is an example of an inbound rule..                 </li> </ul> <ul style="list-style-type: none"> <li>- <b>Protocol: TCP</b></li> <li>- <b>Port: 8000.</b> Use the database port number when you create the cluster for receiving GaussDB(DWS) client connections.</li> </ul> | Automatic creation |

| Parameter             | Description  | Example Value        |
|-----------------------|--|----------------------|
|                       | <ul style="list-style-type: none"> <li>- <b>Source:</b> Select <b>IP address</b> and use the host IP address of the client host, for example, <b>192.168.0.10/32</b>.</li> </ul> <p>The security group of a cluster cannot be changed but can be modified. For details, see .</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"> <li>• <b>Do not use:</b> The EIP is not required.</li> <li>• <b>Automatically assign:</b> Specify the EIP bandwidth, and an EIP with dedicated bandwidth will be bound to the cluster. The EIP can be used to access the cluster over the Internet. The name of an automatically assigned EIP starts with the cluster name.</li> <li>• <b>Specify:</b> A specified EIP is bound to the cluster. If no available EIPs are displayed in the drop-down list, click <b>Create EIP</b> to go to the <b>Elastic IP</b> page and create an EIP that satisfies your needs. You can set the bandwidth as needed.</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>• If you use the EIP binding function for the first time in each project of each region, the system prompts you to create the <b>DWSAccessVPC</b> 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.</li> <li>• Only Huawei Cloud accounts or users with <b>Security Administrator</b> permissions can create agencies by default. IAM users under an account do not have the permission for creating agencies by default. Contact a user with the permission and complete the authorization on the current page.</li> <li>• <b>Do not use</b> indicates disabling access to the cluster over the public network. 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. For details, see <a href="#">Creating a Public Network Domain Name</a>.</li> </ul> | Automatically assign |


| Parameter | Description   | Example Value |
|-----------|---|---------------|
| Bandwidth | When <b>EIP</b> is set to <b>Automatically assign</b> , you need to specify the bandwidth of the EIP, which ranges from 1 Mbit/s to 100 Mbit/s. | 50 Mbit/s     |

**Step 8** Configure advanced settings. Select **Default** to keep the default values of the advanced parameters. You can also select **Custom** to modify the values.

**Figure 5-2** Custom advanced parameters



- **Automated Snapshot**

 indicates that the policy is enabled. It is enabled by default. When the automated snapshot policy is enabled, the system automatically creates snapshots based on the preset time and period. Configure the following parameters as required.

**Table 5-5** Snapshot policy parameters

| Parameter        | Description  |
|------------------|--|
| Retention Days   | Retention days of the snapshots that are automatically created. The value ranges from 1 to 31 days.<br><b>NOTE</b><br>Snapshots that are automatically created cannot be deleted manually. The system automatically deletes these snapshots when their retention duration exceeds the threshold. |
| Execution Period | Interval for creating automated snapshots. You can specify the weekly and daily frequency.   |

- **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 2 to the number of cluster nodes minus 1. The maximum value is **20** and the default value is **3**. In a large-scale cluster, you are advised to deploy multiple CNs.

- **Parameter Template**

A parameter template is a set of parameters for data warehouses. You need to select a parameter template from the drop-down list of **Parameter Template** and associate it with the cluster during cluster creation. You can select the default parameter template or a customized parameter template. By default, the cluster is associated with the default database parameter template.

For details about parameter templates, see [Managing Parameter Templates](#).

- **Tag**

A tag is a key-value pair used to identify a cluster. For details about the keys and values, see [Table 5-6](#). By default, no tag is added to the cluster.

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

**Table 5-6** Tag parameters

| Parameter | Description  | Example Value |
|-----------|--|---------------|
| Tag key   | <p>You can perform the following operations:</p> <ul style="list-style-type: none"><li>– Select a predefined tag key or an existing resource tag key from the drop-down list of the text box.</li></ul> <p><b>NOTE</b><br/>To add a predefined tag, you need to create one on TMS and select it from the drop-down list of <b>Tag key</b>. You can click <b>View predefined tags</b> to enter the <b>Predefined Tags</b> page of TMS. Then, click <b>Create Tag</b> to create a predefined tag. For more information, see <b>Management &gt; Predefined Tags &gt; Creating Predefined Tags</b> in the <i>Tag Management Service User Guide</i>.</p> <ul style="list-style-type: none"><li>– Enter a tag key in the text box. The tag key can contain a maximum of 36 characters and cannot be an empty string. Only digits, letters, underscores (_), and hyphens (-) are allowed.</li></ul> <p><b>NOTE</b><br/>A key must be unique in a given cluster.</p> | key01         |

| Parameter | Description  | Example Value |
|-----------|--|---------------|
| Tag value | <p>You can perform the following operations:</p> <ul style="list-style-type: none"> <li>Select a predefined tag value or resource tag value from the drop-down list of the text box.</li> <li>Enter a tag value in the text box. The tag key can contain a maximum of 43 characters and cannot be an empty string. Only digits, letters, underscores (_), periods (.), and hyphens (-) are allowed.</li> </ul> | value01       |

- **Encrypt DataStore**



indicates that database encryption is disabled. This is the default setting.



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

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 a **DWSAccessKMS** agency so that GaussDB(DWS) can 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.

Only Huawei Cloud accounts or users with **Security Administrator** permissions can create agencies by default. IAM users under an account do not have the permission for creating agencies by default. Contact a user with the 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 9** Click **Create Now**. The **Confirm** page is displayed.

**Step 10** Click **Submit**.

After the submission is successful, the creation starts. Click **Back to Cluster List** to go back to the **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

# 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 following steps:
  - a. [\(Optional\) Configuring SSL Connection](#)
  - b. [\(Optional\) Downloading the SSL Certificate](#)
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 gsql Client to Connect to a Cluster](#)
    - [Using the Data Studio GUI Client to Connect to a Cluster](#)
  - Use a JDBC or ODBC driver to connect to the cluster.
    - [Using a JDBC Driver to Connect to a Database](#)
    - [Using an ODBC Driver to Connect to a Database](#)

## 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 Connections Page](#)
- [Obtaining the Cluster Access Addresses on the Basic Information Page](#)

## Obtaining the Cluster Connection Address on the Connections Page

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

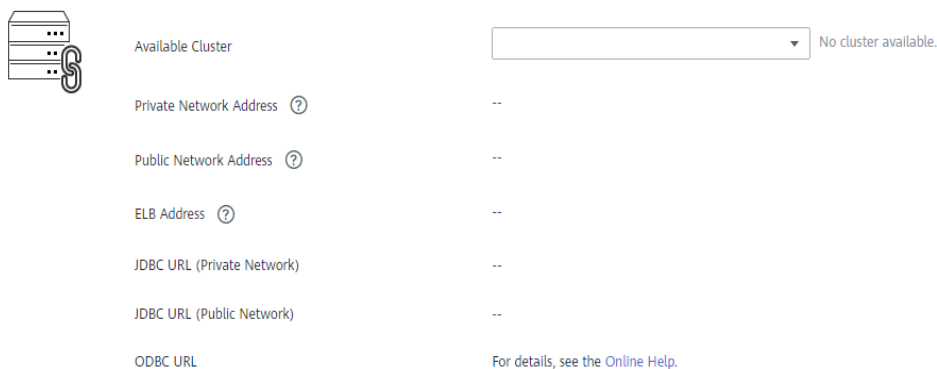
**Step 2** In the navigation pane on the left, click **Connections**.

**Step 3** In the **Data Warehouse Connection Information** area, select an available cluster.

You can only select clusters in the **Available** state.

**Figure 6-1** Data warehouse connection information

Data Warehouse Connection Information



**Step 4** View and obtain the cluster connection information.

- **Private Network Address**
- **Public Network 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 Basic Information Page

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

- Step 2** In the navigation tree on the left, click **Clusters**.
- Step 3** In the cluster list, click the name of the target cluster. The **Basic Information** page is displayed.
- Step 4** In the **Database Attribute** area, view and obtain the cluster's access address information, including the private network address and public network address.

**Table 6-1** Database attribute parameters

| Parameter                   | Description   |
|-----------------------------|---|
| Default Database            | Database name specified when the cluster is created. When you connect to the cluster for the first time, connect to the default database.   |
| Initial Administrator       | 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.   |
| Port                        | Port for accessing the cluster database over the public network or private network. The host port is specified when a cluster is created and used to listen to client connections.  |
| Connection String           | Connection string. You can click <b>View Details</b> to check the data warehouse connection information. Its value can be: <ul style="list-style-type: none"><li>● <b>JDBC URL (Private Network)</b>. In the private network environment, you can use the JDBC URL (private network) to connect to the cluster when developing applications.</li><li>● <b>JDBC URL (Public Network)</b>. In the public network environment, you can use the JDBC URL (public network) to connect to the cluster when developing applications.</li><li>● <b>ODBC URL</b>. In GaussDB(DWS), you can use an ODBC driver to connect to the database. The driver can connect to the database on an ECS or over the Internet.</li></ul> |
| Private Network Domain Name | Name of the domain for accessing the database in the cluster over the private network. The private network domain address is automatically generated when a cluster is created. When you access a data warehouse cluster using a domain name, the domain name resolver provides the load balancing function.<br><br>You can click <b>Modify</b> 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.  |

| Parameter                  | Description  |
|----------------------------|--|
| Private Network IP Address | IP address for accessing the database in the cluster over the private network.<br><b>NOTE</b> <ul style="list-style-type: none"><li>• A private IP address is automatically generated when you create a cluster. The IP address is fixed.</li><li>• The number of private IP addresses equals the number of CNs. You can log in to any node to connect to the cluster.</li><li>• If you access a fixed IP address over the internal network, all the workloads will be processed on a single CN.</li></ul> |
| Public Network Domain Name | Name of the domain for accessing the database in the cluster over the public network.  |
| Public Network IP Address  | IP address for accessing the database in the cluster over the public network.<br><b>NOTE</b> <ul style="list-style-type: none"><li>• If no EIP is assigned during cluster creation and <b>Public Network IP Address</b> is empty, click <b>Bind EIP</b> to bind an EIP to the cluster.</li><li>• If an EIP is bound during cluster creation, click <b>Unbind EIP</b> to unbind the EIP.</li></ul>  |

----End

## 6.3 Using the gsql CLI Client to Connect to a Cluster

### 6.3.1 Downloading the 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:

- **Database connection tool gsql and the script for testing sample data**  
gsql is a command line client running on the Linux operating system. It is used to connect to the database in a data warehouse cluster.

The script for testing sample data is used when you start an example.

### Downloading the Client

- Step 1** Log in to the GaussDB(DWS) console. For details, see [Accessing the GaussDB\(DWS\) Management Console](#).
- Step 2** In the navigation pane on the left, click **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.

- The **Redhat x86\_64** client can be used on the following OSs:
  - RHEL 6.4 to RHEL 7.6
  - CentOS 6.4 to CentOS 7.4
  - EulerOS 2.3
- The **SUSE x86\_64** client can be used on the following OSs:
  - SLES 11.1 to SLES 11.4
  - SLES 12.0 to SLES 12.3
- The **Euler Kunpeng\_64** client can be used on the following OS:
  - EulerOS 2.8

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

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

 **NOTE**

- In the cluster list on the **Clusters** page, click the name of the specified cluster and click the **Basic Information** tab to view the cluster version.

**Table 6-2** lists the files and folders in the downloaded tool package.

**Table 6-2** Files and folders in the downloaded tool package

| File or Folder | Description  |
|----------------|--|
| bin            | This folder contains the executable files of gsql on Linux, including the tools gsql, GDS, gs_dump, gs_dumpall, and gs_restore. For details, see "Server Tool".                                      |
| 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 <b>lib</b> library required for executing the gsql client.  |

| File or Folder | Description  |
|----------------|--|
| sample         | This folder contains the following directories and files: <ul style="list-style-type: none"><li>• <b>setup.sh</b>: script file for configuring the AK/SK before using gsql to import sample data</li><li>• <b>tpcds_load_data_from_obs.sql</b>: script file for importing the TPC-DS sample data using the gsql client</li><li>• <b>query_sql</b> directory: script file for querying the TPC-DS sample data</li></ul> |
| gsql_env.sh    | Script file for configuring environment variables before running the gsql client.  |

----End

### 6.3.2 Preparing an ECS as the gsql Client Host

The gsql command line client provided by GaussDB(DWS) runs on the Linux OS. Before using it to remotely connect to a GaussDB(DWS) cluster, you need to prepare a Linux host for installing and running the gsql client. If you use a public network address to access the cluster, you can install the gsql client on your own Linux host. Ensure that the Linux host has a public network address. For your convenience, you are advised to create a Linux ECS. This section describes how to prepare an ECS. If you already have a qualified ECS, skip this section.

#### Preparing an ECS

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:

There is no special requirement for the image's specifications. 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 to RHEL 7.6
  - CentOS 6.4 to CentOS 7.4
  - EulerOS 2.3
- The **SUSE x86\_64** client can be used on the following OSs:
  - SLES 11.1 to SLES 11.4
  - SLES 12.0 to SLES 12.3
- The **Euler Kunpeng\_64** client can be used on the following OS:



- EulerOS 2.8

- 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/Application:** The value must contain **TCP**, for example, **TCP** and **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 cluster to be connected. For example, set this parameter to **0.0.0.0/0** or the specific connection address of the GaussDB(DWS) cluster.

- The security group rules of the GaussDB(DWS) 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/Application:** The value must contain **TCP**, for example, **TCP** and **All**.
- **Port:** Set this parameter to the database port that provides services in the GaussDB(DWS) cluster, for example, **8000**.
- **Source:** The IP address set here must contain the IP address of the GaussDB(DWS) client host, for example, **192.168.0.10/32**.

### 6.3.3 Using the 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 gsql client that matches the cluster version for you to access the cluster using the cluster's public or private network address.

#### Using the gsql CLI Client to Connect to a Cluster

**Step 1** Prepare a Linux ECS to install and run the gsql client.

For details, see [Preparing an ECS as the gsql Client Host](#).

**Step 2** Download the gsql client by referring to [Downloading the Client](#), and use an SSH file transfer tool (such as WinSCP) to upload the client to a target Linux host.

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 3** Use the SSH tool to remotely log in to 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 4** (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 5** 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 6** Run the following command to configure the GaussDB(DWS) client:

```
source gsql_env.sh
```

If the following information is displayed, the GaussDB(DWS) client is successfully configured:

```
All things done.
```

**Step 7** 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> -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 database 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*.

## (Optional) Importing TPC-DS Sample Data Using gsql

GaussDB(DWS) users can import data from external sources to data warehouse clusters. This section describes how to import sample data from OBS to a data warehouse cluster and perform querying and analysis operations on the sample data. The sample data is generated based on the standard TPC-DS benchmark test.

TPC-DS is the benchmark for testing the performance of decision support. With TPC-DS test data and cases, you can simulate complex scenarios, such as big data set statistics, report generation, online query, and data mining, to better understand functions and performance of database applications.

- Step 1** Use the SSH remote connection tool to log in to the host where the gsql client is installed and go to the gsql directory. The **/opt** directory is used as an example for storing the gsql client.

```
cd /opt
```

- Step 2** Switch to the specified directory and set the AK and SK for importing sample data and the OBS access address.

If the following information is displayed, the settings are successful:

```
setup successfully!
```

### NOTE

<Access\_Key\_Id> and <Secret\_Access\_Key>: indicate the AK and SK, respectively. For details about how to obtain the AK and SK, see "Data Import > Concurrently Importing Data from OBS > Creating Access Keys (AK and SK)" in the *Data Warehouse Service (DWS) Developer Guide*. Then, replace the parameters in the statements with the obtained values.

- Step 3** Go back to previous directory and run the gsql environment variables.

```
cd ..  
source gsql_env.sh  
cd bin
```

- Step 4** Import the sample data to the data warehouse.

Command format:

```
gsql -d <Database name> -h <Public network address of the cluster> -U <Administrator> -p <Data warehouse port number> -f <Path for storing the sample data script> -r
```

Sample command:

```
gsql -d gaussdb -h 10.168.0.74 -U dbadmin -p 8000 -f /opt/sample/tpcds_load_data_from_obs.sql -r
```

**NOTE**

In the preceding command, sample data script **tpcds\_load\_data\_from\_obs.sql** is stored in the sample directory (for example, **/opt/sample/**) of the GaussDB(DWS) client.

After you enter the database administrator password and successfully connect to the database in the cluster, the system will automatically create a foreign table to associate the sample data outside the cluster. Then, the system creates a target table for saving the sample data and imports the data to the target table using the foreign table.

The time required for importing a large dataset depends on the current GaussDB(DWS) cluster specifications. Generally, the import takes about 10 to 20 minutes. If information similar to the following is displayed, the import is successful.

```
Time:1845600.524 ms
```

**Step 5** In the Linux command window, run the following commands to switch to a specific directory and query the sample data:

```
cd /opt/sample/query_sql/  
/bin/bash tpcds100x.sh
```

**Step 6** Enter the cluster's public network IP address, access port, database name, user who accesses the database, and password of the user as prompted.

- The default database name is **gaussdb**.
- Use the database administrator username and password configured during cluster creation as the username and password for accessing the database.

After the query is complete, a directory for storing the query result, such as **query\_output\_20170914\_072341**, will be generated in the current query directory, for example, **sample/query\_sql/**.

----End

## 6.3.4 Establishing Secure TCP/IP Connections in SSL Mode

If the client or JDBC/ODBC driver needs to use SSL connection, you must 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 resides and specify the path of the certificate on the client.

**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 database customer service.

For more information about SSL certificates, see [\(Optional\) Downloading the SSL Certificate](#). The following parts are included in this section:

- [Configuring Digital Certificate Parameters Related to SSL Authentication on the gsql Client](#)

- [SSL Authentication Modes and Client Parameters](#)

## 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 and click **Connections** to download the SSL certificate.

For more information about SSL certificates, see [\(Optional\) Downloading the SSL Certificate](#).

- Step 2** 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 3** 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 4** 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. Different authentication modes require different client environment variables. 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 5** 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 6-3](#) shows the differences between these two modes. You are advised to use bidirectional authentication for security purposes.

**Table 6-3** 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"> <li>PGSSLCERT</li> <li>PGSSLKEY</li> <li>PGSSLROTCERT</li> <li>PGSSLMODE</li> </ul> | This authentication mode is applicable to scenarios that require high data security. When using this mode, you are advised to set the <b>PGSSLMODE</b> client variable to <b>verify-ca</b> for network data security purposes.                 |
| 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"> <li>PGSSLROTCERT</li> <li>PGSSLMODE</li> </ul>                                      | 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 <b>PGSSLMODE</b> variable to <b>verify-ca</b> on the client. |

Configure environment variables related to SSL authentication on the client. For details, see [Table 6-4](#).

 NOTE

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

Table 6-4 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:<br><code>export PGSSLCERT='/home/dbadmin/dws_ssl/sslcert/client.crt'</code><br>(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:<br><code>export PGSSLKEY='/home/dbadmin/dws_ssl/sslcert/client.key'</code><br>(No default value)   |
| PGSSLMODE            | Specifies whether to negotiate with the server about SSL connection and specifies the priority of the SSL connection.   | Values and meanings: <ul style="list-style-type: none"> <li>• <b>disable</b>: only tries to establish a non-SSL connection.</li> <li>• <b>allow</b>: tries to establish a non-SSL connection first, and then an SSL connection if the first attempt fails.</li> <li>• <b>prefer</b>: tries to establish an SSL connection first, and then a non-SSL connection if the first attempt fails.</li> <li>• <b>require</b>: 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 <b>verify-ca</b>.</li> <li>• <b>verify-ca</b>: tries to establish an SSL connection and check whether the server certificate is issued by a trusted CA.</li> <li>• <b>verify-full</b>: GaussDB(DWS) does not support this mode.</li> </ul> Default value: <b>prefer</b> |

| Environment Variable | Description   | Value Range   |
|----------------------|---|---|
| PGSSLROOTCERT        | Specifies the root certificate file for issuing client certificates. The root certificate is used to verify the server certificate.                                   | The absolute path of the files must be specified, for example:<br><code>export PGSSLROOTCERT='/home/dbadmin/dws_ssl/sslcert/certca.pem'</code><br>Default value: null |
| 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. | The absolute path of the files must be specified, for example:<br><code>export PGSSLCRL='/home/dbadmin/dws_ssl/sslcert/sslcrfile.crt'</code><br>Default value: null   |

### 6.3.5 (Optional) Configuring SSL Connection

GaussDB(DWS) supports connections in SSL authentication mode so that data transmitted between the GaussDB(DWS) client and the database can be encrypted. 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. For security purposes, you are advised to enable SSL connection. If you want to use SSL connection, enable **Require SSL Connection** for the cluster.

On the **Security Settings** page of the cluster, you can enable or disable **Require SSL Connection**.

#### NOTE

- After you have changed the security setting parameters and the settings 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 value of **Task Information** cannot be **Creating snapshot**, **Scaling out**, **Configuring**, or **Restarting**.

The following parts are included in this section:

- [Configuring SSL Connection](#)
- [Combinations of SSL Connection Parameters on the Client and Server](#)

### Configuring SSL Connection

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



**Step 2** In the navigation pane on the left, click **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 set to **Synchronized**, which indicates that the latest database result is displayed.

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



indicates that the server requires SSL connection.



indicates that no SSL connection is required (default).

 **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

## 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 described 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** for clusters whose version is 1.3.1 or later, and you cannot set this parameter on the GaussDB(DWS) management console.
- For clusters whose version is earlier than 1.3.1, the default value is **on**. You can set this parameter in the **SSL Connection** area on the cluster's **Security Settings** page of 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** for clusters whose version is 1.3.1 or later. 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.

- For clusters whose version is earlier than 1.3.1, the default value is **off**, and you cannot set this parameter 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 6-5** 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.   |
|              | 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.   |

| ssl (Server) | sslmode (Client) | require_ssl (Server) | Result  |
|--------------|------------------|----------------------|---|
|              | 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. |

### 6.3.6 (Optional) Downloading the SSL Certificate

GaussDB(DWS) supports the standard SSL (TLS 1.2). 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**.

By default, the SSL function is enabled for a data warehouse cluster (the server) to allow SSL and non-SSL connections from the client. In addition, the certificate, private key, and root certificate of the server have been configured by default.

If the client or JDBC/ODBC driver needs to use SSL connection, you must 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 resides and specify the path of the certificate on the client.

#### 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 database customer service.

This section describes how to download an SSL certificate.

## Downloading the SSL Certificate File

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane on the left, click **Connections**.
- Step 3** In the **Driver** area, click **download an SSL certificate**.

----End

## 6.4 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

- An EIP has been bound to the GaussDB(DWS) cluster.
- You have obtained the database administrator username and password for logging in to 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 to which the data warehouse cluster belongs and added a 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 SDK, and install it by following the installation guidance.

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

**Step 3** Click **Connections**.

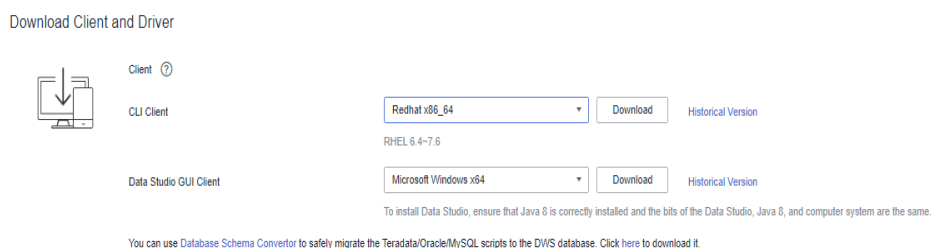
**Step 4** On the **Download Client and Driver** page, download **Data Studio GUI Client**.

- Select **Windows x86** or **Windows x64** based on the OS type and click **Download** to download a Data Studio version that matches the current cluster.

If clusters of different versions are available, you will download the Data Studio 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.

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

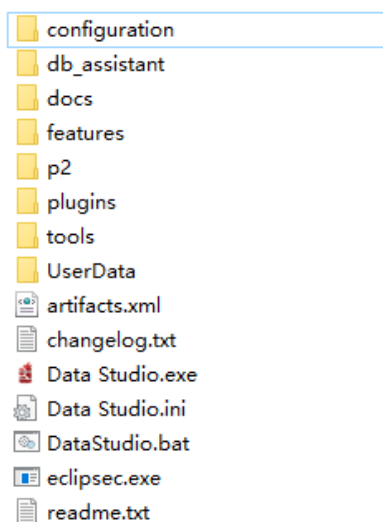
**Figure 6-2** Downloading clients



**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 6-3](#).

**Figure 6-3** Starting the client

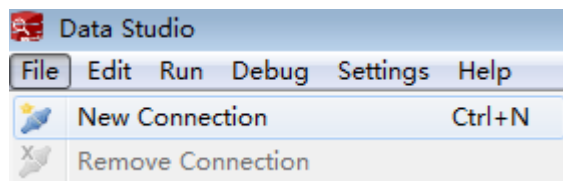


**NOTE**

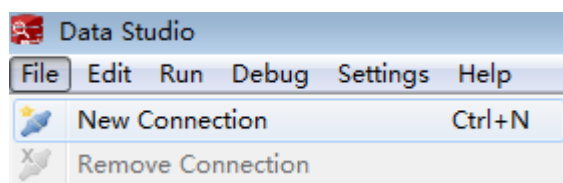
If your computer blocks the running of the application, you can unlock the **Data Studio.exe** file to start the application.

**Step 7** Choose **File > New Connection** from the main menu. See [Figure 6-4](#).

**Figure 6-4** New connection



**Figure 6-5** New connection



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

**Table 6-6** Connection parameters

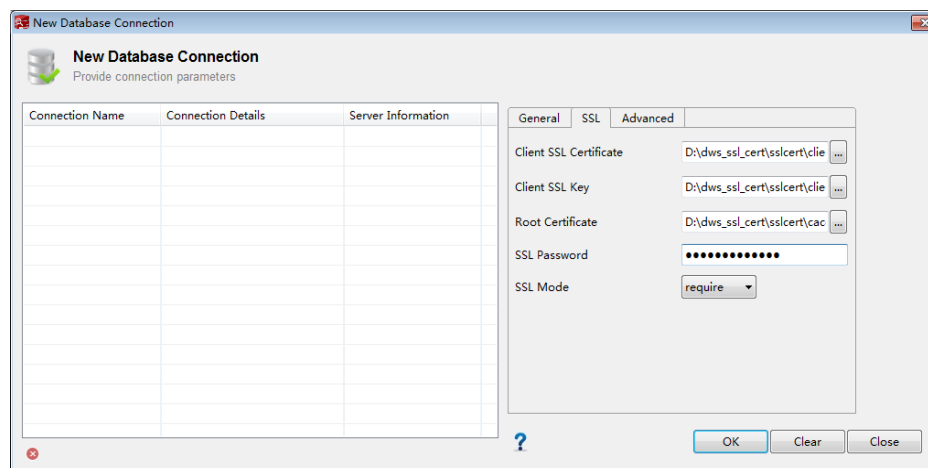
| Field           | Description   | Example Value |
|-----------------|---|---------------|
| Database Type   | Select <b>GaussDB A</b>   | GaussDB A     |
| Connection Name | Name of the connection  | dws-demo      |
| Host            | IP address (IPv4) or domain name of the cluster to be connected   | -             |
| Port Number     | Database port   | 8000          |
| Database Name   | Database name   | gaussdb       |
| Username        | 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"> <li>● <b>Current Session Only</b>: The password is saved only in the current session.</li> <li>● <b>Do Not Save</b>: The password is not saved.</li> </ul> | -             |
| Enable SSL      | If <b>Enable SSL</b> is selected, the client can use SSL to encrypt connections. The SSL connection 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 **(Optional) Downloading the SSL Certificate**. Click the **SSL** tab and configure the following parameters:

**Table 6-7** Configuring SSL parameters

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

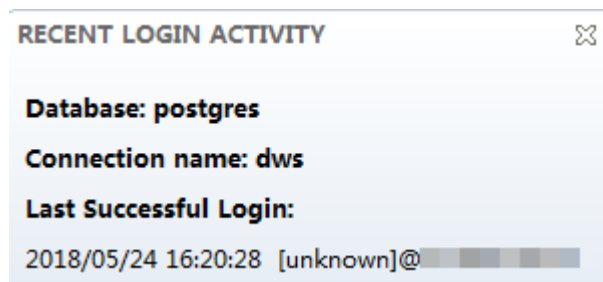
**Figure 6-6** Configuring SSL parameters



**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 6-7** 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.

----End

## 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 and ODBC drivers: PostgreSQL JDBC 9.3-1103 or later; 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 pane on the left, click **Connections**.

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

- **JDBC Driver**

Method 1:

Select **DWS JDBC Driver** and click **Download** to download the JDBC driver matching the current cluster version. 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.

The ODBC driver is applicable to the following operating systems only:

- Windows Server 2008 or Windows 7 or later
- x86 servers: RHEL 6.4 to RHEL 7.6
- x86 servers: CentOS 6.4 to CentOS 7.4
- x86 servers: SUSE 11.1 to SUSE 11.4; SUSE 12.0 to SUSE 12.3
- Kunpeng servers: EulerOS 2.8

 **NOTE**

Windows drivers can only be 32-bit and can be used in 32-bit or 64-bit operating systems. However, the applications must be 32-bit.

----End

### 6.5.3 Using a JDBC Driver to Connect to a Database

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 Huawei Cloud 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.

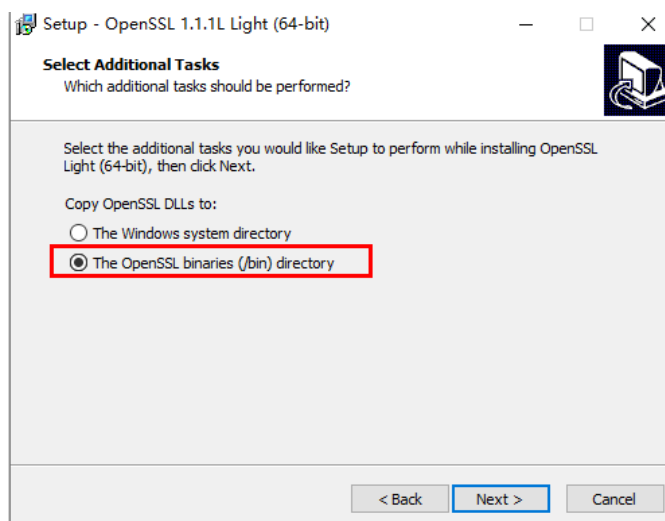
## Prerequisites

- 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 [\(Optional\) Downloading the SSL Certificate](#).

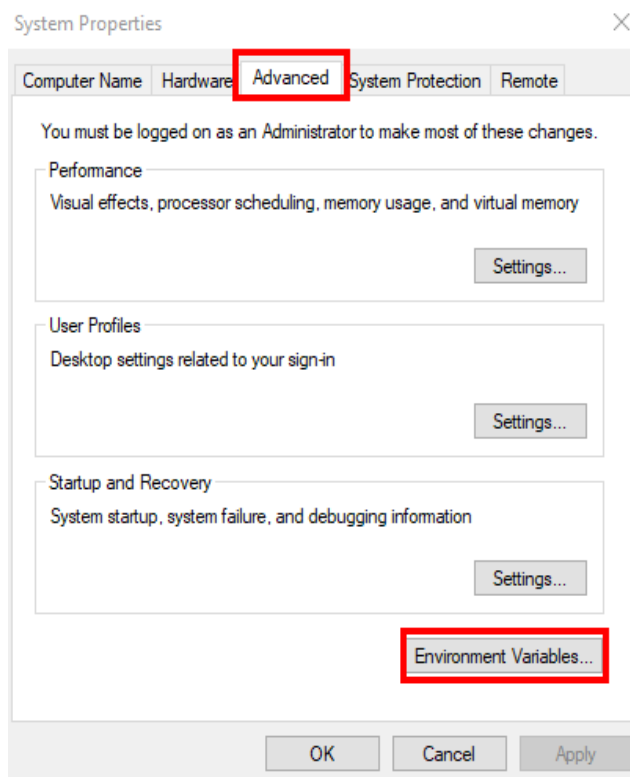
## Using a JDBC Driver to Connect to a Database

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.

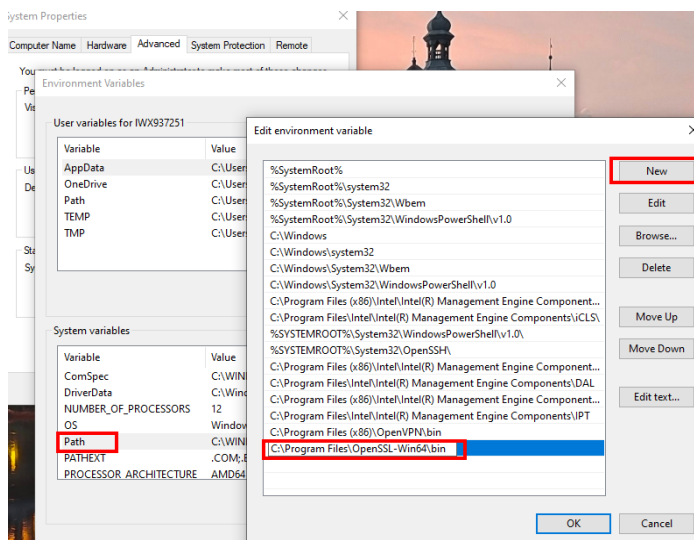
- Step 1** Determine whether you want to use the SSL mode to connect to the GaussDB(DWS) cluster.
- If yes, enable SSL connection by referring to [\(Optional\) Configuring SSL Connection](#). SSL connection is enabled by default. Then go to [Step 2](#).
  - If no, disable SSL connection by referring to [\(Optional\) 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 toolkit for Windows at <https://slproweb.com/products/Win32OpenSSL.html>. OpenSSL 3.0.0 is currently not supported. Download Win64 OpenSSL v1.1.1L Light instead.
  2. Double-click the installation package **Win64OpenSSL\_Light-1\_1\_1L.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.



5. 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.

6. 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**.

7. 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.

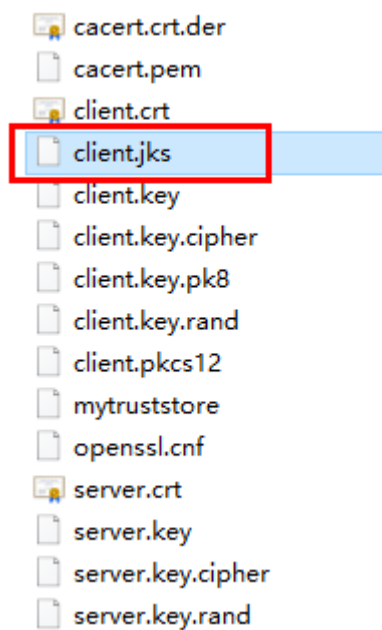
8. Import the private key to the keystore.  

```
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@MppDB -destkeystore client.jks -srckeystore client.pkcs12 -srcstorepass key123 -srcstoretype PKCS12 -alias 1  
Importing keystore client.pkcs12 to client.jks...
```

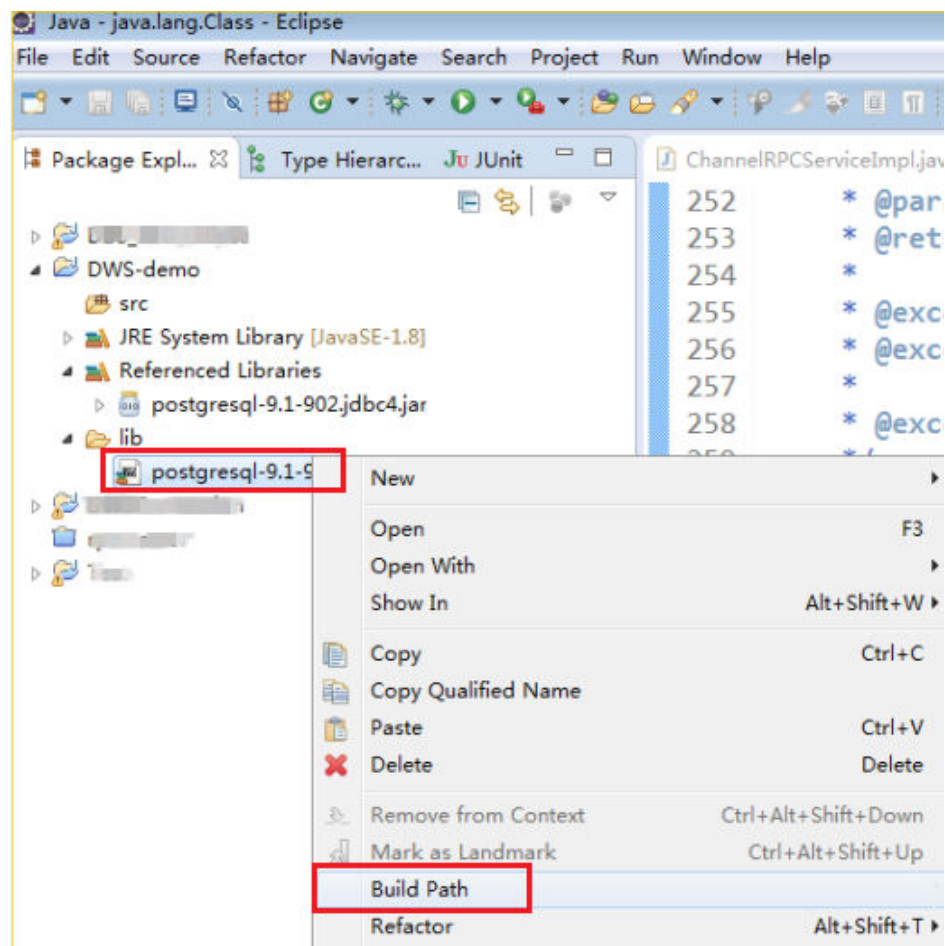


**Step 4** Decompress the downloaded JDBC driver to obtain **gsjdbc4.jar**.

**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 6-8** 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 <a href="#">Obtaining the Cluster Connection Address</a>.</p> <p>The URL format is as follows:</p> <ul style="list-style-type: none"> <li>• jdbc:postgresql:database</li> <li>• jdbc:postgresql://host/database</li> <li>• jdbc:postgresql://host:port/database</li> <li>• jdbc:postgresql://host:port[,host:port][...]/database</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>• If <b>gsjdbc200.jar</b> is used, change <b>jdbc:postgresql</b> to <b>jdbc:gaussdb</b>. <ul style="list-style-type: none"> <li>- <b>database</b> indicates the name of the database to be connected.</li> <li>- <b>host</b> indicates the name or IP address of the database server. If the connected GaussDB(DWS) server and cluster are on the same network, use a private IP address. Otherwise, use a public IP address.</li> <li>- <b>port</b> indicates the port number of the database server. By default, the database running on port 8000 of the local host is connected.</li> <li>- 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: <b>jdbc:postgresql://10.10.0.13:8000,10.10.0.14:8000/database</b></li> </ul> </li> <li>• If JDBC is used to connect to a cluster, only JDBC connection parameters can be configured in a cluster address. Variables cannot be added.</li> </ul> |

| Parameter | Description  |
|-----------|--|
| info      | <p>Specifies database connection properties. Common properties include the following:</p> <ul style="list-style-type: none"> <li>● <b>user</b>: a string type. It indicates the database user who creates the connection task.</li> <li>● <b>password</b>: a string type. It indicates the password of the database user.</li> <li>● <b>ssl</b>: a boolean type. It indicates whether to use the SSL connection.</li> <li>● <b>loggerLevel</b>: string type. It indicates the volume of log data sent to the LogStream or LogWriter specified in the DriverManager. Currently, <b>OFF</b>, <b>DEBUG</b>, and <b>TRACE</b> are supported. <b>DEBUG</b> indicates that only logs of <b>DEBUG</b> or a higher level are printed, generating little log information. <b>TRACE</b> indicates that logs of the <b>DEBUG</b> and <b>TRACE</b> levels are displayed, generating detailed log information. The default value is <b>OFF</b>, indicating that no logs will be displayed.</li> <li>● <b>prepareThreshold</b>: integer type. It indicates the number of <b>PreparedStatement</b> executions required before requests are converted to prepared statements in servers. The default value is 5.</li> <li>● <b>batchMode</b>: boolean type. It indicates whether to connect the database in batch mode.</li> <li>● <b>fetchsize</b>: integer type. It indicates the default fetch size for statements in the created connection.</li> <li>● <b>ApplicationName</b>: string type. It indicates an application name. The default value is <b>PostgreSQL JDBC Driver</b>.</li> <li>● <b>allowReadOnly</b>: boolean type. It indicates whether to enable the read-only mode for connection. The default value is <b>false</b>. If the value is not changed to <b>true</b>, the execution of <b>connection.setReadOnly</b> does not take effect.</li> <li>● <b>blobMode</b>: string type. It is used to set the <b>setBinaryStream</b> method to assign values to different data types. The value <b>on</b> indicates that values are assigned to the BLOB data type and <b>off</b> indicates that values are assigned to the BYTEA data type. The default value is <b>on</b>.</li> <li>● <b>connectionExtraInfo</b>: boolean type. This parameter indicates whether the JDBC driver reports the driver deployment path and process owner to the database.</li> </ul> <p><b>NOTE</b><br/>The value can be <b>true</b> or <b>false</b>. The default value is <b>true</b>. If <b>connectionExtraInfo</b> is set to <b>true</b>, the JDBC driver reports the driver deployment path and process owner to the database and displays the information in the <b>connection_info</b> parameter. In this case, you can query the information from <b>PG_STAT_ACTIVITY</b> or <b>PGXC_STAT_ACTIVITY</b>.</p> |
| user      | Specifies the database user.   |

| Parameter | Description                                  |
|-----------|--|
| 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" + '/'
        + "gaussdb";
    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(Exception e)
    {
        e.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 "Tutorial: Development Using JDBC or ODBC" in the *Data Warehouse Service (DWS) 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 Using an ODBC Driver to Connect to a Database

GaussDB(DWS) allows you to use an ODBC driver to 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\_odbc\_driver\_for\_linux.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 [\(Optional\) Downloading the 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
```

- Modify the configuration.

