Data Warehouse Service

User Guide

Issue 01

Date 2025-11-11

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


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

Notice

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

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

Contents

1 Service Overview

1.1 What Is DWS?

DWS is an online data analysis and processing database built on the Huawei Cloud infrastructure and platform. It offers scalable, ready-to-use, and fully managed analytical database services, and is compatible with ANSI/ISO SQL-92, SQL-99, and SQL:2003 syntax. Additionally, DWS is interoperable with other database ecosystems such as PostgreSQL, Oracle, Teradata, and MySQL. This makes it a competitive option for petabyte-scale big data analytics across diverse industries.

Logical Cluster Architecture

Figure 1-1 shows the logical architecture of a DWS cluster. For details about the instance, see Table 1-1.

Figure 1-1 Logical cluster architecture

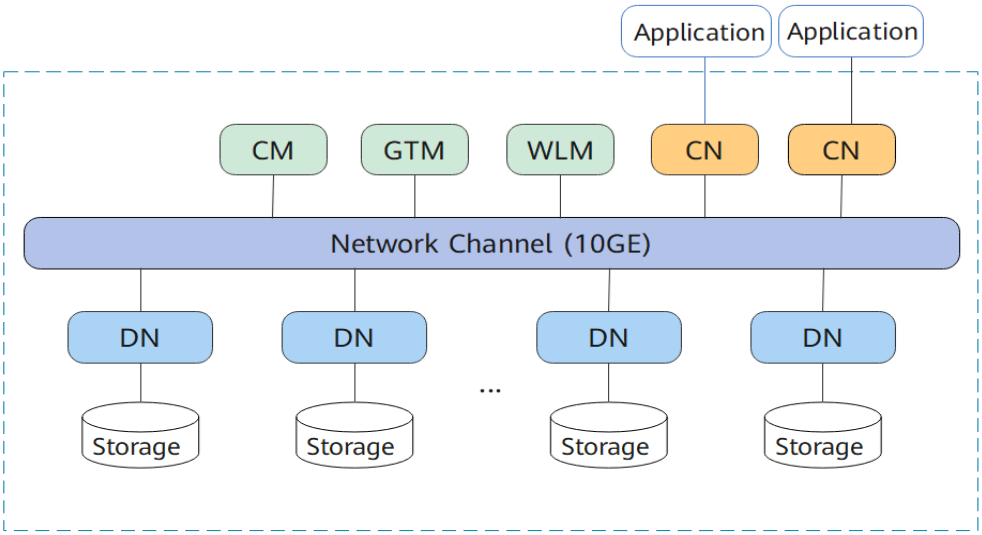


Table 1-1 Cluster architecture description

Name	Function	Description
Cluster Manager (CM)	Cluster Manager. It manages and monitors the running status of functional units and physical resources in the distributed system, ensuring system stability.	The CM consists of CM Agent, OM Monitor, and CM Server. • CM Agent monitors the running status of primary and standby GTMs, CNs, and primary and standby DNs on the host, and reports the status to CM Server. In addition, it executes the arbitration instruction delivered by CM Server. A CM Agent process runs on each host. • OM Monitor monitors scheduled tasks of CM Agent and restarts CM Agent when CM Agent stops. If CM Agent cannot be restarted, the host cannot be used. In this case, manually rectify this fault. **NOTE** CM Agent cannot be restarted probably because of insufficient system resources, which is not a common situation. • CM Server checks whether the current system is normal according to the instance status reported by CM Agent. In the case of exceptions, CM Server delivers recovery commands to CM Agent. CM Servers are deployed in primary/standby pairs to ensure system high availability. CM Agent connects to the primary CM Server. If the primary CM Server is faulty, the standby CM Server is promoted to primary to prevent a single point of failure (SPOF).
Global Transaction Manager (GTM)	Generates and maintains the globally unique information, such as the transaction ID, transaction snapshot, and timestamp.	The cluster includes only one pair of GTMs: one primary GTM and one standby GTM.
Workload Manager (WLM)	Workload Manager. It controls allocation of system resources to prevent service congestion and system crash resulting from excessive workload.	You do not need to specify names of hosts where WLMs are to be deployed, because the installation program automatically installs a WLM on each host.

Name	Function	Description
Coordinator (CN)	A CN receives access requests from applications, and returns execution results to the client; splits tasks and allocates task fragments to different DNs for parallel processing.	CNs in a cluster have equivalent roles and return the same result for the same DML statement. Load balancers can be added between CNs and applications to ensure that CNs are transparent to applications. If a CN is faulty, the load balancer automatically connects the application to the other CN. For details, see section "Associating and Disassociating ELB". 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. 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.

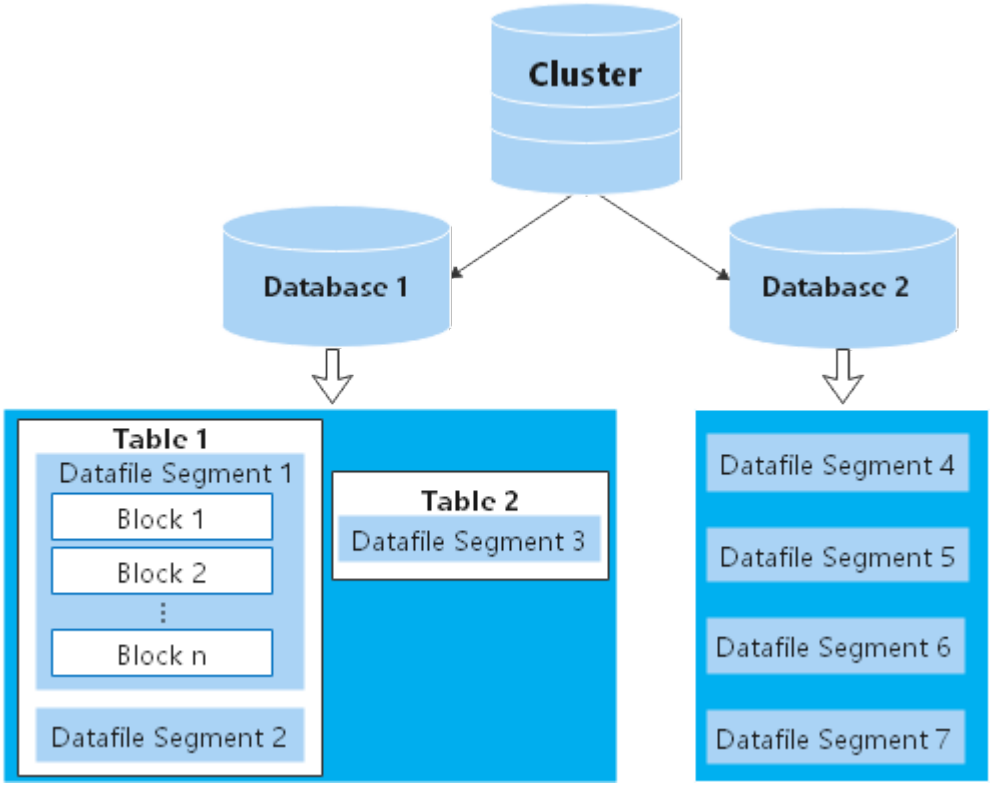
Name	Function	Description
Datanode (DN)	A DN stores data in row-store, column-store, or hybrid mode, executes data query tasks, and returns execution results to CNs.	There are multiple DNs in the cluster. Each DN stores part of data. DWS provides DN high availability: active DN, standby DN, and secondary DN. The working principles of the three are as follows: • During data synchronization, if the active DN suddenly becomes faulty, the standby DN is switched to the active state. • Before the faulty active DN recovers, the new active DN synchronizes data logs to the secondary DN. • After the faulty active DN recovers, it becomes the standby DN and uses data logs stored on the secondary DN to restore data generated during its faulty period. The secondary DN serves exclusively as a backup, never ascending to active or standby status in case of faults. It conserves storage by only holding Xlog data transferred from the new active DN and data replicated during original active DN failures. This efficient approach saves one-third of the storage space compared to conventional tri-backup methods.
Storage	Functions as the server's local storage resources to store data permanently.	-

DNs in a cluster store data on disks. **Figure 1-2** 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.

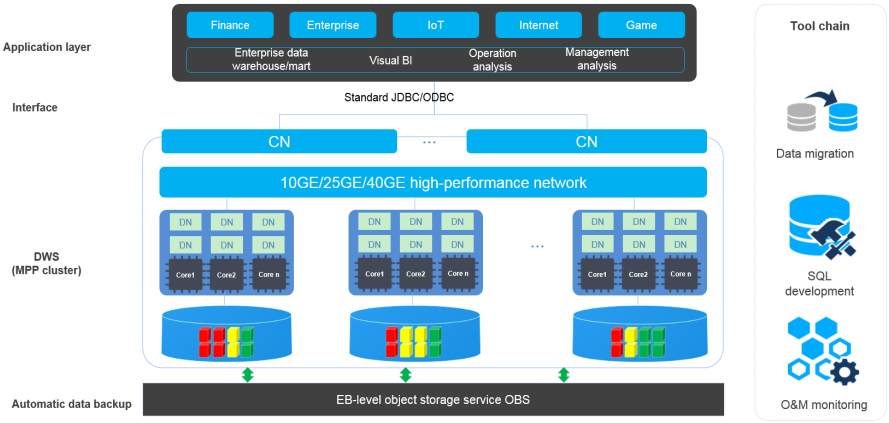
Figure 1-2 Logical database architecture



Storage-Compute Coupled Architecture

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 1-3 Architecture



- **Application layer**

Data loading tools, Extract-Transform-Load (ETL) tools, Business Intelligence (BI) tools, and data mining and analysis tools can be integrated with DWS through standard interfaces. DWS is compatible with the PostgreSQL ecosystem, and the SQL syntax is compatible with Oracle and Teradata. Applications can be smoothly migrated to DWS with only a few changes.

- **API**

  Applications can connect to DWS through standard JDBC and ODBC.

- **DWS**

  A data warehouse cluster contains nodes with the same flavor in the same subnet. These nodes jointly provide services. Datanodes (DNs) in a cluster store data on disks. CNs, or Coordinators, receive access requests from the clients and return the execution results. They also split and distribute tasks to the Datanodes (DNs) for parallel execution.

- **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 DWS at a specified time point. It records all configuration data and service data of the cluster at the specified moment.

- **Tool chain**

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

1.2 Advantages

DWS supports ANSI/ISO SQL-92, SQL-99, and SQL-2003 syntax, as well as the PostgreSQL, Oracle, Teradata, and MySQL database ecosystems. It offers powerful solutions for analyzing massive amounts of data in different industries, even at the petabyte scale.

DWS outperforms conventional data warehouses in hyper-scale data processing and general platform management due to the following features:

Ease of use

- Visualized one-stop management

  DWS helps you easily complete the entire process, from project concept to production deployment. The DWS console allows you to quickly set up a high-performance and highly available enterprise-level data warehouse cluster in just a few minutes, without requiring any data warehouse software or servers.

  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

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

High performance

- Cloud-based distributed architecture

  DWS adopts the MPP architecture so that service data is separately stored on numerous nodes. Data analytics tasks are quickly executed in parallel on the nodes where data is stored.

- Query response to trillions of data records within seconds

  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.

  DWS provides you with a better data compression ratio (column-store), higher index performance (column-store), and better point update and query (row-store) performance.

- Fast data loading

  DWS provides you with GDS, a high-speed parallel bulk data loading tool.

- Data Compression in Column Storage

  To compress old and inactive data to save space and reduce procurement and O&M costs.

  In DWS, data can be compressed using the Delta Value Encoding, Dictionary, RLE, LZ4, and ZLIB algorithms. The system automatically selects a compression algorithm based on data characteristics. The average compression ratio is 7:1. Compressed data can be directly accessed and is transparent to services, greatly reducing the preparation time before accessing historical data.

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. Table-level scale-out ensures service continuity.
- Online upgrade: Upgrading major versions online from 8.1.1 and performing online patch upgrades from 8.1.3 and later versions is now possible without interrupting your services. Any interruptions will only last a few seconds.

Robust reliability

- Transaction management
  - Transaction blocks are supported. You can run **start transaction** to explicitly start a transaction block.
  - Single-statement transactions are supported. If you do not explicitly start a transaction, a single statement is processed as a transaction.
  - Distributed transaction management and global transaction information management are supported. This includes gxid, snapshot, timestamp

management, distributed transaction status management, and gxid overflow processing.

- The atomicity, consistency, isolation, and durability (ACID) feature is supported, which ensures strong data consistency for distributed transactions.
- Deadlocks are prevented in the distributed system. A transaction will be unlocked immediately after a deadlock (if any).

- Comprehensive HA design

  All software processes of DWS are in active/standby mode. Logical components such as the CNs and DNs of each cluster also work in active/standby mode. This ensures data reliability and consistency when any single point of failure (SPOF) occurs.

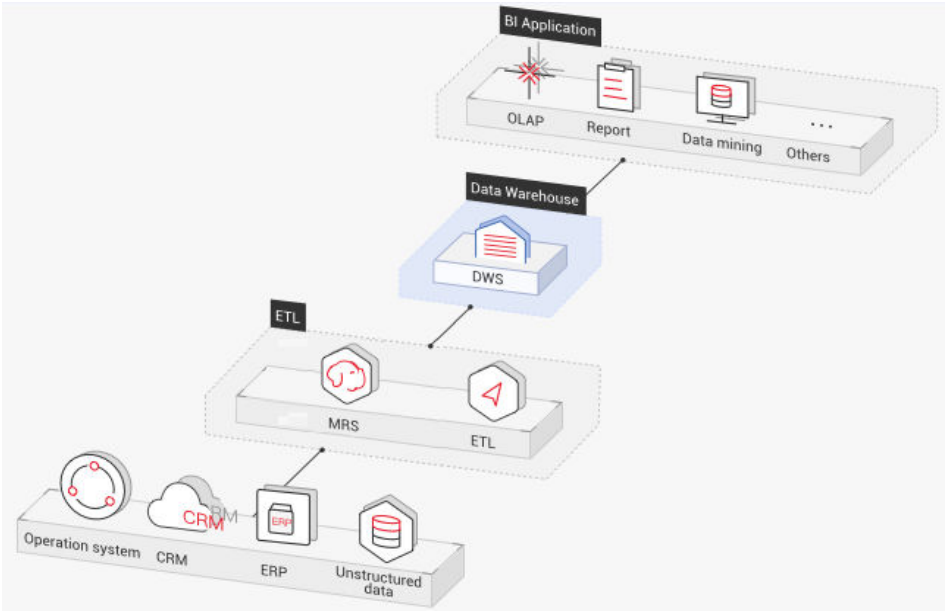
- High security

  DWS supports transparent data encryption and can interconnect with the Database Security Service (DBSS) to better protect user privacy and data security with network isolation and security group rule setting options. In addition, DWS supports automatic full and incremental backup of data for higher reliability.

1.3 Application Scenarios

Enhanced ETL + Real-time BI analysis

Figure 1-4 ETL+BI analysis



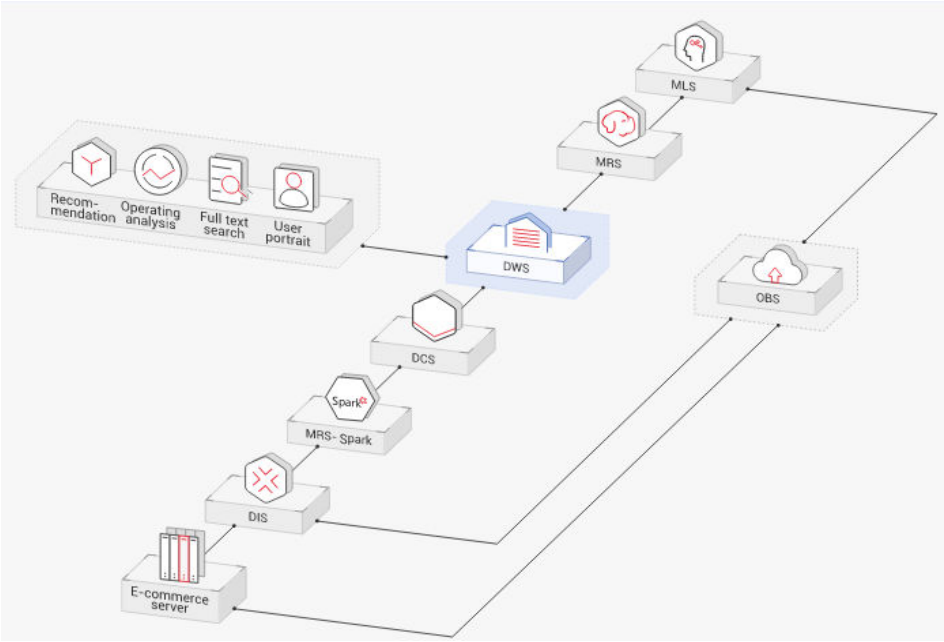
The data warehouse is the pillar of the Business Intelligence (BI) system for collecting, storing, and analyzing massive amounts of data. It provides powerful business analysis support for mobile Internet, gaming, and Online to Offline (O2O) industries.

Advantages of 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



The analyzed data is used for marketing recommendation, operations analysis, full-text search, and customer analysis.

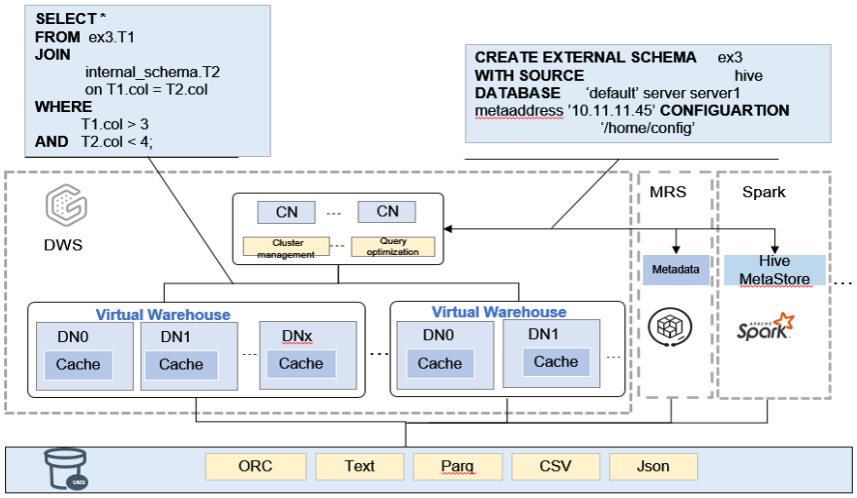
Advantages of DWS are as follows:

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

Lakehouse

- **Seamless access to the data lake**
  - With the interconnection with Hive Metastore metadata management, you can directly access the data table definitions in the data lake. You do not need to create a foreign table. You only need to create an external schema.
  - The following data formats are supported: ORC and Parquet.
- **Convergent query**

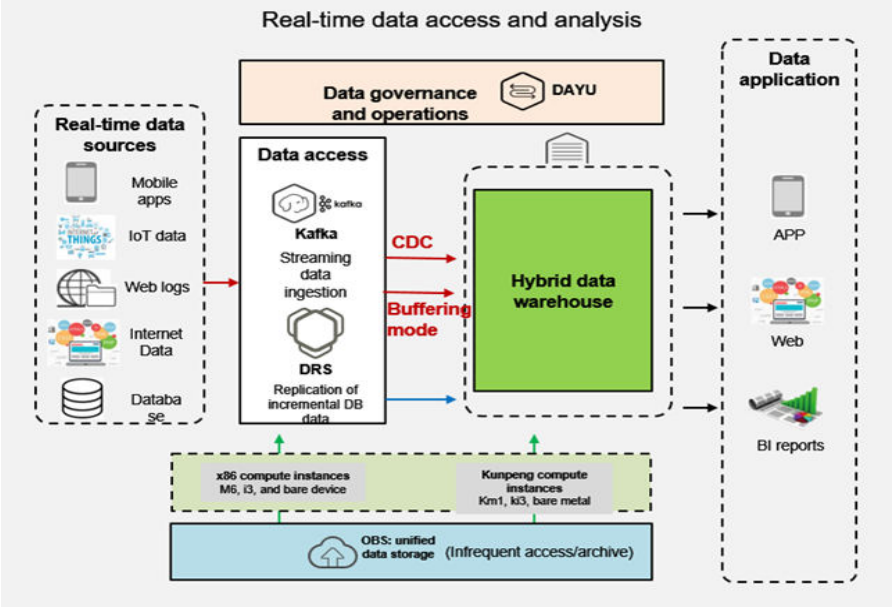
  - Hybrid query of any data in the data lake and warehouse is supported.
  - The query result is directly sent to the warehouse or data lake. No data needs to be transferred or copied.
- **Excellent query performance**
  - High-quality query plans and efficient execution engines
  - Precise load management methods



Real-Time Write

DWS 3.0 utilizes the H-Store storage engine to store micro-batch data locally and syncs it to OBS at regular intervals. It enables high-throughput real-time write and update, as well as large-scale data writes.