```
cd unixODBC-2.3.0  
vi configure
```

Change the value of **LIB\_VERSION** to the following. Save the change and exit.

```
LIB_VERSION="1:0:0"
```

- Compile the code file and install the driver.

```
./configure --enable-gui=no  
make  
make install
```

**Step 4** Replace the driver file.

- Decompress **dws\_odbc\_driver\_for\_linux.zip**.

```
unzip dws_odbc_driver_for_linux.zip
```

- Copy all files in the **lib** directory to **/usr/local/lib**. If there are files with the same name, overwrite them.

- 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 <b>DriverName</b> in <b>odbcinst.ini</b> . | Driver=DWS                |
| Servername | Server IP address.   | 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"><li>● <b>disable</b>: only tries to establish a non-SSL connection.</li><li>● <b>allow</b>: tries establishing a non-SSL connection first, and then an SSL connection if the attempt fails.</li><li>● <b>prefer</b>: tries establishing an SSL connection first, and then a non-SSL connection if the attempt fails.</li><li>● <b>require</b>: 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 <b>verify-ca</b>.</li><li>● <b>verify-ca</b>: tries establishing an SSL connection and checks whether the server certificate is issued by a trusted CA.</li><li>● <b>verify-full</b>: not supported by GaussDB(DWS)</li></ul> <p><b>NOTE</b><br/>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 **Connection Management**. 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.

Currently, because GaussDB(DWS) only provides a 32-bit ODBC driver, it only supports 32-bit application development. Use the 32-bit Driver Manager when you configure the data source. (Assume the Windows system drive is drive C. If another disk drive is used, modify the path accordingly.)

- In a 64-bit Windows operating system, open **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. **C:\Windows\System32\** stores the environment consistent with the current operating system. For technical details, see the Windows technical documents.

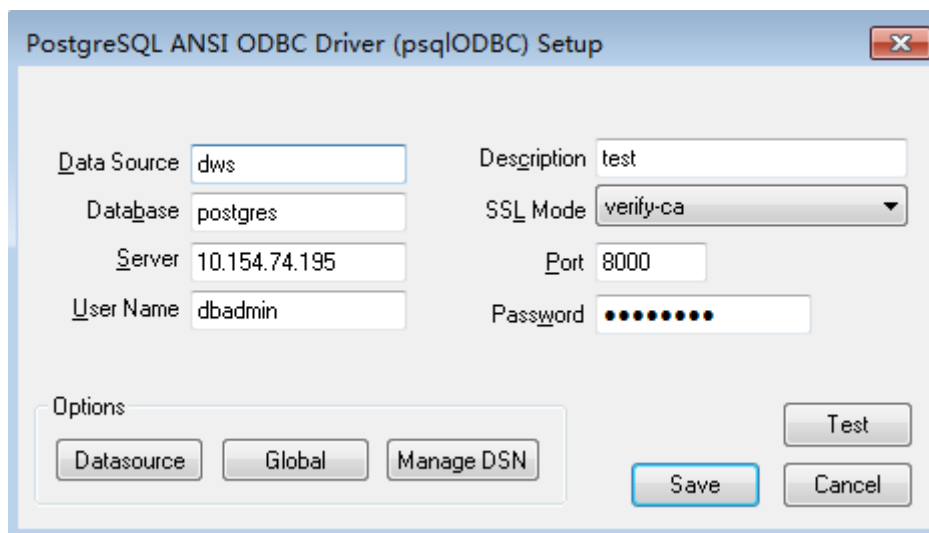
- In a 32-bit Windows operating system, open **C:\Windows\System32\odbcad32.exe**.

You can also open Driver Manager by choosing **Control Panel > System and Security > Administrative Tools > 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 6-9** 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 **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 <sqlxext.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

## 6.5.5 Connecting to a Cluster Using IAM Authentication

### 6.5.5.1 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.

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 process of accessing a database is as follows:

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

### 6.5.5.2 Granting an IAM Account the DWS Database Access Permission

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. Using the **DWS Database Access** permission helps to control access to databases.

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.

## Procedure

**Step 1** Log in to the Huawei Cloud management console and choose **Service List > 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

### 6.5.5.3 Creating an IAM User Credential

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

#### Creating an AK/SK Pair

Log in to the management console, move your cursor over your account in the upper right corner, and choose **My Credential**. Click the **Access Keys** tab. On the **Access Keys** tab page, you can view the existing AKs or click **Add Access Key** to create an AK/SK pair.

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, 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.

### 6.5.5.4 Configuring the JDBC Connection to Connect to a Cluster 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 whose version is 1.3.1 or later and their corresponding JDBC driver provided by GaussDB(DWS) can access the databases in IAM authentication mode. Download the JDBC driver. For details, see [Downloading the JDBC or ODBC Driver](#).

## 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:ae-ad-1/gaussdb? AccessKeyID=XXXXXXXXXXXXXXXXXXXX&amp;SecretAccessKey=XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXX&amp;DbUser=user_test&amp;AutoCreate=true</pre> <p><b>JDBC URL parameters:</b></p> <ul style="list-style-type: none"> <li>• <b>jdbc:dws:iam</b> is a prefix in the URL format.</li> <li>• <b>dws-IAM-demo</b> indicates the name of the cluster containing the database.</li> <li>• <b>ae-ad-1</b> indicates the region where the cluster resides. For details about GaussDB(DWS) regions, visit <a href="#">Regions and Endpoints</a>.</li> <li>• <b>gaussdb</b> indicates the name of the database to which you want to connect.</li> <li>• <b>AccessKeyID</b> and <b>SecretAccessKey</b> are the access key ID and secret access key corresponding to the IAM user specified by <b>DbUser</b>.</li> <li>• Set <b>DbUser</b> to the IAM username. Note that the current version does not support hyphens (-) in the IAM username. <ul style="list-style-type: none"> <li>– If the user specified by <b>DbUser</b> exists in the database, the temporary user credential has the same permissions as the existing user.</li> <li>– If the user specified by <b>DbUser</b> does not exist in the database and the value of <b>AutoCreate</b> is <b>true</b>, a new user named by the value of <b>DbUser</b> is automatically created. The created user is a common database user by default.</li> </ul> </li> <li>• Parameter <b>AutoCreate</b> is optional. The default value is <b>false</b>. This parameter indicates whether to automatically create a database user named by the value of <b>DbUser</b> in the database. <ul style="list-style-type: none"> <li>– The value <b>true</b> indicates that a user is automatically created. If the user already exists, the user will not be created again.</li> <li>– The value <b>false</b> indicates that a user is not created. If the username specified by <b>DbUser</b> does not exist in the database, an error is returned.</li> </ul> </li> </ul> |

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

## Example

```
//The following uses gsjdbc4.jar as an example.  
//The following 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)  
{  
    //Driver class  
    String driver = "org.postgresql.Driver";  
    // Database connection descriptor.  
    String sourceURL = "jdbc:dws:iam://" + clustername + ":" + regionname + "/gaussdb?" +  
"AccessKeyID=" + AK + "&SecretAccessKey=" + SK + "&DbUser=" + username + "&autoCreate=true";  
  
    Connection conn = null;  
  
    try  
    {  
        //Load the driver.  
        Class.forName(driver);  
    }  
    catch( Exception e )  
    {  
        return null;  
    }  
  
    try  
    {  
        //Create a connection.  
        conn = DriverManager.getConnection(sourceURL);  
        System.out.println("Connection succeed!");  
    }  
    catch(Exception e)  
    {
```

```
        return null;
    }

    return conn;
};
```

## 6.6 Using the Python Library `psycopg2` 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   | -   |
|             |                                  | close()  | Y   | -   |
|             | Two-phase commit support methods | <i>xid(format_id, gtrid, bqual)</i>  | Y   | -   |
|             |                                  | tpc_begin( <i>xid</i> )  | Y   | -   |
|             |                                  | tpc_prepare()  | N   | The kernel does not support explicit <b>PREPARE TRANSACTION</b> . |
|             |                                  | tpc_commit( <i>[xid]</i> )   | Y   | -   |
|             |                                  | tpc_rollback( <i>[xid]</i> )   | Y   | -   |
|             |                                  | tpc_recover()  | Y   | -   |
| closed      | Y                                | -  |     |   |



| Class Name | Usage                                       | Function/Member Variable  | Yes | Remarks   |
|------------|---|---|-----|---|
|            |   | cancel()  | Y   | -   |
|            |   | reset()   | N   | <b>DISCARD ALL</b> 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 <b>default_transaction_read_only</b> in a session. |
|            |   | autocommit  | Y   | -   |
|            |   | isolation_level   | Y   | -   |
|            |   | readonly  | N   | The database does not support the setting of <b>default_transaction_read_only</b> in a session. |
|            |   | deferrable  | Y   | -   |
|            |   | set_isolation_level( <i>level</i> )   | Y   | -   |
|            |   | encoding  | Y   | -   |
|            |   | set_client_encoding(enc)  | Y   | -   |
|            |   | notices   | N   | The database does not support <b>listen/notify</b> .  |
|            |   | notifies  | Y   | -   |
|            |   | cursor_factory  | Y   | -   |
|            |   | info  | Y   | -   |
|            |   | status  | Y   | -   |

| Class Name | Usage  | Function/Member Variable        | Yes | Remarks   |
|------------|--|---------------------------------|-----|---|
|            |  | lobject                         | 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   | -   |
|            |  | get_dsn_parameters()            | Y   | -   |
| cursor     | basic  | description                     | Y   | -   |
|            |  | close()                         | Y   | -   |
|            |  | closed                          | Y   | -   |
|            |  | connection                      | Y   | -   |
|            |  | name                            | Y   | -   |
|            |  | scrollable                      | N   | The database does not support <b>SCROLL CURSOR</b> .                              |

| Class Name | Usage  | Function/Member Variable                          | Yes | Remarks  |
|------------|--|---|-----|--|
|            |  | withhold  | N   | The <b>withhold cursor</b> needs to be closed before the commit operation. |
|            | Command 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 <b>SCROLL CURSOR</b> .                       |
|            |  | arraysize   | Y   | -  |
|            |  | itersize  | Y   | -  |
|            |  | rowcount  | Y   | -  |
|            |  | 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   | -   |  |

| Class Name | Usage                                   | Function/Member Variable  | Yes | Remarks |
|------------|---|---|-----|---------|
|            | COPY-related methods                    | <code>copy_from(file, table, sep=' t', null='   N', size=8192, columns=None)</code> | Y   | -       |
|            |   | <code>copy_to(file, table, sep=' t', null='   N', columns=None)</code>              | Y   | -       |
|            |   | <code>copy_expert(sql, file, size=8192)</code>                                      | Y   | -       |
|            | Interoperation with other C API modules | <code>pgresult_ptr</code>   | 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

## CN Retry Is Not Supported If psycopg2 Is Used to Connect 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   | -       |

| PyGreSQL   | Yes | Remarks |
|--|-----|---------|
| 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   | -       |

| PyGreSQL                           |   | Yes | Remarks |
|------------------------------------|---|-----|---------|
|                                    | 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   | -       |
|                                    | 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   | -       |

| PyGreSQL             |  | Yes | Remarks  |
|----------------------|--|-----|--|
|                      | 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 <b>listen/notify</b> .                     |
|                      | inserttable – insert a list into a table                         | Y   | Use double quotation marks (""") to quote \n in the <b>copy</b> 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   | -  |
| The DB wrapper class | Initialization   | Y   | -  |
|                      | pkey – return the primary key of a table                         | Y   | -  |

| PyGreSQL |  | Yes | Remarks |
|----------|--|-----|---------|
|          | 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   | -       |

| PyGreSQL      |  | Yes | Remarks  |
|---------------|--|-----|--|
|               | 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.         |
|               | 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 <b>listen/notify</b> . |
|               | 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   | -  |

| PyGreSQL                       |  | Yes | Remarks   |
|--------------------------------|--|-----|---|
|                                | single/singledict/<br>singlenamed/singlescalar –<br>get single result of a query | Y   | -   |
|                                | listfields – list fields names<br>of previous query result                       | Y   | -   |
|                                | fieldname, fieldnum – field<br>name/number conversion                            | Y   | -   |
|                                | fieldinfo – detailed info<br>about query result fields                           | Y   | -   |
|                                | ntuples – return number of<br>tuples in query object                             | Y   | -   |
|                                | memsize – return number<br>of bytes allocated by query<br>result                 | Y   | -   |
| LargeObject – Large<br>Objects | open – open a large object   | N   | Operations<br>related to<br>large objects                     |
|                                | close – close a large object   | N   | Operations<br>related to<br>large objects                     |
|                                | read, write, tell, seek,<br>unlink – file-like large<br>object handling          | N   | Operations<br>related to<br>large objects                     |
|                                | size – get the large object<br>size  | N   | Operations<br>related to<br>large objects                     |
|                                | export – save a large object<br>to a file  | N   | Operations<br>related to<br>large objects                     |
|                                | Object attributes  | N   | Operations<br>related to<br>large objects                     |
| The Notification Handler       | Instantiating the<br>notification handler  | N   | The database<br>does not<br>support<br><b>listen/notify</b> . |
|                                | Invoking the notification<br>handler   | N   | The database<br>does not<br>support<br><b>listen/notify</b> . |



| PyGreSQL                           |  | Yes | Remarks  |
|------------------------------------|--|-----|--|
|                                    | Sending notifications  | N   | The database does not support <b>listen/notify</b> . |
|                                    | Auxiliary methods  | N   | The database does not support <b>listen/notify</b> . |
| <b>pgdb</b>                        |  |     |  |
| 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   | -  |

| PyGreSQL                             |  | Yes | Remarks |
|--------------------------------------|--|-----|---------|
|                                      | 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   | -       |
|                                      | Methods and attributes that are not part of the standard | Y   | -       |
| Type – Type objects and constructors | 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       | Node Flavor         | Number of CN Connections | Number of DN Connections     |
|-----------------|---------------------|--------------------------|------------------------------|
| max_connections | vCPUs < 16          | 512                      | Number of CN connections x 2 |
|                 | vCPUs > 16 && <= 32 | 1024                     | Number of CN connections x 2 |
|                 | other               | 2048                     | Number of CN connections x 2 |

 NOTE

The policies of **comm\_max\_stream**, **poolsize**, and **max\_prepared\_transactions** are the same as those of **max\_connections**.

## 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](#).

**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 <b>dbadmin</b>.</p> <pre>SELECT ROLNAME,ROLCONNLIMIT FROM PG_ROLES WHERE ROLNAME='dbadmin';</pre> <p>Information similar to the following is displayed. <b>-1</b> indicates that the number of sessions connected to user <b>dbadmin</b> 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 <b>dbadmin</b>.</p> <pre>SELECT COUNT(*) FROM V\$SESSION WHERE USERNAME='dbadmin';</pre> <p>Information similar to the following is displayed. <b>1</b> indicates the number of session connections used by user <b>dbadmin</b>.</p> <pre>count ----- 1 (1 row)</pre>   |

| Description   | Command   |
|---|---|
| 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 <b>gaussdb</b>.</p> <pre>SELECT DATNAME,DATCONNLIMIT FROM PG_DATABASE WHERE DATNAME='gaussdb';</pre> <p>Information similar to the following is displayed. <b>-1</b> indicates that the number of sessions connected to database <b>gaussdb</b> is not limited.</p> <pre>datname   datconnlimit -----+----- gaussdb            -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 <b>gaussdb</b>.</p> <pre>SELECT COUNT(*) FROM PG_STAT_ACTIVITY WHERE DATNAME='gaussdb';</pre> <p>Information similar to the following is displayed. <b>1</b> indicates the number of session connections used by database <b>gaussdb</b>.</p> <pre>count -----       1 (1 row)</pre>  |
| View the number of session connections that have been used by all users.  | <p>Run the following command to view the number of session connections that have been used by all users:</p> <pre>SELECT COUNT(*) FROM V\$SESSION;</pre> <p>Information similar to the following is displayed.</p> <pre>count -----      10 (1 row)</pre>   |

----End


# 7 Clusters


## 7.1 Checking the Cluster Status

On the **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

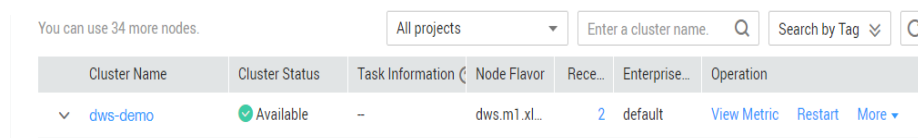
Log in to the GaussDB(DWS) management console. In the navigation pane, choose **Clusters**. The cluster list displays all clusters. If there are a large number of clusters, you can turn pages to view the clusters in any status.

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 7-1](#) lists the cluster list parameters.

**Figure 7-1** Cluster list



**Table 7-1** Cluster list parameters

| Parameter    | Description                                       |
|--------------|---|
| Cluster Name | Cluster name specified when a cluster is created. |

| Parameter        | Description  |
|------------------|--|
| Cluster Status   | Cluster running status. For details, see <a href="#">Cluster Status</a> .  |
| Task Information | Cluster task status. For details, see <a href="#">Cluster Task Information</a> .   |
| 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"><li>● <b>Monitoring Panel:</b> For details, see <a href="#">Databases Monitoring</a>.</li><li>● <b>Restart:</b> Click <b>Restart</b> to restart a cluster. For details, see <a href="#">Cluster Restart</a>.</li><li>● <b>More</b><ul style="list-style-type: none"><li>- <b>View Metric:</b> For details, see <a href="#">Monitoring Clusters Using Cloud Eye</a>.</li><li>- <b>Scale Out:</b> For details, see <a href="#">Cluster Scale-out</a>.</li><li>- <b>Create Snapshot:</b> For details, see <a href="#">Manually Creating a Snapshot</a>.</li><li>- <b>Reset Password:</b> For details, see <a href="#">Password Reset</a>.</li><li>- <b>Delete:</b> Click <b>Delete</b> to delete a cluster. For details, see <a href="#">Deleting Clusters</a>.</li><li>- <b>Switchback:</b> Click <b>Switchback</b> restore the primary/standby relationship of a cluster. For details, see <a href="#">Performing a Primary/Standby Switchback</a>.</li><li>- <b>Manage CN:</b> For details, see <a href="#">CNs</a>.</li></ul></li></ul> |

## Cluster Status

**Table 7-2** Cluster status description

| Status    | Description                               |
|-----------|---|
| Available | Indicates that the cluster runs properly. |

| Status                | Description  |
|-----------------------|--|
| 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"><li>• Use the SQL client tool to connect to the database as the database administrator and run the following command to periodically clear and reclaim the storage space:<br/><code>VACUUM FULL;</code><br/>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.</li><li>• 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 <a href="#">Cluster Scale-out</a>.</li></ul> |
| Unbalanced            | <p>If the role of a GTM or DN in the cluster is different from the initial role, the cluster is in the <b>Unbalanced</b> state. In the <b>Unbalanced</b> 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 <b>Available</b> state during off-peak hours.</p>   |
| Redistributing        | <p>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.</p>  |
| Redistribution failed | <p>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.</p>   |
| Degraded              | <p>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.</p>  |
| Unavailable           | <p>A cluster goes into this state when it cannot provide database services. You are advised to contact technical support.</p>  |
| Creating              | <p>A cluster goes into this state when it is being created.</p>  |
| Creation failed       | <p>A cluster goes into this state when it fails to be created.</p>   |
| Creating, restoring   | <p>A cluster goes into this state when it is being restored from a snapshot.</p>   |

## Cluster Task Information

**Table 7-3** 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.                            |
| 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.                          |

## 7.2 Viewing Basic Cluster Information

Log in to the GaussDB(DWS) management console. In the navigation tree on the left, choose **Clusters**. In the cluster list, locate the required cluster and click its name. The **Basic Information** page is displayed.

On a cluster's **Basic Information** page, you can view the following information:

- **Cluster Information:** [Table 7-4](#) lists the related parameters.
- **Database Attribute:** [Table 7-5](#) lists the related parameters.
- **Node Configuration:** [Table 7-6](#) lists the related parameters.
- **Network:** [Table 7-7](#) lists the related parameters.

**Table 7-4** Cluster information

| Parameter      | Description   |
|----------------|---|
| Cluster Name   | Cluster name specified when a cluster is created.                         |
| Cluster Status | Cluster running status. For details, see <a href="#">Cluster Status</a> . |

| Parameter                      | Description  |
|--------------------------------|--|
| Parameter Configuration Status | Parameter configuration status of a cluster.   |
| Task Information               | Cluster task status. For details, see <a href="#">Cluster Task Information</a> .   |
| Cluster ID                     | ID of the cluster.   |
| Nodes                          | Number of nodes in the cluster.  |
| Cluster Version                | Cluster version information.   |
| Used Storage Capacity          | Used storage capacity of the cluster.  |
| Created                        | Time when the cluster was created.   |
| Last Snapshot Created          | Time when the last snapshot was created.   |
| Maintenance Window             | Maintenance window of the cluster.<br>You can click <b>Configure</b> on the right of <b>Maintenance Window</b> to configure the maintenance window. For more information, see <a href="#">Configuring the Maintenance Window</a> . |

**Table 7-5** Database attribute parameters

| Parameter             | Description   |
|-----------------------|---|
| Default Database      | Database name specified when the cluster is created. When you connect to the cluster for the first time, connect to the default database.   |
| Initial Administrator | 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. |
| Port                  | Port for accessing the cluster database over the public network or private network. The host port is specified when a cluster is created and used to listen to client connections.                                |



| Parameter                   | Description  |
|-----------------------------|--|
| Connection String           | <p>Connection string. You can click <b>View Details</b> to check the data warehouse connection information. Its value can be:</p> <ul style="list-style-type: none"><li>• <b>JDBC URL (Private Network)</b>. In the private network environment, you can use the JDBC URL (private network) to connect to the cluster when developing applications.</li><li>• <b>JDBC URL (Public Network)</b>. In the public network environment, you can use the JDBC URL (public network) to connect to the cluster when developing applications.</li><li>• <b>ODBC URL</b>. In GaussDB(DWS), you can use an ODBC driver to connect to the database. The driver can connect to the database on an ECS or over the Internet.</li></ul> |
| Private Network Domain Name | <p>Name of the domain for accessing the database in the cluster over the private network. The private network domain address is automatically generated when a cluster is created. When you access a data warehouse cluster using a domain name, the domain name resolver provides the load balancing function.</p> <p>You can click <b>Modify</b> 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><b>NOTE</b></p> <ul style="list-style-type: none"><li>• A private IP address is automatically generated when you create a cluster. The IP address is fixed.</li><li>• The number of private IP addresses equals the number of CNs. You can log in to any node to connect to the cluster.</li><li>• If you access a fixed IP address over the internal network, all the workloads will be processed on a single CN.</li></ul>  |
| Public Network Domain Name  | <p>Name of the domain for accessing the database in the cluster over the public network.</p>   |
| Public Network IP Address   | <p>IP address for accessing the database in the cluster over the public network.</p> <p><b>NOTE</b></p> <ul style="list-style-type: none"><li>• If no EIP is assigned during cluster creation and <b>Public Network IP Address</b> is empty, click <b>Bind EIP</b> to bind an EIP to the cluster.</li><li>• If an EIP is bound during cluster creation, click <b>Unbind EIP</b> to unbind the EIP.</li></ul>   |

**Table 7-6** Node configuration

| Parameter           | Description                        |
|---------------------|------------------------------------|
| Node Flavor         | Node flavor of the cluster.        |
| Node Specifications | Specifications of the node flavor. |

**Table 7-7** Network

| Parameter      | Description  |
|----------------|--|
| Region         | Current working zone of the cluster.   |
| AZ             | AZ selected during cluster creation.   |
| VPC            | VPC selected during cluster creation.<br>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 <b>VPC and Subnet</b> in the <i>Virtual Private Cloud User Guide</i> .  |
| Subnet         | Subnet selected during cluster creation.<br>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 <b>VPC and Subnet &gt; Modifying a Subnet</b> in the <i>Virtual Private Cloud User Guide</i> . |
| Security Group | Security group selected during cluster creation.<br>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 <b>Security &gt; Security Group</b> in the <i>Virtual Private Cloud User Guide</i> .                                    |

## 7.3 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.

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 tree on the left, choose **Clusters**.
- Step 3** In the cluster list, find the target cluster and click the cluster name. The **Basic Information** page is displayed.
- Step 4** In the **Database Attribute** 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.

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 tree on the left, choose **Clusters**.

**Step 3** In the cluster list, find the target cluster and click the cluster name. The **Basic Information** page is displayed.

**Step 4** In the **Database Attribute** area, **Public Network Domain Name** and **Public Network IP Address** are empty. Click **Bind EIP** to bind the cluster with an EIP.

**Step 5** Select an EIP from the drop-down list in the **Bind EIP** dialog box.


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 **Database Attribute** area.

**Step 6** In the **Database Attribute** 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 **Database Attribute** 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 tree on the left, choose **Clusters**.
  - Step 3** In the cluster list, find the target cluster and click the cluster name. The **Basic Information** page is displayed.
  - Step 4** Click **Modify** next to the **Public Network Domain Name** in the **Database Attribute** 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**.
  - Step 3** In the cluster list, find the target cluster and click the cluster name. The **Basic Information** page is displayed.
  - Step 4** Click **Release** next to the **Public Network Domain Name** in the **Database Attribute** area.
  - Step 5** In the **Release Domain Name** dialog box, click **Yes**.
- End

## 7.4 Cluster Scale-out

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 **VACUUM**, see "SQL Reference > SQL Syntax > VACUUM" in the *Data Warehouse Service (DWS) Developer Guide*.

### Impact on the System

- Before the scale-out, exit 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, you cannot restart or scale a cluster, create a snapshot, reset a password, or delete a cluster.

- During the 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, create a manual snapshot before scaling. For details about how to create a snapshot, see [Manually Creating a Snapshot](#).

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

**Step 2** Choose **Clusters**.

All clusters are displayed by default.

**Step 3** In the **Operation** column of the target cluster, choose **More > Scale Out**.

**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 256.
- 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 **Custom**, you can enable **Online Scale-out** and **Auto Redistribution**, and set **Redistribution Mode** to **Online**. Click **OK** if a message is prompted.

If you choose **Default**, **Online Scale-out** will be disabled, **Auto Redistribution** will be enabled, and **Redistribution Mode** will be **Offline** by default.

**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

## 7.5 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 lower than that in the balanced state. During off-peak hours, you are advised to perform a switchback to restore the primary/standby relationship of your cluster.

#### NOTE

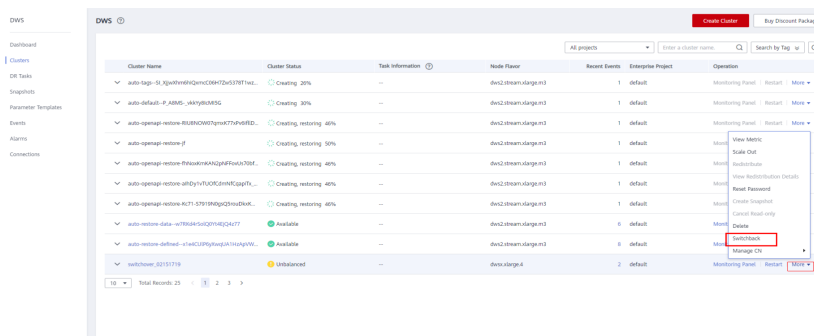
- Only 8.1.1.202 and later versions support primary/standby switchback.
- You are advised to perform a switchback during off-peak hours. Switchback interrupts services for a short period of time. The actual interruption duration depends on your service volume.

### Procedure

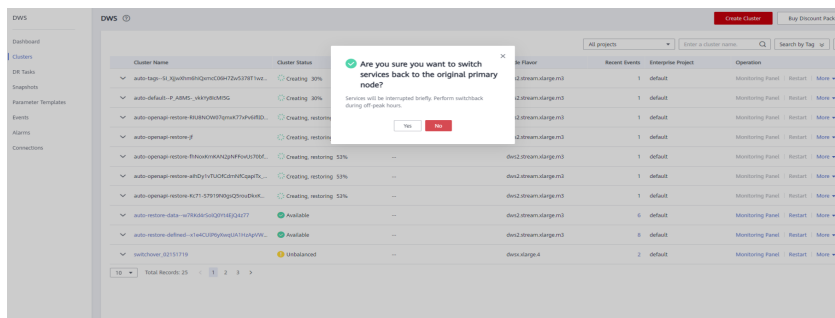
**Step 1** Log in to the GaussDB(DWS) management console.

**Step 2** On the **Clusters** page, find a cluster in **Unbalanced** state.

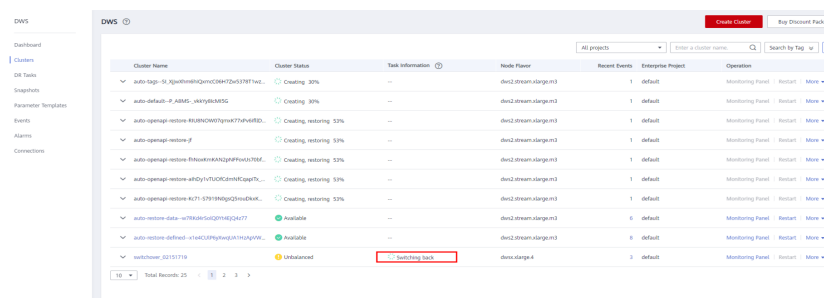
**Step 3** In the **Operation** column of the cluster, choose **More > Switchback**.



**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 switchback request is being processed.



**Step 5** Check the cluster status. During the switchback, the cluster status is **Switching back**. After the switchback, the cluster status will change to **Available**.



----End

## 7.6 Cluster Upgrade

After you create a data warehouse cluster, the system automatically configures a random maintenance window for the cluster. Alternatively, you can customize a maintenance window as required. For details about how to view and configure the maintenance window, see [Configuring the Maintenance Window](#).

The validity period of the maintenance window (maximum maintenance duration) is 4 hours. During this period, you can upgrade the cluster, install operating system patches, and harden the system. If no maintenance tasks are performed within the planned maintenance window, the cluster continues to run properly until the next maintenance window. 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

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, the service automatically upgrades the clusters to the latest version within the maintenance window. During the upgrade, the cluster is automatically restarted and cannot provide services for a short period of time. Therefore, you are advised to set a suitable time range when the number of connected users and the number of active tasks are small.

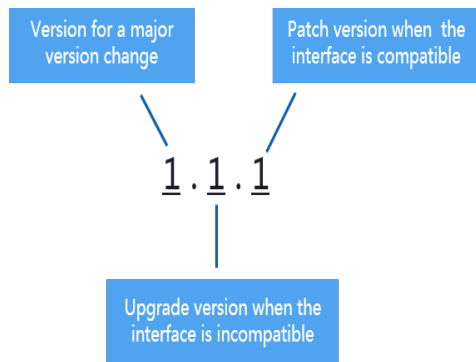


**NOTE**

After the cluster is upgraded, it cannot be rolled back.

The following figure shows the cluster version.

**Figure 7-2** 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, services will be interrupted for 1 to 3 minutes.
- **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: During this period, the database cannot be accessed.

## Configuring the Maintenance Window

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

**Step 2** Click **Clusters**.

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

In the **Cluster Information** area, you can view the maintenance window.

**Step 4** Click **Configure** next to **Maintenance Window**.

**Step 5** In the dialog box that is displayed, configure the maintenance window.

**Step 6** Click **OK**.

----End

## 7.7 Password Reset

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** 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 Modifications** tab 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**.

**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 8 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.
- Cannot be the same as the old password and cannot be the reverse of the old password.
- Cannot use a historical password.

### NOTE

If the default cluster administrator account is deleted or renamed, password resetting fails.

----End

## 7.8 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 in the parameter template, 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** Click **Clusters**.

**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

## 7.9 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 view or set common database parameters. For details, see [Managing Parameter Templates](#). 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.

### Procedure

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

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

**Step 3** In the cluster list, find the target cluster and click the cluster name. The **Basic Information** page is displayed.

**Step 4** Click the **Parameters** tab and modify the parameter values. Then click **Save**.

**Step 5** In the **Modification Preview** dialog box, confirm the modifications and click **Save**.

----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 [Procedure](#).

 NOTE

- The default values of the following parameters are for reference only. For more information, see "Setting GUC Parameters".
- After you modify parameters, restart the cluster to make the new settings take effect. Before the restart, the old parameter settings are still displayed.

**Table 7-8** Parameters

| Parameter              | Description   | Default Value |
|------------------------|---|---------------|
| session_timeout        | Specifies the timeout interval of an idle session, in seconds. The value <b>0</b> indicates that the timeout limit is disabled. The value ranges from 0 to 86400.   | 600           |
| datestyle              | Sets the display format for date and time.  | ISO,MDY       |
| failed_login_attempts  | Sets the number of consecutive incorrect password attempts after which the account is locked. The value <b>0</b> indicates that the number of incorrect password attempts is not limited. The value ranges from 0 to 1000.  | 10            |
| timezone               | Sets the time zone displayed in the time stamps.  | UTC           |
| log_timezone           | Sets the time zone for timestamps in the server log.  | UTC           |
| enable_resource_record | <p>Specifies whether to enable resource recording.</p> <p>If the actual execution time of an SQL statement is greater than the value of <b>resource_track_duration</b> (the default value is 60s; customizable), the monitoring information will be archived.</p> <p>This function will cause storage space expansion and slightly affect system performance. Disable it when it is not required.</p> <p><b>NOTE</b></p> <ul style="list-style-type: none"><li>• Archiving: The monitoring information is stored in the history view and archived in the info table. The archiving time is 3 minutes. After the archiving, records in the history view are cleared.</li><li>• History view <b>GS_WLM_SESSION_HISTORY</b>, which corresponds to info table <b>GS_WLM_SESSION_INFO</b></li><li>• History view <b>GS_WLM_OPERATOR_HISTORY</b>, which corresponds to info table <b>GS_WLM_OPERATOR_INFO</b></li></ul> | off           |

| Parameter               | Description  | Default Value |
|-------------------------|--|---------------|
| query_dop               | Sets the Symmetric Multi-Processing (SMP) degree. <ul style="list-style-type: none"><li>Value <b>0</b> indicates that the SMP is adaptive.</li><li>Value <b>1</b> indicates that the SMP is disabled.</li><li>Value <b>2</b> indicates that the SMP degree is 2.</li></ul>   | 0             |
| resource_track_cost     | Sets the minimum execution cost for resource monitoring on statements. The value <b>-1</b> indicates that resource monitoring is disabled (execution cost less than 10). If the value is greater than or equal to <b>0</b> , and the cost of executing statements exceeds the value and is greater than or equal to 10, resource monitoring is performed.<br>You can run the SQL command <b>Explain</b> to query the estimated execution cost of an SQL statement. | 100000        |
| resource_track_duration | Sets the minimum time for archiving executed statements recorded during real-time monitoring, in seconds. <ul style="list-style-type: none"><li>The value <b>0</b> indicates that all the statements are archived.</li><li>If the value is greater than <b>0</b>, historical statements are archived when the execution time of the statements exceeds this value.</li></ul>   | 60            |
| password_effect_time    | Sets the validity period of the account password. When the password is about to expire or has expired, the system prompts the user to change the password.<br>The value ranges from 0 to 999, in days. If this parameter is set to <b>0</b> , the function is disabled.  | 90            |
| update_lockwait_timeout | Sets the maximum duration that a lock waits for concurrent updates on a row to complete. If the lock wait time exceeds the value, the system will report an error. If this parameter is set to <b>0</b> , an error is reported immediately when a lock appears. The unit is milliseconds.  | 120000        |
| enable_resource_track   | Specifies whether to enable resource monitoring. After you enable this function, SQL statements can be monitored.  | on            |

| Parameter           | Description  | Default Value |
|---------------------|--|---------------|
| password_policy     | <p>Specifies whether to check the password complexity when you create a GaussDB(DWS) account using <b>CREATE ROLE/CREATE USER</b>, or modify the account using <b>ALTER ROLE/ALTER USER</b>.</p> <ul style="list-style-type: none"> <li>• <b>0</b> indicates that no password complexity policy is used.</li> <li>• <b>1</b> indicates that the default password complexity policy is used.</li> </ul>   | 1             |
| password_reuse_time | <p>Specifies whether to check the password reuse interval when you modify the account password using <b>ALTER USER/ALTER ROLE</b>. The value ranges from 0 to 3650, in days.</p> <ul style="list-style-type: none"> <li>• <b>0</b> indicates that the password reuse interval is not restricted.</li> <li>• A positive number indicates that the new password cannot be chosen from passwords in history that are newer than the specified number of days.</li> </ul> <p><b>NOTE</b><br/>The <b>password_reuse_time</b> and <b>password_reuse_max</b> parameters are checked during password change.</p> <ul style="list-style-type: none"> <li>• If the value of either <b>password_reuse_time</b> or <b>password_reuse_max</b> is positive, the password can be reused.</li> <li>• If the value of <b>password_reuse_time</b> is <b>0</b>, password reuse is restricted based on the number of password changes, but not on the time elapsed.</li> <li>• If the values of both parameters are <b>0</b>, password reuse is not restricted.</li> </ul> | 60            |

| Parameter                | Description  | Default Value |
|--------------------------|--|---------------|
| password_reuse_max       | <p>Specifies whether to check the number of password changes when you modify the account password using <b>ALTER USER/ALTER ROLE</b>.</p> <ul style="list-style-type: none"><li>• <b>0</b> indicates that the password is not restricted by the number of password changes.</li><li>• A positive number indicates that the new password cannot be chosen from the specified number of the most recent passwords.</li></ul> <p><b>NOTE</b><br/>The <b>password_reuse_time</b> and <b>password_reuse_max</b> parameters are checked during password change.</p> <ul style="list-style-type: none"><li>• If the value of either <b>password_reuse_time</b> or <b>password_reuse_max</b> is positive, the password can be reused.</li><li>• If the value of <b>password_reuse_max</b> is <b>0</b>, the password is not restricted based on the time elapsed, but not on the number of password changes.</li><li>• If the values of both parameters are <b>0</b>, password reuse is not restricted.</li></ul> | 0             |
| password_lock_time       | <p>Specifies the duration before a locked account is automatically unlocked.</p> <ul style="list-style-type: none"><li>• <b>0</b> indicates that the account is not automatically locked if the password verification fails.</li><li>• A positive number indicates the duration after which a locked account is automatically unlocked.</li></ul>  | 1             |
| password_encryption_type | <p>Specifies the encryption type of user passwords.</p> <ul style="list-style-type: none"><li>• <b>0</b> indicates that passwords are encrypted with MD5.</li><li>• <b>1</b> indicates that passwords are encrypted with SHA-256, which is compatible with the MD5 user authentication method of the PostgreSQL client.</li><li>• <b>2</b> indicates that passwords are encrypted with SHA-256. MD5 is not recommended because it is not a secure encryption algorithm.</li></ul>  | 2             |

| Parameter                     | Description   | Default Value |
|-------------------------------|---|---------------|
| password_notify_time          | <p>Specifies how many days in advance a user is notified before a password expires.</p> <ul style="list-style-type: none"><li>• <b>0</b> indicates that the notification is disabled.</li><li>• A value ranging from 1 to 999 indicates the number of days prior to password expiration that a user will receive a notification.</li></ul>  | 7             |
| enable_stateless_pooler_reuse | <p>Specifies whether to enable the pooler reuse mode. The setting takes effect after the cluster is restarted.</p> <ul style="list-style-type: none"><li>• <b>on</b> indicates that the pooler reuse mode is enabled.</li><li>• <b>off</b> indicates that the pooler reuse mode is disabled.</li></ul> <p><b>NOTE</b><br/>Set this parameter to the same value for CNs and DN. If this parameter is set to <b>off</b> for CNs and <b>on</b> for DNs, the cluster communication fails. Restart the cluster for the setting to take effect.</p>   | off           |
| work_mem                      | <p>Specifies the amount of memory to be used by internal sort operations and hash tables before they write data into temporary disk files, in KB.</p> <p>Sort operations are required for <b>ORDER BY</b>, <b>DISTINCT</b>, and merge joins.</p> <p>Hash tables are used in hash joins, hash-based aggregation, and hash-based processing of <b>IN</b> subqueries.</p> <p>In a complex query, several sort or hash operations may run in parallel; each operation will be allowed to use as much memory as this parameter specifies. If the memory is insufficient, data will be written into temporary files. In addition, several running sessions could be performing such operations concurrently. Therefore, the total memory used may be many times the value of <b>work_mem</b>.</p> | 64MB          |



| Parameter            | Description   | Default Value |
|----------------------|---|---------------|
| maintenance_work_mem | Specifies the maximum amount of memory to be used by maintenance operations, such as <b>VACUUM</b> , <b>CREATE INDEX</b> , and <b>ALTER TABLE ADD FOREIGN KEY</b> , in KB.<br><b>NOTE</b><br>This parameter may affect the execution efficiency of <b>VACUUM</b> , <b>VACUUM FULL</b> , <b>CLUSTER</b> , and <b>CREATE INDEX</b> .  | 128MB         |
| enable_orc_cache     | Specifies whether to reserve 1/4 of <b>cstore_buffers</b> for storing ORC metadata when <b>cstore_buffers</b> is initialized. <ul style="list-style-type: none"> <li>• <b>on</b> indicates that the ORC metadata is cached, which improves the query performance of HDFS tables but occupies column-store caches. As a result, the column-store performance is compromised.</li> <li>• <b>off</b> indicates that the ORC metadata is not cached.</li> </ul> | on            |
| sql_use_spacelimit   | Specifies the space size for files to be flushed to disks when a single SQL statement is executed on a single DN, in KB. The managed space includes the space occupied by ordinary tables, temporary tables, and intermediate result sets to be flushed to disks. <b>-1</b> indicates no limit.   | -1            |
| enable_bitmapscan    | Specifies whether to enable the optimizer's use of bitmap-scan plan types. <ul style="list-style-type: none"> <li>• <b>on</b></li> <li>• <b>off</b></li> </ul>  | on            |
| enable_hashagg       | Specifies whether to enable the optimizer's use of hash aggregation plan types. <ul style="list-style-type: none"> <li>• <b>on</b></li> <li>• <b>off</b></li> </ul>   | on            |
| enable_hashjoin      | Specifies whether enable the optimizer's use of hash join plan types. <ul style="list-style-type: none"> <li>• <b>on</b></li> <li>• <b>off</b></li> </ul>   | on            |
| enable_indexscan     | Specifies whether to enable the optimizer's use of index-scan plan types. <ul style="list-style-type: none"> <li>• <b>on</b></li> <li>• <b>off</b></li> </ul>   | on            |

| Parameter            | Description   | Default Value |
|----------------------|---|---------------|
| enable_indexonlyscan | Specifies whether to enable the optimizer's use of index-only-scan plan types. <ul style="list-style-type: none"> <li>• <b>on</b></li> <li>• <b>off</b></li> </ul>  | on            |
| enable_mergejoin     | Specifies whether the optimizer's use of merge-join plan types. <ul style="list-style-type: none"> <li>• <b>on</b></li> <li>• <b>off</b></li> </ul>   | off           |
| enable_nestloop      | Specifies whether the optimizer's use of nested-loop-join plan types. It is impossible to suppress nested-loop joins entirely, but disabling this parameter encourages the optimizer to choose other methods if available. <ul style="list-style-type: none"> <li>• <b>on</b></li> <li>• <b>off</b></li> </ul>  | off           |
| enable_seqscan       | Specifies whether enable the optimizer's use of sequential-scan plan types. It is impossible to suppress sequential scans entirely, but disabling this parameter encourages the optimizer to choose other methods if available. <ul style="list-style-type: none"> <li>• <b>on</b></li> <li>• <b>off</b></li> </ul>   | on            |
| enable_tidscan       | Specifies whether enable the optimizer's use of TID scan plan types. <ul style="list-style-type: none"> <li>• <b>on</b></li> <li>• <b>off</b></li> </ul>  | on            |
| enable_kill_query    | In CASCADE mode, when a user is deleted, all the objects belonging to the user are deleted. This parameter specifies whether the queries of the objects belonging to the user can be unlocked when the user is deleted. <ul style="list-style-type: none"> <li>• <b>on</b> indicates that the unlocking is allowed.</li> <li>• <b>off</b> indicates that the unlocking is not allowed.</li> </ul> | off           |
| enable_vector_engine | Specifies whether to enable the optimizer's use of vectorized execution engines. <ul style="list-style-type: none"> <li>• <b>on</b></li> <li>• <b>off</b></li> </ul>  | on            |

| Parameter                 | Description   | Default Value |
|---------------------------|---|---------------|
| enable_broadcast          | Specifies whether to enable the optimizer's use of broadcast distribution when it evaluates the cost of stream. <ul style="list-style-type: none"><li>• <b>on</b></li><li>• <b>off</b></li></ul>  | on            |
| skew_option               | Specifies whether to enable an optimization policy. <ul style="list-style-type: none"><li>• <b>off</b> indicates that the policy is disabled.</li><li>• <b>normal</b> indicates that a radical policy is used. All possible skews are optimized.</li><li>• <b>lazy</b> indicates that a conservative policy is used. Uncertain skews are ignored.</li></ul>   | normal        |
| default_statistics_target | Specifies the default statistics target for table columns without a column-specific target set via <b>ALTER TABLE SET STATISTICS</b> . If this parameter is set to a positive number, it indicates the number of samples of statistics information. If this parameter is set to a negative number, percentage is used to set the statistic target. The negative number converts to its corresponding percentage, for example, <b>-5</b> means 5%. | 100           |
| enable_codegen            | Specifies whether to enable code optimization. Currently, LLVM optimization is used. <ul style="list-style-type: none"><li>• <b>on</b></li><li>• <b>off</b></li></ul>   | on            |

| Parameter                | Description  | Default Value |
|--------------------------|--|---------------|
| autoanalyze              | <p>Specifies whether to automatically collect statistics on tables that have no statistics when a plan is generated.</p> <ul style="list-style-type: none"><li>• <b>on</b> indicates that the table statistics are automatically collected.</li><li>• <b>off</b> indicates that the table statistics are not automatically collected.</li></ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"><li>• This parameter is now not available to foreign tables. If you need the statistics, manually perform the analyze operation.</li><li>• This parameter is not available to temporary tables with the <b>ON COMMIT [DELETE ROWS] DROP</b> option. If you need the statistics, manually perform the analyze operation.</li><li>• If an exception occurs in the database during the execution of <b>autoanalyze</b> on a table, after the database is recovered, the system may still prompt you to collect the statistics of the table when you run the statement again. In this case, manually perform <b>ANALYZE</b> on the table to synchronize statistics.</li></ul> | off           |
| enable_sonic_has_hagg    | <p>Specifies whether to enable the hash aggregation operator designed for column-oriented hash tables when certain constraints are met.</p> <ul style="list-style-type: none"><li>• <b>on</b></li><li>• <b>off</b></li></ul>   | on            |
| log_hostname             | <p>By default, connection log messages only show the IP address of the connecting host. The host name can be recorded when this parameter is set to <b>on</b>. It may take some time to parse the host name. Therefore, the database performance may be affected.</p> <ul style="list-style-type: none"><li>• <b>on</b></li><li>• <b>off</b></li></ul>   | off           |
| max_active_statements    | <p>Specifies the maximum number of concurrent jobs. This parameter applies to all the jobs on one CN. The values <b>-1</b> and <b>0</b> indicate that the number of concurrent jobs is not limited.</p>  | 60            |
| enable_resource_tracking | <p>Specifies whether to enable resource monitoring.</p>  | on            |

| Parameter               | Description  | Default Value |
|-------------------------|--|---------------|
| resource_track_level    | <p>Sets the resource monitoring level of the current session. This parameter is valid only when <b>enable_resource_track</b> is set to <b>on</b>.</p> <ul style="list-style-type: none"><li>• <b>none</b> indicates that resources are not monitored.</li><li>• <b>query</b> enables the query-level resource monitoring. If this function is enabled, the plan information (similar to the output information of <b>explain</b>) of SQL statements will be recorded in top SQL statements.</li><li>• <b>perf</b> enables the perf-level resource monitoring. If this function is enabled, the plan information (similar to the output information of <b>EXPLAIN ANALYZE</b>) that contains the actual execution time and the number of execution rows will be recorded in top SQL statements.</li><li>• <b>operator</b> enables the operator-level resource monitoring. If this function is enabled, not only the information including the actual execution time and number of execution rows is recorded in the top SQL statements, but also the operator-level execution information is updated to the top SQL statements.</li></ul> | query         |
| enable_dynamic_workload | <p>Specifies whether to enable dynamic load management.</p> <ul style="list-style-type: none"><li>• <b>on</b></li><li>• <b>off</b></li></ul>   | on            |
| topsql_retention_time   | <p>Specifies the data storage retention period of the <b>gs_wlm_session_info</b> and <b>gs_wlm_operator_info</b> catalogs in historical top SQL statements. The unit is day.</p> <ul style="list-style-type: none"><li>• If it is set to <b>0</b>, the data is stored permanently.</li><li>• If the value is greater than <b>0</b>, the data is stored for the specified number of days.</li></ul>   | 0             |
| track_counts            | <p>Specifies whether to enable collection of statistics on database activities.</p> <ul style="list-style-type: none"><li>• <b>on</b></li><li>• <b>off</b></li></ul>   | off           |

| Parameter                       | Description   | Default Value |
|---------------------------------|---|---------------|
| autovacuum                      | Specifies whether to enable the autovacuum process. <b>track_counts</b> must be set to <b>on</b> for autovacuum to work. <ul style="list-style-type: none"> <li>• <b>on</b></li> <li>• <b>off</b></li> </ul>  | off           |
| autovacuum_mode                 | Specifies the autovacuum mode. <b>autovacuum</b> must be set to <b>on</b> . <ul style="list-style-type: none"> <li>• <b>analyze</b> indicates that only autoanalyze is performed.</li> <li>• <b>vacuum</b> indicates that only autovacuum is performed.</li> <li>• <b>mix</b> indicates that both are performed.</li> <li>• <b>none</b> indicates that neither is performed.</li> </ul> | mix           |
| autoanalyze_timeout             | Specifies the autoanalyze timeout period, in seconds. If the duration of autoanalyze on a table exceeds the value of <b>autoanalyze_timeout</b> , the autoanalyze operation is automatically canceled.  | 5min          |
| autovacuum_io_limits            | Specifies the maximum number of I/Os triggered by the autovacuum process per second. <b>-1</b> indicates that the default cgroup is used.   | -1            |
| autovacuum_max_workers          | Specifies the maximum number of concurrent autovacuum threads. <b>0</b> indicates that autovacuum is disabled.  | 3             |
| autovacuum_naptime              | Specifies the interval between two autovacuum operations, in seconds.   | 10min         |
| autovacuum_vacuum_threshold     | Specifies the threshold for triggering <b>VACUUM</b> . When the number of deleted or updated records in a table exceeds the specified threshold, the <b>VACUUM</b> operation is executed on this table.   | 50            |
| autovacuum_analyze_threshold    | Specifies the threshold for triggering <b>ANALYZE</b> . When the number of deleted, inserted, or updated records in a table exceeds the specified threshold, the <b>ANALYZE</b> operation is executed on this table.  | 50            |
| autovacuum_analyze_scale_factor | Specifies a fraction of the table size added to the <b>autovacuum_analyze_threshold</b> parameter when deciding whether to analyze a table.   | 0.1           |

| Parameter                  | Description  | Default Value |
|----------------------------|--|---------------|
| statement_timeout          | Specifies the statement timeout interval, in milliseconds. When the execution time of a statement exceeds the value (starting from the time when the server receives the command), the statement reports an error and exits.   | 0             |
| deadlock_timeout           | Specifies the deadlock timeout interval, in milliseconds. When the applied lock exceeds the value, the system will check whether a deadlock occurs.  | 1s            |
| lockwait_timeout           | Specifies the maximum wait time for a single lock, in milliseconds. If the lock wait time exceeds the value, the system will report an error.  | 20min         |
| max_query_retry_times      | Specifies the maximum number of automatic retry times when an SQL statement error occurs. Currently, a statement can start retrying if the following errors occur: <b>Connection reset by peer</b> , <b>Lock wait timeout</b> , and <b>Connection timed out</b> . If this parameter is set to <b>0</b> , the retry function is disabled.   | 6             |
| max_pool_size              | Specifies the maximum number of connections between the connection pool of a CN and another CN or DN.  | 800           |
| enable_gtm_free            | Specifies whether the GTM-FREE mode is enabled. In large concurrency scenarios, the snapshots delivered by the GTM increase in number and size. The network between the GTM and the CN becomes the performance bottleneck. The GTM-FREE mode is used to eliminate the bottleneck. In this mode, the CN communicates with DNs instead of the GTM. The CN sends queries to each DN, which locally generates snapshots and XIDs, ensuring external write consistency but not external read consistency. | off           |
| enable_fast_query_shipping | Specifies whether to enable the optimizer's use of a distributed framework.  | on            |
| enable_crc_check           | Specifies whether to enable data checks. Check information is generated when table data is written and is checked when the data is read. You are not advised to modify the settings.   | on            |

| Parameter                     | Description  | Default Value |
|-------------------------------|--|---------------|
| explain_perf_mode             | Specifies the display format of <b>explain</b> . <ul style="list-style-type: none"><li>• <b>normal</b> indicates that the default printing format is used.</li><li>• <b>pretty</b> indicates that the optimized display format of GaussDB(DWS) is used. The new format contains a plan node ID, directly and effectively analyzing performance.</li><li>• <b>summary</b> indicates that analysis of the <b>pretty</b> printed information is added.</li><li>• <b>run</b> indicates that the system exports the printed information specified by <b>summary</b> as a CSV file for further analysis.</li></ul> | pretty        |
| udf_memory_limit              | Specifies the maximum physical memory that can be used when UDFs are executed on each CN and DN, in KB.  | 200MB         |
| default_transaction_read_only | Specifies whether each newly created transaction is read only. <ul style="list-style-type: none"><li>• <b>on</b> indicates the transaction is read only.</li><li>• <b>off</b> indicates the transaction is not read only.</li></ul>  | off           |

## 7.10 MRS Data Sources

### 7.10.1 Importing Data from MRS to GaussDB(DWS)

#### Importing Data from MRS to a Data Warehouse Cluster

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.

#### Import Process

Perform the following operations to import data from MRS to a data warehouse cluster:



1. In the data warehouse cluster, create an MRS data source connection according to [Creating an MRS Data Source Connection](#).

 **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. 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*.
  3. (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](#).

## 7.10.2 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 AZ, 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 > Analytics > 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 AZ, VPC, and subnet of the MRS cluster must be the same as those of the data warehouse cluster.

- **Cluster Type** must be **Analysis Cluster**.
- **Cluster Version** supports **MRS 1.9.2** (recommended).

 **NOTE**

**Cluster Version** can also be set to 1.6.x, 1.7.x, 1.8.x, or 2.0.x.

- In the **Analysis Components** area, select **Hive**, **Tez**, and **Spark2x**.

 **NOTE**

If you enable Kerberos authentication for an MRS cluster, use MRS Manager to create a user for interconnecting GaussDB(DWS) with the system after the MRS cluster is created. The user type must be **Human-Machine** and the user, user group **hadoop**, and role **Manager\_administrator** must be bound together. The user password must be changed on the MRS Manager page after the user is created.

If you already have a qualified MRS cluster, skip this step.

**Step 3** Choose **Service List > Analytics > GaussDB(DWS)**.

**Step 4** On the GaussDB(DWS) management console, click **Clusters**.

**Step 5** In the cluster list, click the name of a cluster. On the page that is displayed, click the **MRS Data Sources** tab.

**Step 6** Click **Create MRS Cluster Connection** and configure parameters.

**Table 7-9** MRS cluster connection parameters

| Parameter              | Description  |
|------------------------|--|
| <b>MRS Data Source</b> | Specifies the MRS cluster to which GaussDB(DWS) can connect. By default, all available analytic MRS clusters that are in the same VPC and subnet as the current data warehouse cluster and in the <b>Available</b> state are displayed.<br>After you select an MRS cluster, the system automatically displays whether Kerberos authentication is enabled for the selected cluster. Click <b>View MRS Cluster</b> to view its detailed information.<br>If the <b>MRS Data Source</b> drop-down list is empty, click <b>Create MRS Cluster</b> to create an MRS cluster. |
| <b>Description</b>     | Describes the connection.  |

**Step 7** 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.

----End

## 7.10.3 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.                            |
| dfs.encrypt.data.transfer                                 | Specifies whether to enable data encryption.<br><b>NOTE</b><br>This parameter is available only for clusters with Kerberos authentication enabled. |
| dfs.encrypt.data.transfer.algorithm                       | Specifies the encryption and decryption algorithm for key transmission.  |
| dfs.encrypt.data.transfer.cipher.suites                   | Specifies the encryption and decryption algorithm for the transmission of actually stored data.  |
| dfs.replication   | Specifies the default number of data copies.   |
| dfs.blocksize   | Specifies the default size of a data block.  |
| hadoop.security.authentication                            | Specifies the security authentication mode.  |
| hadoop.rpc.protection                                     | Specifies the RPC communication protection mode.   |
| dfs.domain.socket.path                                    | Specifies the locally used <b>Domain socket</b> path.  |

## 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, click **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.

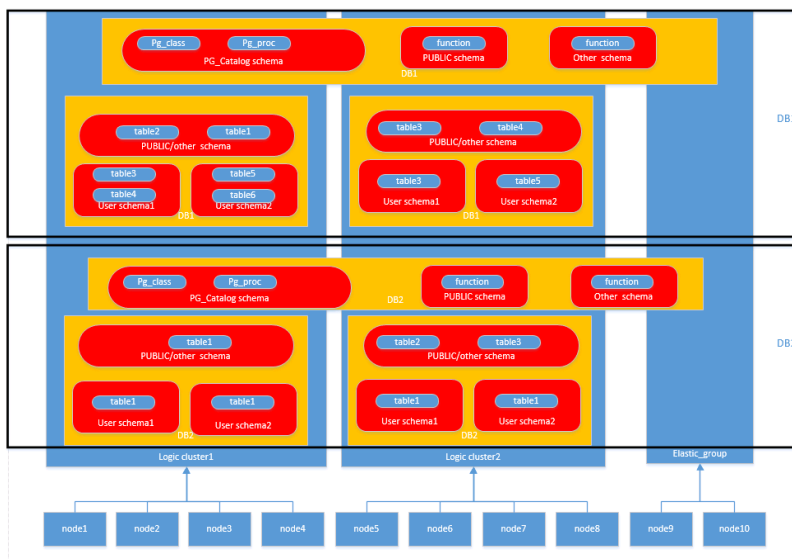
----End

# 7.11 Managing Logical Clusters

## 7.11.1 Overview

A physical cluster can be divided into logical clusters that use the node-group mechanism. Tables in a database can be allocated to different physical nodes by logical cluster. A logical cluster can contain tables from multiple databases. [Figure 7-3](#) shows the relationships between logical clusters, databases, and tables.

**Figure 7-3** 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.

## Permissions (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.
  - If a user is associated with a logical cluster, tables are created in the cluster. Otherwise, table creation abides by the [table creation rules](#) of logical clusters.
  - Users associated with a logical cluster do not need to specify **to group** when creating a table. The associated logical cluster can be changed.
- 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.

## Elastic Cluster

An elastic cluster consists of non-logical cluster nodes in a physical cluster in logical cluster mode. The elastic cluster is named **elastic\_group**, which is a special node group that can contain multiple or no 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.

## 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.

## 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 DNs are deployed.
- A logical cluster cannot be independently backed up or restored.

- A logical cluster cannot be independently upgraded.
- A logical cluster can be restarted, but cannot be independently stopped or started.
- A physical cluster cannot be rolled back to a physical cluster after it is converted to a logical cluster.
- Currently, logical cluster management cannot be used together with workload management.

## 7.11.2 Adding Logical Clusters

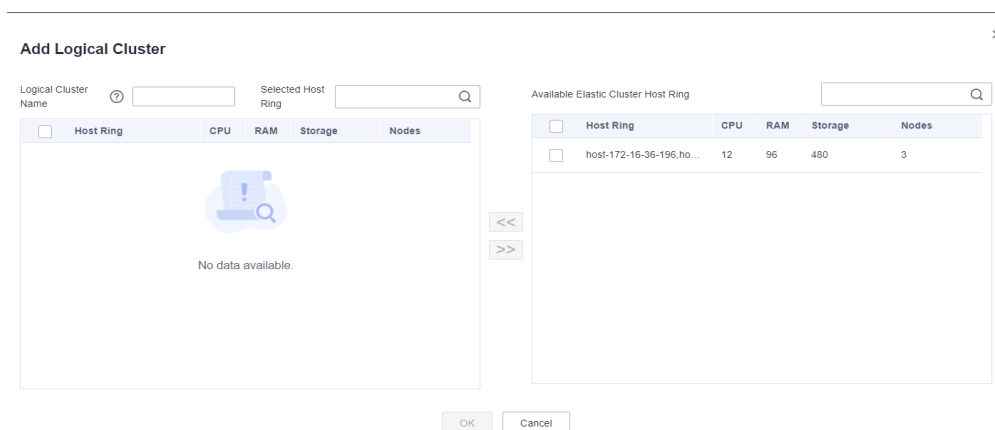
**Step 1** Log in to the GaussDB(DWS) management console.

**Step 2** In the cluster list, click the name of a cluster.

**Step 3** On the **Basic Information** page, enable **Logical Cluster**.

**Step 4** Go to the **Logical Clusters** tab and click **Add Logical Cluster**.

**Step 5** Move the ring you want to add from the right to the left panel, enter the logical cluster name, and click **OK**.



----End

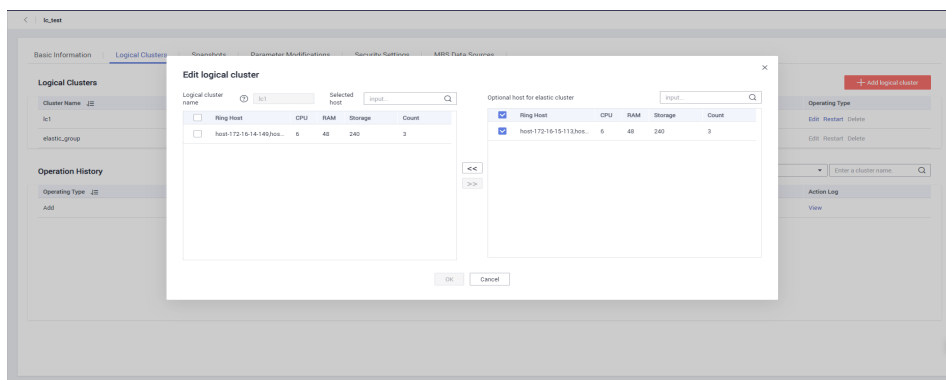
### CAUTION

- 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.

## 7.11.3 Editing Logical Clusters

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

- Step 2** In the cluster list, click the name of a cluster.
- Step 3** Switch to the **Logical Clusters** tab 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**.



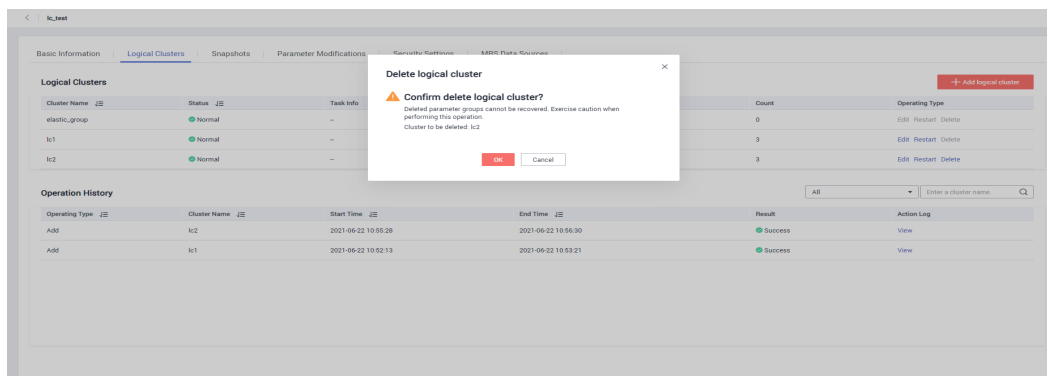
----End

**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.

## 7.11.4 Deleting Logical Clusters

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the cluster list, click the name of a cluster.
- Step 3** Switch to the **Logical Clusters** page. Click **Delete** in the **Operation** column of the target cluster, and click **OK** in the displayed dialog box.



----End



**NOTICE**

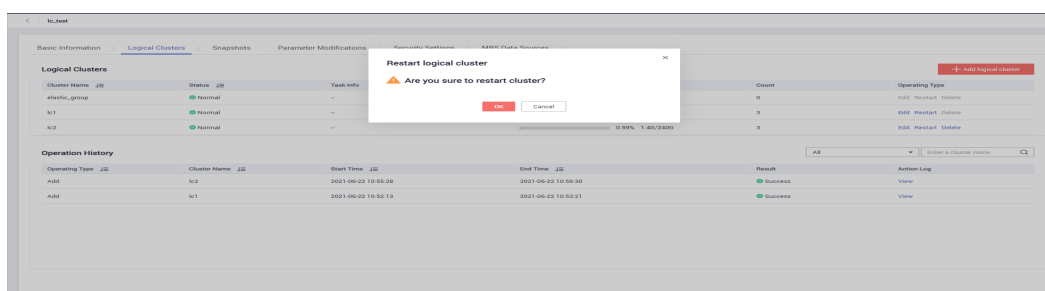
- The first added logical cluster cannot be deleted.
- Nodes of the deleted logical cluster are added to the elastic cluster.

## 7.11.5 Restarting Logical Clusters

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

**Step 2** In the cluster list, click the name of a cluster.

**Step 3** Switch to the **Logical Clusters** page. Click **Restart** in the **Operation** column of the target cluster, and click **OK** in the displayed dialog box.

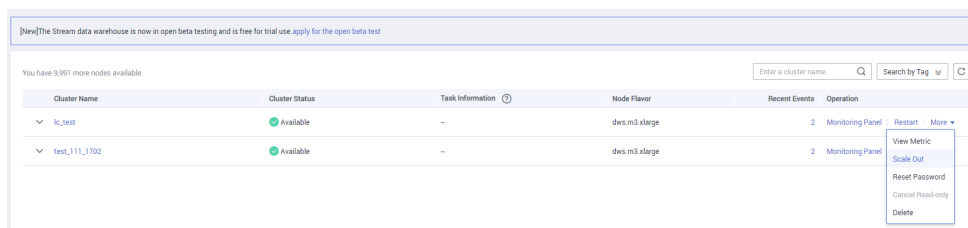


----End

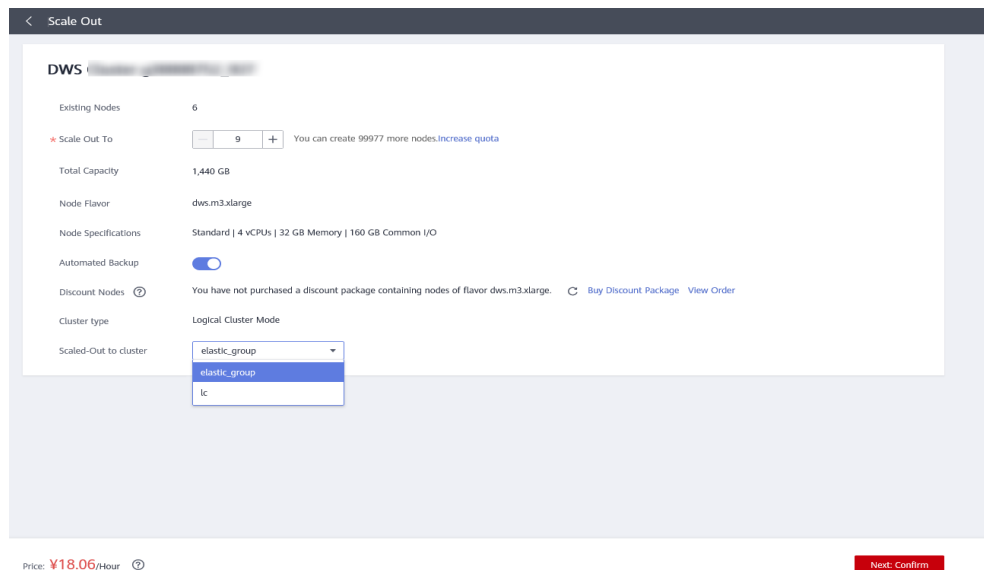
## 7.11.6 Scaling Out Logical Clusters

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

**Step 2** On the displayed **Clusters** page, click **More** in the **Operation** column of the target cluster, and select **Scale Out**.



**Step 3** On the **Scale Out** page, select the logical cluster or elastic cluster to be scaled out and click **Next** to confirm specifications.



----End

## NOTICE

- Logical clusters and elastic clusters cannot be scaled out online.

## 7.12 Managing Tags

### 7.12.1 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 10 tags added to it.

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.

## 7.12.2 Tag Management

This section describes how to search for clusters based on tags and how to add, modify, and delete tags for clusters.

### Adding a Tag to a Cluster

**Step 1** On the **Clusters** page, click the name of the cluster to which a tag is to be added, and click the **Tags** tab.

**Step 2** Click **Add Tag**.

**Step 3** Configure the tag parameters in the displayed dialog box.

**Table 7-10** Tag parameters

| Parameter | Description  | Example Value |
|-----------|--|---------------|
| Tag key   | <p>You can:</p> <ul style="list-style-type: none"><li>• Select a predefined tag key or an existing resource tag key from the drop-down list of the text box.</li></ul> <p><b>NOTE</b><br/>To add a predefined tag, you need to create one on TMS and select it from the drop-down list of <b>Tag key</b>. You can click <b>View predefined tags</b> to enter the <b>Predefined Tags</b> page of TMS. Then, click <b>Create Tag</b> to create a predefined tag. For more information, see "Management &gt; Predefined Tags &gt; Creating Predefined Tags" in the <i>Tag Management Service User Guide</i>.</p> <ul style="list-style-type: none"><li>• Enter a tag key in the text box. The tag key can contain a maximum of 36 characters and cannot be an empty string. Only digits, letters, underscores (_), and hyphens (-) are allowed.</li></ul> <p><b>NOTE</b><br/>A key must be unique in a given cluster.</p> | key01         |

| Parameter | Description  | Example Value |
|-----------|--|---------------|
| Tag value | <p>You can:</p> <ul style="list-style-type: none"><li>• Select a predefined tag value or resource tag value from the drop-down list of the text box.</li><li>• Enter a tag value in the text box. The tag key can contain a maximum of 43 characters and cannot be an empty string. Only digits, letters, underscores (_), periods (.), and hyphens (-) are allowed.</li></ul> | 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** Click **Clusters**.

**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** page, click the name of the cluster for which a tag is to be modified, and click the **Tags** tab.

**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** 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

## 7.13 Deleting Clusters


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** page, locate the cluster to be deleted.

**Step 4** Locate the row that contains the target cluster, and 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** page and delete the cluster.

- Release the EIP bound to the cluster.

If the cluster is bound with an EIP, you can click **Release the EIP bound to the cluster** to release the EIP of the cluster to be deleted.

**Step 6** Click **Yes**.

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 after the cluster is deleted.

----End

## 7.14 Managing Parameter Templates

To facilitate database parameter configuration, GaussDB(DWS) provides the parameter template function. A parameter template contains some common database parameters. You can manage parameter templates on the GaussDB(DWS) management console. After applying a parameter template to a cluster, you can modify parameters on the parameter modification page of the cluster.

The following parts are included in this section:

- [Overview](#)
- [Parameters](#)
- [Creating a Parameter Template](#)
- [Modifying a Parameter Template](#)
- [Applying a Parameter Template to the Cluster](#)
- [Deleting a Parameter Template](#)

### Overview

A parameter template is a set of parameters applicable to data warehouses. All parameters in the template have default values. The parameters include the session timeout interval, date, and time format. For details, see [Parameters](#). You can adjust parameter values to better adapt the database to actual services. When creating a cluster, you can specify a parameter template for it. Parameters in the template will be applied to all databases in the cluster. If you do not specify a parameter template, the system applies the default parameter template to the cluster. After a cluster is created, you can modify the parameters on the **Parameter Modifications** page. Alternatively, select an existing parameter template or create a parameter template on the **Parameter Template Management** page and apply it to the cluster.

GaussDB(DWS) presets a default parameter template to data warehouses of each version. The default parameter template cannot be deleted and modified. If you want to modify parameter values in the template, create a customized parameter template. The parameters in the customized template can be modified. After a customized parameter template is applied to a cluster, it is not associated with the cluster. If you modify the parameter values in the template, the modifications will not be synchronized to the cluster. You need to apply the template to the cluster again, and then the modified parameter values can be applied to the cluster. Similarly, if you modify parameters on the cluster details page, the modifications will not be synchronized to the parameter template.

 **NOTE**

The default values of the following parameters are for reference only. For more information, see "Setting GUC Parameters".

## Parameters

The parameter template only contains three parameters. These parameters will take effect on a cluster during cluster installation. You can check more parameters on the cluster parameter modification page of the console. For details, see [Modifying Database Parameters](#).

**Table 7-11** Parameters

| Parameter                | Description  | Default Value |
|--------------------------|--|---------------|
| timezone                 | Sets the time zone displayed in the time stamps.   | UTC           |
| log_timezone             | Sets the time zone for timestamps in the server log.   | UTC           |
| password_encryption_type | Specifies the encryption type of user passwords. <ul style="list-style-type: none"><li>• <b>0</b> indicates that passwords are encrypted with MD5.</li><li>• <b>1</b> indicates that passwords are encrypted with SHA-256, which is compatible with the MD5 user authentication method of the PostgreSQL client.</li><li>• <b>2</b> indicates that passwords are encrypted with SHA-256. MD5 is not recommended because it is not a secure encryption algorithm.</li></ul> | 1             |

## Creating a Parameter Template

If parameters in the default parameter template cannot meet service requirements, you can customize a parameter template and change the parameter values to better adapt to services.

To create a parameter template, perform the following steps:

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation tree on the left, click **Parameter Templates**.
- Step 3** Click **Create Parameter Template** and set the following parameters:
  - **Database Engine:** Select a database engine.
  - **Database Version:** Select a database version.

- **Name:** Enter the name of the new parameter template.  
Enter 4 to 64 characters. Only letters, digits, hyphens (-), underscores (\_), and periods (.) are allowed. The value must start with a letter. Letters are case-insensitive.
- **Description:** Enter the description of the new parameter template. This parameter is optional.  
The parameter template description contains 0 to 256 characters and does not support special characters !<>='&".

**NOTE**

The **Database Engine** and **Database Version** selected during parameter template creation must be the same as the cluster type and version of the parameter template to be applied.

**Step 4** Click **OK**.

----End

## Modifying a Parameter Template

You can modify the parameter values in a customized parameter template but cannot modify the parameter values in the default parameter template.


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

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

**Step 3** In the **Name** column, click the name of the target parameter template. Its parameter table is displayed.

**Step 4** Enter a new value in the **Value** column of the parameter to be modified. After the modification, click **Save**.

**Figure 7-4** Modifying parameters



| 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.   |

**Step 5** In the **Modification Preview** dialog box, confirm the settings and modifications and click **Save**.

----End

## Applying a Parameter Template to the Cluster

After a cluster is created, you can apply a new parameter template to the cluster so that the values of all parameters in the parameter template can take effect in the cluster.

To apply a parameter template, perform the following steps:

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

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



**Step 3** Select the target parameter template and click **Apply** in the **Operation** column.

**Step 4** In the **Parameter Template Application** dialog box that is displayed, select the target cluster.

You can apply the selected parameter template to the cluster corresponding to the parameter template version.

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

If some parameter values in the new parameter template are different from the original parameter values in the cluster, a window comparing the differences will be displayed.

----End

## Deleting a Parameter Template

You can delete an unnecessary parameter template or a parameter template that is no longer used. The default parameter template cannot be deleted. Deleted parameter templates cannot be restored. Exercise caution when performing this operation.

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

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

**Step 3** In the **Operation** column of the parameter template to be deleted, click **Delete**.

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

----End

## 7.15 Managing Clusters That Fail to Be Created

If a cluster fails to be created, you can go to the **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 and click **Clusters** in the left navigation pane on the left.

**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 codes, 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 and click **Clusters** in the left navigation pane on the left.

**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

# 8 Cluster HA

---

## 8.1 Snapshots

### 8.1.1 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](#).

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 about how to restore a cluster from a snapshot, see [Restoring a Snapshot to a New Cluster](#).

The snapshot backup and restoration rates are as follows. (The statistics are obtained in the lab test and 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 dependencies of the snapshot service are as follows:
  - Only the snapshots stored in OBS can be used to restore data to a new cluster.
- A new cluster created from the snapshot has the same configurations (including the number and flavor of nodes) as those of 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.

### 8.1.2 Manual Snapshots

## 8.1.2.1 Manually Creating a Snapshot

### Prerequisites

A 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 backups, which takes a long time to create.

#### NOTE

- Manual snapshots can be backed up to OBS.