Real-time data is written and calculated, and can be used for dashboard statistics, analysis, monitoring, risk control, and recommendations.



1.4 Functions

DWS provides various methods to access the service, such as the console, client, and REST APIs. This section describes the main functions of DWS.

Enterprise-Level Data Warehouses and Compatibility with Standard SQL

After a DWS 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.

DWS provides high-performance databases that can handle petabytes of data, with the following features:

- MPP computing framework, hybrid row-column storage, and vectorized execution, enabling response to billion-level data correlation analysis within seconds
- Optimized in-memory computing based on Hash Join of Bloom Filter, improving the performance by 2 to 10 times
- Supports the symmetrically distributed, active-active multi-node cluster architecture, ensuring no SPOFs.
- Optimized communication between large-scale clusters based on telecommunication technologies, improving data transmission efficiency between compute nodes
- Cost-based intelligent optimizers, helping generate the optimal plan based on the cluster scale and data volume to improve execution efficiency

DWS has comprehensive SQL capabilities:

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

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

Cluster Management

A data warehouse cluster contains nodes with the same flavor in the same subnet. These nodes jointly provide services. DWS offers a professional, efficient, and centralized management console that enables you to quickly request clusters, manage data warehouses with ease, and concentrate on data and services.

Main functions of cluster management are described as follows:

- Creating Clusters

  To use data warehouse services on the cloud, create a 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 DWS cluster. A snapshot can be used to restore a cluster at a certain time. You can manually create snapshots for a cluster or enable automated snapshot creation (periodic). Automated snapshots have a limited retention period. You can copy automatic snapshots for long-term retention.

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

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

- Managing nodes

  You can check the nodes in a cluster, including the status, specifications, and usage of each node. To prepare for a large scale-out, you can add nodes in batches. To add 180 nodes, add them in three batches of 60 nodes each. If any nodes fail to be added, retry adding them. Once all 180 nodes are added, use them for scaling out. Adding nodes will not interrupt cluster services.

- Scaling out clusters

  As the service volume increases, the current scale of a cluster may not meet service requirements. In this case, you can scale out the cluster by adding compute nodes to it. Services are not interrupted during the scale-out. You can enable 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, DWS supports offline redistribution (default mode).

- Resource management

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

add database users to the corresponding pools to restrict their resource usages.

- Restarting clusters

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

- Deleting Clusters

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

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

- Management console

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

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

- REST APIs

  Use REST APIs provided by DWS to manage clusters. In addition, if you need to integrate 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

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

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

APIs

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

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

High Reliability

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

  DWS integrates with Cloud Eye, allowing you to monitor compute nodes and databases in the cluster in real time. For details, see "Monitoring Clusters Using Cloud Eye" in *Data Warehouse Service (DWS) User Guide*.
- Database Monitoring

  DMS is provided by 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 (DMS) "in the *Data Warehouse Service (DWS) User Guide*.
- Alarms

  You can check and configure alarm rules and subscribe to alarm notifications. Alarm rules display alarm statistics and details of the past week for users to

view tenant alarms. This feature monitors common DWS alarms with pre-set rules and allows users to customize the alarm thresholds based on their service needs. For details, see "Alarms" in the *Data Warehouse Service (DWS) User Guide*.

- Notifying Events

  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
  - DWS can be integrated with Cloud Trace Service (CTS) to audit management console operations and API calls. For details, see "Viewing Audit Logs of Key Operations on the Management Console".
  - DWS records all SQL operations, including connection attempts, query attempts, and database changes. For details, see "Viewing Database Audit Logs" in the *Data Warehouse Service (DWS) User Guide*.

Multiple Database Tools

DWS provides the following self-developed tools. You can download the tool packages on the DWS console. For how to use the tools, see the *Data Warehouse Service (DWS) Tool Guide*.

- gsql

  gsql is a CLI SQL client tool running on the Linux OS. It helps connect to, operate, and maintain the database in a DWS cluster.

- Data Studio

  It is a SQL client tool with a Graphical User Interface (GUI) that runs on Windows. It is utilized to connect to databases in a DWS cluster, manage database objects, edit, run, and debug SQL scripts, and view execution plans.

- GDS

  It is a data service tool offered by DWS that utilizes the foreign table mechanism to achieve fast 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 CLI tool running on the Linux or Windows OS. It is dedicated to providing customers with simple, fast, and reliable application SQL script migration services. It parses the SQL scripts of source database applications using the built-in syntax migration logic, and converts them to SQL scripts applicable to DWS databases.

  The DSC can migrate SQL scripts of Teradata, Oracle, Netezza, MySQL, and DB2 databases.

- **gs_dump** and **gs_dumpall**

  gs_dump exports a single database or its objects. **gs_dumpall** exports all databases or global objects in a cluster.

  To migrate database information, you can use a tool to import the exported metadata to a target database.

- gs_restore

  During database migration, you can export files using **gs_dump tool** and import them to DWS by using **gs_restore**. In this way, metadata, such as table definitions and database object definitions, can be imported.

1.5 Concepts

DWS Management Concepts

- Cluster

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

- Node

  A DWS cluster can contain 3 to 256 nodes. Each node can store and analyze data.

- Type

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

- Snapshot

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

- Project

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

DWS Database Concepts

- Databases

  A database manages data objects and is isolated from other databases. While creating an object, you can specify a tablespace for it. If you do not specify it, the object will be saved to the **PG_DEFAULT** space by default. Objects managed by a database can be distributed to multiple tablespaces.

- OLAP

  OLAP is a major function of DWS 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, DWS distributes service data to different nodes based on the database model and application characteristics. Nodes are connected through the network and collaboratively process computing tasks as a cluster and provide database services that meet service needs.

- Shared-Nothing Architecture

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

- Database Version

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

- Database Connections

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

- Database users and roles

  DWS uses users and roles to control the access to databases. A role can be a database user or a group of database users based on the role setting. In DWS, the difference between roles and users is that a role does not have the **LOGIN** permission by default. In DWS, one user can have only one role, but you can put a user's role under a parent role to grant multiple permissions to the user.

- Instance

  In DWS, instances are a group of database processes running in the memory. An instance can manage one or more databases that form a cluster. A cluster is an area in the storage disk. This area is initialized during installation and composed of a directory. The directory, called data directory, stores all data and is created by **initdb**. Theoretically, one server can start multiple instances on different ports, but DWS manages only one instance at a time. The start and stop of an instance rely on the specific data directory. For compatibility purposes, the concept of instance name may be introduced.

- Tablespaces

  In DWS, a tablespace is a directory storing physical files of the databases the tablespace contains. Multiple tablespaces can coexist. Files are physically isolated using tablespaces and managed by a file system.

- Schema

  DWS schemas logically separate databases. All database objects are created under certain schemas. In DWS, schemas and users are loosely bound. When you create a user, a schema with the same name as the user will be created automatically. You can also create a schema or specify another schema.

- V2 table

  A V2 table refers to a table whose **colversion** defined in the **CREATE TABLE** syntax is 2.0 during table creation, indicating that each column of the column-store table is combined and stored in a file named **relfilenode.C1.0**, and data is stored on the local disk. For storage-compute coupled clusters, if **colversion** is not specified, the created column-store table is a V2 table by default.

- V3 table

  A V3 table refers to a table whose **colversion** defined in the **CREATE TABLE** syntax is 3.0 during table creation, indicating that each column of the column-store table is stored in a file named **C1_field.0**, and data is stored in the OBS file system. For storage-compute coupled clusters, if **colversion** is not specified, the created column-store table is a V3 table by default.

- Transaction management

  In DWS, transactions are managed by multi-version concurrency control (MVCC) and two-phase locking (2PL). It enables smooth data reads and writes. In DWS, MVCC saves historical version data together with the current tuple version. DWS uses the VACUUM process instead of rollback segments to routinely delete historical version data. This does not affect user operations, unless in performance tuning. Transactions are automatically submitted in DWS.

1.6 Related Services

ECS

DWS uses an ECS as a cluster node.

VPC

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

OBS

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 DWS clusters for analysis after the data is processed by Hadoop.

CDM

You can use Cloud Data Migration (CDM) to migrate data from multiple sources to DWS.

Cloud Eye

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

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

LTS

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

SMN

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

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

ELB

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

1.7 DWS Permissions Management

If you need to assign different permissions to employees in your enterprise to access your DWS resources on Huawei Cloud, 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, to allow software developers in your company to use DWS resources while restricting high-risk operations and resource deletion, you can create IAM users tailored for these developers and grant them only the essential permissions for DWS usage.

If your Huawei Cloud account does not require individual IAM users for permissions management, you can 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*.

DWS Permissions

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

DWS is a project-level service deployed and accessed in specific physical regions. To assign 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 DWS, the users need to switch to a region where they have been authorized to use 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 DWS users only the permissions for managing a certain type of DWS resources.

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

Table 1-2 DWS system permissions

Role/Policy Name	Description	Category	Dependencies
DWS ReadOnlyAccess	Read-only permissions for DWS. Users granted these permissions can only view DWS data.	System-defined policy	N/A
DWS FullAccess	Database administrator permissions for DWS. Users granted these permissions can perform all operations on DWS.	System-defined policy	N/A
DWS Administrator	Database administrator permissions for DWS. Users granted these permissions can perform operations on all DWS resources. • Users also granted permissions of the **VPC Administrator** policy can create VPCs or subnets. • Users also granted permissions of the **CES Administrator** policy can view monitoring information of data warehouse clusters. • If you need to create an agency, you also need to configure the **Security Administrator** permission.	System-defined role	Dependent on the **Tenant Guest** and **Server Administrator** policies, which must be assigned in the same project as the **DWS Administrator** policy.

Role/Policy Name	Description	Category	Dependencies
DWS Database Access	Database access permissions for DWS. Users granted these permissions can generate temporary database user credentials based on IAM users to connect to databases in the data warehouse clusters.	System-defined role	Dependent on the **DWS Administrator** policy, which must be assigned in the same project as the **DWS Database Access** policy.

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

NOTE

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

Table 1-3 Common operations supported by system-defined permissions for DWS

Operation	DWS FullAccess	DWS ReadOnlyAccess	DWS Administrator	DWS Database Access
Creating/ Restoring clusters	√	x	√	x
Obtaining the cluster list	√	√	√	x

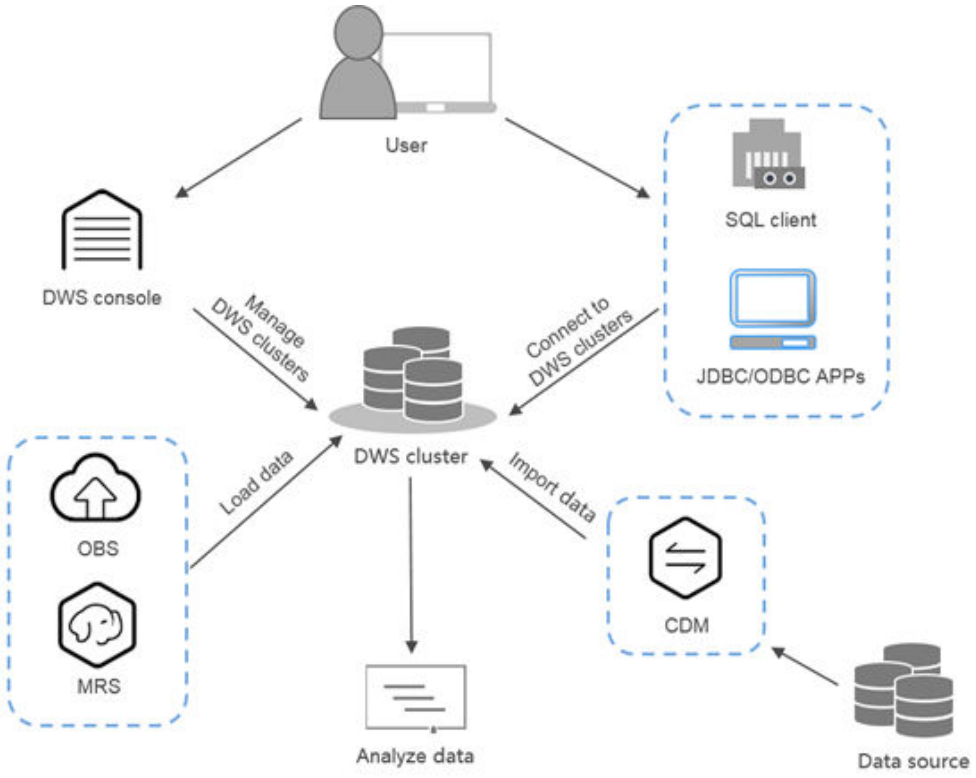
Operation	DWS FullAccess	DWS ReadOnlyAccess	DWS Administrator	DWS Database Access
Obtaining the details of a cluster	√	√	√	x
Setting automated snapshot policy	√	x	√	x
Setting security parameters/ parameter groups	√	x	√	x
Restarting clusters	√	x	√	x
Scaling out clusters	√	x	√	x
Resetting passwords	√	x	√	x
Deleting clusters	√	x	√	x
Configuring maintenance windows	√	x	√	x
Binding EIPs	x	x	√	x
Unbinding EIPs	x	x	√	x
Creating DNS domain names	√	x	√	x
Releasing DNS domain names	√	x	√	x
Modifying DNS domain names	√	x	√	x

Operation	DWS FullAccess	DWS ReadOnlyAccess	DWS Administrator	DWS Database Access
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
Obtaining the snapshot list	√	√	√	√
Deleting snapshots	√	x	√	x
Copying snapshots	√	x	√	x

1.8 Accessing DWS

The following figure shows how to use DWS.

Figure 1-6 Process for using DWS



Accessing a Cluster

DWS provides a web-based management console and HTTPS-compliant APIs for you to manage DWS clusters.

NOTE

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

Accessing the Database in a Cluster

DWS supports database access using the following methods:

- DWS client

  Use the DWS client to access the database in the cluster. For details, see "Cluster Connection" in the *Data Warehouse Service (DWS) User Guide*.

- JDBC and ODBC API calling

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

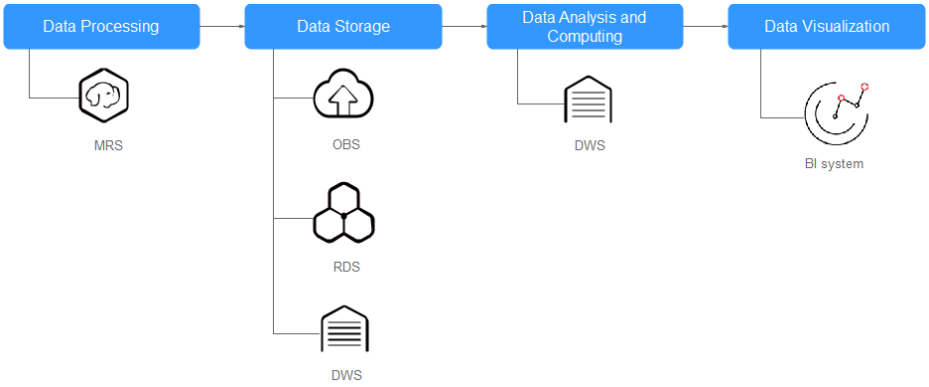
  For details, see "Using the JDBC and ODBC Drivers to Connect to a Cluster" in the *Data Warehouse Service (DWS) User Guide*.

End-to-End Data Analysis Process

DWS has been seamlessly integrated with other services on 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 1-7 End-to-end data analysis process



1.9 Restrictions

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

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

Specification and Performance Limits

Table 1-4 Specification and performance limits

Item	Limit	Description
Multi-AZ	3	• The **Multi-AZ** option is displayed only if the number of AZs in the selected region is greater than or equal to 3. If this condition is not met, only a single-AZ cluster can be created. • For a multi-AZ cluster, only three AZs can be selected at a time so far. Server nodes are evenly distributed among the three AZs. • The numbers of nodes in a multi-AZ cluster must be a multiple of 3. • In a multi-AZ cluster, the number of DNs must be less than or equal to 2.

Item	Limit	Description
Storage	• If you want to increase the upper limit of the storage space of a cloud SSD disk, contact customer service. • The amount of storage space available on the local SSD disk depends on the type of data warehouse flavor you choose.	• Cloud SSD disks offer a cost-effective solution for enterprise systems that require moderate performance. • Local SSD disks do not support disk scale-out.
Number of deployed CNs	The value ranges from 3 to the number of cluster nodes. The maximum value is **20** and the default value is **3**.	In a large-scale cluster, you are advised to deploy multiple CNs.
Maximum number of connections	• Number of CN connections: 100 to 262,143 • Number of DN connections: 100 to 262,143	For details about CNs and DNs, see **Logical Cluster Architecture**.

Quota Limits

Table 1-5 Quota limits

Item	Limit	Description
Number of nodes	256	The number of nodes in a new cluster cannot exceed the quota that can be used by a user or 256. If the node quota is insufficient, click **Increase quota** to submit a service ticket and apply for higher node quota.
Tags	Each cluster can have a maximum of 20 tags.	For more information, see "Overview" in *Data Warehouse Service User Guide*.
Retention period of automated backups	The value ranges from 1 to 31 days. The default value is 7 days.	The system deletes expired snapshots when the retention period ends.