- Snapshots can be created only for clusters in **Available**, **Read-only**, or **Unbalanced** 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** Click **Create Snapshot** and enter 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 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 8-1** Creating a snapshot

**Step 4** Click **OK**.

The 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 becomes **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. Clear the data and recreate a snapshot. For details, see

----End

### 8.1.2.2 Deleting Manual Snapshots

On the **Snapshots** page, you can delete a snapshot in the **Unavailable** state or delete an available snapshot to release the storage space.

**CAUTION**

Deleted snapshots cannot be restored. Exercise caution when performing this operation.

#### Procedure

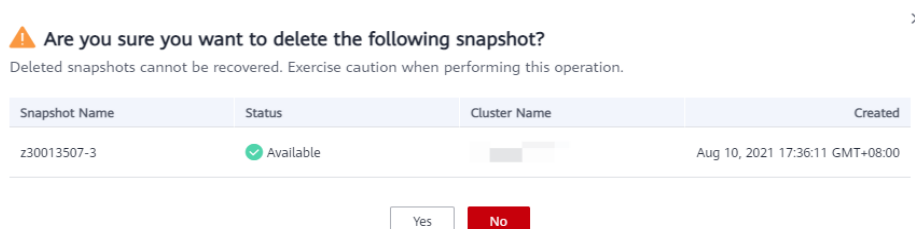
- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane, choose **Snapshots**. All snapshots are displayed by default.
- Step 3** In the **Operation** column of the snapshot that you want to delete, click **Delete**.

| Snapshot Name | Status    | Cluster Name | Type   | Storage Medium | Created                         | Operation           |
|---------------|-----------|--------------|--------|----------------|---------------------------------|---------------------|
| z30013507-3   | Available |              | Manual | OBS            | Aug 10, 2021 17:36:11 GMT+08:00 | Restore Delete Copy |

**NOTE**

You can delete snapshots that are manually created only.

**Step 4** In the dialog box that is displayed, confirm the information and click **Yes** to delete the snapshot.



----End

## 8.1.3 Automated Snapshots

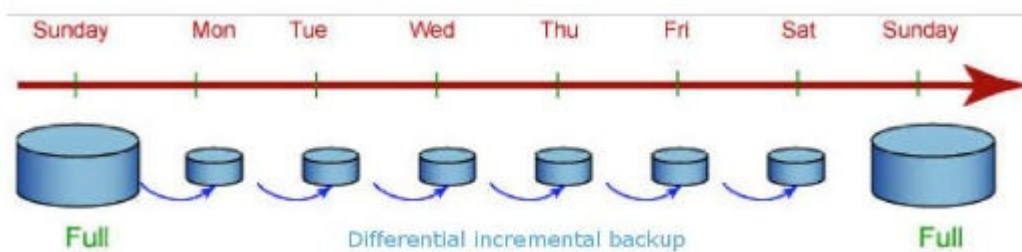
### 8.1.3.1 Automated Snapshot Overview

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. This can prevent data loss.

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, when the deletion of the snapshot will not hinder incremental data backup and restoration.

**Figure 8-2** 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 replicate an automated snapshot, see [Copying Automated Snapshots](#).

 **CAUTION**



If you disable the automated snapshot function for an existing cluster, all its automated snapshots will be deleted. However, manual snapshots will not be deleted.

### 8.1.3.2 Configuring an Automated Snapshot Policy

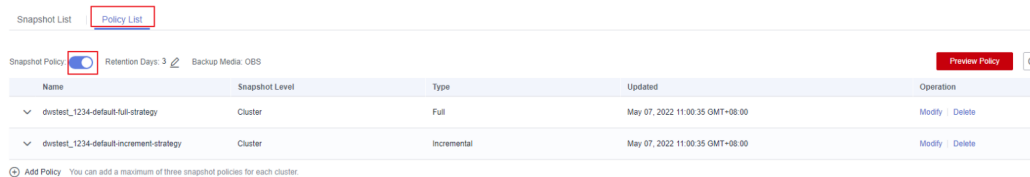
You can select a snapshot type and set one or more automated snapshot policies for a cluster. After an automated snapshot policy is enabled, the system automatically creates snapshots based on the preset time, period, and snapshot type according to the policy.

Perform the following steps to configure an automated snapshot policy.

#### Procedure

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation pane on the left, choose **Clusters**.
- Step 3** Click the name of a cluster.
- Step 4** Click the **Snapshots** tab page and click **Policy List**. All the 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 policy is disabled. If this policy is disabled, historical automated snapshots will be automatically deleted.

**Figure 8-3** Policy list



- Step 5** After this function is enabled, you can set the retention mode for automated snapshots. For more information, see [Table 8-1](#).

**Table 8-1** Automated snapshot parameters


| Parameter      | Description  |
|----------------|--|
| Retention Days | Retention days of the snapshots that are automatically created. The value ranges from 1 to 31 days.<br><b>NOTE</b><br>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 8-2](#).

 **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.

**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   One-time 

---

**Periodic Policy Configurations**

Days  Weekly   Monthly 

Sunday  Monday  Tuesday  Wednesday  Thursday  Friday  Saturday

Time  Daily 

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.

- **One-time**: Specify a day and the exact time on the day.



**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   One-time 


---


**One-time Policy Configurations**

Time 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.

- 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.

**Snapshot Policy**

Name

Type  Full  Incremental

Policy  Periodic  One-time

---

**Periodic Policy Configurations**

Days  Weekly  Monthly

|    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|
| 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 |    |    |

Select at least one of the preceding days.

All Clear

Time  Daily

Create a backup at  UTC

**WARNING**

Choosing the days in red (29/30/31) may skip some monthly backups.

**Table 8-2** Parameter description

| 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.   |
| Policy    | You can choose either periodic or one-time snapshots.<br><b>NOTE</b><br>You can select One-off Snapshot only when Snapshot Type is set to Full.  |
| One-time  | You can create a full snapshot at a specified time in the future. The UTC time is used.  |

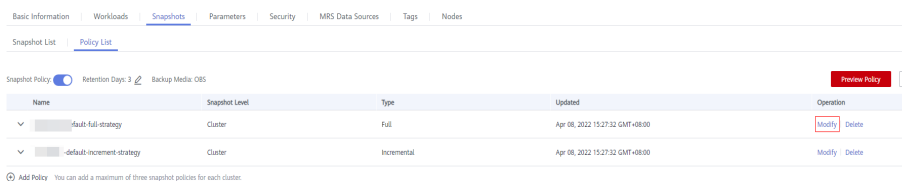
| Parameter                      | Description   |
|--------------------------------|---|
| Periodic Policy Configurations | <p>You can create automated snapshots on a daily, weekly, or monthly basis:</p> <ul style="list-style-type: none"> <li>• <b>Days:</b> Specify days for every week or every month. <b>Weekly</b> and <b>Monthly</b> cannot be selected at the same time. For <b>Monthly</b>, the specified days are applicable only to months that contain the dates. For example, if you select <b>29</b>, no automated snapshot will be created on February, 2022.</li> <li>• <b>Time:</b> 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.</li> </ul> <p><b>NOTICE</b><br/>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**.

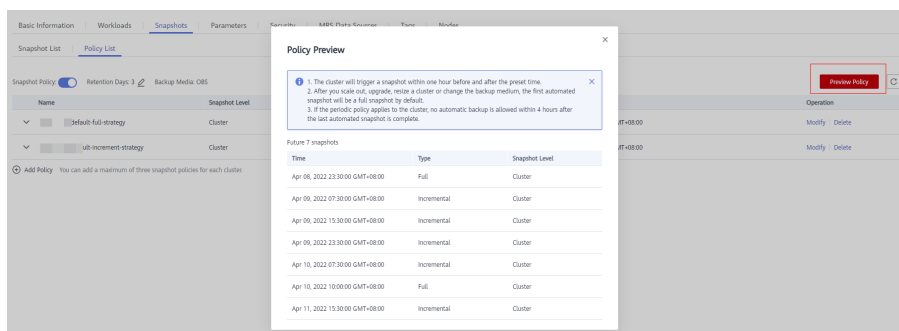
 **NOTE**

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 or 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 automated backup is allowed within 4 hours after the last automated snapshot is complete.
- If the snapshot start time 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

### 8.1.3.3 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 on the left, choose **Snapshots**.

All snapshots are displayed by default. You can copy the snapshots that are automatically created.

**Step 3** In the **Operation** column of the snapshot that you want to copy, click **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 8-4** 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. Clear the data and recreate a snapshot. For details, see

----End

### 8.1.3.4 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 automated snapshot function is disabled for a cluster. For details, see [Configuring an Automated Snapshot Policy](#).
- The cluster is deleted.

---

 **CAUTION**

If you disable automated snapshot, GaussDB(DWS) will stop taking snapshots and delete the existing automated snapshots of the corresponding cluster. Exercise caution when performing this operation.

---


## 8.1.4 Viewing Snapshot Information

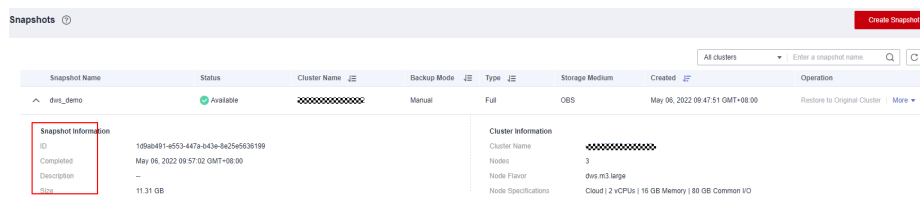
This section describes how to view snapshot information on the **Snapshots** page.

### 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 display the snapshot details.



**Step 3** You can view **Snapshot Name, Snapshot Status, Cluster Name, Snapshot Type, and Snapshot Created** 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 8-3** describes snapshot status.

**Table 8-3** Snapshot status

| Status             | Description   |
|--------------------|---|
| <b>Available</b>   | Indicates that the existing snapshot works properly.          |
| <b>Creating</b>    | Indicates that a snapshot is being created.                   |
| <b>Unavailable</b> | Indicates that the existing snapshot cannot provide services. |

**Table 8-4** describes the snapshot types.

**Table 8-4** Snapshot types

| Type          | Description  |
|---------------|--|
| <b>Manual</b> | 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. |

| Type             | Description  |
|------------------|--|
| <b>Automated</b> | 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. |

**Table 8-5** shows the snapshot media.

**Table 8-5** Storage media

| Storage Medium | Description  |
|----------------|--|
| OBS            | The created snapshot is an OBS snapshot and the backup data is stored on the OBS server. |

----End

## 8.1.5 Restoration Using a Snapshot

### 8.1.5.1 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 configure 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.

**Prerequisites**

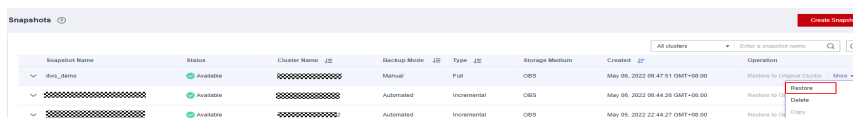
- The resources required for restoring data to a new cluster do not exceed your available resource quota.
- The snapshot is in the **Available** state.

**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** Configure the parameters of the new cluster, as shown in the following figure.

You can modify cluster parameters. For details, see [Table 8-6](#). By default, other parameters are the same as those in the snapshot. For details, see [Table 8-2](#).

**Table 8-6** Parameters for the new cluster

| Category          | Operation   |
|-------------------|---|
| Basic             | Configure the region, AZ, node flavor, cluster name, database port, VPC, subnet, security group, public access, and enterprise project.   |
| Advanced settings | If <b>Custom</b> is selected, configure the following parameters: <ul style="list-style-type: none"> <li>• <b>Parameter Template</b></li> <li>• <b>Tag:</b> If encryption is enabled for the original cluster, you can configure a key name.</li> </ul> |

**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.

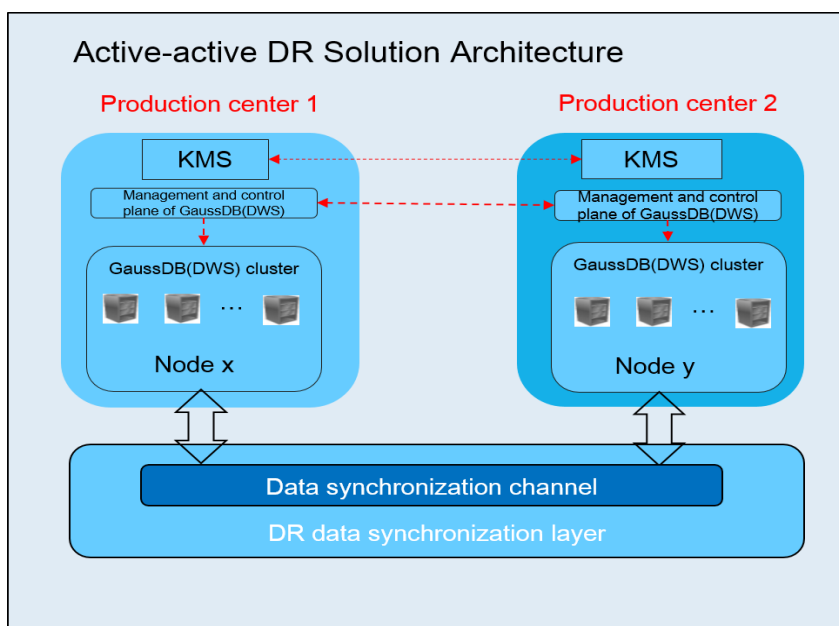
----End

## 8.2 Cluster DR

### 8.2.1 DR Overview

#### Overview

A homogeneous GaussDB(DWS) DR cluster is deployed in another AZ. 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.



#### NOTE

- This feature is supported only in cluster version 8.1.1 or later.

#### DR Features

- Multi-form DR
  - Cross-AZ 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, the 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.
- Logical clusters are not supported.
- DR refers to dual-cluster DR of the same tenant.
- The production cluster and DR cluster are in the same VPC and of the same version.

## 8.2.2 Creating a 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, click **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: Cross-AZ DR**
- **Name:** Enter 4 to 64 case-insensitive characters, starting with a letter. Only letters, digits, hyphens (-), and underscores (\_) are allowed.

|      |   |
|------|---|
| Type | <input type="button" value="Cross-AZ DR"/>  |
| Name | <input type="text" value="dws-demo"/>  |

**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.

### Production Cluster Information

Cluster Name

AZ

#### Step 6 Configure the DR cluster.

- Select the AZ associated with the region where the DR cluster resides.

#### NOTE

- The production cluster AZ will be filtered out from the available DR cluster AZs.
- 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

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: cn-north-7a | Cluster Type: Standard | Node Flavor: dws.m3.large | Nodes: 3 | VPC: vpc-abb5

#### 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.

Advanced Settings

DR Synchronization Period  Unit:

#### 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

## 8.2.3 Viewing DR Information

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

**Step 2** In the navigation pane on the left, click **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 Configurations:** You can view and modify the DR synchronization period.

|   |                                      |                       |  |
|---|--------------------------------------|-----------------------|--|
| <b>DR Information</b>                           |                                      |                       |  |
| DR ID   | a2053443-2c0e-48e2-94c2-2890d8cc3b61 | Type                  | Cross-AZ   |
| Name  |                                      | DR Task Created       | Jan 11, 2021 14:08:45 GMT+08:00  |
| Status  | Not started                          | DR Task Started       | --   |
| <b>Production Cluster Information</b>           |                                      |                       |  |
| AZ  | cn-north-7c                          | Used Storage Capacity | <div style="width: 0.27%;"><div style="width: 0.27%;"></div></div> 0.27% 0.66/240 GB |
| Cluster ID                                      | e57ca035-a184-4b08-9c15-1e31adfc08d3 | Last DR Succeeded     | --   |
| Cluster Name                                    | 1_3                                  | DR Status             | --   |
| <b>DR Cluster Information</b>                   |                                      |                       |  |
| AZ  | cn-north-7b                          | Used Storage Capacity | <div style="width: 0.28%;"><div style="width: 0.28%;"></div></div> 0.28% 0.66/240 GB |
| Cluster ID                                      | f31e86cf-94a3-40be-b520-bd7a82158962 | Last DR Succeeded     | --   |
| Cluster Name                                    | dr_test_0111_2                       | DR Status             | --   |
| <b>DR Configurations</b> <a href="#">Modify</a> |                                      |                       |  |
| DR Synchronization Period  60 Minute            |                                      |                       |  |

----End

## 8.2.4 DR Management

### Starting a DR Task

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

**Step 2** In the navigation pane on the left, click **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, and node replacement, on the production cluster and 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) management console.

**Step 2** In the navigation pane on the left, click **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) management console.

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

- You can perform a DR switchover when the DR task is in the **Running** or **Abnormal** state.
- During a switchover, the original production cluster is not available.
- RPO of DR switchover:
  - 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

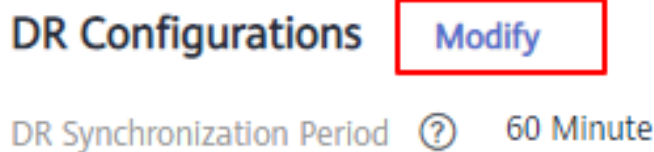
## Updating DR Configurations

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

**Step 2** In the navigation pane on the left, click **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) management console.

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

## 8.2.5 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) management console.

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

## 8.3 Associating and Disassociating ELB

### Overview

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. If a private domain name is used for connection, connection failures can be avoided by polling. However, private domain names cannot be used for public network access, and requests cannot be forwarded in the case of a CN failure. Therefore, ELB is used to avoid single CN failures.

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 *Elastic Load Balance User Guide*.

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.

The following ELB operations are supported:

- [Associating ELB](#)
- [Disassociating ELB](#)

 **NOTE**

- This feature is supported only in cluster version 8.1.1 and later.
- For load balancing and high availability purposes, and to prevent single CN failures, a cluster must be bound to ELB.

### 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).
- The ELB to be associated must use TCP and has a private IP address.
- When creating a load balancer, configure the specifications by service access traffic. ELB specifications cannot be modified on the GaussDB(DWS) management console.
- 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. Delete the load balancer in time to prevent unnecessary costs.
- 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** Click **Clusters**. All clusters are displayed by default.

**Step 3** Click the name of the target cluster.

**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                     |
|-------------------|----------------|------------------|-------------|---------------|--------------------|-------------------------------|
| 00028188747811200 | Available      | Associating ELB  | dws2.2large | 7             | default            | Monitoring Panel Restart More |

**Step 6** Log in to the ELB management console, click the name of the associated ELB, switch to the **Backend Server Groups** tab, and check whether the cluster CNs are associated with the load balancer.

| Name | Status  | Private IP Address | Health Check Result | Weight | Backend Port |
|------|---------|--------------------|---------------------|--------|--------------|
| ...  | Running | Etenmen NIC        | Healthy             | 1      | 8000         |
| ...  | Running | Etenmen NIC        | Healthy             | 1      | 8000         |
| ...  | Running | Etenmen NIC        | Healthy             | 1      | 8000         |

**Step 7** Open the **Basic Information** tab of the cluster. The **ELB Address** will be used for connecting to the cluster.

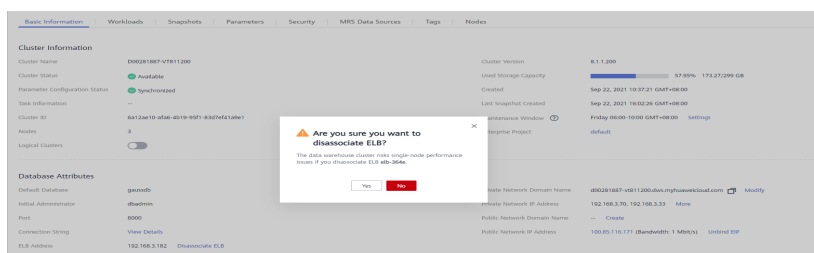
| Cluster Information            | Database Attributes                  |
|--------------------------------|--------------------------------------|
| Cluster Name                   | gaussdb                              |
| Cluster Status                 | Available                            |
| Parameter Configuration Status | Synchronized                         |
| Task Information               |                                      |
| Cluster ID                     | albfda5d-acea-49f2-8037-38d7a07135bd |
| Nodes                          | 3                                    |
| Logical Clusters               |                                      |
| Default Database               | gaussdb                              |
| Initial Administrator          | dbadmin                              |
| Port                           | 8000                                 |
| Connection String              | More Details                         |
| ELB Address                    | Disassociate ELB                     |



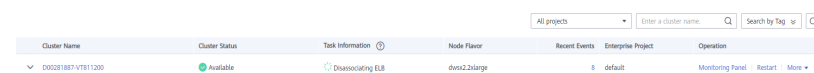
----End

## Disassociating ELB

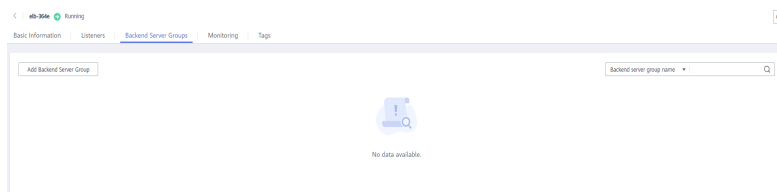
- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** Click **Clusters**. All clusters are displayed by default.
- Step 3** Click the name of the target cluster.
- Step 4** On the **Basic Information** page that is displayed, click **Disassociate ELB**.



- Step 5** After the request is delivered, go back to the **Clusters** page. Task information **Dissociating ELB** of the cluster is displayed. The process takes some time.



- 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.



----End

## 8.4 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.

## 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.
- Stop services when adding or deleting a CN.
- If a fault occurs when you add or delete a CN and the rollback also fails, log in to the background to rectify the fault. For details, see .


## Adding CNs

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

**Step 2** On the **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** On the displayed page, set the number of CNs as required and click **OK**.

 **Add CN Node**

Number of deployed CN 4

\* CN quantity after adjustment

### NOTICE

- Before adding a CN, ensure that the cluster is in the **Available**, **Unbalanced**, or **Redistributing** state.
- The number of CNs after adjustment must be greater than the number of deployed CNs, less than or equal to the number of nodes, and less than or equal to 20.

----End

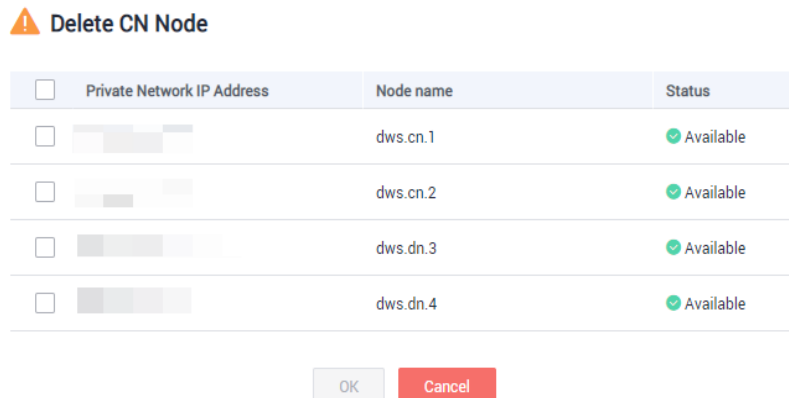
## Deleting CNs

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

**Step 2** On the **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 one CN must be reserved when you delete a CN.
- 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

# 9 Monitoring and Alarms

---

## 9.1 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](#)
- [Cluster Monitoring Metrics](#)
- [Dimensions](#)
- [Cluster and Node Monitoring Information](#)
- [Comparing the Monitoring Metrics of Multiple Nodes](#)

### 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 9-1](#) describes GaussDB(DWS) monitoring metrics.

**Table 9-1** GaussDB(DWS) monitoring metrics

| Metric ID                         | Name                   | Description   | Value Range | Monitored Object       | Monitoring Period (Raw Data) |
|-----------------------------------|------------------------|---|-------------|------------------------|------------------------------|
| dws001_shared_buffer_hit_ratio    | Cache Hit Ratio        | Percentage of data volume obtained from memory, expressed in percentage     | 0% to 100%  | Data warehouse cluster | 4 minutes                    |
| dws002_in_memory_sort_ratio       | In-memory Sort Ratio   | Percentage of data volume that is sorted in memory, expressed in percentage | 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                    |
| 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 size of data in the database, in 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                    |

| Metric ID              | Name                      | Description  | Value Range      | Monitored Object    | Monitoring Period (Raw Data) |
|------------------------|---------------------------|--|------------------|---------------------|------------------------------|
| dws010_cpu_usage       | CPU Usage                 | CPU usage of each node in a cluster, in percentage                                 | 0% to 100%       | Data warehouse node | 1 minute                     |
| dws011_mem_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            | $\geq 0$         | Data warehouse node | 1 minute                     |
| dws013_bytes_in        | Network Input Throughput  | Data input to each node in the cluster per second over the network<br>Unit: byte/s | $\geq 0$ 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<br>Unit: byte/s  | $\geq 0$ 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<br>Unit: GB                           | 100 to 2000 GB   | Data warehouse node | 1 minute                     |

| Metric ID                     | Name                        | Description  | Value Range      | Monitored Object    | Monitoring Period (Raw Data) |
|-------------------------------|-----------------------------|--|------------------|---------------------|------------------------------|
| dws017_disk_used_size         | Used Disk Space             | Used disk space of each node in the cluster<br>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<br>Unit: byte/s  | $\geq 0$ 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<br>Unit: byte/s | $\geq 0$ 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<br>Unit: second         | $> 0s$           | Data warehouse node | 1 minute                     |
| dws021_avg_disk_sec_per_write | Average Time per Disk Write | Average time used each time when data is written to a disk<br>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   | $\geq 0$         | Data warehouse node | 1 minute                     |

## Dimensions

| Key          | Value                     |
|--------------|---------------------------|
| datastore_id | Data warehouse cluster ID |


| Key             | Value                  |
|-----------------|------------------------|
| dws_instance_id | Data warehouse node ID |

## Cluster and Node Monitoring Information

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

**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, 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 9-2** Configuration example

| Parameter     | Example Value       |
|---------------|---------------------|
| Resource Type | DWS                 |
| Dimension     | Data Warehouse Node |



| Parameter        | Example Value   |
|------------------|---|
| Monitored Object | dws-demo-dws-cn-cn-2-1<br>dws-demo-dws-cn-cn-1-1<br>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

## Creating Alarm Rules

Setting GaussDB(DWS) alarm rules allows you to customize the monitored objects and notification policies and determine the running status of your GaussDB(DWS) at any time.

A GaussDB(DWS) alarm rule includes the alarm rule name, monitored object, metric, threshold, monitoring interval, and whether to send a notification. This section describes how to set GaussDB(DWS) alarm rules.

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

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

**Step 3** Locate the row containing the target cluster, click **More > View Metric** in the **Operation** column to enter the Cloud Eye management console and view the GaussDB(DWS) monitoring information.

The status of the target cluster must be **Available**. Otherwise, you cannot create alarm rules.


**Step 4** In the left navigation pane of the Cloud Eye management console, choose **Alarm Management > Alarm Rules**.

**Step 5** On the **Alarm Rules** page, click **Create Alarm Rule** in the upper right corner.

**Step 6** On the **Create Alarm Rule** page, set parameters as prompted.

1. Configure the rule name and description.
2. Configure the alarm parameters as prompted.

**Table 9-3** Configuring alarm parameters

| Parameter        | Description  | Example Value          |
|------------------|--|------------------------|
| Resource Type    | Name of the cloud service resource for which the alarm rule is configured.   | Data Warehouse Service |
| Dimension        | Metric dimension of the alarm rule. You can select <b>Data Warehouse Nodes</b> or <b>Data Warehouses</b> .   | Data Warehouse Node    |
| Monitoring Scope | Resource scope to which an alarm rule applies. Select <b>Specific resources</b> and select one or more monitoring objects. Select the ID of the cluster instance or node you have created. Click  to synchronize the monitoring objects to the right pane.  | Specific resources     |
| Method           | Select <b>Use template</b> or <b>Create manually</b> as required. <ul style="list-style-type: none"><li>- If no alarm template is available, set <b>Method</b> to <b>Create manually</b> and configure related parameters to create an alarm rule.</li><li>- If you have available alarm rule templates, set <b>Method</b> to <b>Use template</b>, so that you can use a template to quickly create alarm rules.</li></ul> | Create manually        |
| Template         | This parameter is valid only when <b>Use template</b> is selected.<br>Select the template to be imported. If no alarm template is available, click <b>Create Custom Template</b> to create one that meets your requirements.   | -                      |
| Alarm Policy     | This parameter is valid only when <b>Create manually</b> is selected.<br>Set the policy that triggers an alarm. For example, trigger an alarm if the CPU usage equals to or is greater than 80% for 3 consecutive periods.<br><a href="#">Table 9-1</a> describes the GaussDB(DWS) monitoring metrics.   | -                      |

| Parameter      | Description  | Example Value |
|----------------|--|---------------|
| Alarm Severity | Severity of an alarm. Valid values are <b>Critical, Major, Minor, and Informational.</b> | Major         |

3. Configure the alarm notification parameters as prompted.

**Table 9-4** Configuring alarm notifications

| Parameter           | Description  | Example Value |
|---------------------|--|---------------|
| Alarm Notification  | Whether to notify users when alarms are triggered. Notifications can be sent as emails or text messages, or HTTP/HTTPS requests sent to the servers.<br>You can enable (recommended) or disable <b>Alarm Notification.</b>   | Enable        |
| Validity Period     | Cloud Eye sends notifications only within the validity period specified in the alarm rule.<br>For example, if <b>Validity Period</b> is set to <b>00:00-8:00</b> , Cloud Eye sends notifications only within 00:00-8:00.   | -             |
| Notification Object | Name of the topic to which the alarm notification is sent.<br>If you enable <b>Alarm Notification</b> , you need to select a topic. If no desired topics are available, create one first, whereupon the SMN service is invoked. For details about how to create a topic, see the <i>Simple Message Notification User Guide</i> . | -             |
| Trigger Condition   | Condition for triggering the alarm. You can select <b>Generated alarm, Cleared alarm</b> , or both.  | -             |

4. After the configuration is complete, click **Next**.  
After the alarm rule is created, if the metric data reaches the specified threshold, Cloud Eye will immediately inform you that an exception has occurred.

----End

## 9.2 Databases Monitoring

## 9.2.1 Database Monitoring Overview

### Overview

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.

### Entering the Database Monitoring Page

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** On the **Clusters** page, locate the target cluster.
- Step 3** In the **Operation** column of the target cluster, choose **Monitoring Panel**. The database monitoring page is displayed.

----End

## 9.2.2 Monitoring Metrics

You can check the status and available resources of a cluster and learn about its real-time resource consumption through the GaussDB(DWS) monitoring items.

[Table 9-5](#) describes GaussDB(DWS) monitoring metrics.

**Table 9-5** GaussDB(DWS) monitoring metrics

| Monitor ed Object     | Metric         | Description   | Value Range                          | Monitor ing Period (Raw Data) |
|-----------------------|----------------|---|--------------------------------------|-------------------------------|
| Cluster Overvie w     | Cluster Status | Status of a cluster.  | Normal/<br>Abnormal<br>/<br>Degraded | 30s                           |
|                       | Nodes          | Number of available nodes and total number of nodes (Available/Total) in a cluster. | ≥ 0                                  | 60s                           |
|                       | CNs            | Number of CNs in a cluster.   | ≥ 0                                  | 60s                           |
|                       | Databases      | Number of created databases in a cluster.   | ≥ 0                                  | 90s                           |
| Resource Consum ption | CPU Usage      | Average real-time CPU usage of all nodes in a cluster.                              | 0% to 100%                           | 30s                           |

| Monitor ed Object                            | Metric               | Description   | Value Range      | Monitor ing Period (Raw Data) |
|--|----------------------|---|------------------|-------------------------------|
|  | Memory Usage         | Average real-time memory usage of all nodes in a cluster.   | 0% to 100%       | 30s                           |
|  | Disk Usage           | Average real-time disk usage of all nodes in a cluster.   | 0% to 100%       | 30s                           |
|  | Disk I/O             | Average real-time disk I/O of all nodes in a cluster.   | ≥ 0 KB/s         | 30s                           |
|  | Network I/O          | Average real-time network I/O of all NICs in a cluster.   | ≥ 0 KB/s         | 30s                           |
| Top 5 Time-Consuming Queries                 | Query ID             | ID of a query, which is automatically generated by the database.  | ≥ 0              | 180s                          |
|  | SQL Statement        | Query statement executed by a user.   | Character string | 180s                          |
|  | Execution Time       | Execution time of a query statement (unit: ms).   | ≥ 0 ms           | 180s                          |
| Top 5 Queries with Most Data Written to Disk | Query ID             | ID of a query, which is automatically generated by the database.  | ≥ 0              | 180s                          |
|  | SQL Statement        | Query statement executed by a user.   | Character string | 180s                          |
|  | Data Written to Disk | Data to be written to disks after a user runs a statement (unit: MB).   | ≥ 0 MB           | 180s                          |
| Cluster Resource Metrics                     | CPU Usage            | Average CPU usage and skew ratio of all nodes in the cluster. The formula for calculating the skew is (max-avg)/max.    | 0% to 100%       | 30s                           |
|  | Memory Usage         | Average memory usage and skew ratio of all nodes in the cluster. The formula for calculating the skew is (max-avg)/max. | 0% to 100%       | 30s                           |

| Monitor ed Object          | Metric                 | Description   | Value Range                 | Monitor ing Period (Raw Data) |
|----------------------------|------------------------|---|-----------------------------|-------------------------------|
|                            | Disk Usage             | Average usage and skew ratio of all disks in the cluster. The formula for calculating the skew is (max-avg)/max.    | 0% to 100%                  | 30s                           |
|                            | Disk I/O Usage         | Average I/O usage and skew rate of all disks in the cluster. The formula for calculating the skew is (max-avg)/max. | 0% to 100%                  | 30s                           |
|                            | Network I/O Usage      | Average I/O usage and skew rate of all NICs in the cluster. The formula for calculating the skew is (max-avg)/max.  | 0% to 100%                  | 30s                           |
| Key Databas e Metrics      | Cluster Status         | Cluster running status.   | Normal/ Degraded / Abnormal | 30s                           |
|                            | Cluster Abnormal CNs   | Number of abnormal CNs in the cluster   | ≥ 0                         | 60s                           |
|                            | Cluster Read-only      | Whether the cluster is in the read-only state   | Yes/No                      | 30s                           |
|                            | Concurrent Sessions    | Number of concurrent sessions in a cluster within a specified period.   | ≥ 0                         | 30s                           |
|                            | Concurrent Queries     | Number of concurrent queries in a cluster within a specified period.  | ≥ 0                         | 30s                           |
| Node Monitor ing-Overvie w | Node Name              | Name of a node in a cluster.  | Character string            | 30s                           |
|                            | CPU Usage              | CPU usage of a host.  | 0% to 100%                  | 30s                           |
|                            | Memory Usage           | Memory usage of a host.   | 0% to 100%                  | 30s                           |
|                            | Average Disk Usage (%) | Disk usage of a host.   | 0% to 100%                  | 30s                           |

| Monitor ed Object       | Metric                                 | Description   | Value Range      | Monitor ing Period (Raw Data) |
|-------------------------|--|---|------------------|-------------------------------|
|                         | IP Address                             | Service IP address of a host.                           | Character string | 30s                           |
|                         | Disk I/O                               | Disk I/O of a host (unit: KB/s)                         | ≥ 0 KB/s         | 30s                           |
|                         | TCP Protocol Stack Retransmission Rate | Retransmission rate of TCP packets per unit time.       | 0% to 100%       | 30s                           |
|                         | Status                                 | Running status of a host                                | Online/Offline   | 30s                           |
| Node Monitoring-Disks   | Node Name                              | Name of a node in a cluster.                            | Character string | 30s                           |
|                         | Disk Name                              | Name of a disk on a host.                               | Character string | 30s                           |
|                         | Disk Capacity                          | Disk capacity of the host (unit: GB)                    | ≥ 0 GB           | 30s                           |
|                         | Disk Usage                             | Disk usage of a host.                                   | 0% to 100%       | 30s                           |
|                         | Disk Read Rate                         | Disk read rate of the host (unit: KB/s)                 | ≥ 0 KB/s         | 30s                           |
|                         | Disk Write Rate                        | Disk write rate of the host (unit: KB/s)                | ≥ 0 KB/s         | 30s                           |
|                         | I/O Wait Time (await, ms)              | Average waiting time for each I/O request (unit: ms)    | ≥ 0 ms           | 30s                           |
|                         | I/O Service Time (svctm, ms)           | Average processing time for each I/O request (unit: ms) | ≥ 0 ms           | 30s                           |
|                         | I/O Utility (util, %)                  | Disk I/O usage of a host.                               | 0% to 100%       | 30s                           |
| Node Monitoring-Network | Node Name                              | Name of a node in a cluster.                            | Character string | 30s                           |
|                         | NIC Name                               | Name of the NIC on a host.                              | Character string | 30s                           |
|                         | NIC Status                             | NIC status.   | up/down          | 30s                           |

| Monitor ed Object     | Metric  | Description   | Value Range      | Monitor ing Period (Raw Data) |
|-----------------------|---|---|------------------|-------------------------------|
|                       | NIC Speed                                       | Working rate of a NIC, in Mbit/s.   | $\geq 0$         | 30s                           |
|                       | Received Packets                                | Number of received packets of a NIC.  | $\geq 0$         | 30s                           |
|                       | Sent Packets                                    | Number of sent packets of a NIC.  | $\geq 0$         | 30s                           |
|                       | Lost Packets Received                           | Number of received lost packets of a NIC.                                     | $\geq 0$         | 30s                           |
|                       | Receive Rate                                    | Number of bytes received by a NIC per unit of time (KB/s).                    | $\geq 0$ KB/s    | 30s                           |
|                       | Transmit Rate                                   | Number of bytes sent by a NIC per unit of time (unit: KB/s)                   | $\geq 0$ KB/s    | 30s                           |
| Databas e Monitor ing | Database Name                                   | Name of the database created by a user in a cluster.                          | Character string | 60s                           |
|                       | Usage   | Used capacity of the current database (unit: GB).                             | $\geq 0$ GB      | 86400s                        |
|                       | Users   | Number of users in the current database.                                      | $\geq 0$         | 30s                           |
|                       | Sessions  | Number of sessions in the current database.                                   | $\geq 0$         | 30s                           |
|                       | Applications                                    | Number of applications in the current database.                               | $\geq 0$         | 30s                           |
|                       | Queries   | Number of active queries in the current database.                             | $\geq 0$         | 30s                           |
|                       | Scanning Rows                                   | Number of rows returned by the full table scan query in the current database. | $\geq 0$         | 60s                           |
|                       | Index Query Rows                                | Number of rows returned by the index query in the current database.           | $\geq 0$         | 60s                           |
|                       | Inserted Rows                                   | Number of rows inserted in the current database.                              | $\geq 0$         | 60s                           |
| Updated Rows          | Number of rows updated in the current database. | $\geq 0$  | 60s              |                               |



| Monitor ed Object        | Metric                  | Description   | Value Range | Monitor ing Period (Raw Data) |
|--------------------------|-------------------------|---|-------------|-------------------------------|
|                          | Deleted Rows            | Number of rows deleted from the current database.   | $\geq 0$    | 60s                           |
|                          | Executed Transactions   | Number of transaction executions on the current database.   | $\geq 0$    | 60s                           |
|                          | Transaction Rollbacks   | Number of transactions in the current database that have been rolled back.  | $\geq 0$    | 60s                           |
|                          | Deadlocks               | Number of deadlocks detected in the current database.   | $\geq 0$    | 60s                           |
|                          | Physical Read Times     | Number of disk blocks read in the current database.   | $\geq 0$    | 60s                           |
|                          | Logical Read Times      | Number of times that disk blocks are found in the cache.  | $\geq 0$    | 60s                           |
|                          | Temporary Files         | Number of temporary files created in the current database.  | $\geq 0$    | 60s                           |
|                          | Temporary File Capacity | Size of temporary files written by the current database, in GB.   | $\geq 0$    | 60s                           |
| Perform ance Monitor ing | Cluster CPU Usage       | Historical trend of the average CPU usage and skew of all nodes in the cluster. The formula for calculating the skew is $(\max-\text{avg})/\max$ .    | 0% to 100%  | 30s                           |
|                          | Cluster Memory Usage    | Historical trend of the average memory usage and skew of all nodes in the cluster. The formula for calculating the skew is $(\max-\text{avg})/\max$ . | 0% to 100%  | 30s                           |
|                          | Cluster Disk Usage      | Historical trend of the average disk usage and skew of all nodes in the cluster. The formula for calculating the skew is $(\max-\text{avg})/\max$ .   | 0% to 100%  | 30s                           |

| Monitor ed Object | Metric                   | Description  | Value Range                          | Monitor ing Period (Raw Data) |
|-------------------|--------------------------|--|--------------------------------------|-------------------------------|
|                   | Cluster Disk I/O         | Historical trend of the average disk I/O and skew of all disks in the cluster. The formula for calculating the skew is (max-avg)/max.                        | 0% to 100%                           | 30s                           |
|                   | Cluster Network I/O      | Historical trend of the average network I/O value and skew of all NICs in the cluster. The formula for calculating the skew is (max-avg)/max.                | 0% to 100%                           | 30s                           |
|                   | Cluster Status           | Historical trend of the cluster status.  | Normal/<br>Abnormal<br>/<br>Degraded | 30s                           |
|                   | Cluster Read-only        | Historical trend of the cluster read-only status change trend.   | Yes/No                               | 30s                           |
|                   | Cluster Abnormal CNs     | Historical trend of the number of abnormal CNs in the cluster.   | ≥ 0                                  | 60s                           |
|                   | Cluster Abnormal DNs     | Historical trend of the number of abnormal DNs in the cluster.   | ≥ 0                                  | 60s                           |
|                   | Cluster CPU Usage of DNs | Historical trends of the average CPU usage and skew ratio changes of all DNs in the cluster. The formula for calculating the skew ratio is (max - avg)/ max. | 0% to 100%                           | 60s                           |
|                   | Cluster Sessions         | Historical trend of the number of sessions in a cluster.   | ≥ 0                                  | 30s                           |
|                   | Cluster Queries          | Historical change trend of the number of queries in the cluster.   | ≥ 0                                  | 30s                           |
|                   | Cluster Deadlocks        | Historical trend of the number of deadlocks in a cluster.  | ≥ 0                                  | 60s                           |

| Monitor ed Object | Metric                          | Description   | Value Range | Monitor ing Period (Raw Data) |
|-------------------|---------------------------------|---|-------------|-------------------------------|
|                   | Cluster TPS                     | Average number of transactions per second of all databases in a cluster. Formula: (delta_xact_commit + delta_xact_rollback)/ current_collect_rate | ≥0          | 60s                           |
|                   | Cluster QPS                     | Average number of concurrent requests per second of all databases in a cluster. Formula: delta_query_count/ current_collect_rate                  | ≥ 0         | 60s                           |
|                   | Database Sessions               | Historical trend of the number of sessions on a single database in a cluster.   | ≥ 0         | 30s                           |
|                   | Database Queries                | Historical trend of the number of queries on a single database in a cluster.  | ≥ 0         | 30s                           |
|                   | Database Submitted Transactions | Historical trend of the number of transactions submitted on a single database in a cluster.   | ≥ 0         | 60s                           |
|                   | Database Rollback Transactions  | Historical trend of the number of rollback transactions on a single database in a cluster.  | ≥ 0         | 60s                           |
|                   | Cluster Scanning Rows           | Historical trend of the number of rows returned by a full table scan on a single database in a cluster.   | ≥ 0         | 60s                           |
|                   | Database Index Query Rows       | Historical trend of the number of rows returned by an index query in a single database of a cluster.  | ≥ 0         | 60s                           |
|                   | Database Inserted Rows          | Historical trend of the number of rows inserted into a single database in a cluster.  | ≥ 0         | 60s                           |

| Monitor ed Object  | Metric                                       | Description   | Value Range      | Monitor ing Period (Raw Data) |
|--------------------|--|---|------------------|-------------------------------|
|                    | Database Updated Rows                        | Historical trend of the number of updated rows in a single database in a cluster.   | $\geq 0$         | 60s                           |
|                    | Database Deleted Rows                        | Historical trend of the number of deleted rows in a single database in a cluster.   | $\geq 0$         | 60s                           |
|                    | Database Capacity                            | Historical trend of the capacity in a single database in a cluster.   | $\geq 0$         | 86400s                        |
|                    | Database Length of the Request Waiting Queue | Historical trend of the waiting queue length on a single database in a cluster.   | $\geq 0$         | 30s                           |
|                    | Database TPS                                 | Number of transactions per second of each database in a cluster. Formula: $(\text{delta\_xact\_commit} + \text{delta\_xact\_rollback}) / \text{current\_collect\_rate}$ . | $\geq 0$         | 60s                           |
| Session Monitoring | Session ID                                   | ID of the current session (query thread ID).  | Character string | 30s                           |
|                    | User Name                                    | Name of the user who executes the current session.  | Character string | 30s                           |
|                    | Database Name                                | Name of the database connected to the current session.  | Character string | 30s                           |
|                    | Session Duration                             | Duration of the current session (unit: ms).   | $\geq 0$ ms      | 30s                           |
|                    | Application Name                             | Name of the application that creates the current session.   | Character string | 30s                           |
|                    | Queries                                      | Number of SQL statements executed in the current session.   | $\geq 0$         | 30s                           |
|                    | Latest Query Duration                        | Duration for executing the previous SQL statement in the current session.   | $\geq 0$ ms      | 30s                           |

| Monitor ed Object          | Metric            | Description  | Value Range        | Monitor ing Period (Raw Data) |
|----------------------------|-------------------|--|--------------------|-------------------------------|
|                            | Client IP Address | IP address of the client that initiates the current session.   | Character string   | 30s                           |
|                            | Connected CN      | Connected CN of the current session.   | Character string   | 30s                           |
|                            | Session Status    | Execution status of the current session.   | Running/Idle/Retry | 30s                           |
| Query Monitoring-Real-Time | Query ID          | Query ID of a current query statement, which is a unique identifier allocated by the kernel to each query statement. | Character string   | 30s                           |
|                            | User Name         | Name of the user who submits the current query statement.  | Character string   | 30s                           |
|                            | Database Name     | Name of the database corresponding to the current query statement.   | Character string   | 30s                           |
|                            | Application Name  | Name of the application corresponding to the current query statement.  | Character string   | 30s                           |
|                            | Workload Queue    | Name of the workload queue that carries the current query statement.   | Character string   | 30s                           |
|                            | Submitted         | Timestamp when the current query statement is submitted.   | Character string   | 30s                           |
|                            | Blocking Time     | Waiting time before the current query statement is executed, in ms.  | ≥ 0                | 30s                           |
|                            | Execution Time    | Execution time of the current query statement, in ms.  | ≥ 0                | 30s                           |
|                            | CPU Time          | Total CPU time spent by the current query statement on all DNs, in ms.   | ≥ 0                | 30s                           |
|                            | CPU Time Skew     | CPU time skew of the current query statement among all DNs.  | 0% to 100%         | 30s                           |

| Monitor ed Object        | Metric               | Description  | Value Range             | Monitor ing Period (Raw Data) |
|--------------------------|----------------------|--|-------------------------|-------------------------------|
|                          | Average Written Data | Average data size of the current query statement flushed to disks on all DN's, in MB.                                      | ≥ 0                     | 30s                           |
|                          | Statement            | Query statement that is being executed.  | Character string        | 30s                           |
|                          | Connected CN         | Name of the CN that submits the current query statement.   | Character string        | 30s                           |
|                          | Client IP Address    | IP address of the client that submits the current query statement.   | Character string        | 30s                           |
|                          | Lane                 | Lane where the current query statement is located.   | Fast Lane/<br>Slow Lane | 30s                           |
|                          | Query Status         | Query status of the statement that is being executed.  | Character string        | 30s                           |
|                          | Session ID           | Session ID of the current query statement, which is a unique identifier allocated by the kernel to each client connection. | Character string        | 30s                           |
|                          | Queuing Status       | Status of the current query execution in the database, indicating whether the query is queued in the workload queue.       | Yes/No                  | 30s                           |
| Query Monitoring-History | Query ID             | Query ID of a query statement, which is a unique identifier allocated by the kernel to each query statement.               | Character string        | 180s                          |
|                          | User Name            | Name of the user who submits a query statement.  | Character string        | 180s                          |
|                          | Application Name     | Application name corresponding to a query statement.   | Character string        | 180s                          |

| Monitor ed Object        | Metric                                 | Description  | Value Range      | Monitor ing Period (Raw Data) |
|--------------------------|--|--|------------------|-------------------------------|
|                          | Database Name                          | Name of the database corresponding to a query statement.                     | Character string | 180s                          |
|                          | Workload Queue                         | Name of the workload queue that carries the current query statement.         | Character string | 180s                          |
|                          | Submitted                              | Timestamp when a query statement is submitted.                               | Character string | 180s                          |
|                          | Blocking Time                          | Waiting time before the query statement is executed, in ms.                  | $\geq 0$         | 180s                          |
|                          | Execution Time                         | Execution time of the query statement, in ms.                                | $\geq 0$         | 180s                          |
|                          | CPU Time                               | Total CPU time spent by the query statement on all DNs, in ms.               | $\geq 0$         | 180s                          |
|                          | CPU Time Skew                          | CPU time skew of a query statement executed on all DNs.                      | 0% to 100%       | 180s                          |
|                          | Average Written Data                   | Average data size of the query statement flushed to disks on all DNs, in MB. | $\geq 0$         | 180s                          |
|                          | Statement                              | Query statements to be parsed  | Character string | 180s                          |
| Slow Instance Monitoring | Slow Instance                          | Number of slow instances detected at the current time point.                 | $\geq 0$         | 240s                          |
|                          | Detected                               | Time when a slow instance is detected for the first time.                    | Character string | 240s                          |
|                          | Node Name                              | Name of the node where the slow instance is deployed.                        | Character string | 240s                          |
|                          | Instance                               | Name of an instance.   | Character string | 240s                          |
|                          | Slow Node Detections (within 24 hours) | Number of times that a slow instance is detected within 24 hours.            | $\geq 0$         | 240s                          |

| Monitor ed Object         | Metric                             | Description   | Value Range      | Monitor ing Period (Raw Data) |
|---------------------------|------------------------------------|---|------------------|-------------------------------|
| Workload Queue Monitoring | Workload Queue                     | Name of the workload queue in the cluster.                          | Character string | 120s                          |
|                           | CPU Usage                          | Real-time CPU usage of the workload queue.                          | 0% to 100%       | 120s                          |
|                           | CPU Resource                       | CPU usage quotas of the workload queue.                             | 0% to 100%       | 120s                          |
|                           | Real-Time Concurrent Short Queries | Number of real-time concurrent simple queries in a workload queue.  | $\geq 0$         | 120s                          |
|                           | Concurrent Short Queries           | Concurrent simple query quotas of a workload queue.                 | $\geq 0$         | 120s                          |
|                           | Real-Time Concurrent Queries       | Number of real-time concurrent complex queries in a workload queue. | $\geq 0$         | 120s                          |
|                           | Query Concurrency                  | Concurrent complex query quotas of a workload queue.                | $\geq 0$         | 120s                          |
| Waiting Queries           | User                               | Name of the user of waiting queries                                 | Character string | 120s                          |
|                           | Application                        | Name of the application to be queried.                              | Character string | 120s                          |
|                           | Database                           | Name of the database to be queried.                                 | Character string | 120s                          |
|                           | Queuing Status                     | Execution status of a query in the database (CCN/CN/DN).            | Character string | 120s                          |
|                           | Wait Time                          | Waiting time for a waiting query (unit: ms).                        | $\geq 0$ ms      | 120s                          |
|                           | Workload Queue                     | Workload queue to which the waiting query belongs.                  | Character string | 120s                          |
|                           | Statement                          | Query statement for the waiting status.                             | Character string | 120s                          |
| Circuit Breaking Queries  | Query ID                           | Query ID of the circuit breaking query statement.                   | Character string | 120s                          |
|                           | Query Statement                    | Query statement for the circuit breaking status.                    | Character string | 120s                          |



| Monitor ed Object | Metric             | Description   | Value Range           | Monitor ing Period (Raw Data) |
|-------------------|--------------------|---|-----------------------|-------------------------------|
|                   | Blocking Time      | Blocking time before the query statement triggers circuit breaking, in ms.                        | ≥ 0                   | 120s                          |
|                   | Execution Time     | Execution time before the query statement triggers circuit breaking, in ms.                       | ≥ 0                   | 120s                          |
|                   | CPU Time           | Average CPU time consumed by each DN before the query statement triggers circuit breaking, in ms. | ≥ 0                   | 120s                          |
|                   | CPU Skew           | Skew rate of CPU time consumed by each DN before the query statement triggers circuit breaking.   | 0% to 100%            | 120s                          |
|                   | Exception Handling | Handling method after the query statement triggers circuit breaking.                              | Abort/ Degrade        | 120s                          |
|                   | Status             | Circuit breaking handling status of a query statement.  | Executing / Completed | 120s                          |
| SQL Tuning        | Query ID           | IP address of the current query (query logic ID).   | Character string      | 180s                          |
|                   | Database           | Name of the database where the current query is executed.   | Character string      | 180s                          |
|                   | Schema Name        | Name of the current query schema.   | Character string      | 180s                          |
|                   | User Name          | Name of the user who performs the query.  | Character string      | 180s                          |
|                   | Client             | Name of the client that initiates the current query.  | Character string      | 180s                          |
|                   | Client IP Address  | IP address of the client that initiates the current query.  | Character string      | 180s                          |
|                   | Running Time       | Execution time of the current query, in ms.   | ≥ 0                   | 180s                          |

| Monitor ed Object | Metric            | Description                           | Value Range      | Monitor ing Period (Raw Data) |
|-------------------|-------------------|---------------------------------------|------------------|-------------------------------|
|                   | CPU Time          | CPU time of the current query, in ms. | ≥ 0              | 180s                          |
|                   | Scale-Out Started | Start time of the current query.      | Timestam p       | 180s                          |
|                   | Completed         | End time of the current query.        | Timestam p       | 180s                          |
|                   | Details           | Details about the current query.      | Character string | 180s                          |
| INODE             | Inode Usage       | Disk inode usage.                     | 0% to 100%       | 30s                           |
| SCHEMA            | Schema Usage      | Database schema usage.                | 0% to 100%       | 3600s                         |

**Table 9-6** Restrictions on monitoring metrics

| Type     | Metric                    | ECS Cluster | BMS Cluster           |
|----------|---------------------------|-------------|-----------------------|
| Netwo rk | NIC nominal speed         | None        | 100/1000/10000/50000M |
|          | NIC working mode (duplex) | None        | full/half             |

## 9.2.3 Cluster Overview

### Cluster Overview

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** On the **Clusters** page, locate the target cluster.
- Step 3** In the **Operation** column of the target cluster, click **Monitoring Panel**. The database monitoring page is displayed.
- Step 4** In the navigation pane on the left, click **Cluster Overview**.

On the page that is displayed, you can view the cluster status, real-time resource consumption, top SQL statements, cluster resource consumption, and key database metrics.

----End

## Cluster Status

In the **Cluster Status** area, you can view the status of the current cluster and the number of available resources, including the numbers of nodes, CNs, and databases.

### Cluster Status



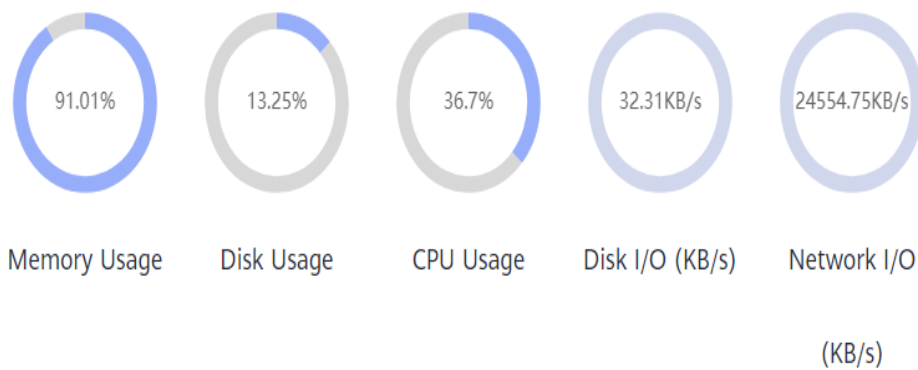
Normal

|           |                 |     |
|-----------|-----------------|-----|
| Nodes     | Available/Total | 3/3 |
| CNs       | Total           | 1   |
| Databases | Total           | 2   |

## Resource Consumption

In **Resource Consumption**, you can view the real-time resource consumption of the current cluster, including **Memory Usage**, **Disk Usage**, **CPU Usage**, **Disk I/O (KB/s)**, and **Network I/O (KB/s)**.

### Resource Consumption



## TOP SQL

In the **TOP SQL** area, you can query the SQL statements that take the longest time and have the largest amount of data flushed to disks in the current cluster.

Top 5 Time-Consuming Queries

| Query ID           | SQL Statement                                       | Execution Time (ms) |
|--------------------|---|---------------------|
| 150870587517664307 | with v1 as( select L.category_id,brand_s,store_n... | 2260                |
| 150870587517664311 | select c.last_name,c.first_name,c.salutation...     | 2067                |
| 150870587517664362 | select eseeeding.mk,fl,product_name,best_pe...      | 2009                |
| 150870587517664323 | select c.last_name,c.first_name,ca.city,boog...     | 1976                |
| 150870587517664350 | select ca.state,cd.gender,cd.marital_status,c...    | 48                  |

Top 5 Queries with Most Data Written to Disk

| Query ID           | SQL Statement                                       | Data Written to Disk (MB) |
|--------------------|---|---------------------------|
| 150870587517664307 | with v1 as( select L.category_id,brand_s,store_n... | 0                         |
| 150870587517664311 | select c.last_name,c.first_name,c.salutation...     | 0                         |
| 150870587517664323 | select c.last_name,c.first_name,ca.city,boog...     | 0                         |
| 150870587517664350 | select ca.state,cd.gender,cd.marital_status,c...    | 0                         |
| 150870587517664362 | select eseeeding.mk,fl,product_name,best_pe...      | 0                         |

**NOTE**

Top 5 SQL statement query is implemented by **max\_ctime**, which displays the query duration and the amount of data written to disks in the current collection period. If no query is executed during off-peak hours, the top 5 time-consuming query page will not be refreshed.

## Key Metrics

On the **Cluster Overview** page, you can also view the database metrics of the current cluster, including **Sessions** and **Queries**.



## 9.2.4 Monitoring

### 9.2.4.1 Node Monitoring

#### Node Monitoring

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** On the **Clusters** page, locate the target cluster.
- Step 3** In the **Operation** column of the target cluster, click **Monitoring Panel**. The database monitoring page is displayed.
- Step 4** In the navigation pane on the left, choose **Monitoring > Node Monitoring**.

On the page that is displayed, view the real-time consumption of nodes, memory, disks, disk I/O, and network I/O.

----End

## Overview

On the **Overview** tab page, you can view the key resources of a specified node based on the node name, including:

- Node Name
- CPU Usage (%)
- Memory Usage (%)
- Average Disk Usage (%)
- IP Address
- Disk I/O (KB/s)
- TCP Protocol Stack Retransmission Rate (%)
- Network I/O (KB/s)
- Status

| Node Name         | CPU Usage (%) | Memory Usage (%) | Average Disk Usage (%) | Service IP Address | Disk I/O (KB/s) | TCP Protocol Stack Retransmission Rate (%) | Network I/O (KB/s) | Status |
|-------------------|---------------|------------------|------------------------|--------------------|-----------------|--|--------------------|--------|
| host-172-16-3-135 | 61            | 92.38            | 25.43                  | 172.16.120.238     | 79212.3         | 0  | 17494.62           | Online |
| host-172-16-35-17 | 80.11         | 93.28            | 27.26                  | 172.16.64.121      | 34777.1         | 0  | 3241.75            | Online |
| host-172-16-8-33  | 36.46         | 92.38            | 24.65                  | 172.16.87.213      | 382.85          | 0  | 52.96              | Online |

## Disks

On the **Disks** tab page, view the real-time disk resource consumption of a node by node name and disk name, including:

- Node Name
- Disk Name
- Disk Capacity (GB)
- Disk Usage (%)
- Disk Read Rate (KB/s)
- Disk Write Rate (KB/s)
- I/O Wait Time (await, ms)
- I/O Service Time (svctm, ms)
- I/O Utility (util, %)

| Node Name         | Disk Name | Disk Capacity (GB) | Disk Usage (%) | Disk Read Rate (KB/s) | Disk Write Rate (KB/s) | I/O Wait Time (await, ms) | I/O Service Time (svctm, ms) | I/O Utility (util, %) |
|-------------------|-----------|--------------------|----------------|-----------------------|------------------------|---------------------------|------------------------------|-----------------------|
| host-172-16-3-135 | vda       | 39.12              | 14.59          | 3.23                  | 22.61                  | 0.22                      | 0.16                         | 3.2                   |
| host-172-16-3-135 | vdb       | 79.96              | 36.21          | 51264.6               | 189.8                  | 81.34                     | 1.88                         | 2019.1                |
| host-172-16-3-135 | vdc       | 99.95              | 1.63           | 4.84                  | 0                      | 0.2                       | 0.1                          | 0                     |
| host-172-16-3-135 | vdd       | 79.96              | 49.08          | 51449.7               | 125.49                 | 75.87                     | 1.42                         | 2019.1                |
| host-172-16-35-17 | vda       | 39.12              | 20.65          | 54.5                  | 39.27                  | 0.24                      | 0.09                         | 4.4                   |
| host-172-16-35-17 | vdb       | 79.96              | 36.59          | 48912.2               | 77.35                  | 23.42                     | 1.51                         | 1570.6                |
| host-172-16-35-17 | vdc       | 79.96              | 48.61          | 51230.5               | 139.98                 | 17.68                     | 1.17                         | 1801.5                |
| host-172-16-35-17 | vdd       | 99.95              | 2.87           | 869.74                | 2.9                    | 0.26                      | 0.01                         | 1                     |
| host-172-16-8-33  | vda       | 39.12              | 14.18          | 0                     | 5.71                   | 0.05                      | 0.05                         | 0.4                   |

 NOTE

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

Run the Linux **df** command to collect the disk capacity information, as shown in the following figure.