Naming Rules

Table 1-6 Naming rules

Item	Description
Cluster name	Enter 4 to 64 characters. Only letters (case-insensitive), digits, hyphens (-), and underscores (_) are allowed. The name must start with a letter.
Administrator account	• The username can contain only lowercase letters, digits, and underscores (_). • The username must start with a lowercase letter or an underscore (_). • The username should contain 6 to 64 characters. • The username cannot be a keyword of the DWS database. For details about the keywords of the DWS database, see "SQL Reference" > "Keyword" in the *Data Warehouse Service Developer Guide*.
Role name	The value must start with a letter and can contain a maximum of 63 characters, including letters, digits, and underscores (_).
Username	The value must start with a letter and can contain a maximum of 63 characters, including letters, digits, and underscores (_).
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 policy 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 (_).
Alarm rule name	The rule name contains 6 to 64 characters and can contain only letters, digits, and slashes (/).
DR name	The DR name is a string of 4 to 64 case-insensitive characters and must start with a letter. It can contain only letters, digits, hyphens (-), and underscores (_).

Restrictions on Basic Cluster Operations

Table 1-7 Restrictions on basic cluster operations

Item	Description
Binding a load balancer	• To bind a load balancer to a DWS cluster, ensure that the load balancer is in the same region, VPC, and enterprise project as the cluster. • Only dedicated load balancers can be bound to DWS. • When you unbind a load balancer from a cluster, related cluster information is cleared on DWS but the load balancer is not deleted.
Adding or deleting a CN node	• 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. • If one of your CNs is abnormal, you can only delete this abnormal CN. If two or more CNs are abnormal, you can delete CNs only after the CNs are recovered from faults.
Managing resource load	• Resources cannot be managed during offline scale-out. If a resource management plan is enabled, stop it before performing offline scale-out.
Managing logical clusters	• You are advised to allocate tables in a database to the same logical cluster.
Restarting a cluster	A cluster cannot provide services during the restart. Therefore, before the restart, ensure that no task is running and all data is saved. If a cluster is processing transactional data, for example, importing data, querying data, creating snapshots, or restoring snapshots, files may be damaged or the cluster may fail to be restarted once the cluster is restarted. You are advised to stop all cluster tasks before restarting a cluster.
Starting or stopping a cluster	After the cluster is stopped, ECS basic resources (vCPUs and memory) are no longer reserved. When you start the service again, it may fail to be started due to insufficient resources. In this case, wait for a while and try again later.

Restrictions on Cluster O&M Operations

Table 1-8 Cluster O&M operation restrictions

Item	Description
Scaling out a cluster	• The cluster will be intermittently disconnected during scale-out. Exercise caution when performing this operation. • Certain cluster functions, including restarting, stopping, and starting, modifying specifications, adding or removing CNs, creating snapshots, and resetting the database administrator's password, cannot be performed while scaling out the cluster. • This function can be manually enabled only when the cluster task information displays **To be redistributed** after scale-out.
Scaling in a cluster	• When scaling in a cluster, several functions are disabled, including cluster restart, cluster scale-out, snapshot creation, node management, intelligent O&M, resource management, parameter modification, security configurations, log service, database administrator password resetting, and cluster deletion.
Performing elastic specification changes	• Elastic specification change is only supported by storage-compute coupled clusters that use ECSs and EVS disks. Clusters with local ECSs do not have this capability. • Stop the VM before changing the flavor. The flavor change can only be done offline and it takes 5 to 10 minutes.
Performing classic specification changes	• A cluster can have up to 240 nodes. The old and new clusters can have up to 480 nodes in total. • Logical clusters do not support the **Change all specifications** option.
Backing up and restoring a cluster	• **Backing up the cluster is essential for maintaining data reliability, especially when the service provider cannot restore data through upstream re-import. This helps prevent data loss caused by human or other factors.** • 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.

Item	Description
Upgrading a cluster	• If you are using a version of 8.1.3 or earlier, you will not be able to roll back or submit upgrade tasks until the cluster upgrade is finished. • DR cannot be established after a hot patch is installed in a cluster.
Managing DR tasks	• If the DR task is stopped or encounters an abnormal situation, but the DR cluster remains normal, it can still provide read services. Once the DR switchover is completed, the DR cluster can provide both read and write services. • After creating the DR cluster, the snapshot function of the production cluster remains normal, but the snapshot function of the DR cluster is disabled, along with the restoration function for both the production and DR clusters. • Logical clusters and resource pools are not supported.
Managing logs	• This function cannot be used if OBS is not available. • When CNs are changed, such as modifying specifications or adding/deleting CNs, there is a risk of data loss. To mitigate this risk, disable audit log dump while performing the change.

1.10 DWS Technical Specifications

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

Table 1-9 Technical specifications of DWS 8.1.3 – 9.1.0

Technical Specifications	Maximum Value of 8.1.3	Maximum Value of 8.2.0	Maximum Value of 8.2.1	Maximum Value of 8.3.0	Maximum Value of 9.1.0
Number of cluster nodes	2048	2048	2048	2048	• Storage-compute integration: 2048 • Decoupled storage and compute: The multi-VW technology is used to support up to 256 VWs, each with up to 1,024 DNs. It is recommended that you limit VWs to 32 or fewer and DNs to 128 or fewer per VW. The total number of DNs of all VWs cannot exceed 2,048.

Technical Specifications	Maximum Value of 8.1.3	Maximum Value of 8.2.0	Maximum Value of 8.2.1	Maximum Value of 8.3.0	Maximum Value of 9.1.0
Number of concurrent connections	Number of concurrent complex queries in minutes: 80 Number of short queries in seconds: 500 Number of concurrent short transactions in milliseconds: 5000	Number of concurrent complex queries in minutes: 80 Number of short queries in seconds: 500 Number of concurrent short transactions in milliseconds: 5000	Number of concurrent complex queries in minutes: 80 Number of short queries in seconds: 500 Number of concurrent short transactions in milliseconds: 5000	Number of concurrent complex queries in minutes: 80 Number of short queries in seconds: 500 Number of concurrent short transactions in milliseconds: 5000	• Storage-compute integration: Number of concurrent complex queries in minutes: 80 Number of short queries in seconds: 500 Number of concurrent short transactions in milliseconds: 5000 • Decoupled storage and compute: The multi-VW technology can increase the number of concurrent requests. As the number of VWs increases, the number of concurrent requests can be increased accordingly. The total number of concurrent requests in a cluster is affected by

Technical Specifications	Maximum Value of 8.1.3	Maximum Value of 8.2.0	Maximum Value of 8.2.1	Maximum Value of 8.3.0	Maximum Value of 9.1.0
					the GTM/CCN queuing. It is recommended that the number of concurrent requests be no more than 8192.
Cluster data capacity	20 PB	20 PB	20 PB	20 PB	• Storage-compute integration: 20 PB • Decoupled storage and compute: Data is stored on OBS. Theoretically, the capacity can be expanded infinitely.
Size of a single table	1 PB	1 PB	1 PB	1 PB	1 PB
Size of data in each row	1 GB	1 GB	1 GB	1 GB	1 GB
Number of columns in a single table: (excluding Hudi tables)	1600	1600	1600	1600	1600

Technical Specifications	Maximum Value of 8.1.3	Maximum Value of 8.2.0	Maximum Value of 8.2.1	Maximum Value of 8.3.0	Maximum Value of 9.1.0
Number of columns in a Hudi table	N/A	N/A	5000	5000	5000
Number of partitions of the partitioned table	32,768	32,768	32,768	32,768	The maximum value is 32768. It is recommended that the value be no more than 1000.
RTO after a SPOF	60s	60s	60s	60s	60s
RPO after a SPOF	0	0	0	0	0
RTO after cluster DR switchover	60min	60min	60min	60min	60min
RPO after cluster DR switchover	60min	60min	60min	60min	60min

NOTE

Virtual Warehouse (VW): also called compute group. DWS storage-compute decoupling splits a physical cluster into multiple VWs. Different services can be bound to different VWs to isolate service loads and increase the number of concurrent services.

Table 1-10 Technical specifications of DWS 8.0.x-8.1.1

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

Technical Specifications	Maximum Value of 8.0.*x*	Maximum Value of 8.1.0	Maximum Value of 8.1.1
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^{55}	2^{55}	2^{55}
Number of columns in each table	1600	1600	1600
Number of indexes in each table	Unlimited	Unlimited	Unlimited
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	Number of concurrent complex queries in minutes: 60 Number of concurrent short transactions in milliseconds: 5000	Number of concurrent complex queries in minutes: 60 Number of concurrent short transactions in milliseconds: 5000	Number of concurrent complex queries in minutes: 80 Number of concurrent short transactions in milliseconds: 5000
Number of partitions in a partitioned table	32,768	32,768	32,768
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}

NOTE

The maximum number of concurrent connections is based on the data warehouse with the cloud disk flavor of 48 vCPUs or 64 vCPUs. For example, you can choose **dwsk.12xlarge (48 vCPU | 384GB | 24000GB SSD)** or **dwsx2.16xlarge.m7 (64 vCPU | 512GB | 32000GB SSD)** for a storage-compute coupled data warehouse.

2 Getting Started

2.1 Step 1: Starting Preparations

This guide is an introductory tutorial that demonstrates how to create a sample 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 DWS.

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

- **Determining the Cluster Ports**

Determining the Cluster Ports

- When creating a 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 port number ranges 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 DWS to analyze data, create a cluster. A cluster consists of multiple nodes in the same subnet. These nodes jointly provide services. This section describes how to create a DWS cluster.

Creating a Cluster

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

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

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

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

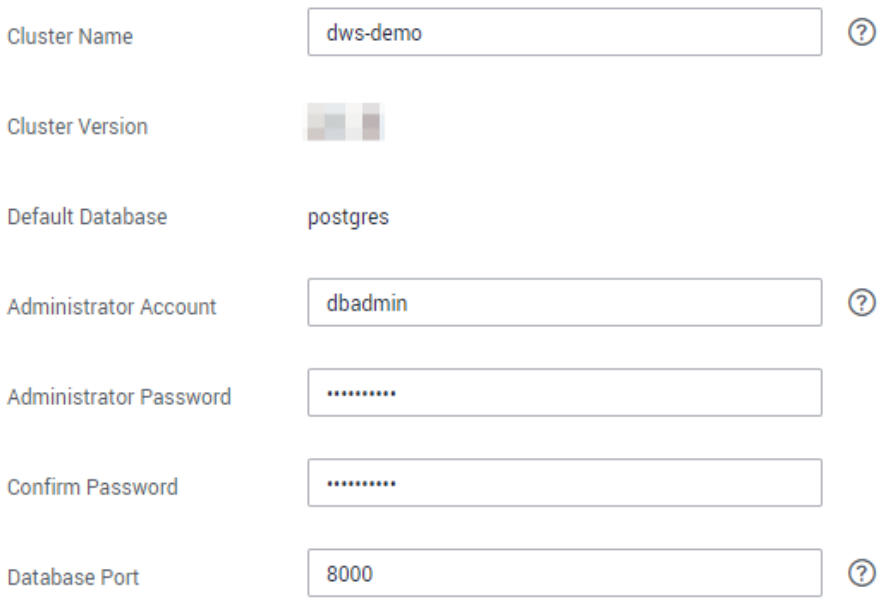
Step 5 Configure node parameters.

- **Version**: Select a data warehouse cluster type as needed, for example, **Coupled storage and compute**.
- **CPU Architecture**: Select a CPU architecture based on your requirements, for example, **x86**.
- **Node Flavor**: Retain the default value.
- **Nodes**: Retain the default value. At least **3** nodes are required.

Step 6 Configure cluster parameters.

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

Figure 2-1 Configuring the cluster



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 DWS 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 **DWS**-<*Cluster name*>-<*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 DWS. **Table 2-1** shows an example. For details about how to add an inbound rule, see "Security > Security Group > Adding a Security Group Rule" in the *Virtual Private Cloud User Guide*.

  Table 2-1 Inbound rule example

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

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

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

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

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

- **Default**: indicates that the following advanced settings use the default configurations.
  - **CNs**: Three CNs are deployed by default.

  - **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 advanced parameters **Automated Snapshot**, **Tag**, **Encrypt DataStore**, and **CNs**.

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

Step 11 Click **Submit**.

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

----End

2.3 Step 3: Connecting to a Cluster

Scenario

This section describes how to use a database client to connect to the database in a 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 **Overview**.

1. Obtain the name, username, and password of the database to be connected.

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

Obtaining the Public Network Address of the Cluster

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

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 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 DWS console.

Step 3 Click **Client Connections**.

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

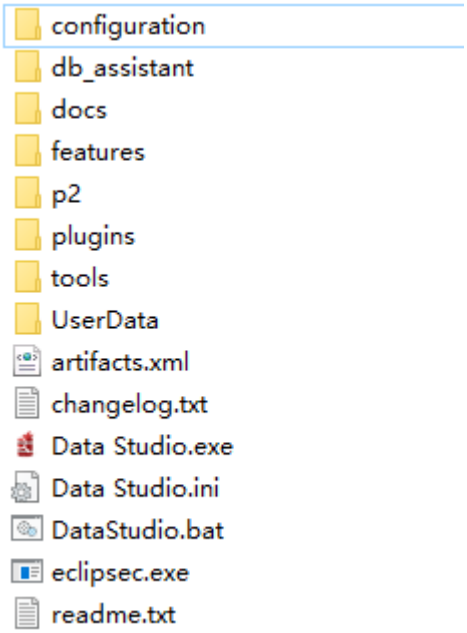
- Select **Windows x86** or **Windows x64** based on the operating system type and click **Download** to download the Data Studio tool matching the current cluster version.

  If clusters of different versions are available, you will download the Data Studio tool matching the earliest cluster version after clicking **Download**. If there is no cluster, you will download the Data Studio tool of the earliest version after clicking **Download**. 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 the Data Studio based on the cluster version.

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

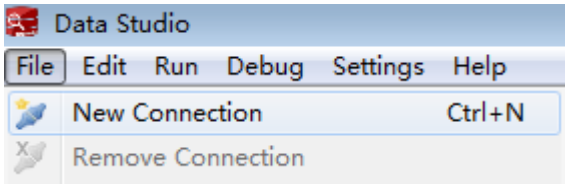
Step 6 Open the installation directory and double-click **Data Studio.exe** to start the Data Studio client. See **Figure 2-2**.

Figure 2-2 Starting the client



Step 7 Choose **File** > **New Connection** from the main menu. See **Figure 2-3**.

Figure 2-3 Creating a connection



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

Table 2-2 Connection parameters

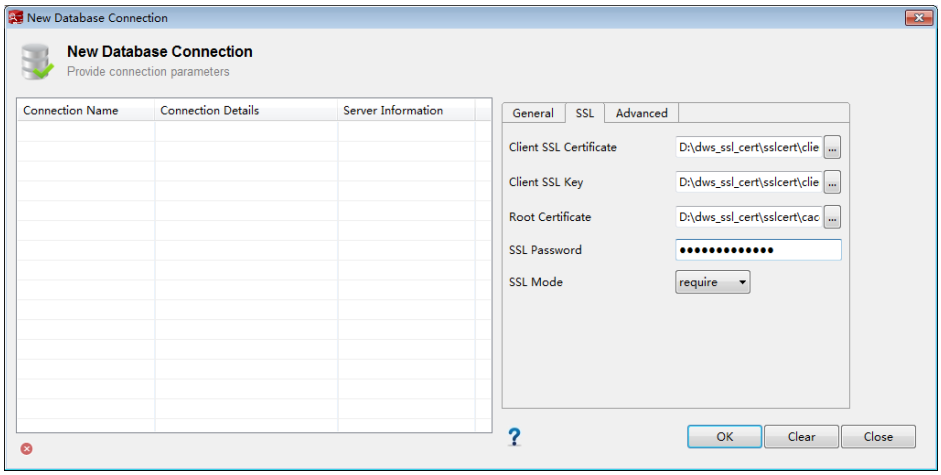
Parameter	Description	Example
Database Type	Select **GaussDB A**.	GaussDB A
Connection Name	Name of a connection	dws-demo
Host	IP address (IPv4) or domain name of the cluster to be connected	-
Host Port	Database port	8000
Database Name	Database name	gaussdb
User Name	Username for connecting to the database	-
Password	Password for logging in to the database to be connected	-
Save Password	Select an option from the drop-down list: • **Current Session Only**: The password is saved only in the current session. • **Do Not Save**: The password is not saved.	-
Enable SSL	If **Enable SSL** is selected, the client can use SSL to encrypt connections. The SSL mode is more secure than common modes, so you are advised to enable SSL connection.	-

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

Table 2-3 Configuring SSL parameters

Parameter	Description
Client SSL Certificate	Select the **sslcert\client.crt** file in the decompressed SSL certificate directory.
Client SSL Key	Only the PK8 format is supported. Select the **sslcert\client.key.pk8** file in the directory where the SSL certificate is decompressed.
Root Certificate	When **SSL Mode** is set to **verify-ca**, the root certificate must be configured. Select the **sslcert\cacert.pem** file in the decompressed SSL certificate directory.
SSL Password	Set the password for the client SSL key in PK8 format.
SSL Mode	DWS supports the following SSL modes: • require • verify-ca DWS does not support the **verify-full** mode.

Figure 2-4 Configuring SSL parameters

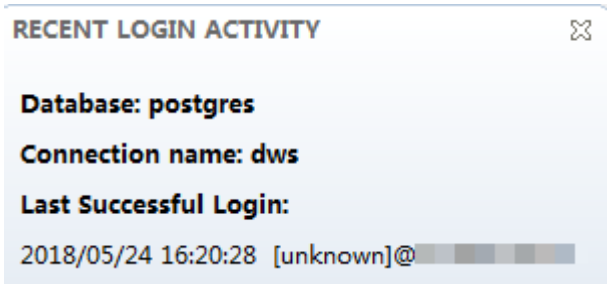


Step 9 Click **OK** to establish the database connection.

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 2-5 Successful login



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

----End

2.4 Step 4: Viewing Other Documents and Deleting Resources

Viewing Other Relevant Documents

After performing the steps in preceding sections, you can refer to the documentation listed as follows for more information about 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 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 the DWS cluster.

On the DWS console, click **Dedicated Clusters** > **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 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 Using DWS

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. DWS provides competitive solutions for PB-level big data analysis in various industries.

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

Procedure

Figure 3-1 Procedure for using DWS

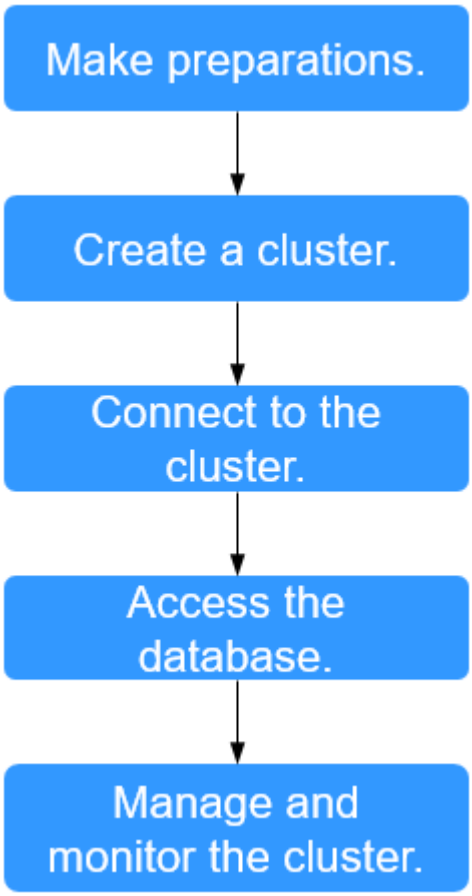


Table 3-1 Procedure description

Process	Task	Description	Operation Instruction
Make preparations.	-	Before using DWS, you need to apply for a Cloud account.	**Preparations**
Create a cluster.	-	Create a cluster before using DWS to execute data analysis tasks. A DWS cluster contains nodes in the same subnet. These nodes jointly provide services. During cluster creation, the system creates a default database.	• **Creating a DWS Storage-Compute Coupled Cluster**

Process	Task	Description	Operation Instruction
Connect to the cluster.	-	After the DWS cluster is created, use the SQL client tool or a third-party driver such as JDBC or ODBC to connect to the database in the cluster. You can download the SQL client tool and JDBC/ODBC driver on the **Client Connections** page of the DWS console.	**Connecting to a DWS Cluster**
Access the database.	-	After connecting to the cluster, you can create and manage databases, manage users and permissions, import and export data, and query and analyze data.	*Data Warehouse Service (DWS) Developer Guide*
Manage and monitor the cluster.	Cluster management	View the cluster status, modify cluster configurations, add cluster tags, and scale out, restart, and delete the cluster.	**DWS Cluster Management**
	Snapshot management	Create snapshots to back up and restore the cluster.	**Backing Up and Restoring a DWS Cluster**
	O&M and monitoring	View the running status and performance of the cluster through cluster monitoring, log auditing, event notification, and alarm management.	• **Viewing DWS Cluster Monitoring Information on Cloud Eye** • **Viewing and Subscribing to DWS Cluster Events** • **Viewing and Subscribing to DWS Cluster Alarms**

Process	Task	Description	Operation Instruction
	Scaling and specification change	• Expand the capacity of an existing cluster on the console if your service requires additional compute or storage resources. • Change the specifications of created clusters on the console.	• **Scaling Nodes**
	Cluster upgrade	Cluster 8.1.1 and later versions allow users to deliver cluster upgrade operations on the console.	**Upgrading a DWS Cluster**
	Resource load management	DWS provides the resource management function. You can put resources (CPU, memory, I/O, and storage space) into different resource pools, which are isolated from each other.	**DWS Resource Load Management**

4 Preparations

4.1 Syntax of Fine-Grained Permissions Policies

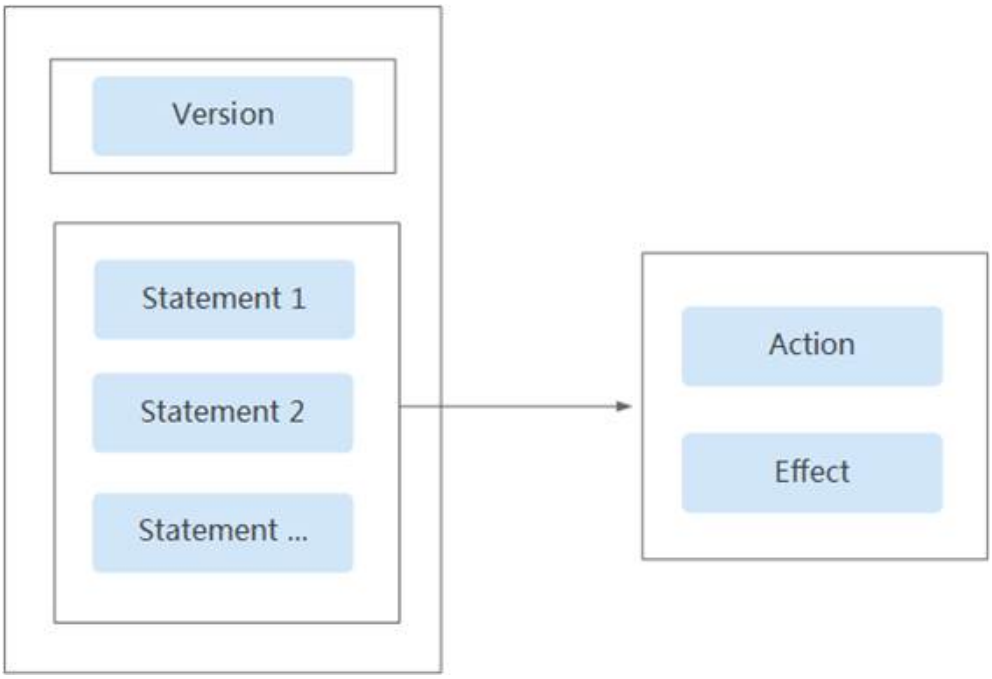
In actual services, you may need to grant different operation permissions on resources to users of different roles. The IAM service provides fine-grained access control. An IAM administrator (a user in the **admin** group) can create a custom policy containing required permissions. After a policy is granted to a user group, users in the group can obtain all permissions defined by the policy. In this way, IAM implements fine-grained permission management.

To control the DWS operations on resources more precisely, you can use the user management function of IAM to grant different operation permissions to users of different roles for fine-grained permission control.