```
[Ruby@host-10-0-16-43 8_1_0]# df -x tmpfs -x devtmpfs
Filesystem      1K-blocks    Used  Available Use% Mounted on
/dev/sda4       569616888 5757444 540228616  2% /
/dev/sda2         999320  107584   822924  12% /boot
/dev/sda1        204580    8368   196212   5% /boot/efi
/dev/sdd        3513495364 390076 3513105288  1% /var/chroot/DWS/data1
/dev/sde        3513495364 274192 3513221172  1% /var/chroot/DWS/data2
/dev/sdb        3513495364 34224 3513461140  1% /var/chroot/DWS/data3
/dev/sdc        3513495364 34224 3513461140  1% /var/chroot/DWS/data4
[Ruby@host-10-0-16-43 8_1_0]#
```

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

- **Filesystem:** path name of the device file corresponding to the file system. Generally, it is a hard disk partition.
- **1K-blocks:** number of data blocks (1024 bytes) in a partition.
- **Used:** number of data blocks used by the disk.
- **Available:** number of available data blocks on the disk.
- **Use%:** percentage of the space used by common users. Even if the space is used up, the partition still reserves the space for system administrators.
- **Mounted on:** mount point of the file system.

## Network

On the **Network** tab page, view the real-time network resource consumption of a node by node name and NIC name, including:

- Node Name
- NIC Name
- NIC Status
- NIC Speed (Mbps)
- Received Packets
- Sent Packets
- Lost Packets Received
- Receive Rate (KB/s)
- Transmit Rate (KB/s)

Overview | Disks | Network

Please select

| Node Name         | NIC Name | NIC Status | NIC Speed (Mbps) | Received Packets | Lost Packets Received | Receive Rate (KB/s) | Transmit Rate (KB/s) |
|-------------------|----------|------------|------------------|------------------|-----------------------|---------------------|----------------------|
| host-172-16-8-33  | eth0     | up         | Unknown          | 7743702          | 0                     | 1.65                | 1.9                  |
| host-172-16-8-33  | eth1     | up         | Unknown          | 122              | 0                     | 0                   | 0                    |
| host-172-16-8-33  | eth2     | up         | Unknown          | 1872360815       | 0                     | 12.58               | 5.98                 |
| host-172-16-8-33  | eth3     | up         | Unknown          | 26202548         | 0                     | 26.35               | 3.33                 |
| host-172-16-35-17 | eth0     | up         | Unknown          | 5240496          | 0                     | 2.14                | 11.24                |
| host-172-16-35-17 | eth1     | up         | Unknown          | 74               | 0                     | 0                   | 0                    |
| host-172-16-35-17 | eth2     | up         | Unknown          | 1062758248       | 0                     | 9.67                | 10.51                |

## 9.2.4.2 Performance Monitoring

### Performance Monitoring

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** On the **Clusters** page, locate the target cluster.
- Step 3** In the **Operation** column of the target cluster, click **Monitoring Panel**.
- Step 4** In the navigation pane on the left, choose **Monitoring > Performance Monitoring**.

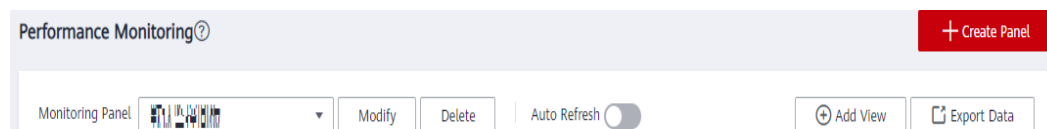
The **Performance Monitoring** page displays the resource consumption trends of clusters and databases.

----End

### Monitoring Panel

You can configure monitoring views by customizing monitoring panels. Monitoring views are bound to users. After logging in to the system, you can view the user-defined monitoring panels.

- Creating a monitoring panel: You can click **Create Panel** to customize a monitoring panel.
- Modifying a monitoring panel: You can click **Modify** to change the name of a monitoring panel.
- Deleting a monitoring panel: You can click **Delete** to delete a monitoring panel. The default monitoring panel cannot be deleted.

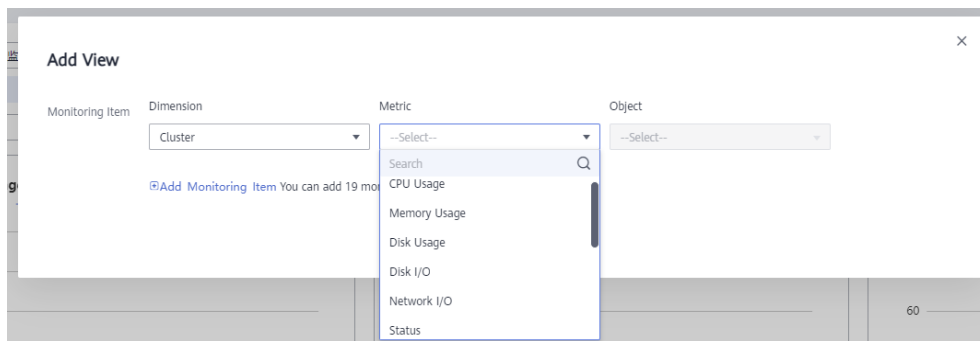


### Adding a Monitoring View

Currently, DMS provides two types of monitoring views: cluster and database. You can click **Add View** to add a monitoring view as required. The monitoring indicators are as follows:

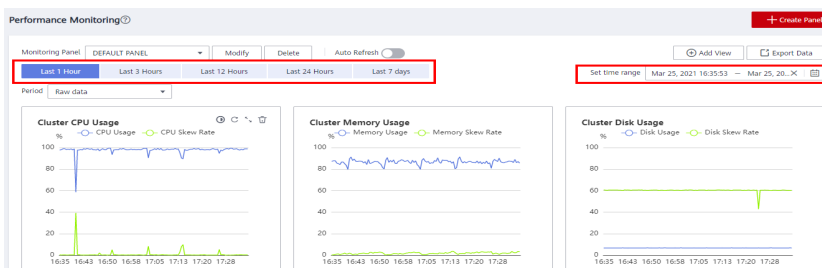
- Cluster: CPU Usage, Memory Usage, Disk Usage, Disk I/O, Network I/O, Status, Abnormal CNs, Read-only, Sessions, Queries, Deadlocks, Abnormal DN, CPU Usage of DN, TPS, and QPS
- Database: Length of the Request Waiting Queue, Sessions, Queries, Submitted Transactions, Rollback Transactions, Scanning Rows, Index Query Rows, Inserted Rows, Updated Rows, Deleted Rows, and Capacity, and TPS





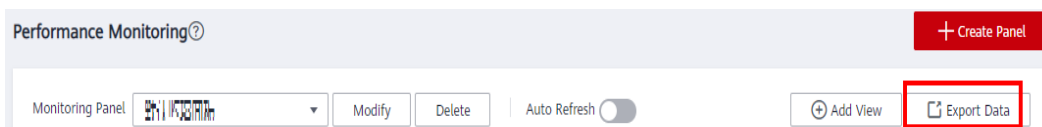
**NOTE**

- A maximum of 20 views can be added to each panel. Adding too many views will increase the number of page requests and the rendering time.
- Performance Monitoring allows you to view data trends in different time ranges in five modes, as shown in the following figure.



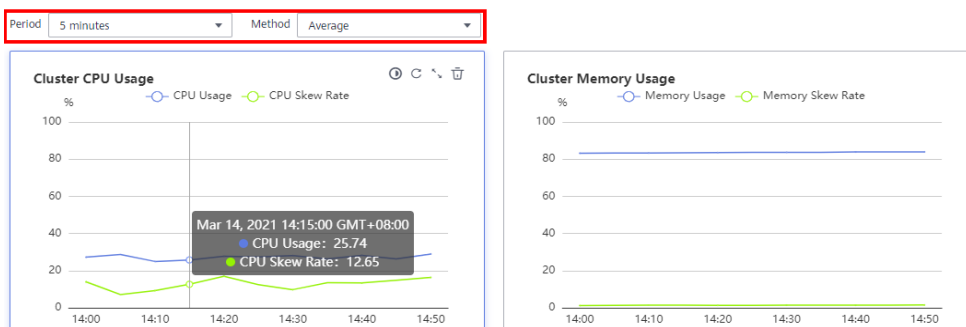
## Exporting Monitoring Data

Performance Monitoring supports data export. You can click **Export Data** to further process data. By default, data in all monitoring views on the current page is exported. The export time range is subject to the selected time range.



**NOTE**

Performance Monitoring allows data aggregation of different periods. You can aggregate raw data based on the corresponding sampling period to display indicator trends of a longer period.





### 9.2.4.3 Database Monitoring

#### Database Monitoring

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** On the **Clusters** page, locate the target cluster.
- Step 3** In the **Operation** column of the target cluster, click **Monitoring Panel**. The database monitoring page is displayed.
- Step 4** In the navigation pane on the left, choose **Monitoring > Database Monitoring**.

The **Database Monitoring** page displays the real-time and historical resource consumption a database.

----End


#### Database Resource Consumption

You can select a database and check its resource usage. For details, see [Monitoring Metrics](#), including:

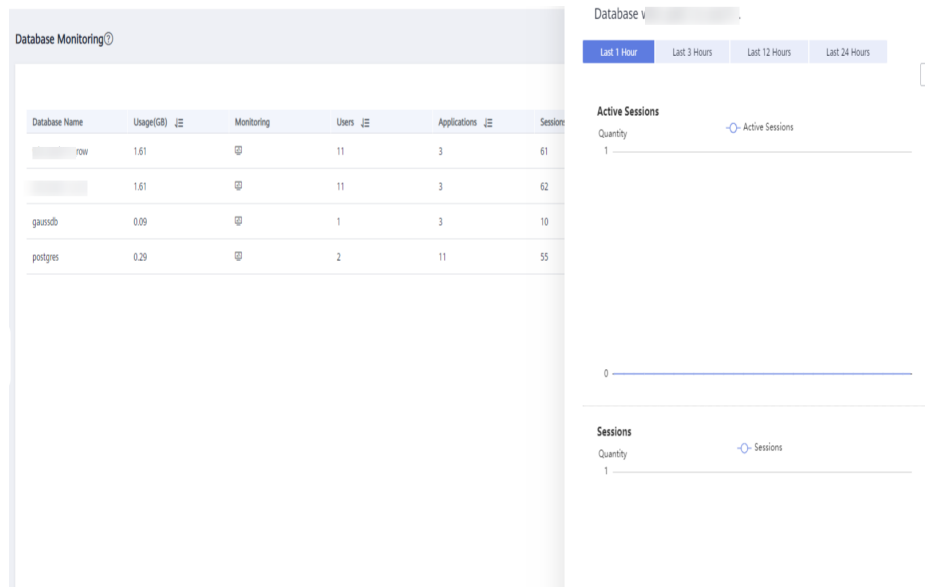
- Database Name
- Usage (GB)
- Monitoring
- Users
- Applications
- Sessions
- Queries
- Scanning Rows
- Index Query Rows
- Inserted Rows
- Updated Rows
- Deleted Rows
- Executed Transactions
- Transaction Rollbacks
- Deadlocks
- Logical Read Times
- Physical Read Times
- Temporary Files
- Temporary File Capacity

| Database Name   | Usage(GB) | Monitoring | Users | Applications | Sessions | Queries | Scanning Rows | Number Of Index Q... |
|-----------------|-----------|------------|-------|--------------|----------|---------|---------------|----------------------|
| wlm_tpch_1c_row | 1.61      | 🔍          | 11    | 3            | 61       | 0       | 78384962368   | 48197395             |
| wlm_tpch_1c_col | 1.61      | 🔍          | 11    | 3            | 62       | 0       | 78386191247   | 48358917             |
| gaussdb         | 0.09      | 🔍          | 1     | 3            | 10       | 0       | 4024111       | 802840               |
| postgres        | 0.29      | 🔍          | 2     | 11           | 55       | 18      | 94203308      | 32210660             |

## Database Trend Monitoring

In the **Monitoring** column of a database, click  to view the performance indicators of the database, including:

- Capacity
- Sessions
- Queries



### 9.2.4.4 Session Monitoring

#### Session Monitoring

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** On the **Clusters** page, locate the target cluster.
- Step 3** In the **Operation** column of the target cluster, click **Monitoring Panel**.
- Step 4** In the navigation pane on the left, choose **Monitoring > Session Monitoring**.

The **Session Monitoring** page displays the session-level real-time database query statistics. You can also select and terminate a session.

----End

#### Query Statistics

You can browse the query statistics of a specified session based on the session ID, including:

- Session ID
- User Name
- Database Name

- Session Duration (ms)
- Application Name
- Queries
- Last Query Duration (ms)
- Client IP Address
- Connected CN
- Session Status

Session Monitoring

Terminate a Session

Please select

Enter a keyword

| Session ID      | User Name | Database Name | Session Duration(ms) | Application Name   | Query Count | Last Query Duration... | Client IP Address | Connected CN | Session Status |
|-----------------|-----------|---------------|----------------------|--------------------|-------------|------------------------|-------------------|--------------|----------------|
| 14012135599936  | Ruby      | postgres      | 0                    | OM                 | 1           | 0                      | -                 | cn_5009      | Running        |
| 140121455085312 | Ruby      | postgres      | 28097                | cn_5009            | 1           | 0                      | 127.0.0.1         | cn_5009      | Idle           |
| 140121551570688 | Ruby      | postgres      | 28231                | cn_agent           | 1           | 0                      | -                 | cn_5009      | Idle           |
| 140121585133312 | Ruby      | postgres      | 28277                | cn_5009            | 1           | 0                      | 127.0.0.1         | cn_5009      | Idle           |
| 140121610303232 | Ruby      | postgres      | 28287                | cn_agent           | 1           | 0                      | -                 | cn_5009      | Idle           |
| 140121515915008 | Ruby      | postgres      | 1                    | OM                 | 1           | 0                      | -                 | cn_5009      | Running        |
| 14012167944576  | Ruby      | postgres      | 28288                | WLM-Meter          | 1           | 0                      | -                 | cn_5009      | Running        |
| 140121694725888 | Ruby      | postgres      | 28288                | WorkloadMonitor    | 1           | 0                      | -                 | cn_5009      | Running        |
| 140121728288512 | Ruby      | postgres      | 28288                | workload           | 1           | 0                      | -                 | cn_5009      | Running        |
| 140121711507200 | Ruby      | postgres      | 28288                | CalculateSpaceInfo | 1           | 0                      | -                 | cn_5009      | Running        |

10 Total Records: 11 < 1 2 >

## Terminating a Session

Select a session to be terminated, click **Terminate a Session**, and confirm your operation.

### NOTE

The fine-grained permission control function is added. Only users with the operate permission are able to terminate sessions. For users with the read-only permission, the **Terminate a Session** button is grayed out.

## 9.2.4.5 Query Monitoring

### Query Monitoring

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** On the **Clusters** page, locate the target cluster.
- Step 3** In the **Operation** column of the target cluster, click **Monitoring Panel**.
- Step 4** In the navigation pane on the left, choose **Monitoring > Query Monitoring**.

The **Query Monitoring** page displays the real-time information about all queries that are running in a cluster and the historical information about the queries that have been run.

----End

### Prerequisites

You need to set GUC parameters before viewing data on the monitoring page. If GUC parameters are not set, real-time or historical query may be unavailable. However, if this parameter is set, the cluster performance may deteriorate.

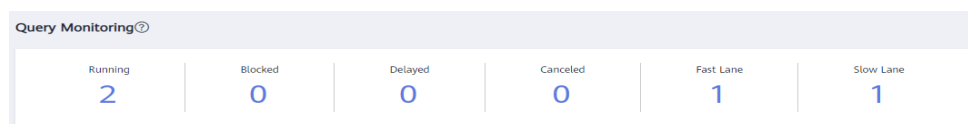
Therefore, you need to balance the settings of related parameters. The following table describes recommended settings. For details about how to modify parameters, see "Modifying Database Parameters". [Setting GUC Parameters](#) provides parameter details.

**Table 9-7** Recommended GUC parameter settings

| GUC Parameter             | CN Configuration | DN Configuration |
|---------------------------|------------------|------------------|
| max_active_statements     | 10               | 10               |
| enable_resource_track     | on               | on               |
| resource_track_level      | query            | query            |
| resource_track_cost       | 0                | 0                |
| resource_track_duration   | 0                | 0                |
| enable_resource_record    | on               | on               |
| session_statistics_memory | 1000MB           | 1000MB           |

## Querying Information

In this area, you can browse the number of queries in different status, including Running, Blocked, Delayed, Canceled, Fast Lane, and Slow Lane.



## Real-Time Query

In the **Real-Time Query** area, you can browse the real-time information about all running queries, including:

- Query ID
- User Name
- Application Name
- Database Name
- Workload Queue
- Submitted
- Blocking Time (ms)
- Execution Time (ms)
- CPU Time (ms)
- CPU Time Skew (%)
- Average Written Data (MB)

- Statement
- Connected CN
- Client IP Address
- Lane
- Query Status
- Session ID
- Queuing Status

| Query ID           | User Name | Database Name | Submitted                       | Execution Time (ms) | Statement                               | Lane | Query Status |
|--------------------|-----------|---------------|---------------------------------|---------------------|---|------|--------------|
| 150070587517664307 | pch_2     | test          | Mar 06, 2021 17:49:56 GMT+08:00 | 2260                | with vt as) select l.category, l.bra... | slow | active       |
| 150070587517664311 | pch_3     | test          | Mar 06, 2021 17:49:57 GMT+08:00 | 2067                | select c.last_name, c.first_name, c...  | fast | active       |
| 150070587517664323 | pch_2     | test          | Mar 06, 2021 17:49:57 GMT+08:00 | 1976                | select c.last_name, c.first_name, c...  | fast | active       |
| 150070587517664350 | pch_3     | test          | Mar 06, 2021 17:49:59 GMT+08:00 | 48                  | select ca.state, cd.gender, cd.mar...   | fast | active       |
| 150070587517664362 | pch_1     | test          | Mar 06, 2021 17:49:59 GMT+08:00 | 2009                | select exceeding_rnk, tl.product_no...  | fast | active       |

#### NOTE

Click the query ID to view the details. However, details cannot be displayed for queries whose ID is **0**. Query **0** indicates that an exception occurs during the query.

## Terminating a Query

Select a query to be terminated, click **Terminate Query**, and confirm your operation.

#### NOTE

The fine-grained permission control function is added. Only users with the operate permission are able to terminate queries. For users with the read-only permission, the **Terminate Query** button is grayed out.

## Historical Query

In the **History** area, you can browse all historical query information based on the specified time period, including:

- Query ID
- User Name
- Application Name
- Database Name
- Workload Queue
- Submitted
- Blocking Time (ms)
- Execution Time (ms)
- CPU Time (ms)
- CPU Time Skew (%)

- Average Written Data (MB)
- Statement
- Connected CN
- Client IP Address
- Query Status
- Completed
- Estimated Execution Time (ms)
- Cancellation Reason

| Query ID           | User Name | Database Name | Submitted                       | Execution Time (ms) | Statement                               | Query Status | Completed                       |
|--------------------|-----------|---------------|---------------------------------|---------------------|---|--------------|---------------------------------|
| 150870587517663870 | pch_2     | test          | Mar 06, 2021 17:49:25 GMT+08:00 | 1510                | select sum(is_net_profit)/sum(is_ex...  | Normal       | Mar 06, 2021 17:49:26 GMT+08:00 |
| 150870587517664161 | pch_2     | test          | Mar 06, 2021 17:49:51 GMT+08:00 | 241                 | select dt_d_year,item_category_id_...   | Normal       | Mar 06, 2021 17:49:51 GMT+08:00 |
| 150870587517663800 | pch_2     | test          | Mar 06, 2021 17:49:21 GMT+08:00 | 3194                | select ca_state,cf_gender,cf_marit...   | Normal       | Mar 06, 2021 17:49:25 GMT+08:00 |
| 150870587517664166 | pch_3     | test          | Mar 06, 2021 17:49:50 GMT+08:00 | 1136                | select i_item_id,i_item_desc,s_store... | Normal       | Mar 06, 2021 17:49:52 GMT+08:00 |
| 150870587517664037 | pch_3     | test          | Mar 06, 2021 17:49:38 GMT+08:00 | 2108                | select i_item_id,eng(c0_quantity) a...  | Normal       | Mar 06, 2021 17:49:40 GMT+08:00 |
| 150870587517663977 | pch_1     | test          | Mar 06, 2021 17:49:32 GMT+08:00 | 5611                | with liv as (select w_warehouse_na...   | Normal       | Mar 06, 2021 17:49:37 GMT+08:00 |
| 150870587517664366 | pch_1     | test          | Mar 06, 2021 17:50:00 GMT+08:00 | 1625                | select exceeding_mh,i1_i_product_na...  | Normal       | Mar 06, 2021 17:50:01 GMT+08:00 |
| 150870587517664285 | pch_3     | test          | Mar 06, 2021 17:49:56 GMT+08:00 | 537                 | with ss as ( select i_manufact_id,so... | Normal       | Mar 06, 2021 17:49:56 GMT+08:00 |
| 150870587517663844 | pch_1     | test          | Mar 06, 2021 17:49:22 GMT+08:00 | 1662                | select c_last_name,c_first_name,c_...   | Normal       | Mar 06, 2021 17:49:24 GMT+08:00 |
| 150870587517663892 | pch_2     | test          | Mar 06, 2021 17:49:26 GMT+08:00 | 3989                | with liv as (select w_warehouse_na...   | Normal       | Mar 06, 2021 17:49:30 GMT+08:00 |

## Viewing Query Monitoring Details

You can click a query ID to view the query details, including the basic information of query statements, real-time and historical resource consumption, SQL description, and query plan.

**Basic Information**

Username: tpc\_ex\_1      Access Node: cn5001      Blocking Time (ms): 0  
 Database: tpcds10\_row      Application Name: gsql      Execution Time: 178129  
 Scale-Out Started: Mar 11, 2021 14:10:15 GMT+08:00      Estimated Execution Time (ms): 389      Workload Queue: default\_pool  
 Status: active      Estimated Remaining Time (ms): 0

| CPU Time (ms) |       | Memory (MB) |       | Average Written Data (MB) |       | I/O (MB) |       | DN Execution Time (ms) |        |
|---------------|-------|-------------|-------|---------------------------|-------|----------|-------|------------------------|--------|
| Item          | Value | Item        | Value | Item                      | Value | Item     | Value | Item                   | Value  |
| Max           | 6717  | Max         | 4     | Max                       | 0     | Max      | 0     | Max                    | 177968 |
| Min           | 4505  | Min         | 4     | Min                       | 0     | Min      | 0     | Min                    | 177958 |
| Avg           | 5352  | Avg         | 4     | Avg                       | 0     | Avg      | 0     | Avg                    | 177963 |
| Skew (%)      | 20    | Skew (%)    | 0     | Skew (%)                  | 0     | Skew (%) | 0     | Skew (%)               | 0      |

**SQL**      Query Plan (Text)

```
select i_returnflag, i_linestatus, sum(i_quantity) as sum_qty, sum(i_extendedprice) as sum_base_price, sum(i_extendedprice * (1 - i_discount)) as sum_disc_price, sum(i_extendedprice * (1 - i_discount) * (1 + i_tax)) as sum_charge, arg(i_quantity) as arg_qty, arg(i_extendedprice) as arg_price, arg(i_discount) as arg_disc, count(*) as count_order from lineitem where i_shipdate <= date '1998-12-01' - interval '3 day' group by i_returnflag, i_linestatus order by i_returnflag, i_linestatus;
```

### 9.2.4.6 Instance Monitoring

#### Instance Monitoring

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** On the **Clusters** page, locate the target cluster.
- Step 3** In the **Operation** column of the target cluster, click **Monitoring Panel**.
- Step 4** In the navigation pane on the left, choose **Monitoring > Instance Monitoring**.

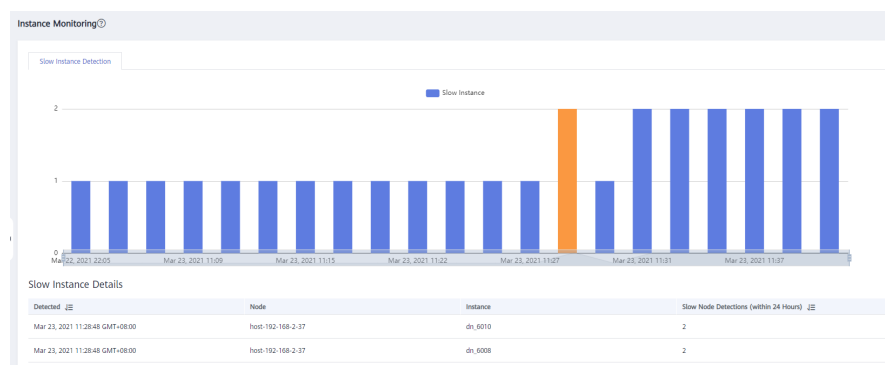
On the **Instance Monitoring** page, you can view the real-time and historical information about detected slow instances.

----End

## Slow Instance Detection

DMS can automatically configure and start the slow instance detection script on cluster CNs, periodically collect the cache table of the script, and report the detected slow instance data. You can view the number of slow instances detected within 24 hours and the distribution status in the time dimension on the GUI to quickly locate the slow nodes in the cluster and analyze the root causes.

The **Instance Monitoring** page consists of two parts. The upper part displays the time distribution chart of detected slow instances, that is, the number of slow instances detected in different detection periods. The lower part displays slow instance details. When you select any bar in the time distribution chart, details about the detection time, node name, instance name, and number of detections (within 24 hours) of slow instances are displayed.



### 9.2.4.7 Load Monitoring

#### Load Monitoring

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** On the **Clusters** page, locate the target cluster.
- Step 3** In the **Operation** column of the target cluster, click **Monitoring Panel**.
- Step 4** In the navigation pane on the left, choose **Monitoring > Load Monitoring**.

On the **Load Monitoring** page, you can view the real-time and historical resource consumption of workload queues.

----End

#### Workload Queues

The DMS displays the user-defined workload queue name, real-time and historical resource consumption, and workload queue resource quotas.

- **Workload Queues:** name of a workload queue

- **Monitoring:** You can click the monitoring icon to display the historical consumption trends of resources such as the CPU, memory, and disk.
- **CPU Usage (%):** real-time CPU usage of a workload queue
- **CPU Resource (%):** CPU usage quotas of a workload queue
- **Real-Time Concurrent Short Queries:** number of concurrent simple queries in a workload queue. Concurrent simple queries are not controlled by the workload queue.
- **Concurrent Short Queries:** simple concurrency quotas of the workload queue
- **Real-Time Concurrent Queries:** number of concurrent complex queries in a workload queue. Concurrent complex queries are controlled by the workload queue.
- **Query Concurrency:** complex concurrency quotas of the workload queue
- Operation

The screenshot shows a 'Real-Time' monitoring interface with a table of query execution details. The table has columns for Query ID, User Name, Database Name, Submitted, Execution Time (ms), Statement, Lane, and Query Status. There are also search and filter controls at the top of the table.

| Query ID           | User Name | Database Name | Submitted                       | Execution Time (ms) | Statement                                    | Lane | Query Status |
|--------------------|-----------|---------------|---------------------------------|---------------------|--|------|--------------|
| 150870587517664307 | pch_2     | test          | Mar 06, 2021 17:49:56 GMT+08:00 | 2260                | with v1 asj select l_category, l_bra...      | slow | active       |
| 150870587517664311 | pch_3     | test          | Mar 06, 2021 17:49:57 GMT+08:00 | 2067                | select c_last_name, c_first_name, c...       | fast | active       |
| 150870587517664323 | pch_2     | test          | Mar 06, 2021 17:49:57 GMT+08:00 | 1976                | select c_last_name, c_first_name, c...       | fast | active       |
| 150870587517664350 | pch_3     | test          | Mar 06, 2021 17:49:59 GMT+08:00 | 48                  | select ca_state, cd_gender, cd_mari...       | fast | active       |
| 150870587517664362 | pch_1     | test          | Mar 06, 2021 17:49:59 GMT+08:00 | 2009                | select ascending mkt, i1, i2, product, na... | fast | active       |

## Exception Handling Rules

You can click the drop-down list of any workload queue to view the exception handling rule configured for the workload queue.

- **Name:** type of the supported exception handling rule, including:
  - **blocktime:** query blocking time. The unit is seconds.
  - **elapsedtime:** execution duration of a query. The unit is seconds.
  - **allcputime:** total CPU time spent in executing a query on all DNs. The unit is seconds.
  - **cpuskewpercent:** CPU time skew of a query executed on DNs. The value depends on the setting of **qualificationtime**.
  - **qualificationtime:** interval for checking the CPU skew. The unit is seconds. This parameter must be set together with **cpuskewpercent**.
  - **spillsize:** amount of query data written to disks on DNs. The unit is MB.
  - **broadcastsize:** size of broadcast operators of a query on DNs. The unit is MB.
  - **mem\_limit:** maximum memory used by a query on a single instance. The unit is MB.
- **Type:** supported exception handling rule type. The options can be **abort** and **penalty**.
- **Value:** rule value, which ranges from 0 to `UINT_MAX`.



| Workload Queues | Monitoring | CPU Usage(%) | Memory Usage(%) | Disk Usage(%) | Concurrent Simple Queries | Concurrent Complex Queries | Operation     |
|-----------------|------------|--------------|-----------------|---------------|---------------------------|----------------------------|---------------|
| ^ pdh_queue_1   | ⊞          | 0            | 0               | 0             | 0                         | 0                          | Configuration |

| Name          | Type  | Value  |
|---------------|-------|--------|
| BlockTime     | Abort | 1200 s |
| ElapsedTime   | Abort | 2400 s |
| SpillSize     | Abort | 256 MB |
| BroadcastSize | Abort | 100 MB |

## Waiting Queries

You can view the waiting queries in a workload queue in real time to identify the service pressure.


- **User:** user name of a query statement
- **Application:** application name of a query statement
- **Database:** name of the database to which a query statement is connected
- **Queuing Status:** queuing status of a query statement in a workload queue
- **Wait Time:** waiting time before a query statement is executed, in ms
- **Workload Queue:** workload queue to which a query statement belongs
- **Query Statement:** details of a query statement submitted by a user

Load Monitoring 0

---

Workload Queues | **Waiting Queries** | Circuit Breaking Queries

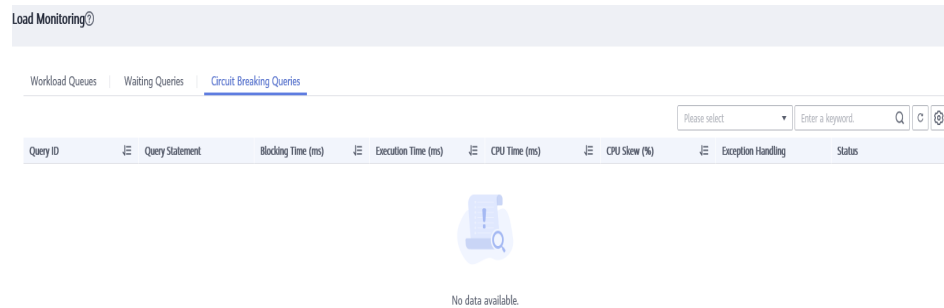
Please select Enter a keyword Q C ⊞

| User  | Application | Database | Statement Attribute | Wait Time (ms) | Workload Queue | Query Statement |
|---|-------------|----------|---------------------|----------------|----------------|-----------------|
| <br>No data available. |             |          |                     |                |                |                 |

## Circuit Breaking Queries

You can view the status of a triggered circuit breaking query in a workload queue.

- **Query ID:** ID of a circuit breaking query
- **Query Statement:** circuit breaking query statement
- **Blocking Time (ms):** blocking time of a circuit breaking statement, in ms
- **Execution Time (ms):** execution time of a circuit breaking statement, in ms
- **CPU Time (ms):** CPU time consumed by a circuit breaking statement, in ms
- **CPU Skew (%):** CPU skew of a circuit breaking statement on each DN
- **Exception Handling:** exception handling method of a circuit breaking statement
- **Status:** real-time status of a circuit breaking statement



## 9.2.5 Utilities

### 9.2.5.1 SQL Diagnosis

#### Prerequisites

To enable SQL diagnosis, enable monitoring on real-time and historical queries on the **Queries** and **History** tabs, respectively. For details, see .

#### Viewing SQL Diagnosis

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

**Step 2** On the **Clusters** page, locate the target cluster.

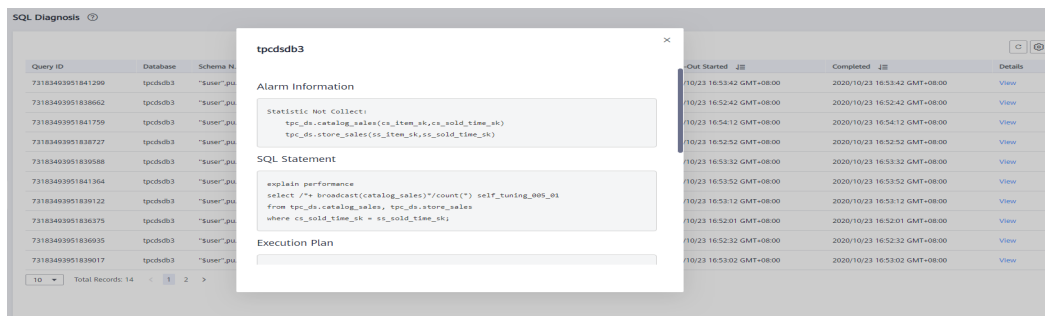
**Step 3** In the **Operation** column of the target cluster, click **Monitoring Panel**.

**Step 4** In the navigation pane on the left, choose **Utilities > SQL Diagnosis**. The metrics include:

- Query ID
- Database
- Schema Name
- User Name
- Client
- Client IP Address
- Running Time (ms)
- CPU Time (ms)
- Scale-Out Started
- Completed
- Details

**Step 5** On the **SQL Diagnosis** page, you can view the SQL diagnosis information. In the **Details** column of a specified query ID, click **View** to view the detailed SQL diagnosis result, including:

- Diagnosis Type
- Alarm Information
- SQL Statement
- Execution Plan



----End

## Setting GUC Parameters

GUC parameters related to SQL diagnosis are as follows. For details, see "GUC Parameters" in the *Data Warehouse Service (DWS) Developer Guide*.

- **enable\_resource\_track**
  - Value range: boolean
  - Default value: **on**
  - Expected DMS value: **on** (for reference only)
  - Function: Specifies whether to enable the real-time resource monitoring function.

### NOTICE

If this parameter is enabled without other GUC-related parameters correctly configured, real-time resource consumption cannot be recorded.

- **resource\_track\_cost**
  - Value range: an integer ranging from -1 to INT\_MAX
  - Default value: **100000**
  - Expected DMS value: **0** (for reference only)
  - Function: Specifies the minimum execution cost of statement resource monitoring for the current session. This parameter is valid only when **enable\_resource\_track** is **on**.

### NOTICE

If this parameter is set to a small value, more statements will be recorded, causing record expansion and affecting cluster performance.

- **resource\_track\_level**
  - Value range: enumerated type
  - Default value: **query**
  - Expected DMS value: **query** (for reference only)
  - Function: Specifies the resource monitoring level for the current session. This parameter is valid only when **enable\_resource\_track** is **on**.

---

**NOTICE**

If the resource monitoring is set to operator-level, performance will be greatly affected.

---

- **resource\_track\_duration**
    - Value range: an integer ranging from 0 to INT\_MAX, in seconds
    - Default value: **60**.
    - Expected DMS value: **0** (for reference only)
    - Function: Specifies the minimum statement execution time that determines whether information about jobs of a statement recorded in the real-time view will be dumped to a historical view after the statement is executed. That is, only statements whose execution time exceeds the specified time are recorded in the historical view. This parameter is valid only when **enable\_resource\_track** is **on**.
- 

**NOTICE**

If this parameter is set to a small value, the batch processing mechanism for dumping kernel statements becomes invalid, affecting the kernel performance.

---

- **topsql\_retention\_time**
    - Value range: an integer ranging from 0 to 3650, in days
    - Default value: **0**
    - Expected DMS value: **1** (for reference only)
    - Function: Specifies the aging time of **pgxc\_wlm\_session\_info** data in the view.
- 

**NOTICE**

If this parameter is set to **0**, data will not be aged, which will cause storage expansion.

---

- **enable\_resource\_record**
  - Value range: boolean
  - Default value: **off**
  - Expected DMS value: **on** (for reference only)
  - Function: Specifies whether to enable the archiving function for resource monitoring records. When this function is enabled, records in the history views (**GS\_WLM\_SESSION\_HISTORY** and **GS\_WLM\_OPERATOR\_HISTORY**) are archived to the info views (**GS\_WLM\_SESSION\_INFO** and **GS\_WLM\_OPERATOR\_INFO**) every 3 minutes. After the archiving, records in the history views are deleted.

**NOTICE**

When this parameter is enabled, you are advised to set **topsql\_retention\_time** properly to configure the aging time. Otherwise, data in the **GS\_WLM\_SESSION\_INFO** or **GS\_WLM\_OPERATOR\_INFO** table will expand.

## 9.2.6 Settings

The **Monitoring** page displays the collection period and data aging period of monitoring metrics.

**NOTE**

- The cluster monitoring function is enabled by default.
- Disable the function if the cluster is being recovered. Enable the function when the fault is rectified.
- When a node in the cluster is powered off or the management IP address of the cluster is unavailable, the cluster monitoring switch and the button for configuring cluster indicator collection are unavailable.

### Monitoring Collection

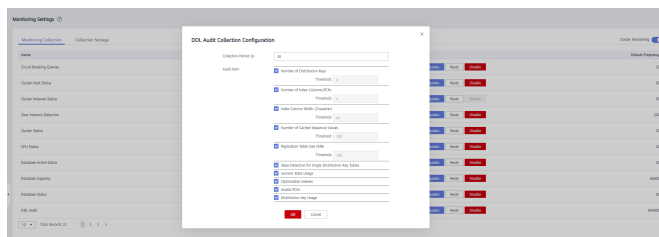
- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** On the **Clusters** page, locate the target cluster.
- Step 3** In the **Operation** column of the target cluster, choose **Monitoring Panel**. The database monitoring page is displayed.
- Step 4** In the navigation pane on the left, choose **Settings > Monitoring**. You can reconfigure the collection frequency or disable the collection of the monitoring item.

| Name                     | Description  | Collection Frequency (s) | Default Frequency |
|--------------------------|--|--------------------------|-------------------|
| Circuit Breaking Queries | Realtime status collection of triggered circuit breaking query in a... | 120                      | 120               |
| Cluster Host Status      | Cluster host status indicator collection                               | 60                       | 60                |
| Cluster Instance Status  | Cluster instance status indicator collection                           | 60                       | 60                |
| Slow Instance Detection  | Locating and status collection of slow instances in a cluster          | 240                      | 240               |
| Cluster Status           | Cluster status indicator collection                                    | 30                       | 30                |
| CN Availability          | Indicator collection of CN abnormal status                             | 60                       | 60                |
| CPU Status               | CPU status indicator collection  | 30                       | 30                |
| Database Active Status   | Database active status indicator collection                            | 30                       | 30                |
| Database Capacity        | Database capacity indicator collection                                 | 86400                    | 86400             |
| Database Status          | Database status indicator collection                                   | 60                       | 60                |

Total Records: 21

**NOTE**

Click **Update** of **DDL Audit** to reset the automatic audit frequency or audit items.



----End

## Collection Storage

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** On the **Clusters** page, locate the target cluster.
- Step 3** In the **Operation** column of the target cluster, choose **Monitoring Panel**. The database monitoring page is displayed.
- Step 4** In the navigation pane on the left, choose **Settings > Monitoring** and switch to the **Collection Storage** tab page. Update the retention days.

| Name                    | Description                                  | Retention Days |        |
|-------------------------|--|----------------|--------|
| Memory Status           | Memory status indicator collection           | 7              | Update |
| Disk Status             | Disk file system status indicator collection | 7              | Update |
| Disk I/O Status         | Disk I/O status indicator collection         | 7              | Update |
| Network I/O Status      | Network status indicator collection          | 7              | Update |
| Cluster Status          | Cluster status indicator collection          | 7              | Update |
| Cluster Host Status     | Cluster host status indicator collection     | 7              | Update |
| Cluster Instance Status | Cluster instance status indicator collection | 7              | Update |
| Database Status         | Database status indicator collection         | 7              | Update |
| Database Active Status  | Database active status indicator collection  | 7              | Update |
| Database Capacity       | Database capacity indicator collection       | 7              | Update |

10 Total Records: 19 < 1 2 >

----End

## 9.2.7 Typical Scenarios

### 9.2.7.1 SQL Diagnosis

#### Symptom

The execution of SQL statements takes a long time, resulting in great resource consumption.

#### Troubleshooting Process

If the execution efficiency of SQL statements is low, optimization suggestions are provided after the kernel executes the SQL statements. You can query the

execution history to retrieve optimization suggestions and further optimize SQL statements to improve query efficiency.

## Troubleshooting Procedure

- Step 1** On the **SQL Diagnosing** page, select a time period that does not seem right.
- Step 2** Search for SQL statements based on indicators such as the start time, end time, and running duration of the statement.
- Step 3** Click **Details** to view SQL optimization suggestions.
- Step 4** Optimize the SQL statement based on suggestions.

----End

### 9.2.7.2 Top Time-Consuming SQL Statements Viewing

#### Symptom

Time-consuming SQL statements exist.

#### Troubleshooting Process

On the **Top 5 Time-Consuming Queries** page directed from the **Cluster Overview** page, record the change of top 5 time-consuming queries.

Analyze the frequency of top 5 queries to locate slow queries.

## Troubleshooting Procedure

- Step 1** On the **Cluster Overview** page, click and view the **Top5 Time-Consuming Queries** page.
- Step 2** Find the IDs of time-consuming queries and query the pid field (session\_id) in the database view **PGXC\_WLM\_SESSION\_STATISTICS**.
- Step 3** On the **Session Monitoring** page, locate the session\_id and kill the time-consuming SQL statement.

----End

## 9.3 Event Notifications

### 9.3.1 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 9-8** 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.  |
| 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 | configureMRSExtDataSources | Normal         | Configuration of MRS external data source for the cluster started. |



| Event Type | Event Name                        | Event Severity | Event   |
|------------|-----------------------------------|----------------|---|
| Management | configureMRSExtDataSourcesFail    | Warning        | Failed to configure the MRS external data source for the cluster. |
| Management | configureMRSExtDataSourcesSuccess | Normal         | MRS external data source configured successfully for the cluster. |
| Management | deleteMRSExtDataSources           | Normal         | Deletion of MRS external data source for the cluster started.     |
| Management | deleteMRSExtDataSourcesFail       | Warning        | Failed to delete the MRS external data source for the cluster.    |
| Management | deletedMRSExtDataSourcesSuccess   | Normal         | MRS external data source deleted successfully for the cluster.    |
| Management | bindEipToCluster                  | Normal         | Bound an EIP to the cluster.                                      |
| Management | bindEipToClusterFail              | Warning        | Failed to bind an EIP to the cluster.                             |
| Management | unbindEipToCluster                | Normal         | Unbound an EIP from the cluster.                                  |
| Management | unbindEipToCluster-Fail           | Warning        | Failed to unbind an EIP from the cluster.                         |
| Management | refreshEipToCluster               | Normal         | Refreshed the cluster's EIP.                                      |
| Management | refreshEipToCluster-Fail          | 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.              |

| Event Type | Event Name                  | Event Severity | Event  |
|------------|-----------------------------|----------------|--|
| Security   | updateConfiguration-Fail    | Warning        | Failed to update security parameters of the cluster.     |
| Security   | updateConfiguration-Success | 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 9-9** 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. |

## 9.3.2 Subscribing to Event Notifications

After subscribing to GaussDB(DWS) event notification, you will receive notifications by text message, email, or application when management, monitoring, or security events occur in a specific cluster or snapshot.



### Creating a Subscription

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

- Step 2** In the navigation tree on the left, click **Event Management**.
- Step 3** On the **Event Management** page, choose **Subscription > Create Subscription**.
- Step 4** In the **Subscription Settings** area, set basic subscription information and event filtering.

The **Subscribed Event List** area displays the events filtered by the system based on the subscription settings.

**Table 9-10** Subscription parameters

| Parameter             | Description   |
|-----------------------|---|
| Notification          | <p>Enable or disable event subscription.</p> <p> indicates that event subscription is enabled.  indicates that event subscription is disabled. This function is enabled by default. After the function is disabled, the system stops sending notifications of subscribed events but does not delete the subscription.</p> |
| Subscription Name     | <p>Enter the name of a subscription.</p> <ul style="list-style-type: none"> <li>The name can contain letters (upper or lower case), digits, hyphens (-), and underscores (_) and must start with a letter or digit.</li> <li>The name must be between 1 and 256 characters in length.</li> </ul>  |
| Event Type            | Select the type of the event to be subscribed. Possible values are <b>Management, Monitoring, and Security</b> .  |
| Event Severity        | Select the alarm severity of the event. Possible values are <b>Normal</b> and <b>Warning</b> .  |
| Event Source Category | Select the event source category: cluster, snapshot,.   |

- Step 5** Select a message notification topic from the **Message Notification Topic** drop-down list.
- The selected topic must have granted GaussDB(DWS) the permission to publish messages to the topic.  
If GaussDB(DWS) has not been authorized to publish messages to the selected topic, go to the topic management page of the SMN console to configure topic authorization. For details, see **Topic Management > Configuring Topic Policies** in the *Simple Message Notification User Guide*. When configuring the topic policy, select **GaussDB(DWS)** for **Services that can publish messages to this topic**.
  - To create a topic, click **Create Topic**. The SMN console is displayed. For details, see **Topic Management > Creating a Topic** in the *Simple Message Notification User Guide*.

**Step 6** Click **OK** to complete the subscription.

----End

## Modifying the Subscription

**Step 1** On the **Event Management** page of the GaussDB(DWS) management console, click the **Subscription** tab.

**Step 2** In the **Operation** column of the row containing the specified subscription, click **Edit** to enter the **Edit Subscription** page.

**Step 3** On the **Edit Subscription** page, set the parameters to be modified. For details, see [Step 4](#) to [Step 6](#) in section "Creating a Subscription".

----End

## Deleting the Subscription

**Step 1** On the **Event Management** page of the GaussDB(DWS) management console, click the **Subscription** tab.

**Step 2** In the **Operation** column of the row containing the specified subscription, click **Delete**. The **Delete Subscription** dialog box is displayed.

**Step 3** Click **Yes** to delete the subscription.

----End


### 9.3.3 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

## 9.4 Alarms

## 9.4.1 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.

### 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** On the page that is displayed:

- **Existing Alarm Statistics**

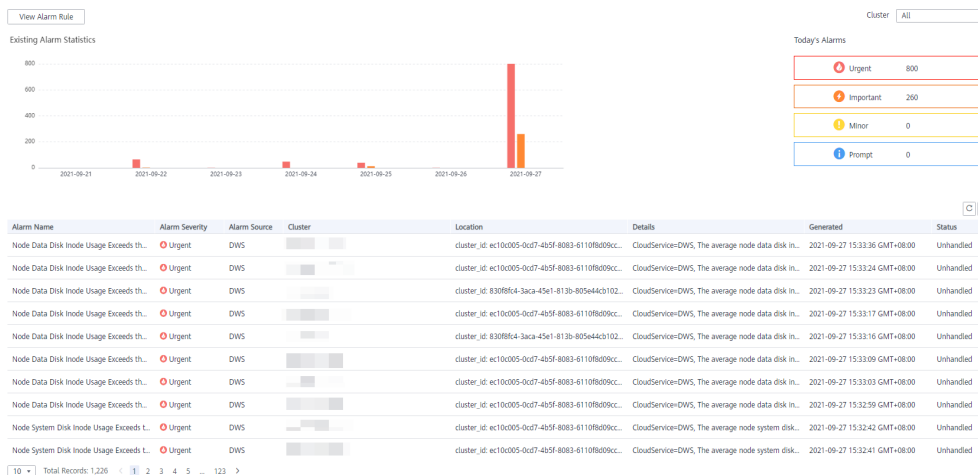
Statistics of the existing alarms in the past seven days are displayed by alarm severity in a bar chart. In this way, you can see clearly the number and category of the alarms generated in the past week.

- **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 9-11** Threshold alarms of DMS alarm sources

| Type    | Name   | Severity  | Description   |
|---------|--|---|---|
| Default | Node CPU Usage Exceeds the Threshold         | <b>Urgent</b>                                   | 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 System CPU Usage Exceeds the Threshold  | <b>Urgent</b>                                   | This alarm is generated if the threshold of system CPU 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 system CPU usage is lower than the threshold and the constraint is not met.                   |
| Default | Node Swap Usage Exceeds the Threshold        | <b>Urgent</b>                                   | This alarm is generated if the threshold of swap 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 swap usage is lower than the threshold and the constraint is not met.                               |
| Default | Node System Disk Usage Exceeds the Threshold | <b>Urgent:</b> > 85%;<br><b>Important:</b> >75% | This alarm is generated if the threshold of system disk (/) 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 system disk (/) usage is lower than the threshold and the constraint is not met.         |

| Type    | Name   | Severity  | Description   |
|---------|--|---|---|
| Default | Node Log Disk Usage Exceeds the Threshold        | <b>Urgent:</b> > 85%;<br><b>Important:</b> >75% | This alarm is generated if the threshold of log disk ( <code>/var/chroot/DWS/manager</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 log disk ( <code>/var/chroot/DWS/manager</code> ) usage is lower than the threshold and the constraint is not met.   |
| Default | Node Data Disk Usage Exceeds the Threshold       | <b>Urgent:</b> > 85%;<br><b>Important:</b> >75% | 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 System Disk I/O Usage Exceeds the Threshold | <b>Urgent</b>                                   | This alarm is generated if the threshold of system disk ( <code>/</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 system disk ( <code>/</code> ) I/O usage ( <code>util</code> ) is lower than the threshold and the constraint is not met.   |
| Default | Node Log Disk I/O Usage Exceeds the Threshold    | <b>Urgent</b>                                   | This alarm is generated if the threshold of log disk ( <code>/var/chroot/DWS/manager</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 log disk ( <code>/var/chroot/DWS/manager</code> ) I/O usage ( <code>util</code> ) is lower than the threshold and the constraint is not met.   |
| Default | Node Data Disk I/O Usage Exceeds the Threshold   | <b>Urgent</b>                                   | 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. |

| Type    | Name   | Severity  | Description   |
|---------|--|---|---|
| Default | Node System Disk Latency Exceeds the Threshold     | <b>Important</b>                                | This alarm is generated if the threshold of system disk (/) I/O latency ( <b>await</b> ) 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 system disk (/) I/O latency ( <b>await</b> ) is lower than the threshold and the constraint is not met.   |
| Default | Node Log Disk Latency Exceeds the Threshold        | <b>Important</b>                                | This alarm is generated if the threshold of log disk (/var/chroot/DWS/manager) I/O latency ( <b>await</b> ) 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 log disk (/var/chroot/DWS/manager) I/O latency ( <b>await</b> ) is lower than the threshold and the constraint is not met.   |
| Default | Node Data Disk Latency Exceeds the Threshold       | <b>Important</b>                                | This alarm is generated if the threshold of data disk (/var/chroot/DWS/data[n]) I/O latency ( <b>await</b> ) 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 (/var/chroot/DWS/data[n]) I/O latency ( <b>await</b> ) is lower than the threshold and the constraint is not met. |
| Default | Node System Disk Inode Usage Exceeds the Threshold | <b>Urgent:</b> > 85%;<br><b>Important:</b> >75% | This alarm is generated if the threshold of system disk (/) 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 system disk (/) inode usage is lower than the threshold and the constraint is not met.   |
| Default | Node Log Disk Inode Usage Exceeds the Threshold    | <b>Urgent:</b> > 85%;<br><b>Important:</b> >75% | This alarm is generated if the threshold of log disk (/var/chroot/DWS/manager) 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 log disk (/var/chroot/DWS/manager) inode usage is lower than the threshold and the constraint is not met.                                     |



| Type    | Name   | Severity  | Description   |
|---------|--|---|---|
| Default | Node Data Disk Inode Usage Exceeds the Threshold                   | <b>Urgent:</b> > 85%;<br><b>Important:</b> >75% | 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 | <b>Urgent</b>                                   | 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           | <b>Urgent</b>                                   | 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.   |
| Custom  | <i>Name of the user-defined threshold alarm</i>                    | <i>User-defined alarm severity</i>              | <i>Alarm description</i>  |

## 9.4.2 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 apply. When an alarm rule is enabled, it is added to the check list of the alarm engine and can be triggered normally. Disabled rules are not in the check list.
  - Delete: deletes an alarm rule. You can delete only user-defined rules. Default alarm rules cannot be deleted.

## 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             |
|--|-------------|--------------------|-----------|---|-----------------------|
| 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 exce...     | Modify Disable Delete |
| Node System CPU Usage Exceeds the Threshold      | Enable      | All                | Default   | This alarm is generated if the threshold of system CPU usage of any node in the cluster is exceeded with...     | Modify Disable Delete |
| Node Swap Usage Exceeds the Threshold            | Enable      | All                | Default   | This alarm is generated if the threshold of swap usage of any node in the cluster is exceeded within the ...    | Modify Disable Delete |
| Node System Disk Usage Exceeds the Threshold     | Enable      | All                | Default   | This alarm is generated if the threshold of system disk (I) usage of any node in the cluster is exceeded w...   | Modify Disable Delete |
| Node Log Disk Usage Exceeds the Threshold        | Enable      | All                | Default   | This alarm is generated if the threshold of log disk (/var/chronos/DWS/manager) usage of any node in th...      | Modify Disable Delete |
| Node Data Disk Usage Exceeds the Threshold       | Enable      | All                | Default   | This alarm is generated if the threshold of data disk (/var/chronos/DWS/data[1]) usage of any node in th...     | Modify Disable Delete |
| Node System Disk I/O Usage Exceeds the Threshold | Enable      | All                | Default   | This alarm is generated if the threshold of system disk (I) I/O usage (util) of any node in the cluster is e... | Modify Disable Delete |
| Node Log Disk I/O Usage Exceeds the Threshold    | Enable      | All                | Default   | This alarm is generated if the threshold of log disk (/var/chronos/DWS/manager) I/O usage (util) of any n...    | 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 (/var/chronos/DWS/data[1]) I/O usage (util) of any n...   | Modify Disable Delete |
| Node System Disk Latency Exceeds the Threshold   | Enable      | All                | Default   | This alarm is generated if the threshold of system disk (I) I/O latency (await) of any node in the cluster i... | 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**
- **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 % | Immediate  | 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**.

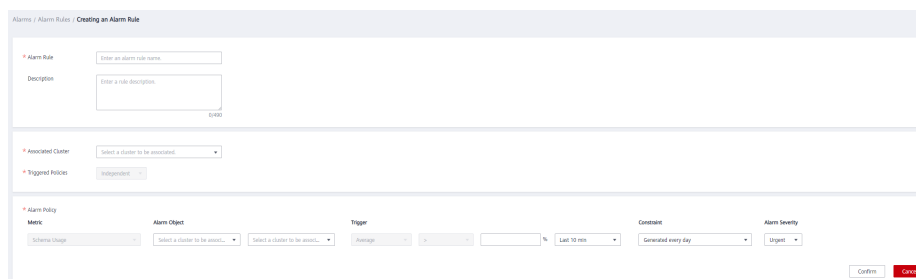
**Step 3** Click **View Alarm Rule** in the upper left corner.

**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**
- **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 9-1 Creating an alarm rule



**NOTE**

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

----End

## 9.4.3 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 9-12** Subscription parameters

| Parameter         | Description  |
|-------------------|--|
| Status            | Whether to enable the alarm subscription.<br><br> 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"> <li>Contains only letters, digits, hyphens (-), and underscores (_), and must start with a letter or digit.</li> <li>Contains 1 to 256 characters.</li> </ul>   |
| Alarm Severity    | Severity of the alarm you want to subscribe to: <b>Urgent, Important, Minor, or Prompt</b>   |

**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, .

Subscribed Alarms

| Alarms   | Alarm Severity |
|--|----------------|
| Node Swap Usage Exceeds the Threshold              | Urgent         |
| Node System Disk Inode Usage Exceeds the Threshold | Important      |
| Node Log Disk I/O Usage Exceeds the Threshold      | Urgent         |
| Node System Disk Usage Exceeds the Threshold       | Important      |
| Node Log Disk Inode Usage Exceeds the Threshold    | Important      |
| Node System Disk I/O Usage Exceeds the Threshold   | Urgent         |
| Remaining Database Disk Capacity Is Insufficient   | Urgent         |
| Node System Disk Latency Exceeds the Threshold     | Important      |
| Node Data Disk I/O Usage Exceeds the Threshold     | Urgent         |
| Node Data Disk Usage Exceeds the Threshold         | Urgent         |

10 Total Records: 28 < 1 2 3 >

\* SMN Topic  [Create Topic](#)

**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**.

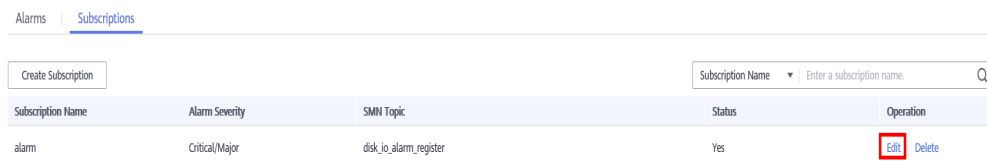
----End

## Modifying a Subscription

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

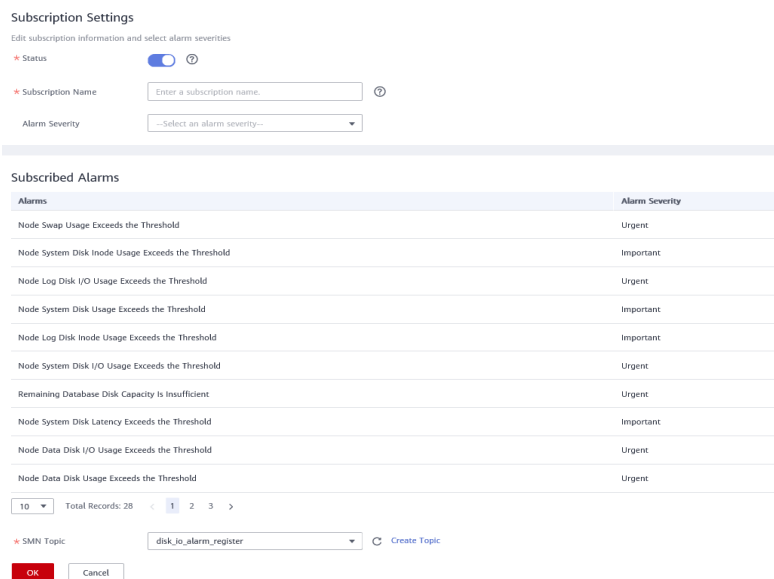
**Step 2** In the navigation pane on the left, choose **Alarms** and click the **Subscriptions** tab.

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



| Subscription Name | Alarm Severity | SMN Topic              | Status | Operation          |
|-------------------|----------------|------------------------|--------|--------------------|
| alarm             | Critical/Major | disk_io_alarm_register | Yes    | <b>Edit</b> Delete |

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



**Subscription Settings**  
Edit subscription information and select alarm severities

\* Status  ⓘ

\* Subscription Name  ⓘ

Alarm Severity

**Subscribed Alarms**

| Alarms   | Alarm Severity |
|--|----------------|
| Node Swap Usage Exceeds the Threshold              | Urgent         |
| Node System Disk Inode Usage Exceeds the Threshold | Important      |
| Node Log Disk I/O Usage Exceeds the Threshold      | Urgent         |
| Node System Disk Usage Exceeds the Threshold       | Important      |
| Node Log Disk Inode Usage Exceeds the Threshold    | Important      |
| Node System Disk I/O Usage Exceeds the Threshold   | Urgent         |
| Remaining Database Disk Capacity Is Insufficient   | Urgent         |
| Node System Disk Latency Exceeds the Threshold     | Important      |
| Node Data Disk I/O Usage Exceeds the Threshold     | Urgent         |
| Node Data Disk Usage Exceeds the Threshold         | Urgent         |

10 Total Records: 28 < 1 2 3 >

\* SMN Topic  ⓘ Create Topic

**OK** Cancel

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

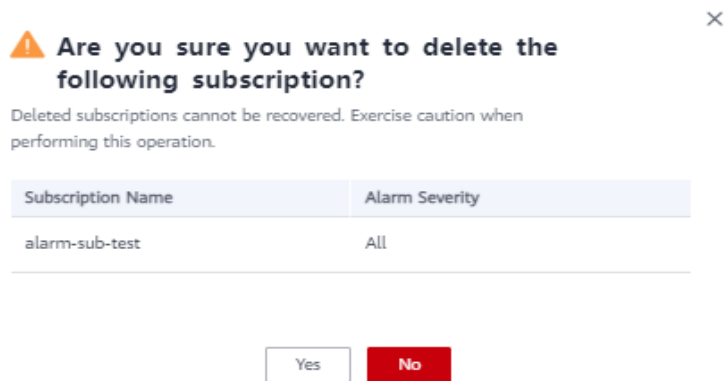
----End

## 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

# 10 Cluster Security Management

---

## 10.1 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 have 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** or **Unbalanced**.
- The value of **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, click **Clusters**.



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

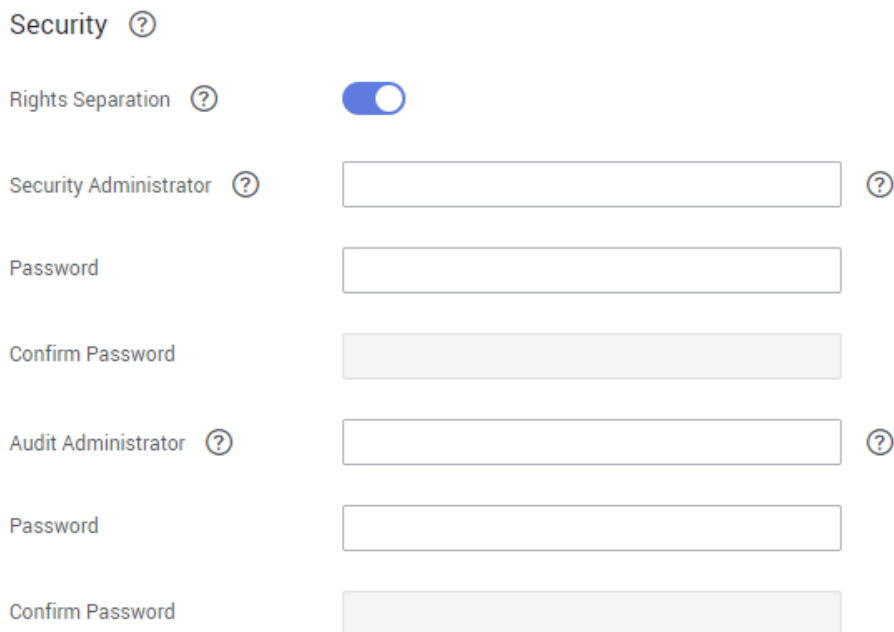
By default, **Configuration Status** is **Synchronized**, which indicates that the latest database result is displayed.

**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.

 indicates that **Rights Separation** is disabled. **Rights Separation** is disabled by default.

**Figure 10-1** Security configuration



Security ?

Rights Separation ?

Security Administrator ?  ?

Password

Confirm Password

Audit Administrator ?  ?

Password

Confirm Password

**Table 10-1** Security parameters

| Parameter              | Description  | Example Value  |
|------------------------|--|----------------|
| Security Administrator | <p>The username must meet the following requirements:</p> <ul style="list-style-type: none"> <li>• Consists of lowercase letters, digits, or underscores.</li> <li>• Starts with a lowercase letter or an underscore.</li> <li>• Contains 6 to 64 characters.</li> <li>• Cannot be a keyword of the GaussDB(DWS) database. For details about the keywords of the GaussDB(DWS) database, see "SQL Syntax Reference &gt; Keyword" in the <i>Data Warehouse Service (DWS) Developer Guide</i>.</li> </ul> | security_admin |
| Password               | <p>The password complexity requirements are as follows:</p> <ul style="list-style-type: none"> <li>• Contains 8 to 32 characters.</li> <li>• Cannot be the username or the username spelled backwards.</li> <li>• Must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters (~! '? , ; - _ ' " ( ) { } [ ] / &lt; &gt; @ # % ^ &amp; * +   \ =)</li> <li>• Passes the weak password check.</li> </ul>                         | -              |
| Confirm Password       | Enter the password of the security administrator again.  | -              |
| Audit Administrator    | <p>The username must meet the following requirements:</p> <ul style="list-style-type: none"> <li>• Consists of lowercase letters, digits, or underscores.</li> <li>• Starts with a lowercase letter or an underscore.</li> <li>• Contains 6 to 64 characters.</li> <li>• Cannot be a keyword of the GaussDB(DWS) database. For details about the keywords of the GaussDB(DWS) database, see "SQL Syntax Reference &gt; Keyword" in the <i>Data Warehouse Service (DWS) Developer Guide</i>.</li> </ul> | audit_admin    |

| Parameter        | Description   | Example Value |
|------------------|---|---------------|
| Password         | The password complexity requirements are as follows: <ul style="list-style-type: none"><li>• Contains 8 to 32 characters.</li><li>• Cannot be the username or the username spelled backwards.</li><li>• Must contain at least 3 of the following character types: uppercase letters, lowercase letters, digits, and special characters ~!@#%^&amp;*()-_+= [[{}];:,&lt;.&gt;/?</li><li>• Passes the weak password check.</li></ul> | -             |
| 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

## 10.2 Encrypting Databases

### 10.2.1 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 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.
- 

## 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.

## 10.2.2 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, click **Clusters**.
- Step 3** In the cluster list, find the target cluster and click the cluster name. The **Basic 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

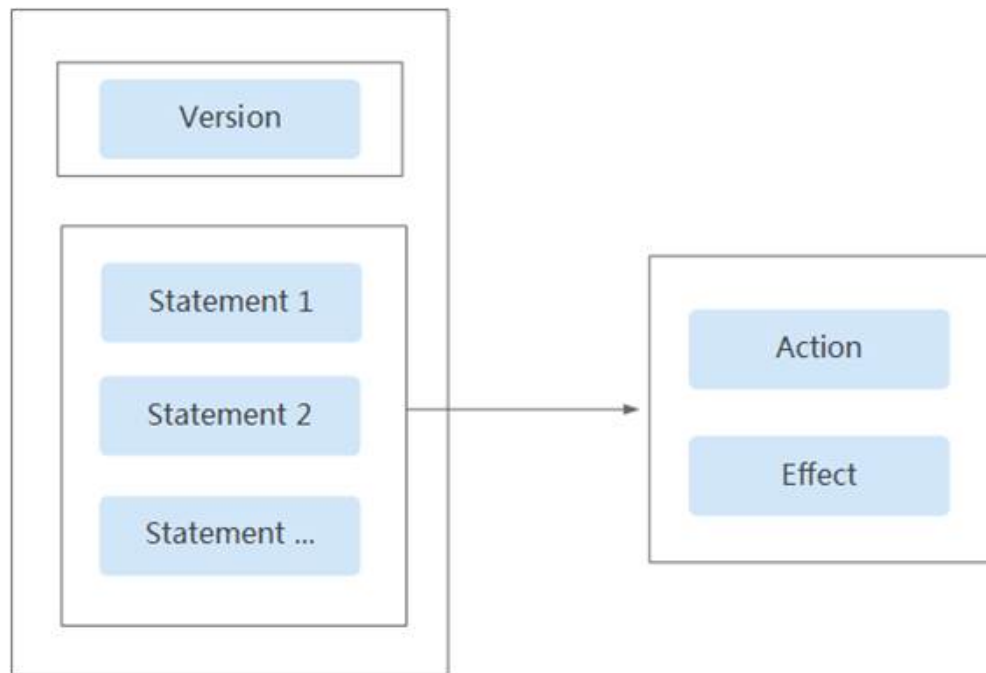
## 10.3 Permissions

## 10.3.1 Syntax of Fine-Grained Permissions Policies

### Policy Structure

A fine-grained policy consists of a Version and a Statement. Each policy can have multiple statements.

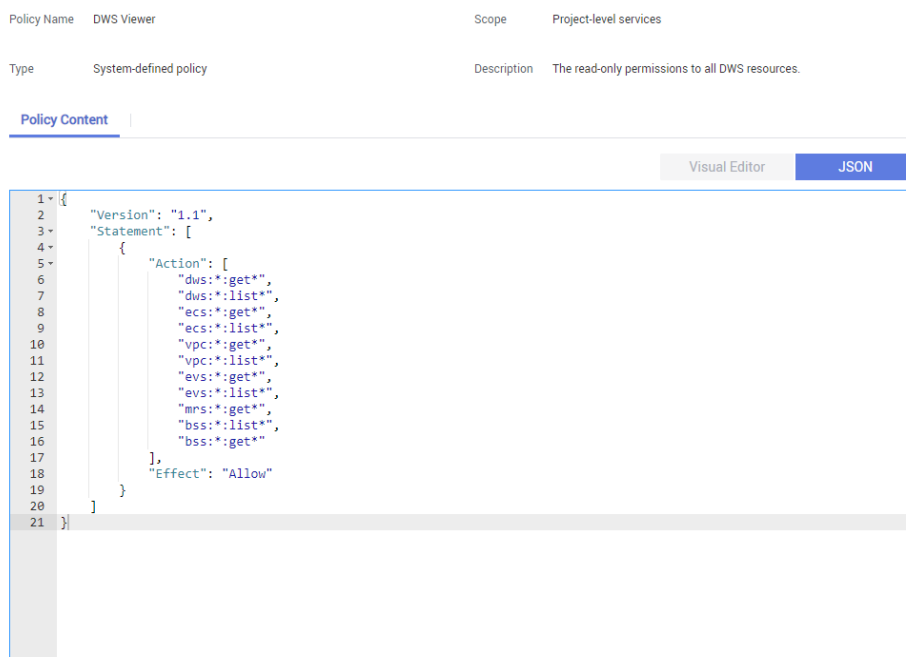
**Figure 10-2** Policy structure



### 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 10-3** 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*",
        "mrs:*:get*",
        "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.

## Multi-Action Policy

A custom policy can contain actions of multiple services that are all of the global or project-level type. The following is a policy with multiple statements:

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:cloudServers:resize",
        "ecs:cloudServers:delete",
        "ecs:cloudServers:rebuild"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "dws:*:get*",
        "dws:*:list*",
        "dws:cluster:create"
      ]
    }
  ]
}
```