Policy Structure

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

Figure 4-1 Policy structure



Policy Syntax

In the navigation pane on the IAM console, click **Policies** and then click the name of a policy to view its details. The **DWS ReadOnlyAccess** policy is used as an example to describe the syntax of fine-grained policies.

```
{
    "Version": "1.1",
    "Depends": [],
    "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.

- **Depends**: dependency item.
- **Statement**: Permissions defined by a policy, including Effect and Action.
  - Effect

    The value of **Effect** can be **Allow** or **Deny**. System policies contain only **Allow** statements. For custom policies containing both **Allow** and **Deny** statements, **Deny** statements take precedence over **Allow** statements.
  - Action

    Actions allowed on resources. An action is in the format of *Service name*:*Resource type*:*Action*. A policy can contain one or more actions. You can use a wildcard (*) to indicate all services, resource types, or actions.

    Example: **dws:cluster:create** (permission for create data warehouse clusters)

List of Supported Actions

When creating a custom policy on IAM, you can add the operations on 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 DWS permissions.

- **REST API**

  For details about REST API actions supported by DWS, see "Permissions Policies and Supported Actions" in *Data Warehouse Service (DWS) API Reference*.
- **Management console operations**

  Table 4-1 describes the DWS operations on resources and corresponding permissions.

  NOTE

  - Some DWS permissions depend on the actions of ECS, VPC, EVS, ELB, MRS, and OBS. Grant DWS the required service admin permissions.
  - The table shows frequently used DWS APIs, but some only allow project-based authentication (IAM authentication) and not enterprise project authentication. To use these APIs, they must be configured on the IAM authentication page.

Table 4-1 DWS permissions

Operation	Permission	Dependent Permission	Scope
Creating a cluster	"dws:cluster:create"	"dws:*:get*", "dws:*:list*", "ecs:*:get*", "ecs:*:list*", "ecs:*:create*", "vpc:*:get*", "vpc:*:list*", "vpc:*:create*", "vpc:securityGroupRules:delete", "vpc:ports:update", "evs:*:get*", "evs:*:list*", "evs:*:create*",	• Scope: – Project
Obtaining the cluster list	"dws:cluster:list"	--	• Scope: – Project
Obtaining the details of a cluster	"dws:cluster:getDetail"	"dws:*:get*", "dws:*:list*", "vpc:vpcs:list", "vpc:securityGroups:get"	• Scope: – Project
Setting automated snapshot policy	"dws:cluster:setAutomatedSnapshot"	"dws:backupPolicy:list"	• Scope: – Project
Setting security parameters/ parameter groups	"dws:cluster:setSecuritySettings"	"dws:*:get*", "dws:*:list*",	• Scope: – Project
Restarting a Cluster	"dws:cluster:restart"	"dws:*:get*", "dws:*:list*",	• Scope: – Project

Operation	Permission	Dependent Permission	Scope
Scaling out clusters	"dws:cluster:scaleOut"	"dws:*:get*", "dws:*:list*", "dws:cluster:scaleOutOrOpenAPIResize", "ecs:*:get*", "ecs:*:list*", "ecs:*:create*", "vpc:*:get*", "vpc:*:list*", "vpc:*:create*", "vpc:*:update*", "evs:*:get*", "evs:*:list*", "evs:*:create*",	• Scope: – Project
Scaling out or resizing a cluster via API	"dws:cluster:scaleOutOrOpenAPIResize"	"dws:*:get*", "dws:*:list*", "vpc:vpcs:list", "vpc:ports:create", "vpc:ports:get", "vpc:ports:update", "vpc:subnets:get", "vpc:subnets:update", "vpc:subnets:create", "vpc:routers:get", "vpc:routers:update", "vpc:networks:create", "vpc:networks:get", "vpc:networks:update", "ecs:serverInterfaces:use", "ecs:serverInterfaces:get", "ecs:cloudServerFlavors:get"	• Scope: – Project – Enterprise Project

Operation	Permission	Dependent Permission	Scope
Resetting Your Password	"dws:cluster:resetPassword"	"dws:*:get*", "dws:*:list*",	• Scope: – Project
Deleting a cluster	"dws:cluster:delete"	"dws:*:get*", "dws:*:list*", "ecs:*:get*", "ecs:*:list*", "ecs:*:delete*", "vpc:*:get*", "vpc:*:list*", "vpc:*:delete*", "evs:*:get*", "evs:*:list*", "evs:*:delete*",	• Scope: – Project
Configuring maintenance windows	"dws:cluster:setMaintainceWindow"	"dws:*:get*", "dws:*:list*",	• Scope: – Project
Binding EIPs	"dws:eip:operate"	"dws:*:get*", "dws:*:list*", "eip:*:get*", "eip:*:list*"	• Scope: – Project
Unbinding EIPs	"dws:eip:operate"	"dws:*:get*", "dws:*:list*", "eip:*:get*", "eip:*:list*"	• Scope: – Project

Operation	Permission	Dependent Permission	Scope
Creating MRS connections	"dws:MRSConnection:create"	"dws:*:get*", "dws:*:list*", "mrs:*:get*", "mrs:*:list*", "mrs:cluster:create", "ecs:*:get*", "ecs:*:list*", "ecs:*:create*", "vpc:*:get*", "vpc:*:list*", "vpc:*:create*", "evs:*:get*", "evs:*:list*", "evs:*:create*"	• Scope: – Project
Updating MRS connections	"dws:MRSConnection:update"	"dws:*:get*", "dws:*:list*", "mrs:*:get*", "mrs:*:list*", "mrs:cluster:create", "ecs:*:get*", "ecs:*:list*", "ecs:*:create*", "vpc:*:get*", "vpc:*:list*", "vpc:*:create*", "evs:*:get*", "evs:*:list*", "evs:*:create*"	• Scope: – Project

Operation	Permission	Dependent Permission	Scope
Deleting MRS connections	"dws:MRSConnection:delete"	"dws:*:get*", "dws:*:list*", "mrs:*:get*", "mrs:*:list*", "mrs:cluster:create" "ecs:*:get*", "ecs:*:list*", "ecs:*:delete*", "vpc:*:get*", "vpc:*:list*", "vpc:*:delete*", "evs:*:get*", "evs:*:list*", "evs:*:delete*",	• Scope: – Project
MRS data source list	"dws:MRSSource:list"	"mrs:cluster:list", "mrs:tag:listResource", "mrs:tag:list", "dws:*:get*", "dws:*:list*"	• Scope: – Project
Adding/Deleting tags	"dws:tag:addAndDelete"	"dws:*:get*", "dws:*:list*", "dws:openAPITag:update", "dws:openAPITag:getResourceTag",	• Scope: – Project
Editing tags	"dws:tag:edit"	"dws:*:get*", "dws:*:list*", "dws:openAPITag:update", "dws:openAPITag:getResourceTag",	• Scope: – Project
Creating a snapshot	"dws:snapshot:create"	"dws:*:get*", "dws:*:list*",	• Scope: – Project
Obtaining the snapshot list	"dws:snapshot:list"	--	• Scope: – Project

Operation	Permission	Dependent Permission	Scope
Viewing the snapshot list of a cluster	"dws:clusterSnapshot:list"	"dws:cluster:list", "dws:openAPICluster:getDetail"	• Scope: – Project
Deleting snapshots	"dws:snapshot:delete"	"dws:snapshot:list"	• Scope: – Project
Copying snapshots	"dws:snapshot:copy"	"dws:snapshot:list", "dws:snapshot:create"	• Scope: – Project
Restoring data to a new cluster	"dws:cluster:restore"	"dws:*:get*", "dws:*:list*", "ecs:*:get*", "ecs:*:list*", "ecs:*:create*", "vpc:*:get*", "vpc:*:list*", "vpc:*:create*", "evs:*:get*", "evs:*:list*", "evs:*:create*"	• Scope: – Project
Resizing a cluster	"dws:cluster:resize"	"dws:*:get*", "dws:*:list*", "ecs:*:get*", "ecs:*:list*", "ecs:*:create*", "ecs:*:delete*", "vpc:*:get*", "vpc:*:list*", "vpc:*:create*", "vpc:*:delete*", "evs:*:get*", "evs:*:list*", "evs:*:create*", "evs:*:delete*"	• Scope: – Project
Switchback	"dws:cluster:switchover"	"dws:*:get*", "dws:*:list*"	• Scope: – Project

Operation	Permission	Dependent Permission	Scope
Querying the ELB list	"dws:elb:list"	"dws:*:get*", "dws:*:list*", "elb:*:get*", "elb:*:list*",	• Scope: – Project
Associating ELB	"dws:elb:bind"	"dws:*:get*", "dws:*:list*", "ecs:*:get*", "ecs:*:list*", "vpc:*:get*", "vpc:*:list*", "evs:*:get*", "evs:*:list*", "elb:*:get*", "elb:*:list*", "elb:*:delete*", "elb:*:create*",	• Scope: – Project
Disassociating ELB	"dws:elb:unbind"	"dws:*:get*", "dws:*:list*", "ecs:*:get*", "ecs:*:list*", "vpc:*:get*", "vpc:*:list*", "evs:*:get*", "evs:*:list*", "elb:*:get*", "elb:*:list*", "elb:*:delete*",	• Scope: – Project
Querying snapshot configurations	"dws:snapshotConfig:list"	"dws:*:get*", "dws:*:list*",	• Scope: – Project
Updating a snapshot policy	"dws:backupPolicyDetail:update"	"dws:*:get*", "dws:*:list*",	• Scope: – Project
Deleting a snapshot policy	"dws:backupPolicy:delete"	"dws:*:get*", "dws:*:list*",	• Scope: – Project
Querying a snapshot policy	"dws:backupPolicy:list"	"dws:cluster:list"	• Scope: – Project

Operation	Permission	Dependent Permission	Scope
Querying cluster encryption information	"dws:clusterEncryptInfo:list"	"dws:*:get*", "dws:*:list*", "KMS Administrator"	• Scope: – Project
Creating an agent	"dws:createAgency:create"	"dws:*:get*", "dws:*:list*", "security administrator"	• Scope: – Project
Querying OBS bucket information	"dws:queryBuckets:list"	"dws:*:get*", "dws:*:list*",	• Scope: – Project
Adding a node	"dws:expandWithExistedNodes:update"	"dws:*:get*", "dws:*:list*", "ecs:*:get*", "ecs:*:list*", "ecs:*:create*", "vpc:*:get*", "vpc:*:list*", "vpc:*:create*", "vpc:*:update*", "evs:*:get*", "evs:*:list*", "evs:*:create*",	• Scope: – Project
Deleting a DR backup	"dws:disasterRecovery:delete"	"dws:*:get*", "dws:*:list*", "ecs:*:get*", "ecs:*:list*", "ecs:*:delete*", "vpc:*:get*", "vpc:*:list*", "vpc:*:delete*", "evs:*:get*", "evs:*:list*", "evs:*:delete*"	• Scope: – Project

Operation	Permission	Dependent Permission	Scope
Creating a DR backup	"dws:disasterRecovery:create"	"dws:*:get*", "dws:*:list*", "ecs:*:get*", "ecs:*:list*", "ecs:*:create*", "vpc:*:get*", "vpc:*:list*", "vpc:*:create*", "evs:*:get*", "evs:*:list*", "evs:*:create*",	• Scope: – Project
Other DR and backup operations	"dws:disasterRecovery:otherOperate"	"dws:*:get*", "dws:*:list*", "ecs:*:get*", "ecs:*:list*", "ecs:*:create*", "vpc:*:get*", "vpc:*:list*", "vpc:*:create*", "evs:*:get*", "evs:*:list*", "evs:*:create*"	• Scope: – Project
Querying DR and backup operations	"dws:disasterRecovery:get"	"dws:*:get*", "dws:*:list*", "ecs:*:get*", "ecs:*:list*", "vpc:*:get*", "vpc:*:list*", "evs:*:get*", "evs:*:list*"	• Scope: – Project
Adding a CN	"dws:module:install"	"dws:*:get*", "dws:*:list*",	• Scope: – Project
Deleting a CN	"dws:module:uninstall"	"dws:*:get*", "dws:*:list*",	• Scope: – Project
Removing nodes	"dws:clusterNodes:operate"	"dws:*:get*", "dws:*:list*"	• Scope: – Project

Operation	Permission	Dependent Permission	Scope
Updating the node alias	dws:instanceAliasName:update	dws:cluster:list	• Scope: – Project
Redistributing data	"dws:redistribution:operate"	"dws:*:get*", "dws:*:list*",	• Scope: – Project
Querying redistribution	"dws:redistributionInfo:list"	"dws:*:get*", "dws:*:list*",	• Scope: – Project
Stopping redistribution	"dws:redistribution:suspend"	"dws:*:get*", "dws:*:list*",	• Scope: – Project
Resuming redistribution	"dws:redistribution:recover"	"dws:*:get*", "dws:*:list*",	• Scope: – Project
Querying product specifications	"dws:specProduct:list"	"dws:*:get*", "dws:*:list*", "ecs:*:get*", "ecs:*:list*"	• Scope: – Project
Performing a check before cluster creation	"dws:checkCluster:create"	"dws:*:get*", "dws:*:list*", "ecs:*:get*", "ecs:*:list*", "ecs:*:create*", "vpc:*:get*", "vpc:*:list*", "vpc:*:create*", "evs:*:get*", "evs:*:list*", "evs:*:create*",	• Scope: – Project
Binding the management plane IP address	"dws:bindManageIp:operate"	"dws:*:get*", "dws:*:list*"	• Scope: – Project
Obtaining user authorization	"dws:checkAuthorize:operate"	"dws:*:get*", "dws:*:list*", "dws:checkSupport:operate"	• Scope: – Project
Authorizing a user	"dws:authorize:operate"	"dws:*:get*", "dws:*:list*", "dws:checkSupport:operate"	• Scope: – Project

Operation	Permission	Dependent Permission	Scope
Querying user databases	"dws:userDatabase:list"	"dws:*:get*", "dws:*:list*", "dws:checkSupport:operate"	• Scope: – Project
Querying user schemas	"dws:schemas:list"	"dws:*:get*", "dws:*:list*", "dws:checkSupport:operate"	• Scope: – Project
Querying user tables	"dws:tables:list"	"dws:*:get*", "dws:*:list*",	• Scope: – Project
Restoring tables	"dws:tableRestore:operate"	"dws:*:get*", "dws:*:list*",	• Scope: – Project
Checking the name of the table to be restored	"dws:tableRestoreCheck:operate"	"dws:*:get*", "dws:*:list*",	• Scope: – Project
Checking whether a cluster supports fine-grained backup	"dws:checkSupport:operate"	"dws:*:get*", "dws:*:list*",	• Scope: – Project
Querying the list of flavors that can be changed	"dws:supportFlavors:list"	"dws:*:get*", "dws:*:list*",	• Scope: – Project
Changing the node flavor	"dws:specResize:operate"	"dws:*:get*", "dws:*:list*", "ecs:*:get*", "ecs:*:list*", "ecs:*:create*"	• Scope: – Project
Stopping snapshot creation	"dws:snapshot:stop"	"dws:snapshot:list"	• Scope: – Project
Terminating a session	"dws:dmsSession:terminate"	"dws:dmsGrpcOuter:operation"	• Scope: – Project
Workload report operations	"dws:dmsWorkloadDiagnosisReport:create"	"dws:dmsGrpcOuter:operation"	• Scope: – Project

Operation	Permission	Dependent Permission	Scope
Modifying an alarm rule	"dws:dmsAlarmRule:update"	"dws:dmsQuery:list"	• Scope: – Project
Enabling an alarm rule	"dws:dmsAlarmRule:enable"	"dws:dmsQuery:list"	• Scope: – Project
Enabling a cluster alarm	"dws:dmsClusterAlarm:enable"	"dws:dmsQuery:list"	• Scope: – Project
Disabling a cluster alarm	"dws:dmsClusterAlarm:disable"	"dws:dmsQuery:list"	• Scope: – Project
gRPC external service	"dws:dmsGrpcOuter:operation"	"dws:dmsQuery:list", "dws:cluster:setSecuritySettings", "obs:bucket:ListAllMyBuckets"	• Scope: – Project
Adding a SQL probe	"dws:dmsProbe:add"	"dws:dmsGrpcOuter:operation"	• Scope: – Project
Modifying a SQL probe	"dws:dmsProbe:update"	"dws:dmsGrpcOuter:operation"	• Scope: – Project
Deleting a SQL probe	"dws:dmsProbe:delete"	"dws:dmsGrpcOuter:operation"	• Scope: – Project
Enabling or disabling a SQL probe	"dws:dmsProbe:enable"	"dws:dmsGrpcOuter:operation"	• Scope: – Project
Creating a User panel	"dws:dmsUserBoard:create"	"dws:dmsQuery:list"	• Scope: – Project
Modifying a user panel	"dws:dmsUserBoard:update"	"dws:dmsQuery:list"	• Scope: – Project
Deleting a user panel	"dws:dmsUserBoard:delete"	"dws:dmsQuery:list"	• Scope: – Project
Terminating a query	"dws:dmsQuery:terminate"	"dws:dmsGrpcOuter:operation"	• Scope: – Project
Enabling or disabling DMS	"dws:dmsService:enableOrDisable"	"dws:dmsQuery:list"	• Scope: – Project

Operation	Permission	Dependent Permission	Scope
Modifying DMS storage configurations	"dws:dmsStorageConfig:modify"	"dws:dmsQuery:list"	• Scope: – Project
Obtaining, or creating a DDL review	"dws:dmsDdlExamine:getOrCreate"	"dws:dmsGrpcOuter:operation"	• Scope: – Project
Workload snapshot operations	"dws:dmsWorkloadDiagnosisSnapshot:create"	"dws:dmsGrpcOuter:operation"	• Scope: – Project
Creating an alarm rule	"dws:dmsAlarmRule:add"	"dws:dmsQuery:list"	• Scope: – Project
Deleting an alarm rule	"dws:dmsAlarmRule:delete"	"dws:dmsQuery:list"	• Scope: – Project
Executing a SQL probe	"dws:dmsProbe:execute"	"dws:dmsGrpcOuter:operation"	• Scope: – Project
Deleting a monitoring item	"dws:dmsPerformanceMonitor:delete"	"dws:dmsQuery:list"	• Scope: – Project
Enabling or disabling DMS monitoring metrics	"dws:dmsCollectItem:enableOrDisable"	"dws:dmsGrpcOuter:operation"	• Scope: – Project
Modifying DMS monitoring configurations	"dws:dmsCollectConfig:modify"	"dws:dmsGrpcOuter:operation"	• Scope: – Project
OpenAPI Conditional Query	"dws:dmsOpenapiQuery:list"	"dws:cluster:list"	• Scope: – Project
Disabling an alarm rule	"dws:dmsAlarmRule:disable"	"dws:dmsQuery:list"	• Scope: – Project
Deleting an alarm record	"dws:dmsAlarmRecord:delete"	"dws:dmsQuery:list"	• Scope: – Project
Checking SQL probes	"dws:dmsProbe:check"	"dws:dmsGrpcOuter:operation"	• Scope: – Project
Adding a monitoring item	"dws:dmsPerformanceMonitor:add"	"dws:dmsQuery:list"	• Scope: – Project

Operation	Permission	Dependent Permission	Scope
Modifying monitoring metrics	"dws:dmsPerformanceMonitor:update"	"dws:dmsQuery:list"	• Scope: – Project
Downloading historical monitoring trend	"dws:dmsTrendHistory:down"	"dws:dmsQuery:list"	• Scope: – Project
Obtaining cluster ring information	"dws:ring:list"	"dws:*:get*", "dws:*:list*"	• Scope: – Project
Obtaining the cluster process topology	"dws:processTopo:list"	"dws:*:get*", "dws:*:list*"	• Scope: – Project
Querying intelligent O&M information	"dws:operationalTask:get"	"dws:*:get*", "dws:*:list*"	• Scope: – Project
Intelligent O&M Operations	"dws:operationalTask:operate"	"dws:*:get*", "dws:*:list*"	• Scope: – Project
Elastic logical cluster planning	"dws:logicalClusterPlan:operate"	"dws:*:get*", "dws:*:list*", "dws:logicalCluster:*", "dws:cluster:scaleOut", "iam:agencies:*", "iam:permissions:*Agency*"	• Scope: – Project
Creating an endpoint service	"dws:vpcEndpointService:create"	"dws:*:get*", "dws:*:list*"	• Scope: – Project
Querying the resource management list	"dws:workLoadManager:get"	"dws:*:get*", "dws:*:list*"	• Scope: – Project
Resource management operations	"dws:workLoadManager:operate"	"dws:*:get*", "dws:*:list*"	• Scope: – Project
LTS operations	"dws:ltsAccess:operate"	"dws:*:get*", "dws:*:list*"	• Scope: – Project

Operation	Permission	Dependent Permission	Scope
Querying LTS Information	"dws:ltsAccess:get"	"dws:*:get*", "dws:*:list*"	• Scope: – Project
Querying events	"dws:event:list"	"dws:*:get*", "dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project
Querying event specifications	"dws:event:list"	"dws:*:get*", "dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project
Querying event subscriptions	"dws:eventSub:list"	"dws:*:get*", "dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project
Creating an event subscription	"dws:eventSub:create"	"dws:*:get*", "dws:*:list*",	• Not supported – Enterprise Project • Scope: – Project
Updating an event subscription	"dws:eventSub:update"	"dws:*:get*", "dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project

Operation	Permission	Dependent Permission	Scope
Deleting an event subscription	"dws:eventSub:delete"	"dws:*:get*", "dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project
Querying alarm statistics	"dws:alarmStatistic:list"	"dws:*:get*", "dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project
Querying alarm details	"dws:alarmDetail:list"	"dws:*:get*", "dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project
Querying alarm configurations	"dws:alarmConfig:list"	"dws:*:get*", "dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project
Querying alarm subscriptions	"dws:alarmSub:list"	"dws:*:get*", "dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project

Operation	Permission	Dependent Permission	Scope
Creating an alarm subscription	"dws:alarmSub:create"	"dws:*:get*", "dws:*:list*",	• Not supported – Enterprise Project • Scope: – Project
Updating an alarm subscription	"dws:alarmSub:update"	"dws:*:get*", "dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project
Deleting an alarm subscription	"dws:alarmSub:delete"	"dws:*:get*", "dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project
Delivering cluster upgrade operations (upgrade, rollback, submission, and retry)	"dws:cluster:doUpdate"	"dws:*:get*", "dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project
Querying the available upgrade paths of a cluster	"dws:cluster:getUpgradePaths"	"dws:*:get*", "dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project

Operation	Permission	Dependent Permission	Scope
Querying cluster upgrade records	"dws:cluster:getUpgradeRecords"	"dws:*:get*", "dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project
Starting a cluster	"dws:cluster:startCluster"	"dws:*:get*", "dws:*:list*", "ecs:*:get*", "ecs:*:list*", "ecs:*:start", "ecs:*:stop"	• Not supported – Enterprise Project • Scope: – Project
Stopping a cluster	"dws:cluster:stopCluster"	"dws:*:get*", "dws:*:list*", "ecs:*:get*", "ecs:*:list*", "ecs:*:start", "ecs:*:stop"	• Not supported – Enterprise Project • Scope: – Project
Obtaining tags	"dws:openAPItag:list"	"dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project
Service EPS list	"dws:service:listEps"	"dws:*:list*"	• Supported: – Project
Obtaining the DR information	"dws:disasterRecovery:get"	"dws:*:*"	• Not supported – Enterprise Project • Scope: – Project

Operation	Permission	Dependent Permission	Scope
Cluster restoration check	"dws:cluster:checkRestore"	"dws:*:*"	• Not supported – Enterprise Project • Scope: – Project
Static alarm list	"dws:alarmStatistic:list"	"dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project
Obtaining static resource information	"dws:service:getResourceStatistics"	"dws:*:*"	• Not supported – Enterprise Project • Scope: – Project
Alarm details list	"dws:alarmDetail:list"	"dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project
Obtaining the cluster details	"dws:openAPICluster:getDetail"	"dws:*:*"	• Not supported – Enterprise Project • Scope: – Project

Operation	Permission	Dependent Permission	Scope
Cluster event specifications	"dws:eventSpec:list"	"dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project
Cluster DR list	"dws:cluster:listDisasterRecovery"	"dws:*:list*",	• Not supported – Enterprise Project • Scope: – Projects
Checking the alarm data overview	"dws:alarm:listStatistics"	"dws:*:list*",	• Not supported: – Enterprise projects • Supported: – Projects
Querying Schemas in a DWS Cluster	"dws:monitor:listClusterOverview"	"dws:*:get*", "dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project
Querying Historical Monitoring Data	"dws:monitor:getHistoryMetrics"	"dws:*:get*", "dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project

Operation	Permission	Dependent Permission	Scope
Column Display Configuration in the Query List	"dws:cluster:listQueryForDMS"	"dws:*:get*", "dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project
Adding or Modifying a Column in the List	"dws:cluster:listQueryForDMS"	"dws:*:get*", "dws:*:list*"	• Not supported – Enterprise Project • Scope: – Project

Authorization Using the Fine-Grained Permission Policy

Step 1 Log in to the IAM console as and create a user-defined policy.

For details, see "Fine-Grained Policy Management > Creating a Custom Policy" in the *Identity and Access Management User Guide*.

Refer to the following to create the policy:

- Use the IAM administrator account, that is, the user in the admin user group, because only the IAM administrator has the permissions to create users and user groups and modify user group permissions.
- DWS is a project-level service, so its **Scope** must be set to **Project-level service**. If this policy is required to take effect for multiple projects, authorization is required to each project.
- Two 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 DWS.
  - **DWS Viewer**: has the read-only permission on DWS.
- You can add permissions corresponding to DWS operations or RESTful APIs listed in **List of Supported Actions** to the action list in the policy authorization statement, so that the policy can obtain the permissions.

  For example, if **dws:cluster:create** is added to the action list of a policy statement, the policy has the permission to create 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 DWS cluster, you need to configure the VPC to which the cluster belongs. To obtain the VPC list, add permission **vpc:*:get*** to the policy statement.

Step 2 Create a user group.

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

Step 3 Add users to the user group and grant the new custom policy to the user group so that users in it can obtain the permissions defined by the policy.

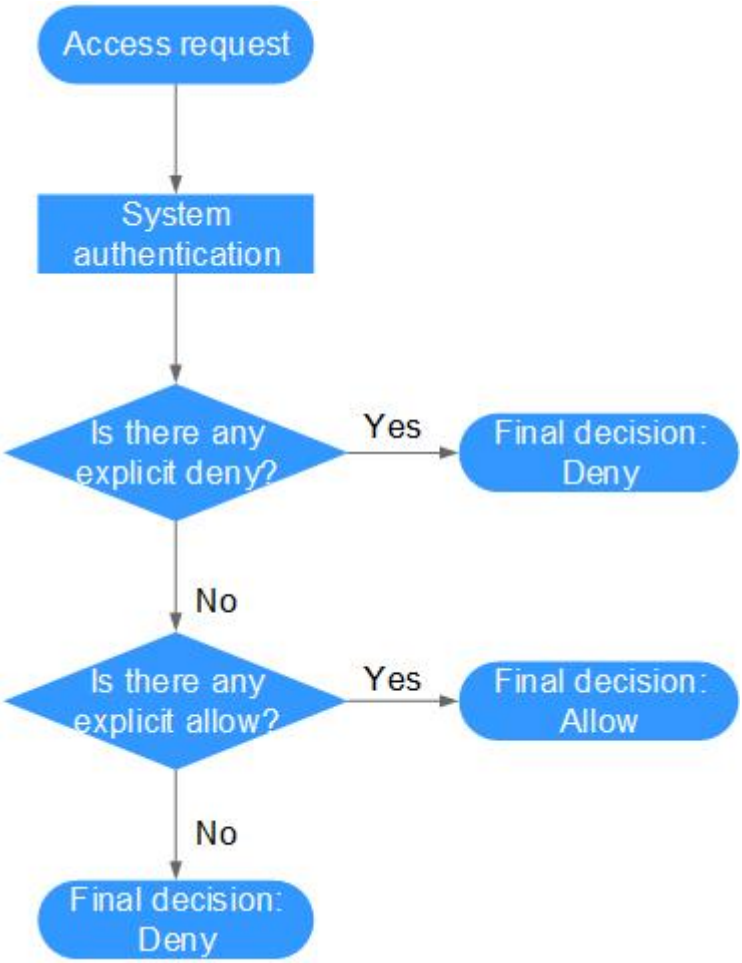
For details, see "User and User Group Management > Viewing or Modifying User Group Information" in the *Identity and Access Management User Guide*.

----End

Authentication Logic

If a user is granted permissions of multiple policies or of only one policy containing both Allow and Deny statements, then authentication starts from the Deny statements. The following figure shows the authentication logic for resource access.

Figure 4-2 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.

4.2 RBAC Syntax of RBAC Policies

Policy Structure

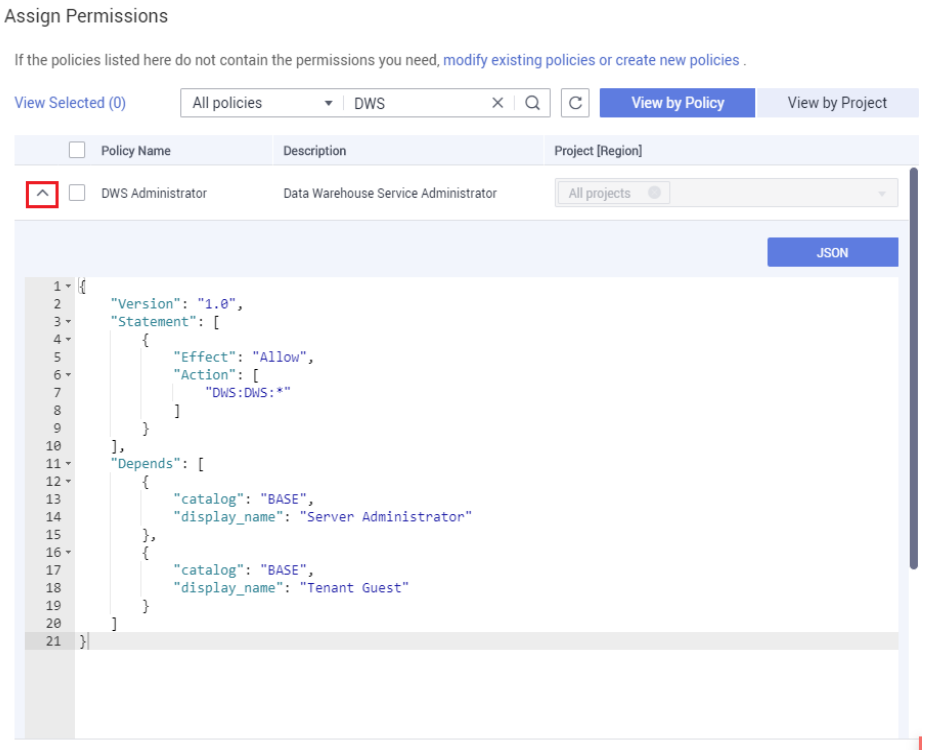
An RBAC policy consists of a Version, a Statement, and Depends.

Figure 4-3 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 4-4 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"
        }
    ]
}
```

Table 4-2 Syntax of RBAC Policies

Parameter	Meaning	Value
Version	Policy version	The value is fixed to **1.0**.

Parameter		Meaning	Value
Statement	Action	Operations to be performed on DWS	Format: *Service name:Resource type:Operation*. **dws:dws:***: Permissions for performing all operations on all resource types in DWS. **dws** indicates the service name. The asterisk (*) is a wildcard.
	Effect	Whether the operation defined in an action is allowed	• Allow • Deny
Depends	catalog	Name of the service to which dependencies of a policy belong	Service name Example: **BASE**
	display_name	Name of a dependent policy	Policy name Example: **Server Administrator**

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.

5 Creating a DWS Cluster

5.1 Creating a Dedicated DWS Cluster

5.1.1 Creating a DWS Storage-Compute Coupled Cluster

To use Huawei Cloud DWS, create a data warehouse cluster first.

This section describes how to create a data warehouse cluster on the DWS console.

Preparations Before Creating a Cluster

- You have evaluated the flavor of cluster nodes.

  You can select the number of nodes by data volume, service load, and performance. More nodes bring you stronger storage and compute capabilities.

  When first using DWS, you can create a cluster with a smaller flavor. Then, you can adjust the cluster scale and node flavor based on the data volume and service load changes without interrupting services. For details, see **Scaling Out a Cluster**.

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

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

  The number of available nodes depends on the selected product type. Three or more nodes are available.

Creating a Cluster

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

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

Step 4 Choose **Region** and select the actual working region of the cluster node.

For more information about regions, visit **Regions and Endpoints**.

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

For more information, see **How Do I Select a DWS Region and AZ?**.

NOTE

- Multi-AZ clusters are supported only by clusters of version 8.2.0.100 or later.
- The **Multi-AZ** option is displayed only if the number of AZs in the selected region is greater than or equal to 3. If this condition is not met, only a single-AZ cluster can be created.
- For a multi-AZ cluster, only three AZs can be selected at a time so far. Server nodes are evenly distributed among the three AZs.
- The numbers of nodes in a multi-AZ cluster must be a multiple of 3.
- A multi-AZ cluster only allows for up to 2 DNs on a single node.

Step 6 Set data warehouse parameters. For details, see **Table 5-1**.