## Deny Policy

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

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

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "dws:cluster:delete"
      ]
    }
  ]
}
```

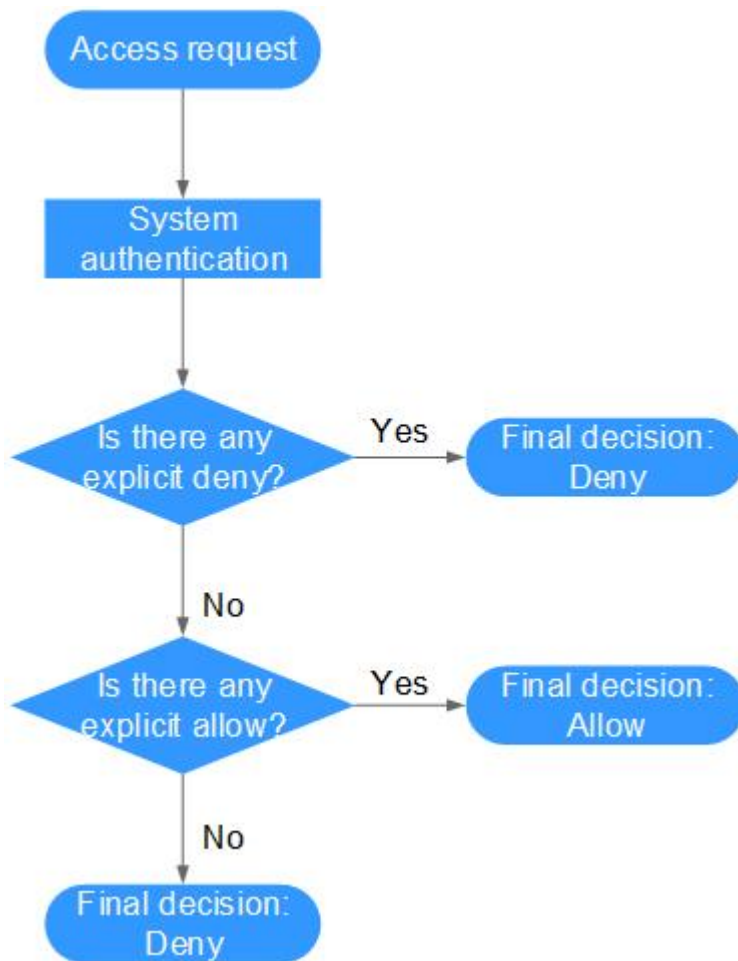


```
}  
  ]  
}
```

## 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 10-4** 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.

## 10.3.2 RBAC Syntax of RBAC Policies


### Policy Structure

An RBAC policy consists of a Version, a Statement, and Depends.

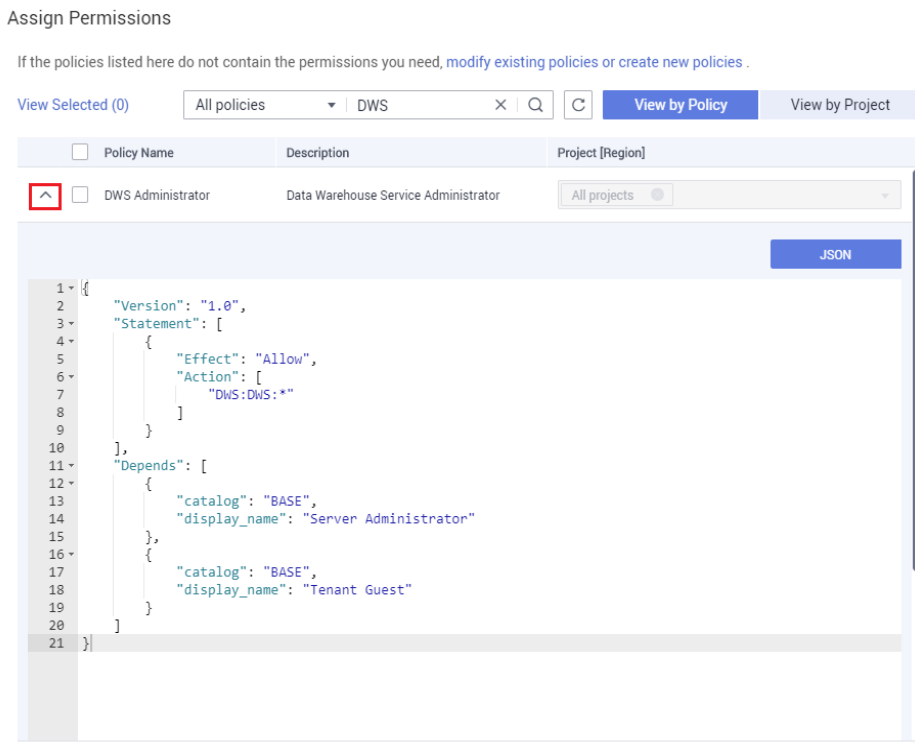
Figure 10-5 RBAC policy structure



### Policy Syntax

When selecting a policy for a user group, click  below the policy to view the details of the policy. The **DWS Administrator** policy is used as an example to describe the syntax of RBAC policies.

**Figure 10-6** Syntax of RBAC Policies



```

{
  "Version": "1.0",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dws:dws:*"
      ]
    }
  ],
  "Depends": [
    {
      "catalog": "BASE",
      "display_name": "Server Administrator"
    },
    {
      "catalog": "BASE",
      "display_name": "Tenant Guest"
    }
  ]
}
    
```

| Parameter |        | Meaning                                    | Value  |
|-----------|--------|--|--|
| Version   |        | Policy version                             | The value is fixed to <b>1.0</b> .   |
| Statement | Action | Operations to be performed on GaussDB(DWS) | Format: <i>Service name:Resource type:Operation</i> .<br><br><b>dws:dws:*</b> : Permissions for performing all operations on all resource types in GaussDB(DWS). |

| Parameter |              | Meaning  | Value  |
|-----------|--------------|--|--|
|           | Effect       | Whether the operation defined in an action is allowed        | <ul style="list-style-type: none"><li>• Allow</li><li>• Deny</li></ul> |
| Depends   | catalog      | Name of the service to which dependencies of a policy belong | Service name<br>Example: <b>BASE</b>                                   |
|           | display_name | Name of a dependent policy                                   | Policy name<br>Example: <b>Server Administrator</b>                    |

#### NOTE

When using RBAC for authentication, pay attention to the **Depends** parameter and grant other dependent permissions at the same time.

For example, the **DWS Administrator** permission depends on the **Server Administrator** and **Tenant Guest** permissions. When granting the **DWS Administrator** permission to users, you also need to grant the two dependent permissions to the users.

## 10.4 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.

### GaussDB(DWS) Permissions in Fine-Grained Policies

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.

- **RESTful APIs**  
For details about GaussDB(DWS) REST API actions, see "Permissions Policies and Supported Actions" in the *Data Warehouse Service (DWS) API Reference*.
- **Management console operations**  
[Table 10-2](#) describes the GaussDB(DWS) operations on resources and corresponding permissions.

**Table 10-2** GaussDB(DWS) permissions

| Operation  | Permission                         | Dependent Permission   | Scope  |
|--|------------------------------------|--|--|
| Creating/<br>Restoring<br>clusters                     | "dws:cluster:create"               | "dws:*:get*",<br>"dws:*:list*",<br>"ecs:*:get*",<br>"ecs:*:list*",<br>"ecs:*:create*",<br>"vpc:*:get*",<br>"vpc:*:list*",<br>"vpc:*:create*",<br>"evs:*:get*",<br>"evs:*:list*",<br>"evs:*:create*", | <ul style="list-style-type: none"> <li>Includes:                             <ul style="list-style-type: none"> <li>– Project</li> </ul> </li> </ul> |
| Obtaining the<br>cluster list                          | "dws:cluster:list"                 | "dws:*:get*",<br>"dws:*:list*",  | <ul style="list-style-type: none"> <li>Includes:                             <ul style="list-style-type: none"> <li>– Project</li> </ul> </li> </ul> |
| Obtaining the<br>details of a<br>cluster               | "dws:cluster:getDetail"            | "dws:*:get*",<br>"dws:*:list*",  | <ul style="list-style-type: none"> <li>Includes:                             <ul style="list-style-type: none"> <li>– Project</li> </ul> </li> </ul> |
| Setting<br>automated<br>snapshot policy                | "dws:cluster:setAutomatedSnapshot" | "dws:*:get*",<br>"dws:*:list*",  | <ul style="list-style-type: none"> <li>Includes:                             <ul style="list-style-type: none"> <li>– Project</li> </ul> </li> </ul> |
| Setting security<br>parameters/<br>parameter<br>groups | "dws:cluster:setSecuritySettings"  | "dws:*:get*",<br>"dws:*:list*",  | <ul style="list-style-type: none"> <li>Includes:                             <ul style="list-style-type: none"> <li>– Project</li> </ul> </li> </ul> |
| Restarting<br>clusters                                 | "dws:cluster:restart"              | "dws:*:get*",<br>"dws:*:list*",  | <ul style="list-style-type: none"> <li>Includes:                             <ul style="list-style-type: none"> <li>– Project</li> </ul> </li> </ul> |
| Scaling out<br>clusters                                | "dws:cluster:scaleOut"             | "dws:*:get*",<br>"dws:*:list*",<br>"ecs:*:get*",<br>"ecs:*:list*",<br>"ecs:*:create*",<br>"vpc:*:get*",<br>"vpc:*:list*",<br>"vpc:*:create*",<br>"evs:*:get*",<br>"evs:*:list*",<br>"evs:*:create*", | <ul style="list-style-type: none"> <li>Includes:                             <ul style="list-style-type: none"> <li>– Project</li> </ul> </li> </ul> |

| Operation                                | Permission                         | Dependent Permission   | Scope  |
|--|------------------------------------|--|--|
| Resetting passwords                      | "dws:cluster:resetPassword"        | "dws:*.get*",<br>"dws:*.list*",  | <ul style="list-style-type: none"> <li>Includes: <ul style="list-style-type: none"> <li>Project</li> </ul> </li> </ul> |
| Applying parameter templates to clusters | "dws:cluster:changeParameterGroup" | "dws:*.get*",<br>"dws:*.list*",  | <ul style="list-style-type: none"> <li>Includes: <ul style="list-style-type: none"> <li>Project</li> </ul> </li> </ul> |
| Deleting clusters                        | "dws:cluster:delete"               | "dws:*.get*",<br>"dws:*.list*",<br>"ecs:*.get*",<br>"ecs:*.list*",<br>"ecs:*.delete*",<br>"vpc:*.get*",<br>"vpc:*.list*",<br>"vpc:*.delete*",<br>"evs:*.get*",<br>"evs:*.list*",<br>"evs:*.delete*", | <ul style="list-style-type: none"> <li>Includes: <ul style="list-style-type: none"> <li>Project</li> </ul> </li> </ul> |
| Configuring maintenance windows          | "dws:cluster:setMaintenanceWindow" | "dws:*.get*",<br>"dws:*.list*",  | <ul style="list-style-type: none"> <li>Includes: <ul style="list-style-type: none"> <li>Project</li> </ul> </li> </ul> |
| Binding EIPs                             | "dws:eip:operate"                  | "dws:*.get*",<br>"dws:*.list*",  | <ul style="list-style-type: none"> <li>Includes: <ul style="list-style-type: none"> <li>Project</li> </ul> </li> </ul> |
| Unbinding EIPs                           | "dws:eip:operate"                  | "dws:*.get*",<br>"dws:*.list*",  | <ul style="list-style-type: none"> <li>Includes: <ul style="list-style-type: none"> <li>Project</li> </ul> </li> </ul> |
| Creating DNS domain names                | "dws:dns:create"                   | "dws:*.get*",<br>"dws:*.list*",  | <ul style="list-style-type: none"> <li>Includes: <ul style="list-style-type: none"> <li>Project</li> </ul> </li> </ul> |
| Releasing DNS domain names               | "dws:dns:release"                  | "dws:*.get*",<br>"dws:*.list*",  | <ul style="list-style-type: none"> <li>Includes: <ul style="list-style-type: none"> <li>Project</li> </ul> </li> </ul> |
| Modifying DNS domain names               | "dws:dns:edit"                     | "dws:*.get*",<br>"dws:*.list*",  | <ul style="list-style-type: none"> <li>Includes: <ul style="list-style-type: none"> <li>Project</li> </ul> </li> </ul> |
| Creating MRS connections                 | "dws:MRSCONNECTION:create"         | "dws:*.get*",<br>"dws:*.list*",  | <ul style="list-style-type: none"> <li>Includes: <ul style="list-style-type: none"> <li>Project</li> </ul> </li> </ul> |
| Updating MRS connections                 | "dws:MRSCONNECTION:update"         | "dws:*.get*",<br>"dws:*.list*",  | <ul style="list-style-type: none"> <li>Includes: <ul style="list-style-type: none"> <li>Project</li> </ul> </li> </ul> |
| Deleting MRS connections                 | "dws:MRSCONNECTION:delete"         | "dws:*.get*",<br>"dws:*.list*",  | <ul style="list-style-type: none"> <li>Includes: <ul style="list-style-type: none"> <li>Project</li> </ul> </li> </ul> |

| Operation                    | Permission                  | Dependent Permission           | Scope                    |
|------------------------------|-----------------------------|--------------------------------|--------------------------|
| Adding/Deleting tags         | "dws:tag:addAndDelete"      | "dws:*.get*",<br>"dws:*.list*" | ● Includes:<br>– Project |
| Editing tags                 | "dws:tag:edit"              | "dws:*.get*",<br>"dws:*.list*" | ● Includes:<br>– Project |
| Creating snapshots           | "dws:snapshot:create"       | "dws:*.get*",<br>"dws:*.list*" | ● Includes:<br>– Project |
| Obtaining the snapshot list  | "dws:snapshot:list"         | "dws:*.get*"                   | ● Includes:<br>– Project |
| Deleting snapshots           | "dws:snapshot:delete"       | "dws:snapshot:list"            | ● Includes:<br>– Project |
| Copying snapshots            | "dws:snapshot:copy"         | "dws:snapshot:list"            | ● Includes:<br>– Project |
| Creating parameter templates | "dws:parameterGroup:create" | "dws:*.get*",<br>"dws:*.list*" | ● Includes:<br>– Project |
| Deleting parameter templates | "dws:parameterGroup:delete" | "dws:*.get*",<br>"dws:*.list*" | ● Includes:<br>– Project |
| Changing parameter templates | "dws:parameterGroup:edit"   | "dws:*.get*",<br>"dws:*.list*" | ● Includes:<br>– Project |

## Authorization Using the Fine-Grained Permission Policy

**Step 1** Log in to the IAM console and create a custom policy.

For details, see "User Guide > Fine-grained Policy Management > Creating Custom Policies" 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 [GaussDB\(DWS\) Permissions in Fine-Grained Policies](#) 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.

Policy example:

- Example in which multiple operation permissions are supported

For example, the following policy has the permissions to create/restore/restart/delete a cluster, set security parameters, and reset passwords.

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dws:cluster:create",
        "dws:cluster:restart",
        "dws:cluster:delete",
        "dws:cluster:setSecuritySettings",
        "dws:cluster:resetPassword",
        "ecs:*:get*",
        "ecs:.list*",
        "ecs:*:create*",
        "ecs:*:delete*",
        "vpc:*:get*",
        "vpc:.list*",
        "vpc:*:create*",
        "vpc:*:delete*",
        "evs:*:get*",
        "evs:.list*",
        "evs:*:create*",
        "evs:*:delete*"
      ]
    }
  ]
}
```

- Example of wildcard (\*) usage

For example, the following policy has all operation permissions on GaussDB(DWS) snapshots.

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dws:snapshot:*",
        "ecs:*:get*",
        "ecs:.list*",
        "vpc:*:get*",
        "vpc:.list*"
      ]
    }
  ]
}
```



```
}  
  ]  
}
```

**Step 2** Create a user group.

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

----End

# 11 Audit Logs

---

## 11.1 Overview

- Tenant database audit logs:

GaussDB(DWS) allows you to record the audit logs of specific operations, involving audit log retention policies, unauthorized access, as well as DML, DDL, **SELECT** and **COPY** operations performed on the stored procedures and database objects.

After configuring audit logs, you can query audit information to troubleshoot, or historical operation records for a malfunctioning GaussDB(DWS) cluster.

For details about how to view the database audit logs, see "Database Security Management > Querying Audit Results" in the *Data Warehouse Service (DWS) Developer Guide*.

---

### NOTICE

- You can configure database audit logs on the **Security Settings** page of the cluster. For details, see [Configuring the Database Audit Logs](#).
- Enabling audit logs will affect performance. The degree of impact depends on the number of audit items enabled.

- 
- Management console audit logs:

GaussDB(DWS) allows you to record key operation events of the management console, such as creating a cluster, creating a snapshot, and restarting a cluster. The logs can be used in common application scenarios such as security analysis, compliance audit, resource tracing, and fault locating.

For details about how to view the audit information on the management console, see [Viewing Audit Logs of Key Operations on the Management Console](#).

## 11.2 Viewing Audit Logs of Key Operations on the Management Console

This section is organized as follows:

- [Enabling CTS](#)
- [Disabling the Audit Log Function](#)
- [Key Operations](#)
- [Viewing Traces](#)

### 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** In the navigation tree on the left, click **Trackers**.

**Step 3** 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 "Tracker Management > 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.

**Table 11-1** 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        | growCluster          |
| Restarting a cluster         | cluster        | rebootCluster        |
| Creating a snapshot          | backup         | createBackup         |
| Deleting a snapshot          | backup         | deleteBackup         |
| Setting security parameters  | configurations | updateConfigurations |
| Creating an MRS data source  | dataSource     | createExtDataSource  |
| Deleting an MRS data source  | dataSource     | deleteExtDataSource  |
| Updating an MRS data source  | dataSource     | updateExtDataSource  |

## 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.

**Figure 11-1** Viewing a trace

```
{
  "service_type": "DWS",
  "user": {
    "name": " ",
    "id": "740d2bc5ad464cec856eeb589c82be07",
    "domain": {
      "name": "dt_user_dws1",
      "id": " "
    }
  },
  "time": "2019-05-15 15:56:17 GMT+08:00",
  "code": 400,
  "resource_type": "cluster",
  "resource_name": "tenant",
  "resource_id": " ",
  "source_ip": " ",
  "trace_name": "createCluster",
  "trace_type": "ConsoleAction",
  "api_version": "v1.0",
  "record_time": "2019-05-15 15:56:17 GMT+08:00",
  "trace_id": "eb48c040-76e6-11e9-8f5d-286ed48919a8",
  "trace_status": "normal"
}
```

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

## 11.3 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** Click **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 results are displayed.

**Step 4** In the **Audit Settings** area, configure the audit log retention policy.

[Table 11-2](#) describes the detailed information.

**Table 11-2** Audit log retention policy

| Parameter                      | Description   |
|--------------------------------|---|
| Audit Log Retention Policy     | Specifies the audit log retention policy. Possible values are: <ul style="list-style-type: none"><li>• <b>Space priority</b>: Audit logs will be automatically deleted if the size of audit logs on a single node exceeds 1 GB.</li><li>• <b>Time priority</b>: Audit logs will be retained within the minimum retention period. After this period expires, audit logs will be automatically deleted if the size of audit logs on a single node exceeds 1 GB.</li></ul> <b>Space priority</b> is preferred. |
| Minimum Retention Period (day) | This parameter is valid when <b>Audit Log Retention Policy</b> is set to <b>Time priority</b> .<br>The value ranges from 0 to 730 days. The default value is 90 days.   |

**Step 5** Enable the audit function for the following operations if necessary.

[Table 11-3](#) describes the detailed information about the audit items.

**Table 11-3** Audit items

| Parameter                        | Description  |
|----------------------------------|--|
| Audit Unauthorized Access        | Specifies whether to record unauthorized operations. This parameter is disabled by default.  |
| Audit DML Execution              | Specifies whether to record <b>INSERT</b> , <b>UPDATE</b> , and <b>DELETE</b> operations on tables. This parameter is disabled by default.   |
| Audit SELECT Execution           | Specifies whether to record the <b>SELECT</b> operation. This parameter is disabled by default.  |
| Audit Stored Procedure Execution | Specifies whether to record operations when executing the stored procedure or user-defined functions. This parameter is disabled by default. |
| Audit COPY Execution             | Specifies whether to record the <b>COPY</b> operation. This parameter is disabled by default.  |

| Parameter           | Description   |
|---------------------|---|
| Audit DDL Execution | Specifies whether to record the <b>CREATE</b> , <b>DROP</b> , and <b>ALTER</b> operations of specified database objects. <b>DATABASE</b> , <b>SCHEMA</b> , and <b>USER</b> are selected by default. |

Except the audit items listed in [Table 11-3](#), key audit items in [Table 11-4](#) are enabled by default on GaussDB(DWS).

**Table 11-4** 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 <b>SET</b> operation.   |

**Step 6** Enable or disable audit log dumps.

For more information, see [Enabling Audit Log Dumps](#).

**Step 7** 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

## 11.4 Dumping the Database Audit Logs

GaussDB(DWS) records information (audit logs) about connections and user activities in your database. With the information, you can monitor the database to ensure security and facilitate fault troubleshooting and historical operation record locating. These audit logs are stored in the database by default. You can also dump them to OBS so that users who are responsible for monitoring the database can view the logs more conveniently.

You can perform the following operations on the GaussDB(DWS) console:

- [Enabling Audit Log Dumps](#)

- [Modifying Audit Log Dump Configurations](#)
- [Viewing Audit Log Dumps](#)
- [Disabling Audit Log Dumps](#)

## Enabling Audit Log Dumps

After a data warehouse cluster is created, you can enable audit log dump for it to dump audit logs to OBS.

Before enabling audit log dump, ensure that the following conditions are met:

- You have created an OBS bucket for storing the audit logs. For details, see "Console Operation Guide > Managing Buckets > Creating a Bucket" in the *Object Storage Service User Guide*.



The procedure is as follows:

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

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

**Step 3** In the cluster list, click the name of the cluster for which you want to enable audit log dump. On the page that is displayed, click **Security Settings**.

**Step 4** In the **Audit Settings** area, enable **Audit Log Dump**.

 indicates that the function is enabled.  indicates that the function is disabled.

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 Huawei Cloud accounts or users with **Security Administrator** permissions can create agencies. IAM users under an account do not have the permission for creating agencies by default. Contact a user with the permission and complete the authorization on the current page.

- **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 ranges from 5 to 43200. The unit is minute.

**Step 5** Click **Apply**.

If **Configuration Status** is **Applying**, the system is saving the settings.



Wait for a moment and then refresh **Configuration Status**. When **Configuration Status** is **Synchronized**, the configuration is saved and takes effect.

----End

## Modifying Audit Log Dump Configurations

After audit log dump is enabled, you can modify the dump configurations, for example, modifying the OBS bucket, path, and dump interval.

The procedure is as follows:

- Step 1** Log in to the GaussDB(DWS) management console.
- Step 2** In the navigation tree on the left, click **Clusters**.
- Step 3** In the cluster list, click the name of the cluster for which you want to modify the audit log dump configurations. On the page that is displayed, click **Security Settings**.
- 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.

Wait for a moment and then refresh **Configuration Status**. When **Configuration Status** is **Synchronized**, the configuration is saved and takes 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 tree on the left, click **Clusters**.
- Step 3** In the cluster list, click the name of the target cluster. On the page that is displayed, click **Security Settings**.
- 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 11-5** Log file fields

| Name | Description                   |
|------|-------------------------------|
| time | Indicates the operation time. |

| Name            | Description   |
|-----------------|---|
| type            | Indicates the operation type.                               |
| result          | Indicates the operation result.                             |
| username        | Indicates the name of the user who initiates the operation. |
| database        | Indicates the database name.                                |
| client_conninfo | Indicates the client connection information.                |
| object_name     | Indicates the operation object name.                        |
| detail_info     | Indicates the detailed information about the operation.     |
| node_name       | Indicates the node name.                                    |
| thread_id       | Indicates the thread ID.                                    |
| local_port      | Indicates the local port.                                   |
| remote_port     | Indicates the remote port.                                  |

----End

## Disabling Audit Log Dumps

You can disable audit log dumps 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 tree on the left, click **Clusters**.
- Step 3** In the cluster list, click the name of the cluster for which you want to disable audit log dump. On the page that is displayed, click **Security Settings**.
- Step 4** In the **Audit Settings** area, disable audit log dump.



indicates that the function is disabled.

- Step 5** Click **Apply**.

If **Configuration Status** is **Applying**, the system is saving the settings.

Wait for a moment and then refresh **Configuration Status**. When **Configuration Status** is **Synchronized**, the configuration is saved and takes effect.

----End

# 12 FAQs

---

## 12.1 General Problems

### 12.1.1 Why Are Data Warehouses Necessary?

#### Status Quo and Requirements

Much data (orders, stocks, materials, and payments) is generated in the business operation systems and background (transactional) database of enterprises.

Decision makers categorize and analyze the data for business decision-making.

#### Difficulties

Data categorization and analysis involve the concurrent access to the data in multiple database tables. That is, multiple tables being updated by different transactions may be locked at the same time, which may cause complications to the database systems during peak hours.

- Locking multiple tables increases the latency of complex query.
- The transactions that are updating the database tables are blocked, causing delays or interruptions.

#### Solution

Data warehouses excel in data aggregation and association, so users mine more data, get more information, and make better decisions. The mining requires complex queries that involve data on multiple tables.

The ETL process copies data in business operation databases to data warehouses for analysis and computing. Data can be aggregated from multiple business operation systems into one data warehouse for better association, analysis, and actionable insights.

Data warehouses and standard transaction-oriented databases such as Oracle, Microsoft SQL Server, and MySQL use different design modes. Data warehouses

are optimized in terms of data aggregation and association but the transaction or data adding and deleting functions or performance may not be guaranteed. Therefore, data warehouses and databases apply to different scenarios. Transactional databases are dedicated to transaction processing (business operation of enterprises) whereas data warehouses excel at complex data analysis. In conclusion, databases apply to data updates whereas data warehouses apply to data analysis.

## 12.1.2 What Are the Differences Between a Data Warehouse and the Hadoop Big Data Platform?

The Hadoop big data platform can be regarded as a next-generation data warehousing system. It has the characteristics of modern data warehouses and is widely used by enterprises. Because of the scalability of MPP, the MPP-based data warehousing system is sometimes classified as a big data platform.

However, data warehouses greatly differ from the Hadoop platform in function and user experience in different scenarios. For details, see the following table.

**Table 12-1** Feature comparison between data warehouses and the Hadoop big data platform

| Feature                     | Hadoop   | Data Warehouse             |
|-----------------------------|--|----------------------------|
| Number of compute nodes     | 1000s  | Max 256                    |
| Data volume                 | Over 10 PB   | Max 10 PB                  |
| Data type                   | Relational, semi-relational, unstructured (voice, images, and video) | Relational only            |
| Latency                     | Medium to high   | Low                        |
| Application ecosystem       | Innovative/AI  | Traditional/BI             |
| Application development API | SQL and other programming language APIs, such as MapReduce           | Standard database SQL      |
| Scalability                 | Unlimited, with comprehensive programming APIs                       | Limited, supported by UDFs |
| Transaction support         | Limited  | Comprehensive              |

Data warehouses and the Hadoop platform work together in different scenarios. GaussDB(DWS) can seamlessly integrate with Hadoop-based MRS to provide the SQL-over-Hadoop data sharing across platforms and services. GaussDB(DWS) serves as a data warehouse for managing massive data while relishing the openness, convenience, and innovation of the Hadoop platform. You can also

enjoy the upper-layer applications of conventional data warehouses, especially BI applications, using GaussDB(DWS).

### 12.1.3 Why Should I Use GaussDB(DWS)?

Conventional data warehouses are not practical for smaller enterprises due to high cost, time-consuming device and system selection and procurement, and complex scale-out.

GaussDB(DWS) is the better choice:

- This cloud service of distributed MPP data warehousing is very open, efficient, compatible, scalable, and is easy to O&M.
- Developed on the FusionInsight LibrA data warehouse kernel, it empowers enterprises with better and consistent experience on and off the cloud.

FusionInsight LibrA is a next-generation distributed data warehousing system with independent intellectual property rights. Currently, it is widely used in government, finance, and carriers. FusionInsight LibrA is compatible with mainstream open-source Postgres databases, especially in Oracle and Teradata SQL statements. Our data warehouse engineers have designed a kernel of hybrid row-column stores not only for faster analysis but also for data processing, such as adding, deleting, modifying data. FusionInsight LibrA features the cost optimizer and warehouse technologies, including machine code vector computing and inter/intra-parallelism for operators and nodes. It uses LLVM to optimize the local code in compilation query plans. More powerful data query and analysis addresses service pain points and improves user experience.

- Application is out of the box and takes just a few minutes so you no longer need to look for a warehouse to . Simpler procurement and lower cost and barriers give smaller enterprises access to more value from their data for enterprise development and insights.

### 12.1.4 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 data sets through custom code. It allows you to control cluster configurations and software installed in the cluster.

GaussDB(DWS) works better with complex queries of a large amount of structured data. It aims to pool data from different sources together, such as inventory, finance, and retail system. To ensure consistency and accuracy of enterprise reports, GaussDB(DWS) stores data in a highly structured manner. This structure can directly build the data consistency rule to the database table. Additionally, GaussDB(DWS) is highly compatible with standard SQL statements and the syntax of conventional transaction-supported databases.

GaussDB(DWS) is preferred when you want to perform complex query of a large amount of structured data with high performance.

### 12.1.5 What Are the Differences Between GaussDB(DWS) and Hive in Functions?

GaussDB(DWS) and Hive have different functions in the following aspects:

1. Hive is a data warehouse based on Hadoop MapReduce. GaussDB(DWS) is a data warehouse based on Postgres MPP.
2. Hive data is stored on HDFS. GaussDB(DWS) data can be stored locally or on OBS in foreign table form.
3. Hive does not support indexes. GaussDB(DWS) supports indexes, so querying is faster.
4. Hive does not support stored procedures. GaussDB(DWS) does, so it has more extensive application scenarios.
5. Hive supports fewer SQL statements than GaussDB(DWS), including functions, customized functions, and stored procedures.
6. Hive does not support transactions. GaussDB(DWS) supports complete transactions.
7. Both Hive and GaussDB(DWS) support backups, so the reliability is the same.
8. GaussDB(DWS) delivers much better performance than Hive.

Based on their respective functions, Hive is useful for offline analysis while GaussDB(DWS) is useful for both online analysis and ad-hoc query.

## 12.1.6 What Is the User Quota?

For HUAWEI CLOUD services, quotas limit the number of resources available to users. If you need more, submit a service ticket to increase your quotas. Once approved, we will update your resource quota accordingly and send you a notification.

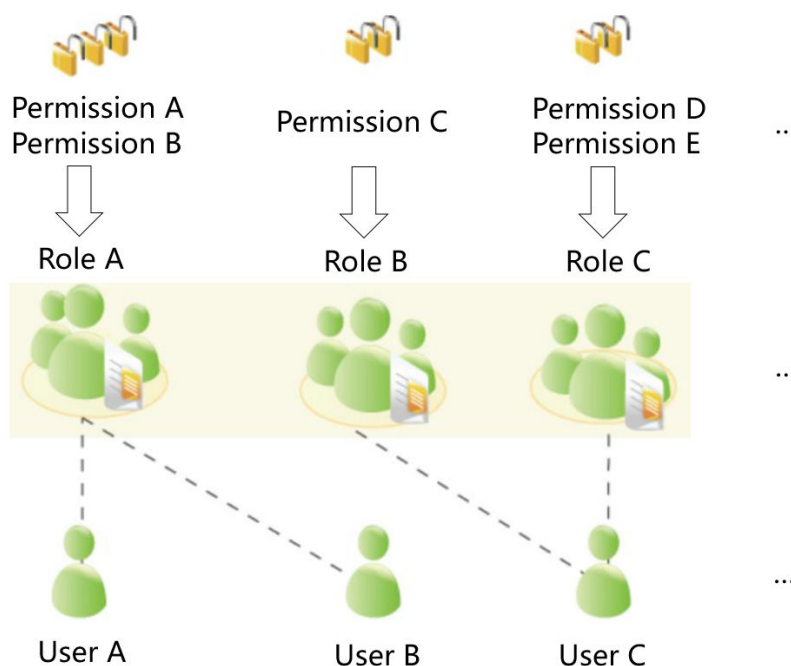
## 12.1.7 What Are the Differences Between Users and Roles?

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.

## 12.1.8 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 | use repl | passwd | valbegin | valuntil |
respool  | parent  | spaclimit  | useconfig | nodegroup | tempspaclimit | spillspaclimit |
-----+-----+-----+-----+-----+-----+-----+-----+-----+
Ruby     |      10 | t          | t        | t        | t        | ***** |          |          | default_pool | 0
dbadmin  |    16393 | f          | f        | f        | f        | ***** |          |          | default_pool | 0
jack     |   451897 | f          | f        | f        | f        | ***** |          |          | default_pool | 0
emma     |   451910 | f          | f        | f        | f        | ***** |          |          | default_pool | 0
jerry    |   457386 | f          | f        | f        | f        | ***** | 2022-05-19 10:31:56+08 |          | default_pool | 0
(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)

```

## 12.1.9 Regions and AZs

### Concepts

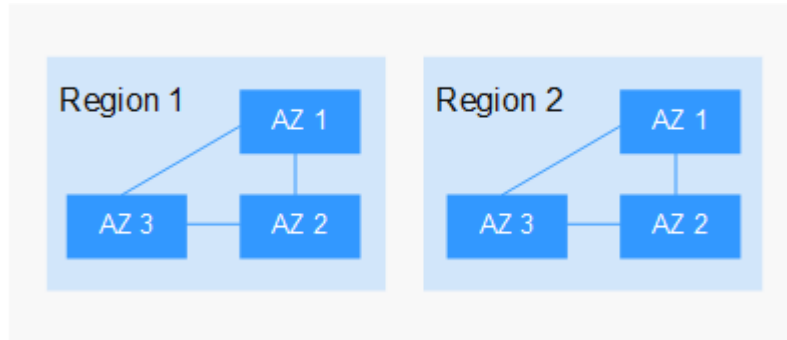
A region and availability zone (AZ) identify the location of a data center. You can create resources in regions and AZs.

- A region is a physical data center. 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 the internal network. The fault of an AZ will not affect other AZs. The internal network provides economical connection with low latency.

**Figure 12-1** shows the relationship between regions and AZs.



**Figure 12-1** Regions and AZs



## Selecting a Region

You are advised to select a region close to you or your target users. This reduces network latency and improves access rate.

## 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 for DR purposes.
- Deploy resources in the same AZ for minimum latency.

## Regions and Endpoints

When you use resources with API calls, you must specify the regional endpoint. For details about regions and endpoints, see [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.

## 12.1.10 Is My Data Secure in GaussDB(DWS)?

In the big data era, data is a user's core asset. data security first.

GaussDB(DWS) is a data warehousing system with telecom-class security to safeguard your data and privacy. GaussDB(DWS) satisfies the data security and privacy requirements of governments, financial organizations, and carriers. The choice of multiple industries, it is certified by the:

- 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 for security, including kernel hardening, installation of the latest patch, permission control, port management, and protocol and port anti-attack.

GaussDB(DWS) provides complete security measures, such as password policies, authentication, session management, user permissions management, and database audit.

## Snapshot Data Security

GaussDB(DWS) backups are snapshots stored in OBS. OBS supports access permission control, key access, and data encryption features. GaussDB(DWS) snapshot data can be used for data backup and restoration only and cannot be accessed by any user. GaussDB(DWS) administrators can view the OBS space occupied by snapshot data on the GaussDB(DWS) console and bills.

## Network Access Security

GaussDB(DWS) is fully isolated between the layer-2 and layer-3 networks to fulfill security requirements of government and financial users.

- GaussDB(DWS) is deployed in the tenant-dedicated ECS environment, which is not shared with other tenants. Therefore, data leakage due to computing resource sharing is impossible physically.
- ECSs in a GaussDB(DWS) cluster are isolated through VPCs, preventing the ECSs from being discovered and intruded on 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.

### 12.1.11 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.

### 12.1.12 Can I Modify the Security Group of a GaussDB(DWS) Cluster?

Yes. When a data warehouse cluster is created, its security group cannot be changed. However, you can add, delete, or modify rules of the current security group.

To edit the cluster security group:

1. Log in to the GaussDB(DWS) management console.
2. In the navigation pane on the left, click **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.

## 12.1.13 What Is a Database/Data Warehouse/Data Lake/Lakehouse?

The evolving Internet and IoT produce massive volumes of data. This data needs to be managed, using concepts like database, data warehouse, data lake, and lakehouse. What are these concepts? What are their relationships? What are the specific products and solutions? This document helps you understand them through comparison.

### Database

A database is where data is organized, stored, and managed by data structure.

Databases have been used in computers since the 1960s, with the two prevailing data models (hierarchical and network), and data and applications were very interdependent. This limited database applications.

A database usually refers to a relational database. A relational database organizes data with a relational model, that is, data is stored in rows and columns. Therefore, database data is well-structured and independent with low redundancy. In 1970, relational databases were born to completely separate data from applications for software and have become an indispensable part of mainstream computer systems. Relational databases are the foundation of database products from all vendors, with relational API support even if a database is non-relational.

Relational databases process basic and routine transactions using OLTP, such as bank transactions.

### Data Warehouse

Database growth has facilitated data growth. OLAP explores the relationship between data and mines more data value. However, it is difficult to share data between different databases, and data integration and analysis also face great challenges.

To overcome these challenges for enterprises, Bill Inmon, proposed the idea of data warehousing in 1990. The data warehouse runs on a unique storage architecture to perform OLAP on a large amount of the OLTP data accumulated over the years. In this way, enterprises can obtain valuable information from massive data quickly and effectively to make informed decisions. Thanks to data warehouses, the information industry has evolved from operational systems based on relational databases to decision support systems.

Unlike a database, a data warehouse has the following features:

- A data warehouse uses themes. It is built to support various services, with data coming from scattered operational data. Therefore, the required data needs to be extracted from multiple heterogeneous data sources, processed and integrated, and reorganized by theme.
- A data warehouse mainly supports enterprise decision analysis and the operations involved are focused on data query. Therefore, it improves the

query speed and cuts the total cost of ownership (TCO) by optimizing table structures and storage modes.

**Table 12-2** Comparison between data warehouses and databases

| Dimension            | Data Warehouse            | Database                         |
|----------------------|---------------------------|----------------------------------|
| Application scenario | OLAP                      | OLTP                             |
| Data source          | Multiple                  | Single                           |
| Data normalization   | Denormalized schemas      | Highly normalized static schemas |
| Data access          | Optimized read operations | Optimized write operations       |

## Data Lake

Data is an important asset for enterprises. Production and operations data are saved and distilled into effective management policies.

The data lake does that. It is a large data warehouse that centrally stores structured and unstructured data. It can store raw data of multiple data sources and types, meaning that data can be accessed, processed, analyzed, and transmitted without being structured first. The data lake helps enterprises quickly complete federated analysis of heterogeneous data sources and explore data value.

A data lake is in essence a solution that consists of a data storage architecture and data processing tools.

- The **storage architecture** must be scalable and reliable enough to store massive data of any type (structured, semi-structured, unstructured data).
- The two types of **processing tools** have separate functions:
  - The first type: migrates data into the lake, including defining sources, formulating synchronization policies, moving data, and compiling catalogs.
  - The second type then uses that data, including analyzing, mining, and using it. The data lake must be equipped with wide-ranging capabilities, such as comprehensive data and data lifecycle management, diversified data analytics, and secure data acquisition and release. These data governance tools help guarantee data quality, which can be compromised by a lack of metadata and turn the data lake into a data swamp.

Now with big data and AI, lake data is even more valuable and plays new roles. It represents more enterprise capabilities. For example, the data lake can centralize data management, helping enterprises build more optimized operation models. It also provides other enterprise capabilities such as prediction analysis and recommendation models. These models can stimulate further growth.

Just like any other warehouse and lake, one stores goods, or data, from one source while the other stores water, or data, from many sources.

**Table 12-3** Comparison between data lakes and data warehouses

| Dimension            | Data Lake   | Data Warehouse   |
|----------------------|---|--|
| Application scenario | Exploratory analytics (machine learning, data discovery, profiling, prediction) | Data analytics (based on historical structured data)     |
| Cost                 | Low initial cost, high subsequent cost  | High initial cost, low subsequent cost                   |
| Data quality         | Massive raw data to be cleaned and normalized before use                        | High quality data that can be used as the basis of facts |
| Target user          | Data scientists and data developers   | Business analysts  |

## Lakehouse

Although the application scenarios and architectures of a data warehouse and a data lake are different, they can cooperate to resolve problems. A data warehouse stores structured data and is ideal for quick BI and decision-making support, while a data lake stores data in any format and can generate larger value by mining data. Therefore, their convergence can bring more benefits to enterprises in some scenarios.

A lakehouse, the convergence of a data warehouse and a data lake, aims to enable data mobility and streamline construction. The key of the lakehouse architecture is to enable the free flow of data/metadate between the data warehouse and the data lake. The explicit-value data in the lake can flow to or even be directly used by the warehouse. The implicit-value data in the warehouse can also flow to the lake for long-term storage at a low cost and for future data mining.

## Intelligent Data Solution

DataArts Studio is a data enablement platform that helps large government agencies and companies customize intelligent data resource management solutions. This solution can import all-domain data into the data lake, eliminating data silos, unleashing the value of data, and empowering data-driven digital transformation.

DataArts Studio features the FusionInsight intelligent data lake as its core. Around it are computing engines such as the database, data warehouse, data lake, and data platform. It provides comprehensive data enablement, covering data collection, aggregation, computing, asset management, and data openness.

Lake, warehouse, and database engines enable agile data lake construction, fast migration of GaussDB databases, and real-time analysis of the data warehouse. For more information, go to:

- Database

- Relational databases include: Relational Database Service (RDS) , GaussDB(for MySQL) , GaussDB , RDS for PostgreSQL , RDS for SQL Server .
- Non-relational database: Document Database Service (DDS), GaussDB NoSQL (including Influx, Redis, Mongo, Cassandra)
- Data warehouse: GaussDB(DWS)
- Data lake or lakehouse: MapReduce Service (MRS), Data Lake Insight (DLI)
- Data governance center: DataArts Studio.

## 12.2 Cluster Management

### 12.2.1 What Do I Do If Creating a GaussDB(DWS) Cluster Failed?

#### Troubleshooting

Check that you have enough quota for creating the cluster.

#### Technical Support

If the fault cannot be identified, submit a service ticket to report the problem: Log in to the management console and choose **Service Tickets > Create Service Ticket**.

### 12.2.2 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**.

## 12.2.3 Why Did the Used Storage Shrink After Scale-out?

### Possible Causes

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. This causes a big drop in capacity.

### 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.

## 12.2.4 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

## 12.2.5 Does GaussDB(DWS) Support a Single Node for a Learning Environment?

Yes. In GaussDB(DWS), you can create a hybrid data warehouse cluster in standalone mode. If the name of the selected node flavor contains **h1** (for example, **dwsx2.h1.xlarge.2.c6**), the hybrid data warehouse only supports standalone deployment, which does not provide HA capabilities. The storage cost

can be reduced by half. A standalone data warehouse can be restored by the automatic reconstruction of ECS, and its data reliability is ensured by the EVS multi-copy mechanism. It is less expensive than other specifications. It is a good choice for lightweight services.

## 12.2.6 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](#).

## 12.2.7 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.

## 12.2.8 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:

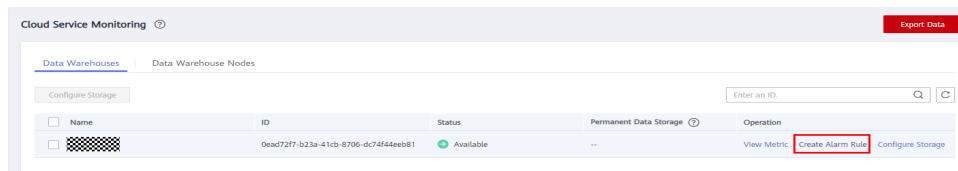
```
show max_active_statements;
```

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%).

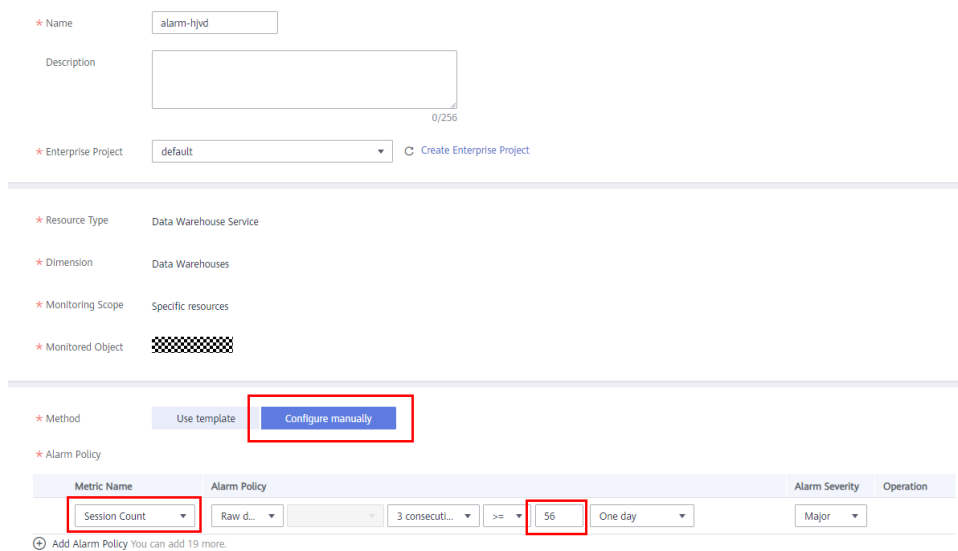


Procedure:

1. Go to the **Clusters** page on the GaussDB(DWS) management console.
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**.



\* Name: alarm-hjvd


Description:

\* Enterprise Project: default

\* Resource Type: Data Warehouse Service

\* Dimension: Data Warehouses

\* Monitoring Scope: Specific resources

\* Monitored Object: 

\* Method:

\* Alarm Policy

| Metric Name   | Alarm Policy | Alarm Severity | Operation |
|---------------|--------------|----------------|-----------|
| Session Count | Raw d...     | 3 consecuti... | >= 56     |

⊕ Add Alarm Policy You can add 19 more.

## 12.2.9 When Should I Add CNs or Scale out a cluster?

### Introduction to CN Concurrency

CN is short for Coordinator Node. A CN is an important component of GaussDB(DWS) and is closely related to users. It provides interfaces to external applications, optimizes global execution plans, distributes 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.

## 12.2.10 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 12-4](#).

**Table 12-4** 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.              |

## 12.3 Database Connections

### 12.3.1 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 12-5** Communication between applications and GaussDB(DWS)

| Scenario              |  | Description  | Supported Connection Type   |
|-----------------------|--|--|---|
| Cloud                 | <b>Service Application and GaussDB(DWS) Are in the Same VPC in the Same Region</b>                             | Two private IP addresses in the same VPC can directly communicate with each other.   | <ul style="list-style-type: none"> <li>• <code>gsq</code></li> <li>• Data Studio</li> <li>• JDBC/ODBC</li> </ul> For more connection modes, see . |
|                       | <b>Service Applications and GaussDB(DWS) are in Different VPCs in the Same Region</b>                          | After a <b>VPC peering</b> connection is created between two VPCs, the two private IP addresses can directly communicate with each other.  |   |
|                       | <b>Service Applications and GaussDB(DWS) Are in Different Regions</b>  | After a <b>cloud connection</b> (CC) is established between two regions, the two regions communicate with each other through private IP addresses.   |   |
| On-premises and cloud | <b>Service applications are deployed in on-premise data centers and need to communicate with GaussDB(DWS).</b> | <ul style="list-style-type: none"> <li>• Use the <b>public IP address or domain name</b> of GaussDB(DWS) for communication.</li> <li>• Use <b>Direct Connect</b> (DC) is for communication.</li> </ul> |   |

#### 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 the 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.

## 12.3.2 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 *Tutorial: Development Using JDBC or ODBC*.

## 12.3.3 Can I Connect to GaussDB(DWS) Cluster Nodes Using SSH?

No, direct access is not supported. VMs at the bottom layer of GaussDB(DWS) serve as the compute nodes for data analysis. Access cluster databases using the private or public network access address instead.

## 12.3.4 What Should I Do If I Cannot Connect to a Data Warehouse Cluster?

### Troubleshooting

Check:

- Whether the cluster status is normal.
- Whether the connection command, username, password, IP address, and port are correct.
- Whether the operating system type and version of the client are correct.
- Whether the client is properly installed.

If cluster connection failed on the , check the following items:

- Whether the ECSs are in the same AZ, VPC, subnet, and security group as the cluster.
- Whether the inbound and outbound rules of the security group are correct.

If cluster connection failed through the Internet, confirm the following items:

- Whether your network is connected to the Internet.
- Whether the firewall blocked access.
- Whether you need to access the Internet through a proxy.

## Technical Support

If the fault cannot be identified, submit a service ticket to report the problem: Log in to the management console and choose **Service Tickets > Create Service Ticket**.

### 12.3.5 Why Was I Not Notified of Failure Unbinding the EIP When GaussDB(DWS) Is Connected Over the Internet?

After the EIP is unbound, the network may be disconnected. 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 duration when the database sends the disconnection message to the client depends on the keepalive settings. The specific algorithm for calculating the duration is:

**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.

## 12.3.6 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**.

The whitelist will be added.

## 12.4 Data Import and Export

### 12.4.1 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 file formats: CSV, TEXT, ORC, and CARBONDATA. The default format is TEXT.

GDS file formats: CSV and TEXT. The default format is TEXT.

### 12.4.2 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.

### 12.4.3 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 the "Import Modes" section in *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 from MRS with MRS as the ETL.
  - Importing data with the **COPY FROM STDIN** command
- Run the **COPY FROM STDIN** command to write data to a table.

## 12.4.4 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.

## 12.4.5 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 `gsutil` 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 '|');
```

## 12.4.6 Can I Export GaussDB(DWS) Data from/to OBS Across Regions?

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.

## 12.4.7 How Do I Import GaussDB(DWS)/Oracle/MySQL/SQL Server Data to GaussDB(DWS) (Whole Database Migration)?

To import heterogeneous data to GaussDB(DWS), use CDM to migrate the entire Oracle, MySQL, or SQL Server database. Or migrate the entire GaussDB(DWS) database to another GaussDB(DWS) database. For details, see "Entire DB Migration" in the *Cloud Data Migration User Guide*.

Alternatively, you can store data in OBS and then dump the data to GaussDB(DWS). For details, see "About Parallel Data Import from OBS" in the *Data Warehouse Service (DWS) Developer Guide*.

## 12.4.8 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.

## 12.4.9 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

## 12.5 Account, Password, and Permissions

### 12.5.1 How Does GaussDB(DWS) Implement Workload Isolation?

#### Workload Isolation

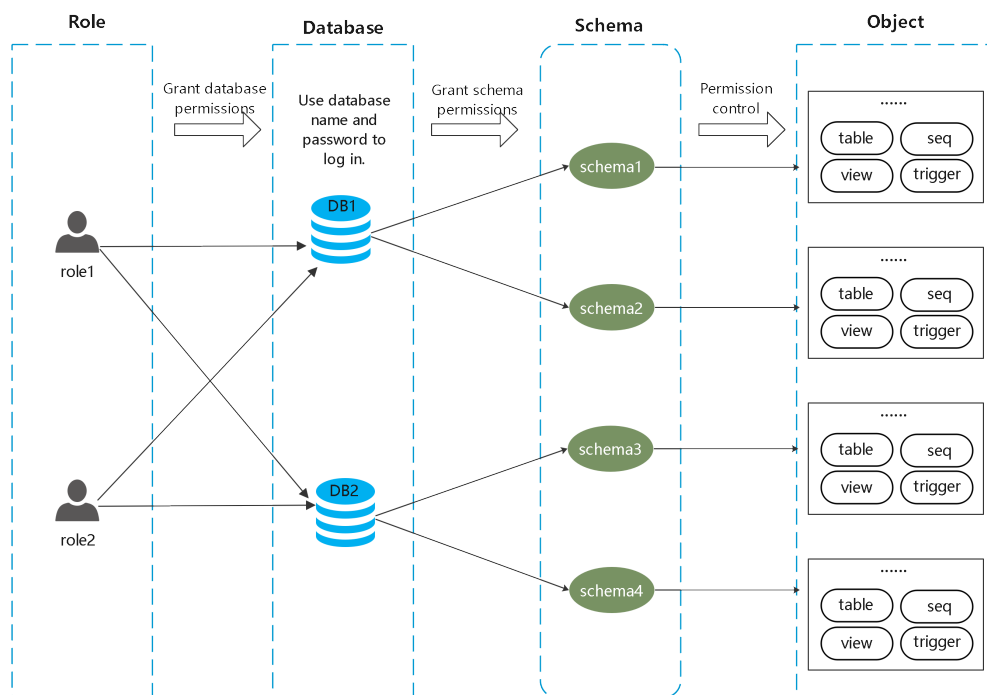
In GaussDB(DWS), you can isolate workloads through database and schema configurations. Their differences are as follows:

- 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 12-2** 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 '*****1';
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 '*****1';
SET
```

Query the table **myschema.t1**.

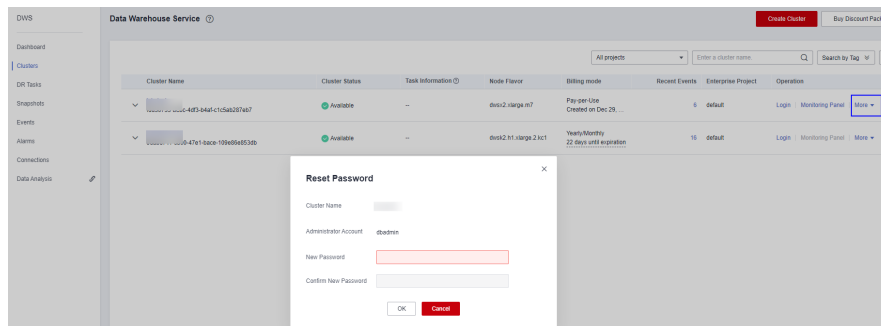
```
gaussdb=> SELECT * FROM myschema.t1;
a | b
---+---
(0 rows)
```

----End

## 12.5.2 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 12-3** 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;`

### 12.5.3 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.

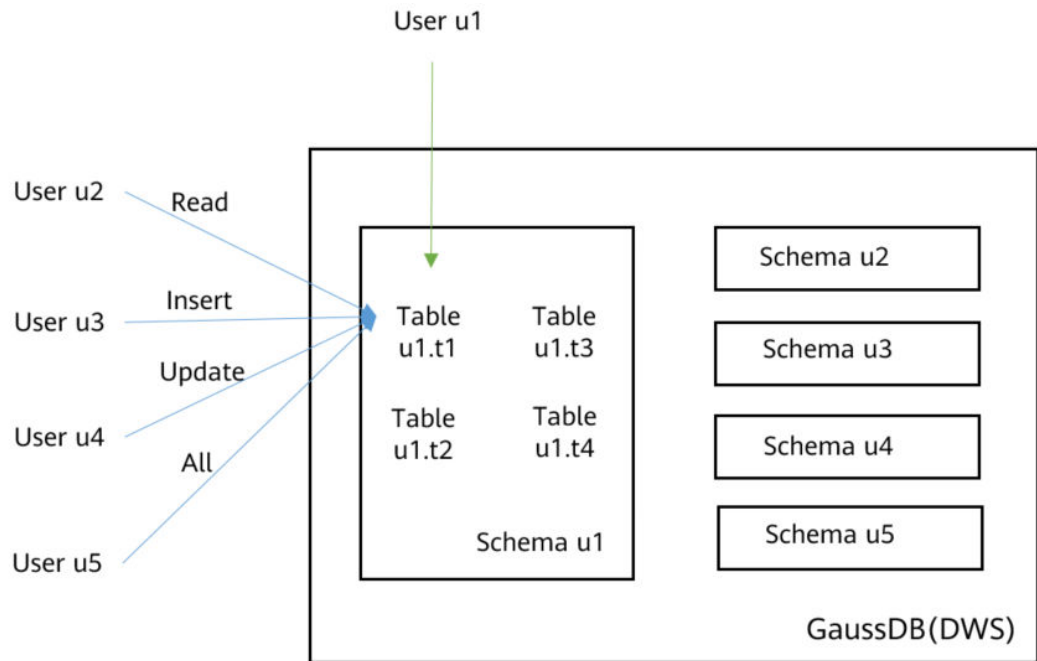
#### Syntax

```
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 12-6** Permissions of the u1.t1 table

| Us er | Ty pe             | GRANT Statement  | Qu ery | Ins ert | Up dat e | Del ete |
|-------|-------------------|--|--------|---------|----------|---------|
| u1    | Ow ner            | -  | √      | √       | √        | √       |
| u2    | Re ad-onl y use r | GRANT SELECT ON u1.t1 TO u2;   | √      | x       | x        | x       |
| u3    | INS ER T use r    | GRANT INSERT ON u1.t1 TO u3;   | x      | √       | x        | x       |
| u4    | UP DA TE use r    | GRANT SELECT,UPDATE ON u1.t1 TO u4;<br><b>NOTICE</b><br>The UPDATE permission must be granted together with the SELECT permission, or information leakage may occur. | √      | x       | √        | x       |



| Us<br>er | Ty<br>pe  | GRANT Statement                      | Qu<br>ery | Ins<br>ert | Up<br>dat<br>e | Del<br>ete |
|----------|---|--------------------------------------|-----------|------------|----------------|------------|
| u5       | Us<br>ers<br>wit<br>h<br>all<br>per<br>mis<br>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;
```

```
gaussdb=> SELECT * FROM u1.t1;  
 c1 | c2  
----+----  
  1 |  3  
  1 |  3  
  1 |  3  
(3 rows)  
  
gaussdb=> INSERT INTO u1.t1 VALUES (1,20);  
INSERT 0 1  
gaussdb=> UPDATE u1.t1 SET c2 = 3 WHERE c1 =1;  
UPDATE 4  
gaussdb=> DELETE FROM u1.t1;  
DELETE 4
```

**Step 11** In the session started by user **dbadmin**, execute the `has_table_privilege` function to query user permissions.

```
SELECT * FROM pg_class WHERE relname = 't1';
```

Check the **relacl** column in the command output. `rolename=xxxx/yyyy` indicates that `rolename` has the `xxxx` permission on the table and the permission is obtained from `yyyy`.

The following figure shows the command output.

- **u1=arwdDxtA/u1** indicates that **u1** is the owner and has full permissions.
- **u2=r/u1** indicates that **u2** has the read permission.
- **u3=a/u1** indicates that **u3** has the insert permission.
- **u4=rw/u1** indicates that **u4** has the read and update permissions.
- **u5=arwdDxtA/u1** indicates that **u5** has full permissions.

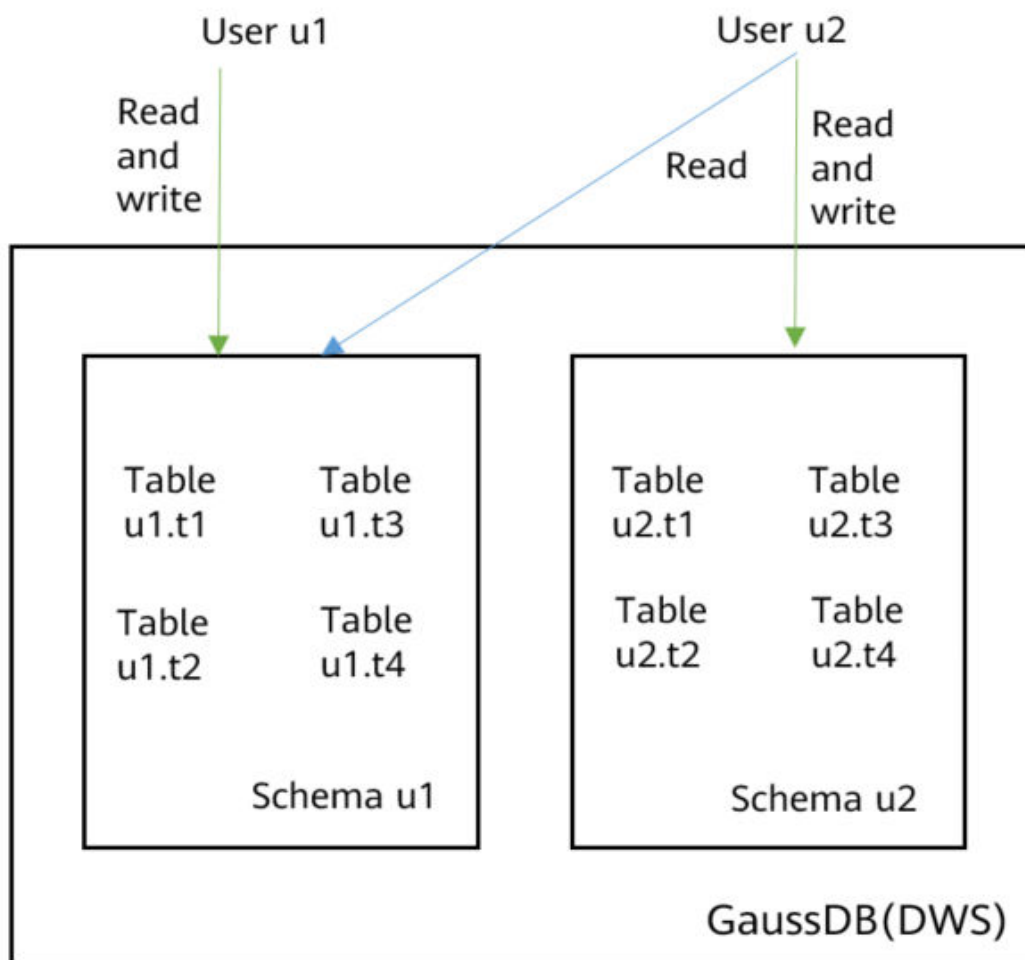
----End

## 12.5.4 How Do I Grant Schema Permissions to a User?

This section describes how to grant the query permission for a schema as an example. For more information, see "How Do I Grant Table Permissions to a User?" in *FAQs*.

- Permission for a table in a schema
- Permission for all the tables in a schema
- Permission for tables to be created in the schema

Assume that there are users **u1** and **u2**, and two schemas named after them. User **u2** needs to access tables in schema **u1**.



**Step 1** Connect to your database as **dbadmin**. Run the following statements to create users **u1** and **u2**. Two schemas will be created and named after the users by default.

```
CREATE USER u1 PASSWORD '{password}';
CREATE USER u2 PASSWORD '{password}';
```

**Step 2** Create tables **u1.t1** and **u1.t2** in schema **u1**.

```
CREATE TABLE u1.t1 (c1 int, c2 int);
CREATE TABLE u1.t2 (c1 int, c2 int);
```

**Step 3** Grant the access permission of schema **u1** to user **u2**.

```
GRANT USAGE ON SCHEMA u1 TO u2;
```

**Step 4** Grant user **u2** the permission to query table **u1.t1** in schema **u1**.

```
GRANT SELECT ON u1.t1 TO u2;
```

**Step 5** Start a new session and connect to the database as user **u2**. Verify that user **u2** can query the **u1.t1** table but not the **u1.t2** table.

```
SELECT * FROM u1.t1;
SELECT * FROM u1.t2;
```

```
gaussdb=> SELECT * FROM u1.t1;
 c1 | c2
----+----
(0 rows)

gaussdb=> SELECT * FROM u1.t2;
ERROR: permission denied for relation t2
```

**Step 6** In the session started by user **dbadmin**, grant user **u2** the permission to query all the tables in schema **u1**.

```
GRANT SELECT ON ALL TABLES IN SCHEMA u1 TO u2;
```

**Step 7** In the session started by user **u2**, verify that **u2** can query all tables.

```
SELECT * FROM u1.t1;
SELECT * FROM u1.t2;
```

```
gaussdb=> SELECT * FROM u1.t1;SELECT * FROM u1.t2;
 c1 | c2
----+----
(0 rows)

 c1 | c2
----+----
(0 rows)
```

**Step 8** In the session started by user **dbadmin**, create table **u1.t3**.

```
CREATE TABLE u1.t3 (c1 int, c2 int);
```

**Step 9** In the session started by user **u2**, verify that user **u2** does not have the query permission for **u1.t3**. It indicates that user **u2** has the permission to access all the existing tables in schema **u1**, but not the tables to be created in the future.

```
SELECT * FROM u1.t3;
```

```
gaussdb=> SELECT * FROM u1.t3;
ERROR: permission denied for relation t3
```

**Step 10** In the session started by user **dbadmin**, grant user **u2** the permission to query the tables to be created in schema **u1**. Create table **u1.t4**.

```
ALTER DEFAULT PRIVILEGES FOR ROLE u1 IN SCHEMA u1 GRANT SELECT ON TABLES TO u2;
CREATE TABLE u1.t4 (c1 int, c2 int);
```

**Step 11** In the session started by user **u2**, verify that user **u2** can access table **u1.t4**, but does not have the permission to access **u1.t3**. To let the user access table **u1.t3**, you can grant permissions by performing [Step 4](#).

```
SELECT * FROM u1.t4;
```

```
gaussdb=> SELECT * FROM u1.t4;
 c1 | c2
----+----
(0 rows)
```

----End

## 12.5.5 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

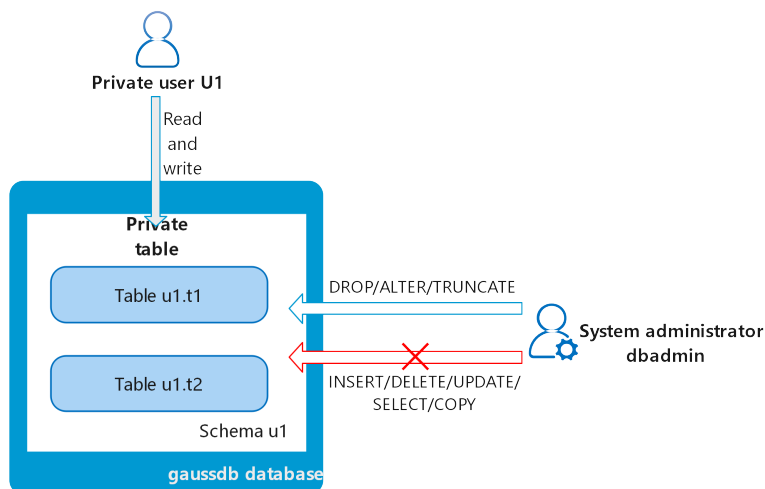
## 12.5.6 How Do I Create Private Database Users and Tables?

### 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 12-4 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

## 12.5.7 How Do I Revoke the CONNECT ON DATABASE Permission from a User?

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** and **CREATE TEMP TABLE** permissions on databases, **EXECUTE** permission on functions, and **USAGE** permission on 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. For security purposes, create an object and set its permissions in the same transaction, so that the object will not be accessible to any other users until you configure its permissions and end the transaction. In addition, you can run the **ALTER DEFAULT PRIVILEGES** statement to modify the default permissions.

The following shows an example of revoking the **CONNECT ON DATABASE** permission from a user.

**Step 1** Connect to the default database **gaussdb** of a GaussDB(DWS) cluster.

```
gsql -d gaussdb -h 192.168.0.89 -U dbadmin -p 8000 -r
```

If the following information is displayed after you enter the password as prompted, the connection is successful.

```
gaussdb=>
```

**Step 2** Create user **u1**.

```
CREATE USER u1 IDENTIFIED BY 'password';  
CREATE USER
```

**Step 3** Check whether access of **u1** is normal.

```
gsql -d gaussdb -h 192.168.0.89 -U u1 -p 8000 -W password -r  
gsql ((GaussDB 8.1.0 build be03b9a0) compiled at 2021-03-12 14:18:02 commit 1237 last mr 2001 release)  
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_128_GCM_SHA256, bits: 128)  
Type "help" for help.
```



**Step 4** Revoke the **CONNECT ON DATABASE** permission from **public**.

```
gsql -d gaussdb -h 192.168.0.89 -U dbadmin -p 8000 -r
gaussdb=>
REVOKE CONNECT ON database gaussdb FROM public;
REVOKE
```

 **NOTE**

**revoke connect on database postgres from u1** cannot be used directly because the **CONNECT** permission is granted to **public**.

**Step 5** Verify the result. If the following information is displayed, the **CONNECT ON DATABASE** permission has been revoked from user **u1**.

```
gsql -d gaussdb -h 192.168.0.89 -U u1 -p 8000
gsql: FATAL: permission denied for database "gaussdb"
DETAIL: User does not have CONNECT privilege.
```

----End

## 12.5.8 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 12-7** 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_dat<br>a | Type of the granted permission. The value can be <b>SELECT</b> , <b>INSERT</b> , <b>UPDATE</b> , <b>DELETE</b> , <b>TRUNCATE</b> , <b>REFERENCES</b> , <b>ANALYZE</b> , <b>VACUUM</b> , <b>ALTER</b> , <b>DROP</b> , or <b>TRIGGER</b> . |

| Column         | Data Type | Description  |
|----------------|-----------|--|
| is_grantable   | yes_or_no | Indicates if the permission can be granted to other users. <b>YES</b> indicates that the permission can be granted to other users, and <b>NO</b> 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 <b>SELECT</b> , <b>YES</b> is displayed. Otherwise, <b>NO</b> 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';
```



```
select * from has_table_privilege('joe','t1','select');
```

```
gaussdb=> select * from has_table_privilege('joe','t1','select');
has_table_privilege
-----
t
(1 row)
```

----End

## 12.5.9 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 | usersysid | usecreatedb | usesuper | usecatupd | use repl | passwd | valbegin | valuntil | respool | parent | spaclimit | useconfig | nodegroup | temppaclimit | spillspa
celimit
-----
obadmin | 16384 | f | f | f | f | ***** | | | default_pool | 0 | | | | |
Ruby | 18 | t | t | t | t | ***** | | | default_pool | 0 | | | | |
user_1 | 24594 | 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)
```

## 12.6 Database Usage

### 12.6.1 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's will be poor, affecting the overall query performance. You also need to consider how to adjust the distribution column of existing tables to improve 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's 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 DNs to execute, reducing the amount of data transferred between the DNs.
- 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, rebuild 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 = 'Jimy' where c_customer_sk = 6885;
```

```
gaussdb=> update customer_t1 set c_last_name = 'Jimy' 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 = 'Jimy' where c_customer_sk = 6885;
```

```
gaussdb=> update customer_t1 set c_last_name = 'Jimy' where c_customer_sk = 6885;
UPDATE 1
```

----End

## 12.6.2 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 **gaussdb**.

Value range: an existing database name. If this is not specified, the system copies **template1** by default. Its value cannot be **gaussdb**.

---

**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;
```

## 12.6.3 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 **DBCOMPATIBILITY** parameter to the compatible database type. The value of **DBCOMPATIBILITY** 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).

To avoid such conversion, set the database to the MySQL compatibility mode, which is the only mode that supports the date type. The compatibility mode can be configured only during database creation. 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 DBCOMPATIBILITY='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
```

## 12.6.4 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 in "VACUUM" and "ANALYZE | ANALYSE" in the *Developer Guide*.



 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 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, the database space will bloat, causing performance deterioration or even 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 12-9** 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 <b>VACUUM</b> 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.   |

### 12.6.5 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.

### 12.6.6 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*.

### 12.6.7 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.

## 12.6.8 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

## 12.6.9 How Can I Export All Tables and Views from a Database?

You can use **pg\_tables** and **pg\_views** to query all table information and views in a database. Example:

```
SELECT * FROM pg_tables;  
SELECT * FROM pg_views;
```

## 12.6.10 How Can I Delete Table Data Efficiently?

Yes. **TRUNCATE** is more efficient than **DELETE** for deleting massive data.

For details, see "TRUNCATE" in the *Data Warehouse Service (DWS) Developer Guide*.

### Function

**TRUNCATE** quickly removes all rows from a database table.

It has the same effect as the unconditional **DELETE**, but **TRUNCATE** is faster, especially for large tables, because it does not scan tables.

### Features

- **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.



## 12.6.12 If No Distribution Column Is Specified, How Will Data Be Stored?

### NOTE

For clusters of 8.1.2 or later, you can use the GUC parameter **default\_distribution\_mode** to query and set the default table distribution mode.

If no distribution column is specified during table creation, data is stored as follows:

- Scenario 1

If the primary key or unique constraint is included during table creation, hash distribution is selected. The distribution column is the column corresponding to the primary key or unique constraint.

```
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)
```

## 12.6.13 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)
```

## 12.6.14 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)
```

## 12.6.15 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 partitioned table **my\_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.relcudescrid) and b.relname="my_table_p1";
 relname
-----
pg_delta_part_60317
pg_cudesc_part_60317
(2 rows)

```

## 12.6.16 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 |           |           |         | 17.97
 2 | -> Nested Loop (3,5)       |    26 |           | 1MB      |         | 11.97
 3 | -> Streaming(type: BROADCAST) |     2 |           | 2MB      |         | 2.78
 4 | -> Seq Scan on public.t2   |     1 |           | 1MB      |         | 2.62
 5 | -> Index Scan using t1_b_idx on public.t1 |    26 |           | 1MB      |         | 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 augments. If the one-augment 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-augment 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.

## 12.6.17 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 .

## 12.6.18 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;
```

| nspname                                   | nspowner | nsptime | nspacl   | permspace | usedspace |
|---|----------|---------|--|-----------|-----------|
| pg_toast                                  | 10       | 0       |  | -1        | 0         |
| ctstore                                   | 10       | 0       |  | -1        | 0         |
| gs_logical_cluster                        | 10       | 0       |  | -1        | 0         |
| sys                                       | 10       | 0       |  | -1        | 0         |
| dbes_job                                  | 10       | 0       | {Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}  | -1        | 24576     |
| pg_catalog                                | 10       | 0       | {Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}  | -1        | 13888896  |
| public                                    | 10       | 0       | {Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}  | -1        | 82252     |
| 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_tov                                   | 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       | {Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}  | -1        | 8192      |
| u1  | 24954    | 0       |  | -1        | 0         |
| u2  | 24955    | 0       |  | -1        | 0         |
| u3  | 24962    | 0       |  | -1        | 0         |
| u4  | 24968    | 0       |  | -1        | 0         |
| s1  | 16833    | 0       | {dbadminUC/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       | {dbadminUC/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_5003_4_1_281471119284272       | 10       | 0       |  | -1        | 0         |
| pg_toast_temp_cn_5003_4_1_281471119284272 | 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.

## 12.6.19 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)

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 TBALE 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. For details about the compatibility differences, see "Syntax Compatibility Differences Among Oracle, Teradata, and MySQL" in *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 vary according to the database compatibility configuration item `behavior_compat_options`. For details about the parameter options, see "GUC Parameters > Miscellaneous Parameters > `behavior_compat_options`" in *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 DN 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.

## 12.6.20 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;
```

## 12.6.21 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 displays **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 | 14012787723904  | 77687093572141691 | active              | select test sleep1(); | 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 **RELEASES** messages, the stored procedure triggers multiple exceptions. In this case, you have to analyze whether the logic of the stored procedure is proper.

## 12.7 Database Performance

### 12.7.1 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 perform **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** Perform **ANALYZE** again. For details, see "ANALYZE | ANALYSE" in the *Developer Guide*.

```
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);
```

### 12.7.2 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 DNs 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 DNs concurrently must centralize the final summarization on one CN. In this case, most of the summarization work has been completed on DNs, 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 many bottlenecks. 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.

### 12.7.3 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 "Real-time TopSQL" in *Data Warehouse Service Development Guide*.
- For details about how to query historical SQL statements, see section "Historical TopSQL" in *Data Warehouse Service Development Guide*.

## 12.8 Snapshot Backup and Restoration

### 12.8.1 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.

### 12.8.2 Where Can a Snapshot Be Restored To?

A snapshot can only be restored to a new cluster. If data is restored to an existing cluster, data may be overwritten and incompliance issues may occur.

For more information, see .

### 12.8.3 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*.

# A Change History

---

| Release Date | Description                                |
|--------------|--|
| 2022-12-12   | This issue is the second official release. |
| 2021-01-07   | This issue is the first official release.  |