Table 5-1 Node configuration parameters

Parameter	Description	Example Value
Resource	Product type. It can be: • **Coupled storage and compute**: The storage-compute coupled data warehouse provides enterprise-level data warehouse services with high performance, high scalability, high reliability, high security, low latency, and easy O&M. It is capable of data analysis at a scale of 2,048 nodes and 20 petabytes of data and is suitable for converged analysis services that integrate databases, warehouses, marts, and lakes.	-
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

Parameter	Description	Example Value
Hot storage	Available storage capacity of each node. **NOTE** • The storage capacity you apply for has the necessary file system overhead, which includes index nodes and the space required for database running. The storage space must be an integer multiple of 100. • 200 GB per node is the actual storage capacity for service data. For example, if the number of nodes is set to 3, the total resource capacity is 600 GB. • By default, tablespaces are automatically created when you configure cold and hot data storage. You do not need to manually create tablespaces. This feature is supported only in clusters of 8.1.3 and later versions.	-
Nodes	Specify the number of nodes in the cluster. The number of nodes ranges from 3 to 256.	3
Total	Displays the total capacity of a cluster. The storage capacity of each flavor is the actual database space used for storing data. The displayed storage capacity has deducted the disk space consumed by backups and RAIDs.	-

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

Step 8 Configure the network.

Table 5-2 Network parameters

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

Parameter	Description	Example Value
Security Group	Specify a VPC security group. A security group restricts access rules to enhance security when DWS and other services access each other. ● Automatic creation If **Automatic creation** is selected, the system automatically creates a default security group. This option is selected by default. The rule of the default security group is as follows: The outbound allows all access requests, while the inbound is open only to the database port that you set to connect to the DWS cluster. The format of the default security group's name is dws-<*cluster name*>-<*database port of the DWS cluster*>, for example, **dws-dws-demo-8000**. **NOTE** If the quotas of the security group and the security group rule are insufficient, an error message will be displayed after you submit the cluster creation application. Select an existing group and retry. ● Manual creation You can also log in to the VPC management console to manually create a security group. Then, go back to the page for creating DWS clusters, click next to the **Security Group** drop-down list to refresh the page, and select the new security group. To enable the DWS client to connect to the cluster, add an inbound rule to the new security group to allow access to the DWS cluster's database port. The following is an example of an inbound rule. – **Protocol**: **TCP** – **Port**: **8000**. Use the database port set when creating the DWS cluster. This port receives client connections to DWS. – **Source**: Select **IP address** and use the host IP address of the client host, for example, **192.168.0.10/32**. After a DWS cluster is created, you can change the security group. You can also add, delete, or modify security group rules in the current security group. For details, see	Automatic creation

Parameter	Description	Example Value
	"Modifying a Security Group". Changing the security group of a cluster may cause brief service disruption. Exercise caution when performing this operation. For better network performance, do not select more than five security groups.	
EIP	Specify whether users can use a client to connect to a cluster's database over the Internet. The following methods are supported: • **Do not use**: The EIP is not required. • **Automatically assign**: 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 bandwidth name of an automatically assigned EIP starts with the cluster name. • **Specify**: A specified EIP is bound to the cluster. If no available EIPs are displayed in the drop-down list, click **View EIP** to go to the EIP page and create one that meets your needs. You can set the IP address type and bandwidth as required. **NOTE** • If you use the EIP binding function for the first time in each project of each region, the system prompts you to create the **DWSAccessVPC** agency to authorize DWS to access VPC. After the authorization is successful, DWS can switch to a healthy VM when the VM bound with the EIP becomes faulty. • By default, only Huawei Cloud accounts or users with **Security Administrator** permissions can query and create agencies. By default, the IAM users in those accounts cannot query or create agencies. When the users use the EIP, the system makes the binding function unavailable. Contact a user with the **DWS Administrator** permissions to authorize the agency on the current page. • **Do not use** 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 **Creating a Public Network Domain Name**.	Automaticall y assign
Bandwidth	When **EIP** is set to **Automatically assign**, you need to specify the bandwidth of the EIP. The value ranges from 1 Mbit/s to 100 Mbit/s.	50 Mbit/s

Parameter	Description	Example Value
ELB	Specifies whether ELB is bound. With ELB health checks, CN requests of a cluster can be quickly forwarded to normal CNs. If a CN is faulty, the workload can be immediately shifted to a healthy node, minimizing cluster access faults. • **Do not use**: The load balancer is not used. • **Specify**: Specify an ELB to be bound to the cluster. If no available ELBs are displayed in the drop-down list, click **Create ELB** to go to the ELB page and create one as needed. **WARNING** Configure load balancing to ensure load balancing and high availability of the cluster and prevent service interruptions. You are not advised to directly connect services to a single CN.	Specify

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

Step 10 Configure cluster parameters.

Table 5-3 Cluster parameters

Parameter	Description	Example Value
Cluster Name	Set the name of the data warehouse cluster. Enter 4 to 64 characters. Only letters (case-insensitive), digits, hyphens (-), and underscores (_) are allowed. The name must start with a letter.	DWS-demo
Cluster Version	Displays the version of the database instance installed in the cluster. The example version number is for reference only.	-
Default Database	The default database name of the cluster is **gaussdb**. **NOTE** This name cannot be changed.	gaussdb

Parameter	Description	Example Value
Administrator Account	Set the database administrator name. The administrator username must: • Consist of lowercase letters, digits, or underscores. • Start with a lowercase letter or an underscore. • Contain 6 to 64 characters. • The username cannot be a keyword of the DWS database. For details about the keywords of the DWS database, see "SQL Reference" > "Keyword" in the *Data Warehouse Service Developer Guide*.	dbadmin
Administrator Password	Set the password of the database administrator account. The password complexity requirements are as follows: • Consists of 12 to 32 characters. • Cannot be the username or the username spelled backwards. • Must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters (~!?,.:;_(){}[]/<>@#%^&*+|\=-) • Passes the weak password check. **NOTE** Change the password regularly and keep it secure.	-
Confirm Password	Enter the database administrator password again.	-
Database Port	Specify the port used when the client or application connects to the database in the cluster. The port number ranges from 8000 to 30000. **NOTE** The database port of a created cluster cannot be changed. You can specify the database port only when creating a cluster.	8000
Time Zone	You can set the time zone for the tenant cluster, including the system OS time zone and cluster data warehouse time zone.	-

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

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

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

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

- **Backup Device**

  Set the backup device used by the current cluster. For details about the parameter configuration principles, see **Table 5-4**.

  Table 5-4 Automated snapshot parameters

Parameter	Description
Backup Device	Select **OBS** or **NFS** from the drop-down list.
NFS backup file system address (NFS parameter)	NFS shared IP address. Enter the IP address of the SFS shared path. After the mounting is successful, a mount directory is created in the **/var/chroot/nfsbackup** directory of the cluster instance by default.

- **CNs**

  CNs, or Coordinators, receive access requests from the clients and return the execution results. They also split and distribute tasks to the Datanodes (DNs) for parallel execution.

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

- **Tag**

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

  For more information about tags, see **Overview**.

Table 5-5 Tag parameters

Parameter	Description	Example Value
Key	The options are as follows: – Select a predefined tag key or an existing resource tag key from the drop-down list of the text box. **NOTE** To add a predefined tag, you need to create one on TMS and select it from the drop-down list of **Tag key**. You can click **View predefined tags** to enter the **Predefined Tags** page of TMS. Then, click **Create Tag** to create a predefined tag. For more information, see "Predefined Tags" > "Creating Predefined Tags" in the *Tag Management Service User Guide*. – Enter a tag key in the text box. A tag key can contain a maximum of 128 characters. It cannot be an empty string or start or end with a space. The value cannot contain the following characters: *<> \,|/ **NOTE** A key must be unique in a given cluster.	key01
Value	You can select: – Select a predefined tag value or resource tag value from the drop-down list of the text box. – Enter a tag value in the text box. A tag value can contain a maximum of 255 characters, which can be an empty string. It cannot start or end with a space. The value cannot contain the following characters: *<> \,|/	value01

- **Encrypt DataStore**

  If 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 **DWSAccessKMS** to authorize DWS to access KMS. If you click **No**, the encryption function is not enabled. Select the created KMS key from the **KMS Key Name** drop-down list.

NOTICE

- Only users with the Tenant Admin permission can view and toggle the **Encrypt DataStore** switch.
- By default, only Huawei Cloud accounts or users with **Security Administrator** permissions can query and create agencies. IAM users under an account do not have the permission to query or create agencies by default. Contact a user with that permission and complete the authorization on the current page.
- 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.
- After database encryption is enabled, you cannot use open APIs to restore created snapshots.

  - Method 1: Select a key name.
  - Method 2: Enter the key ID. Enter the key ID used for authorizing the current tenant.

    When you grant permissions on the Creating a Grant page, the authorized object must be an account instead of a user. The authorized operations must at least contain **Querying key details**, **Encrypting data**, and **Decrypting data**.

Step 13 Specify whether to enable the IPv6 dual stack for the cluster. If this function is enabled, a client or application can connect to the database using an IPv6 address.

To enable IPv6, the following conditions must be met:

- The subnet configured in **Step 8** is an IPv6 dual-stack subnet.
- The cluster supports IPv6 addresses and a maximum of three NICs.
- The cluster version must be 8.2.1.210 or later.

Step 14 Click **Next: Confirm**.

Step 15 Click **Create Now**

After the submission is successful, the creation starts. Click **Back to Cluster List**. The cluster management 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

Handling the Cluster Creation Failure

If a cluster fails to be created, go to the DWS console and choose **Dedicated Clusters** > **Clusters** to view the cluster status and the cause of failure.

Viewing the cause of creation failure

Step 1 Log in to the DWS console. In the navigation pane, choose **Dedicated Clusters** > **Clusters**.

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 cluster creation failure.

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 DWS console. In the navigation pane, choose **Dedicated Clusters** > **Clusters**.

Step 2 In the cluster list, locate the row containing the failed cluster to be deleted, and choose **More** > **Delete**.

Step 3 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, click **Create Snapshot**. On the snapshot list page, click **Create Snapshot** to create a snapshot for the cluster to be deleted. For details, see **Manual Snapshots**. In the row of a cluster, choose **More** > **Delete**.

- Delete associated resources.
  - Release the EIP bound to the cluster.

    If an EIP is bound to the cluster, you are advised to select **EIP** to release the EIP.
  - Delete automated snapshots.
  - Delete manual snapshots.

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

Step 4 After confirming that the information is correct, enter **DELETE** or click **Auto Enter** and click **OK** to delete the cluster. The cluster status in the cluster list will change to **Deleting** and the cluster deletion progress will be displayed.

If 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

6 Connecting to a DWS Cluster

6.1 Overview

If you have created a 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.

Constraints and Limitations

WARNING

- Avoid having all business operations run under a single database user. Instead, plan different database users according to the business modules.
- For better access control of different business modules, it is better to use multiple users and permissions instead of depending on the system administrator user to run business operations.
- You are not advised to connect services to a single CN. Instead, configure load balancing by referring to **Binding and Unbinding Load Balancers for a DWS Cluster** to ensure that connections to each CN are balanced.
- After connecting to the database and completing required operations, close the database connection in a timely manner to prevent idle connections from continuously occupying resources and consuming connections and public resources.
- In the scenario where the database connection pool is used, after the database GUC parameters are set using the **SET** statement in the service, the parameters must be restored using the **RESET** statement before the connection pool is returned.
- For more information about development and design specifications, see "DWS Development and Design Proposal" in the *Data Warehouse Service (DWS) Developer Guide*.

Connecting to a Cluster

The procedure for connecting to a cluster is as follows:

1. **Obtaining the Connection Address of a DWS Cluster**
2. If SSL encryption is used, perform the operations in **Establishing Secure TCP/IP Connections in SSL Mode**.
3. Connect to the cluster and access the database in the cluster. You can choose any of the following methods to connect to a cluster:

   NOTICE

   - You are advised to use the officially recommended method for connecting to the database.
   - Compatibility with other clients cannot be guaranteed, so it may be necessary to verify it.
   - If an error occurs due to incompatibility with another client and the client cannot be replaced, try replacing the libpq driver on the client. To replace the **libpg.so** file on the client, download and extract the gsql client package by referring to **Downloading the Client**, locate the **gsql** directory, and obtain the file. Then, replace the existing **libpg.so** file in the designated directory on the client.

   – Use the SQL client tool to connect to the cluster.
     - **Using the Linux gsql Client to Connect to a Cluster**
     - **Using the Windows gsql Client to Connect to a Cluster**
     - **Using Data Studio to Connect to a DWS Cluster**
   – Use a JDBC, psycopg2, or PyGreSQL driver to connect to the cluster.
     - **Using JDBC to Connect to a Cluster**
     - **Using ODBC to Connect to a Cluster**
     - **Using the Python Library psycopg2 to Connect to a DWS Cluster**
     - **Using the Python Library PyGreSQL to Connect to a DWS Cluster**

6.2 Obtaining the Connection Address of a DWS Cluster

Scenario

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

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

- **Obtaining the cluster connection address on the Client Connections Page**
- **Obtaining the Cluster Access Addresses on the Cluster Information Page**

Obtaining the cluster connection address on the Client Connections Page

Step 1 Log in to the DWS console.

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

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

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

Step 4 View and obtain the cluster connection information.

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

NOTE

- If no EIP is automatically assigned during cluster creation, **Public Network Address** is empty. If you want to use a public network address (consisting of an EIP and the database port) to access the cluster from the Internet, click **Bind EIP** to bind one.
- If an EIP is bound during cluster creation but you do not want to use the public network address to access the cluster, click **Unbind EIP** to unbind the EIP. After the EIP is unbound, **Public Network Address** is empty.
- If a cluster was not bound to ELB when it was created, the **ELB Address** parameter will be left blank. You can bind the cluster to ELB to avoid single CN failures.
- If a cluster has been bound to ELB, use the ELB address to connect to the cluster for high availability purposes.
- If the IPv6 dual stack is enabled for a DWS cluster, private IPv4 and IPv6 addresses can be used. You can connect to the cluster via IPv4 or IPv6.

----End

Obtaining the Cluster Access Addresses on the Cluster Information Page

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

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

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

Table 6-1 Connection

Parameter	Description
Private Network Domain Name	Domain name for accessing the cluster database through the internal network. The domain name corresponds to all CN IP addresses. The private network domain address is automatically generated when a cluster is created. **NOTE** • If the cluster name does not comply with the domain name standards, the prefix of the default access domain name will be adjusted accordingly. • Load balancing is not supported. You can click **Modify** to change the private network domain name. The access domain name contains 4 to 63 characters, which consists of letters, digits, and hyphens (-), and must start with a letter.
Private Network IP Address	IP address for accessing the database in the cluster over the private network. **NOTE** • A private IP address is automatically generated when you create a cluster. The IP address is fixed. • The number of private IP addresses equals the number of CNs. You can log in to any node to connect to the cluster. • If you access a fixed IP address over the internal network, all the resource pools will run on a single CN. • If IPv6 is enabled for a cluster, both IPv4 and IPv6 private addresses will be displayed. You can use either of them as needed.
Public Network Domain Name	Name of the domain for accessing the database in the cluster over the public network. **NOTE** Load balancing is not supported.
Public Network IP Address	IP address for accessing the database in the cluster over the public network. **NOTE** • If no EIP is assigned during cluster creation and **Public Network IP Address** is empty, click **Edit** to bind an EIP to the cluster. • If an EIP is bound during cluster creation, click **Edit** to unbind the EIP.
Initial Administrator	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 number for accessing the cluster database through the public network or private network. The port number is specified when the cluster is created.
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.

Parameter	Description
ELB Address	To achieve high availability and avoid single-CN failures, a new cluster needs to be bound to ELB. You are advised to use the ELB address to connect to the cluster. **NOTE** If the cluster is an IPv4 cluster, only IPv4 ELB can be manually bound. If the cluster is an IPv6 dual-stack cluster, only IPv6 dual-stack ELB can be manually bound.

----End

6.3 Using a Visualization Tool to Connect to a DWS Cluster

6.3.1 Using Data Studio to Connect to a DWS 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 DWS console. The package can be used without installation after being decompressed.

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

Preparations Before Connecting to a Cluster

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

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

Connecting to the Cluster Database Using Data Studio

Step 1 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 DWS console.

Step 3 Choose **Management** > **Client Connections**.

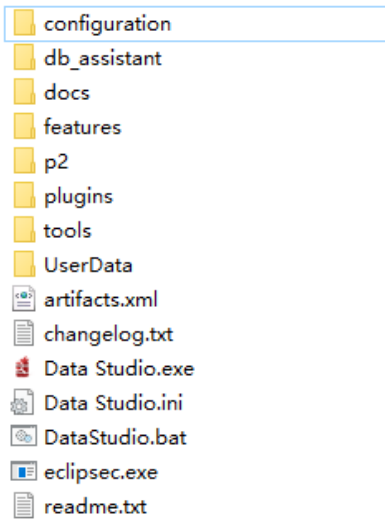
Step 4 On the **Download Client and Driver** page, download **Data Studio GUI Client**.

- Select **Windows x86** or **Windows x64** based on the OS type and click **Download** to download a Data Studio version that matches the current cluster.
- Click **Historical Version** to download the corresponding Data Studio version. You are advised to download Data Studio based on the cluster version.

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

Step 6 Open the installation directory and double-click **Data Studio.exe** to start the Data Studio client. See **Figure 6-1**.

Figure 6-1 Starting the client

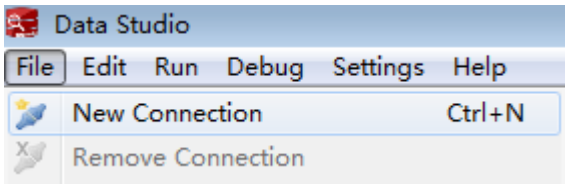


NOTE

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

Step 7 Choose **File** > **New Connection** from the main menu. See **Figure 6-2**.

Figure 6-2 New connection



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

Table 6-2 Connection parameters

Field	Description	Example Value
Database Type	Select **GaussDB A**	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: • **Current Session Only**: The password is saved only in the current session. • **Do Not Save**: The password is not saved.	-
Enable SSL	If **Enable SSL** is selected, the client can use SSL to encrypt connections. The SSL connection mode is more secure than common modes, so you are advised to enable SSL connection.	-

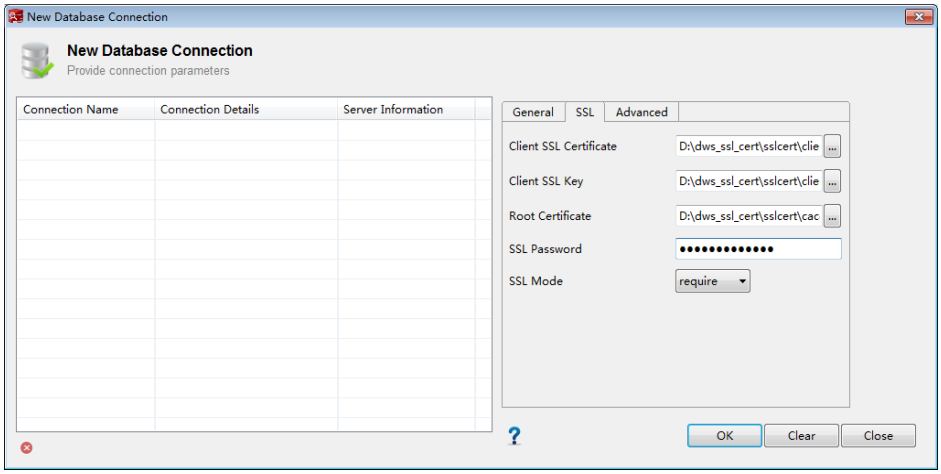
If **Enable SSL** is selected, **download the SSL certificate** and decompress it. Click the **SSL** tab and configure the following parameters:

Table 6-3 Configuring SSL parameters

Field	Description
Client SSL Certificate	Select the **sslcert\client.crt** file in the decompressed SSL certificate directory.
Client SSL Key	Only the PK8 format is supported. Select the **sslcert \client.key.pk8** file in the directory where the SSL certificate is decompressed.

Field	Description
Root Certificate	When **SSL Mode** is set to **verify-ca**, the root certificate must be configured. Select the **sslcert\cacert.pem** file in the decompressed SSL certificate directory.
SSL Cipher	Set the password for the client SSL key in PK8 format.
SSL Mode	DWS supports the following SSL modes: • require • verify-ca DWS does not support the **verify-full** mode.

Figure 6-3 Configuring SSL parameters



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

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

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

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

NOTE

- Data cannot be rolled back after being added, deleted, modified, or queried in Data Studio.
- Data Studio can save connection information, excluding passwords.
- DDL/DDL and data cannot be exported in batches for the following objects:
  - **Export DDL:**

    Connection, database, foreign table, sequence, column, index, constraint, partition, function/procedure group, regular tables group, views group, schemas group, and system catalog group.
  - **Export DDL and Data:**

    Connection, database, namespace, foreign table, sequence, column, index, constraint, partition, function/procedure, view, regular tables group, schemas group, and system catalog group.

----End

6.4 Using the CLI to Connect to a DWS Cluster

6.4.1 Downloading the Client

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

The client tool package contains the following:

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

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

  The script for testing sample data is used to execute the introductory example.
- **Windows gsql**

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

  NOTE

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

Downloading the Client

Step 1 Log in to the DWS console.

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

Step 3 Select the DWS client of the target version from the drop-down list of **CLI Client**.

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

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

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

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

File or Folder	Description
bin	This folder holds the Linux executable files for gsql, which include 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 DWS database.
lib	This folder contains the **lib** library required for executing the gsql client.
sample	This folder contains the following directories and files: – **setup.sh**: script file for configuring the AK/SK before using gsql to import sample data – **tpcds_load_data_from_obs.sql**: script file for importing the TPC-DS sample data using the gsql client – **query_sql** directory: script file for querying the TPC-DS sample data
gsql_env.sh	Script file for configuring environment variables before running the gsql client.

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

  Table 6-5 Files and folders in the Windows gsql tool package

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

NOTE

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

----End

6.4.2 Using the Linux gsql Client to Connect to a Cluster

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

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

(Optional) Preparing an ECS as the gsql Client Server

For 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:

- ECS and DWS clusters must belong to the same region and AZ.
- If you use the gsql client provided by DWS to connect to the DWS cluster, the ECS image must meet the following requirements:

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

  - The **Redhat x86_64** client can be used on the following OSs:
    - RHEL 6.4~7.6
    - CentOS 6.4~7.4
    - EulerOS 2.3
  - The **SUSE x86_64** client can be used on the following OSs:
    - SLES 11.1~11.4
    - SLES 12.0~12.3
  - 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 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 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 DWS cluster uses to provide services.

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

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

  - **Transfer Direction**: **Outbound**
  - Protocol: The protocol must contain TCP. For example, **TCP** or **All**.
  - **Port**: The value must contain the database port that provides services in the data warehouse cluster. For example, set this parameter to **1-65535** or a specific DWS database port.
  - Destination: The IP address set here must contain the IP address of the DWS cluster to be connected. **0.0.0.0/0** indicates any IP address.

- The security group rules of the data warehouse cluster must ensure that DWS can receive network access requests from clients.

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

  - **Transfer Direction**: **Inbound**
  - **Protocol**: The protocol must contain TCP. For example, **TCP** or **All**.
  - **Port**: Set this parameter to the servicing database port of the DWS cluster. Example: 8000.
  - **Source**: The IP address set here must contain the IP address of the DWS client server. Example: 192.168.0.10/32.

Downloading the Linux gsql Client and Connecting to a Cluster

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

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

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

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

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

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

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

Step 4 Run the following commands to decompress the client:

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

In the preceding commands:

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

Step 5 Run the following command to configure the DWS client:

```
source gsql_env.sh
```

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

```
All things done.
```

Step 6 Connect to the database in the DWS cluster using the gsql client.

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

The parameters are 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 Connection Address of a DWS Cluster**. If a public network address is used for connection, set this parameter to **Public Network Address**. If a private network address is used for connection, set this parameter to **Private Network Address**.
- *Database_user*: Enter the username of the cluster's database. If you use the client to connect to the cluster for the first time, set this parameter to the default administrator configured during cluster creation, for example, **dbadmin**.
- *Database_port*: Enter the database port set during cluster creation.

For example, run the following command to connect to the default database **gaussdb** in the 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

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 server 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 how to obtain the AK and SK, see "Importing Data" > "Importing Data from OBS in Parallel" > "Creating Access Keys (AK and SK)" in the *Data Warehouse Service (DWS) Developer Guide*. 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 DWS client.

After you enter the 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 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 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

6.4.3 Using the Windows gsql Client to Connect to a Cluster

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

Procedure

Step 1 Install and run the gsql client on the local Windows server (in Windows CLI). Windows Server 2008/Windows 7 and later are supported.

Step 2 Download the Windows gsql client by referring to **Downloading the Client** and decompress the package to a local folder.

Step 3 On the local server, click **Start**, search for **cmd**, and run the program as the administrator. Alternatively, press **Win**+**R** to open the Windows CLI.

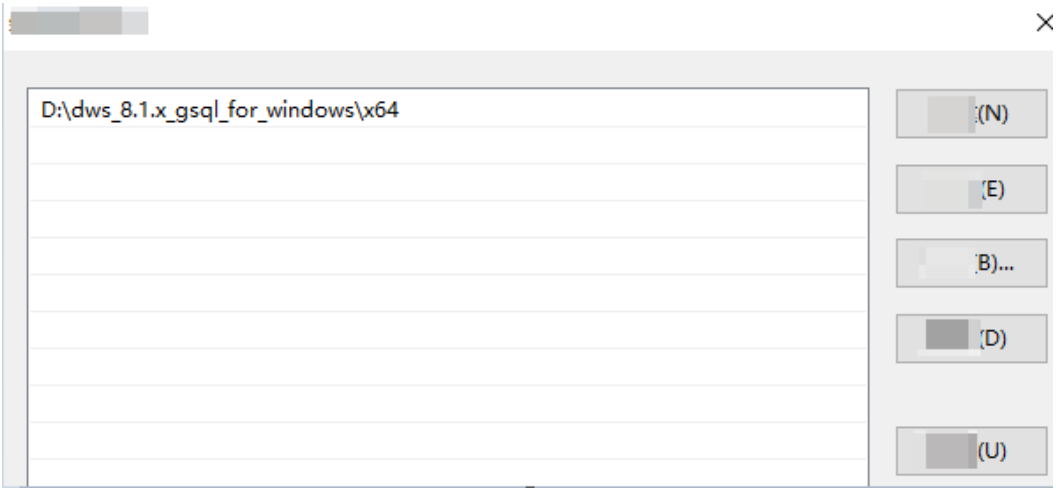
Step 4 Set environment variables. For a 32-bit OS, select the **x86** folder. For a 64-bit OS, select the **x64** folder.

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

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

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

Figure 6-4 Configuring Windows environment variables



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

NOTE

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

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

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

The parameters are as follows:

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

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

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

If the following information is displayed, the connection succeeded:

```
gaussdb=>
```

----End

Precautions

1. The default character encoding of the Windows command prompt is GBK, and the default value of **client_encoding** of Windows gsql is **GBK**. Some characters encoded using UTF-8 cannot be displayed in Windows gsql.

   Suggestion: Ensure the file specified using **-f** uses UTF-8 encoding, and set the default encoding format to **UTF-8** (**set client_encoding='utf-8';**).

2. Paths in Windows gsql must be separated by slashes (/), or an error will be reported. In a meta-command, the backslash (\) indicates the start of a meta-command. If the backslash is enclosed in single quotation marks ('\'), it is used for escape.

   ```
   gaussdb=> \i D:\test.sql
   D:: Permission denied
   postgres=> \i D:/test.sql
   id
   ----
    1
   (1 row)
   ```

3. To use the **\!** metacommand to run a system command in Windows gsql, be sure to use the path separator required by the system command. Generally, the path separator is a backslash (\).

   ```
   gaussdb=> \! type D:/test.sql
   Incorrect syntax.
   gaussdb=> \! type D:\test.sql
   select 1 as id;
   ```

4. Windows gsql does not support the **\parallel** meta-command.

   ```
   gaussdb=> \parallel
   ERROR: "\parallel" is not supported in Windows.
   ```

5. In Linux shell, single quotation marks ('') and double quotation marks ("") can be used to enclose strings. In Windows, only double quotation marks can be used.

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

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

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

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

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

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

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

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

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

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

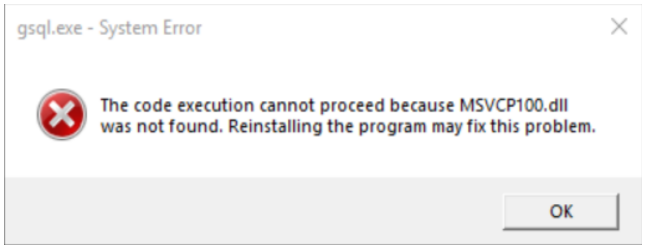

9. The location of the **gsqlrc.conf** file:

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

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

10. **MSVCP100.dll** may be missing in the Windows Server system. When you use **gsql**, the following error message is displayed.

    Figure 6-5 Error message

    Solution: Add the **MSVCP100.dll** file. You can download the C++ redistributable program package and install the **vcredist_x86.exe/ vcredist_x64.exe** package to supplement the required dynamic link library file.

gsql Command Reference

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

6.4.4 Establishing Secure TCP/IP Connections in SSL Mode

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

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

The client or JDBC/ODBC driver needs to use SSL connection. Configure related SSL connection parameters in the client or application code. The DWS console provides the SSL certificate required by the client. The SSL certificate contains the default certificate, private key, root certificate, and private key password encryption file required by the client. Download the SSL certificate to the host where the client is installed, and specify the path of the certificate on the client.

For more information, see **Configuring Digital Certificate Parameters Related to SSL Authentication on the gsql Client** and **SSL Authentication Modes and Client Parameters**.

NOTE

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

Configuring SSL Connection

Prerequisites

- Changes made to security configuration parameters require a cluster restart to take effect. Otherwise, the cluster will be temporarily unavailable.
- To modify the cluster's security configuration, ensure that the following conditions are met:
  - The cluster status is **Available** or **Unbalanced**.
  - The **Task Information** cannot be set to **Creating snapshot**, **Scaling out**, **Configuring**, or **Restarting**.

Procedure

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

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

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

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

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

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

NOTE

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

Step 5 Click **Apply**.

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

----End

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

After a DWS 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 DWS console. In the navigation pane, choose **Client Connections**.

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

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

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

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

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

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

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

The following parameters must be configured for bidirectional authentication:

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

The following parameters must be configured for unidirectional authentication:

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

NOTICE

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

Step 6 Change the client private key permissions.

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

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

----End

SSL Authentication Modes and Client Parameters

There are two SSL authentication modes: bidirectional authentication and unidirectional authentication. Table **Table 6-6** shows the differences between these two modes. You are advised to use bidirectional authentication for security purposes.

Table 6-6 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: • PGSSLCERT • PGSSLKEY • PGSSLROOTCERT • PGSSLMODE	This authentication mode is applicable to scenarios that require high data security. When using this mode, you are advised to set the **PGSSLMODE** client variable to **verify-ca** for network data security purposes.
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: • PGSSLROOTCERT • PGSSLMODE	To prevent TCP-based security attacks, you are advised to use the SSL certificate authentication. In addition to configuring the client root certificate, you are advised to set the **PGSSLMODE** variable to **verify-ca** on the client.

Configure environment variables related to SSL authentication on the client. For details, see **Table 6-7**.

NOTE

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

Table 6-7 Client parameters

Environment Variable	Description	Value Description
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 peer end for data encryption.	The absolute path of the files must be specified, for example: export PGSSLCERT='*/home/dbadmin/dws_ssl/sslcert/client.crt*' (No default value)
PGSSLKEY	Specifies the private key file for the client to decrypt digital signatures and data encrypted using the public key.	The absolute path of the files must be specified, for example: export PGSSLKEY='*/home/dbadmin/dws_ssl/sslcert/client.key*' (No default value)

Environment Variable	Description	Value Description
PGSSLMODE	Specifies whether to negotiate with the server about SSL connection and specifies the priority of the SSL connection.	Values and meanings: • **disable**: only tries to establish a non-SSL connection. • **allow**: tries to establish a non-SSL connection first, and then an SSL connection if the first attempt fails. • **prefer**: tries to establish an SSL connection first, and then a non-SSL connection if the first attempt fails. • **require**: only tries to establish an SSL connection. If there is a CA file, perform the verification according to the scenario in which the parameter is set to **verify-ca**. • **verify-ca**: tries to establish an SSL connection and check whether the server certificate is issued by a trusted CA. • **verify-full**: DWS does not support this mode. Default value: **prefer** **NOTE** When an external client accesses a cluster, the error message "ssl SYSCALL error" is displayed on some nodes. In this case, run **export PGSSLMODE="allow"** or **export PGSSLMODE="prefer"**.
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: export PGSSLROOTCERT='*/home/dbadmin/dws_ssl/sslcert/certca.pem*' 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: export PGSSLCRL='*/home/dbadmin/dws_ssl/sslcert/sslcrl-file.crl*' Default value: null

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 (DWS cluster) parameters **ssl** and **require_ssl**. The parameters are as follows:

- **ssl (Server)**

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

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

- **require_ssl (Server)**

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

  - The default value is **off**. You can set the **require_ssl** parameter in the **Require SSL Connection** area of the cluster's **Security Settings** page on the DWS 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 6-8 Combinations of SSL connection parameters on the client and server

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

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

6.5 Using a Third-Party Database Adapter for DWS Cluster Connection

6.5.1 Using the JDBC and ODBC Drivers to Connect to a DWS Cluster

6.5.1.1 Development Specifications

If the connection pool mechanism is used during application development, comply with the following specifications: If you do not do so, the status of connections in the connection pool will remain, which affects subsequent operations using 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.

6.5.1.2 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 DWS from the console or use the open-source JDBC or ODBC driver.

Open-Source JDBC or ODBC Driver

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 DWS console.

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

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

- **JDBC Driver**

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

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

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

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

- **ODBC Driver**

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

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

  NOTE

  The ODBC driver is incompatible with Windows Server 2016.

----End

6.5.1.3 Using JDBC to Connect to a Cluster

In 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 DWS 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 DWS 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 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**.

  DWS also supports open source JDBC driver: PostgreSQL JDBC 9.3-1103 or later.
- You have downloaded the SSL certificate file. For details, see **Downloading an SSL Certificate**.

Using a JDBC Driver to Connect to a Database

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

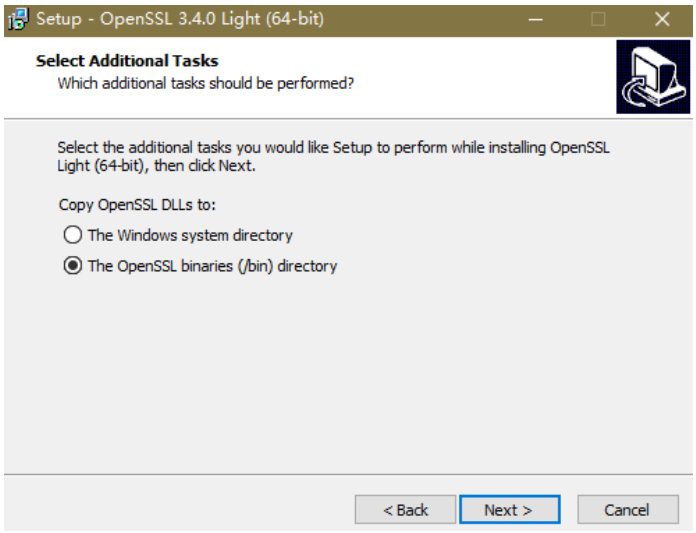
Step 1 Determine whether you want to use the SSL mode to connect to the DWS cluster.

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

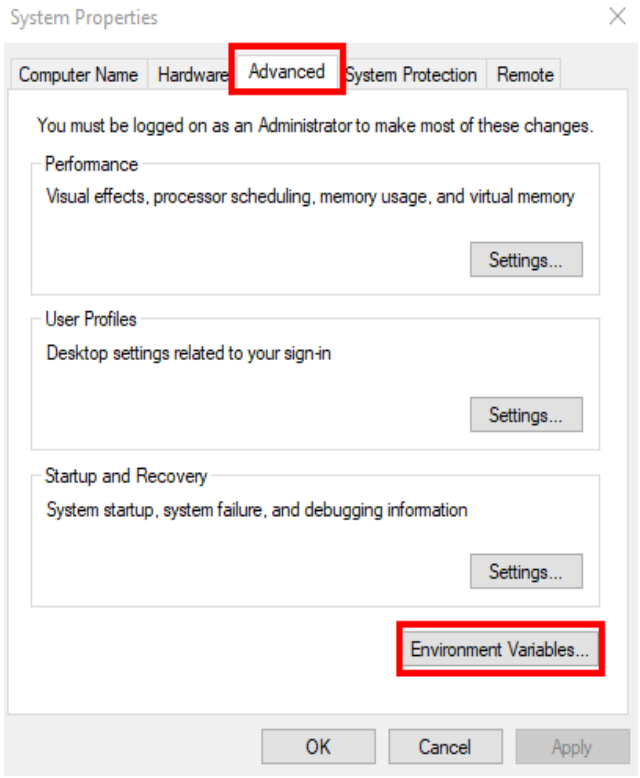
Step 2 (Optional) On Linux, use WinSCP to upload the downloaded SSL certificate file to the Linux environment.

Step 3 Configure the certificate to enable SSL connection.

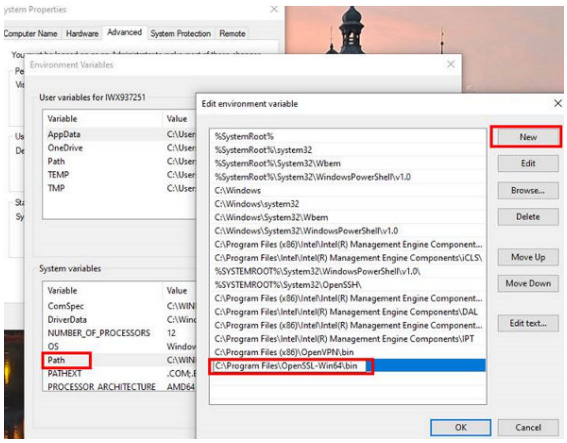
1. Download the OpenSSL tool for Windows at **https://slproweb.com/products/Win32OpenSSL.html**. The latest stable version is 3.4. All earlier versions (including 1.1.1, 1.1.0, 1.0.2, 1.0.0, and 0.9.8) are not supported and should not be used. Download **Win64 OpenSSL v3.4.0 Light**.
2. Double-click the installation package **Win64OpenSSL_Light-3.4.0.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 complete.



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*

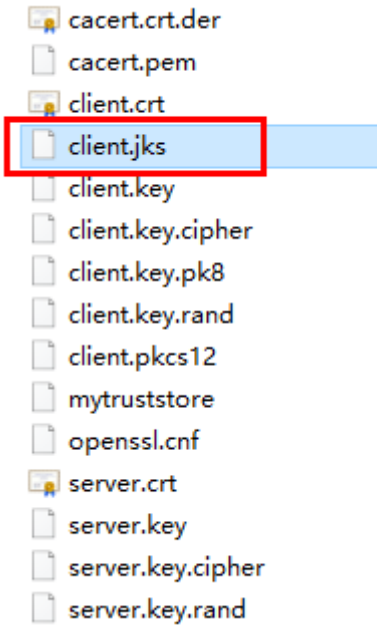
   Enter the client private key password **Gauss@MppDB**. Then enter and confirm the self-defined private key password.

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**.

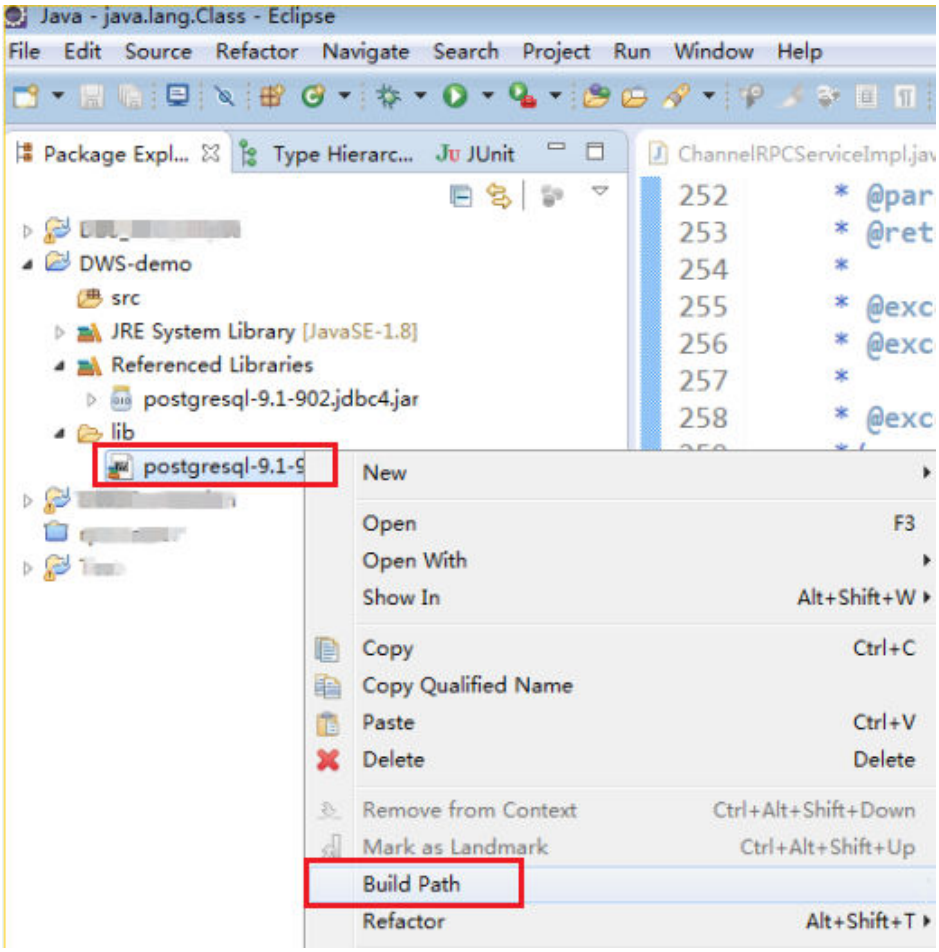


Step 4 Download the driver package **dws_8.1.x_jdbc_driver.zip** and decompress it. There will be two JDBC drive JAR packages, **gsjdbc4.jar** and **gsjdbc200.jar**. Use either of them as required.

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

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

Figure 6-6 Referencing a JAR file



Step 6 Load the driver.

The following methods are available:

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

  NOTE

  The JDBC driver package downloaded on DWScontains **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 Invoke the DriverManager.getConnection() method of JDBC to connect to the DWS database.

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 6-9 Database connection parameters

Parameter	Description
url	Specifies the database connection descriptor, which can be viewed on the management console. For details, see **Obtaining the Connection Address of a DWS Cluster**. The URL format is as follows: • jdbc:postgresql:database • jdbc:postgresql://host/database • jdbc:postgresql://host:port/database • jdbc:postgresql://host:port[,host:port][...]/database **NOTE** • If **gsjdbc200.jar** is used, change **jdbc:postgresql** to **jdbc:gaussdb**. – **database** indicates the name of the database to be connected. – **host** indicates the name or IP address of the database server. If an ELB is bound to the cluster, set **host** to the IP address of the ELB. – **port** indicates the port number of the database server. By default, the database running on port 8000 of the local host is connected. – Multiple IP addresses and ports can be configured. JDBC balances load by random access and failover, and will automatically ignore unreachable IP addresses. Separate multiple pairs of IP addresses and ports by commas (,). Example: **jdbc:postgresql://10.10.0.13:8000,10.10.0.14:8000/database** • If JDBC is used to connect to a cluster, only JDBC connection parameters can be configured in a cluster address. Variables cannot be added.

Parameter	Description
info	Specifies database connection properties. Common properties include the following: • **user**: indicates the database user who creates the connection task. The value is of the string data type. • **password**: indicates the password of the database user. The value is of the string data type. • **ssl**: indicates whether to use the SSL connection. The value is of the Boolean data type. • **loggerLevel**: indicates the log amount recorded in DriverManager for LogStream or LogWriter. The value is of the string data type. Currently, **OFF**, **DEBUG**, and **TRACE** are supported. **DEBUG** indicates that only logs of **DEBUG** or a higher level are printed, generating little log information. **TRACE** indicates that logs of the **DEBUG** and **TRACE** levels are displayed, generating detailed log information. The default value is **OFF**, indicating that no logs will be displayed. • prepareThreshold: It is used to determine the execution times of PreparedStatement before the information is converted into prepared statements on the server. The value is of the integer data type. The default value is **5**. • **batchMode**: indicates whether to connect the database in batch mode. The value is of the Boolean data type. • **fetchsize**: integer type. It indicates the default fetch size for statements in the created connection. • **ApplicationName**: indicates the application name. The value is of the string data type. The parameter is set to **PostgreSQL JDBC Driver** by default. • **allowReadOnly**: indicates whether the read-only mode can be set for a connection. The value is of the Boolean data type. The default value is **false**. If this parameter is not set to **true**, the **connection.setReadOnly** statement will not take effect. • **blobMode**: indicates the data types to which a value is assigned using the setBinaryStream method. The value is of the string data type. If this parameter is set to **on**, a value is assigned to the BLOB data type. If this parameter is set to **off,** a value is assigned to the bytea data type. The default value is **on**. • **currentSchema**: string type. It specifies the schema used for connecting to the database. • **defaultQueryMetaData**: Boolean. It specifies whether to query SQL metadata by default. The default value is **false**. After this function is enabled, raw data operations are supported. However, it is incompatible with the **create table as** and **select into** operations in **PrepareStatement**. • connectionExtraInfo: indicates whether the driver reports the driver deployment path and process owner to the database. The value is of the Boolean data type.

Parameter	Description
	NOTE The value can be **true** or **false**. The default value is **true**. If **connectionExtraInfo** is set to **true**, the JDBC driver reports the driver deployment path and process owner to the database and displays the information in the **connection_info** parameter. In this case, you can query the information from **PG_STAT_ACTIVITY** or **PGXC_STAT_ACTIVITY**. • **TCP_KEEPIDLE=30**: The detection starts after the connection is idle for 30s. This parameter is valid only when **tcpKeepAlive** is set to **true**. • **TCP_KEEPCOUNT=9**: A total of nine detections are performed. This parameter is valid only when **tcpKeepAlive** is set to **true**. • **TCP_KEEPINTERVAL=30**: The detection interval is 30s. This parameter is valid only when **tcpKeepAlive** is set to **true**. • **cnListRefreshSwitch**: indicates whether JDBC automatically detects the live CN list. The value is of the string data type. If this parameter is set to **on**, the function of automatically detecting the live CN list is enabled. If this parameter is set to **off**, the function is disabled. The default value is **off**. • **cnListRefreshDelay**: specifies the start time for scanning the live CN list. This parameter is valid only when **cnListRefreshSwitch** is set to **on**. The value is of the integer data type. The default value is **1800000**, in milliseconds. • **cnListRefreshPeriod**: specifies the interval for scanning the live CN list. This parameter is valid only when **cnListRefreshSwitch** is set to **on**. The value is of the integer data type. The default value is **1800000**, in milliseconds. • **autoReconnect**: indicates whether to enable automatic reconnection of database connections. The value is of the Boolean data type. If this parameter is set to **true**, the automatic reconnection is enabled. If this parameter is set to **false**, the automatic reconnection is disabled. The default value is **false**. • **reConnectCount**: specifies the number of automatic database reconnections. The value is of the integer data type. This parameter is valid only when **autoReconnect** is set to **true**. The default value is **10**. If the number of reconnection attempts exceeds the configured value, the reconnection fails. • **sslCrl**: a string type that sets the path for the revoked certificate used by JDBC. The default value is **null**.
user	Specifies the database user.
password	Specifies the password of the database user.

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

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



```
public static Connection GetConnection(String username, String passwd) {
    // Define the driver class.
    String driver = "org.postgresql.Driver";
        //Set keyStore.
    System.setProperty("javax.net.ssl.trustStore", "mytruststore");
    System.setProperty("javax.net.ssl.keyStore", "client.jks");
    System.setProperty("javax.net.ssl.trustStorePassword", "password");
    System.setProperty("javax.net.ssl.keyStorePassword", "password");

    Properties props = new Properties();
    props.setProperty("user", username);
    props.setProperty("password", passwd);
    props.setProperty("ssl", "true");

    String url = "jdbc:postgresql://" + "10.10.0.13" + ':' + "8000" + '/' + "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 (SQLException throwables) {
        throwables.printStackTrace();
        return null;
    }
    return conn;
}
```

Step 8 Run SQL statements.

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

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

----End

Sample Code

This example illustrates how to develop applications based on the JDBC API provided by 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();
    }

  }

}
```

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

Context

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

To avoid these problems, you can use JDBC to connect to multiple CNs. The following three methods are available:

- Connection using ELB: An ELB distributes access traffic to multiple ECSs for traffic control based on forwarding policies. It improves the fault tolerance capability of application programs.

- To connect to a cluster using JDBC load balancing, include at least one internal IP address of the CN in the URL. The system will scan all CN IP addresses automatically. JDBC load balancing functions like ELB, randomly connecting to a CN.
- Connection in multi-host mode: Use JDBC to configure multiple nodes, which is similar to ELB.

Method 1: Using ELB to Connect to a Cluster

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

Step 2 Configure the driver. For details, see **Downloading the JDBC or ODBC Driver**.

Step 3 Obtain the database connection.

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

----End

Method 2: Using JDBC Load Balancing to Connect to a Cluster (Recommended)

Step 1 Obtain the private IP address. Open the specified cluster topology page on the console and obtain the private IP address of the CN. For details, see **Obtaining the Connection Address of a DWS Cluster**.

Step 2 Configure the driver. For details, see **Downloading the JDBC or ODBC Driver**.

NOTE

Starting from version 8.3.1.200, the JDBC load balancing mode now allows for the driver to connect to the cluster. If you intend to utilize this mode, ensure that your JDBC driver version is 8.3.1.200 or later.

Step 3 Obtain the database connection. For how to set URL parameters, see **Using JDBC to Connect to a Cluster**.

```
private static final String USER_NAME = "dbadmin";
private static final String PASSWORD = "password";
// jdbc:postgresql://host1:port1,host2:port2/dbName"
private static final String URL = "jdbc:postgresql://
100.95.146.194:8000,100.95.148.220:8000,100.93.0.221:8000/gaussdb?
```



```
loadBalanceHosts=true&cnListRefreshSwitch=on&cnListRefreshDelay=100000&cnListRefreshPeriod=5000
;
private static Properties properties = new Properties();
static {
    properties.setProperty("user", USER_NAME);
    properties.setProperty("password", PASSWORD);
}
/**
 * Obtain the database connection.
 */
public static Connection getConnection() {
    Connection connection = null;
    try {
        connection = DriverManager.getConnection(URL, properties);
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return connection;
}
```

----End

Method 3: Connecting to the Cluster in Multi-host Mode

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

Step 2 Configure the driver. For details, see **Downloading the JDBC or ODBC Driver**.

Step 3 Obtain the database connection.

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

----End

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

Overview

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

NOTE

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

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

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

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

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

The process of accessing a database is as follows:

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

Granting an IAM Account the DWS Database Access Permission

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

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

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

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

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

----End

Creating an IAM User Credential

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

Step 1 Log in to the management console.

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

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

The AK/SK pair is so important that you can download the private key file containing the AK/SK information only when you create the pair. On the management console, you can only view the AKs. If you have not downloaded the file, obtain it from your administrator or create an AK/SK pair again.

NOTE

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

----End

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

Configuring JDBC Connection Parameters

Table 6-10 Database connection parameters

Parameter	Description
url	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: jdbc:dws:iam://dws-IAM-demo:eu-west-0/gaussdb?**AccessKeyID**=XXXXXXXXXXXXXXXXXXXX&**SecretAccessKey**=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX&**DbUser**=user_test&**AutoCreate**=true **JDBC URL parameters**: • **jdbc:dws:iam** is a prefix in the URL format. • **dws-IAM-demo** indicates the name of the cluster containing the database. • **eu-west-0** indicates the region where the cluster resides. JDBC accesses the DWS cluster in the corresponding region and delivers the IAM certificate to the cluster for IAM user authentication. The DWS service address has been recorded in the JDBC configuration file. For details about DWS regions, visit **Regions and Endpoints**. • **gaussdb** indicates the name of the database to which you want to connect. • **AccessKeyID** and **SecretAccessKey** are the access key ID and secret access key corresponding to the IAM user specified by **DbUser**. • Set **DbUser** to the IAM username. Note that the current version does not support hyphens (-) in the IAM username. – If the user specified by **DbUser** exists in the database, the temporary user credential has the same permissions as the existing user. – If the user specified by **DbUser** does not exist in the database and the value of **AutoCreate** is **true**, a new user named by the value of **DbUser** is automatically created. The created user is a common database user by default. • Parameter **AutoCreate** is optional. The default value is **false**. This parameter indicates whether to automatically create a database user named by the value of **DbUser** in the database. – The value **true** indicates that a user is automatically created. If the user already exists, the user will not be created again. – The value **false** indicates that a user is not created. If the username specified by **DbUser** does not exist in the database, an error is returned. • **addressType** indicates the type of the address used for the connection. The default value is **auto**. – **auto**: The selection priority is EIP, ELB, and private IP address. If **auto** is selected, the system chooses an IP address for the connection.

Parameter	Description
	– **eip**: The system chooses an EIP for the connection. – **elb**: The system chooses an ELB for the connection. The selection priority is **elb_public** and **elb_private**. – **elb_public**: Use the ELB public IP address for the connection. – **elb_private**: Use ELB private IP address for the connection.
info	Database connection properties. Common properties include the following: ● **ssl**: a boolean type. It indicates whether the SSL connection is used. ● **loglevel**: an integer type. It sets the log amount recorded in DriverManager for LogStream or LogWriter. Currently, **org.postgresql.Driver.DEBUG** and **org.postgresql.Driver.INFO** logs are supported. If the value is **1**, only **org.postgresql.Driver.INFO** (little information) is recorded. If the value is greater than or equal to **2**, **org.postgresql.Driver.DEBUG** and **org.postgresql.Driver.INFO** logs are printed, and detailed log information is generated. Its default value is **0**, which indicates that no logs are printed. ● **charSet**: a string type. It indicates character sets used when data is sent from the database or the database receives data. ● **prepareThreshold**: an integer type. It is used to determine the execution times of PreparedStatement before the information is converted into prepared statements on the server. The default value is **5**.

Example

```
//The following uses gsjdbc4.jar as an example.
// The following code encapsulates the database connection obtaining operations into an API. You can connect to the database by specifying the region where the cluster is located, cluster name, access key ID, secret access key, and the corresponding IAM username.
public static Connection GetConnection(String clustername, String regionname, String AK, String SK,
    String username) {
    // Driver class.
    String driver = "org.postgresql.Driver";
    // Database connection descriptor.
    String sourceURL = "jdbc:dws:iam://" + clustername + ":" + regionname + "/postgresgaussdb?" +
"AccessKeyID="
        + AK + "&SecretAccessKey=" + SK + "&DbUser=" + username + "&autoCreate=true";

    Connection conn = null;

    try {
        // Load the driver.
        Class.forName(driver);
    } catch (ClassNotFoundException e) {
        return null;
    }
    try {
        // Create a connection.
        conn = DriverManager.getConnection(sourceURL);
        System.out.println("Connection succeed!");
```



```
    } catch (SQLException e) {
        return null;
    }
    return conn;
}
```

6.5.1.6 Third-party Connection Pool of the JDBC Configuration Database

Context

DWS does not have its own JDBC connection pool, and the inherited PostgreSQL connection pool is offline. Use third-party connection pools like Druid, HikariCP, or DBCP 2.

NOTE

- The connection pool inherited by JDBC from PostgreSQL has been brought offline and is not recommended.
- Determine the version of the JDBC and driver to be downloaded and how to set the connection pool parameters based on the site requirements.

Configuring the DBCP 2 Connection Pool

Step 1 Download the JDBC driver package. For details, see **Downloading the JDBC or ODBC Driver**.

- Download the **commons-dbcp2** driver package from **https://commons.apache.org/dbcp/download_dbcp.cgi**.
- Download the **commons-logging** driver package from **https://commons.apache.org/proper/commons-logging/download_logging.cgi**.
- Download the **commons-pool2** driver package from **https://commons.apache.org/proper/commons-pool/download_pool**.

Step 2 Add the JDBC driver package and the **commons-dbcp2**, **commons-logging**, and **commons-pool2** driver packages to the project and configure parameters related to the database connection pool.

NOTE

- Enabling **removeAbandoned** allows the connection pool to reclaim and reuse a discarded connection. This occurs when the conditions (getNumIdle() < 2) and (getNumActive() > getMaxTotal() - 3) are met.
  - For example, if **maxTotal** is set to **20**, there are 18 active connections and one connection is restricted. In this case, **removeAbandoned** is triggered.
  - An active connection is deleted only when it is not used for a period of time specified by **removeAbandonedTimeout**. The default value is 300 seconds.
  - Traversing a result set does not count as usage. Creating a statement, prepared statement, callable statement, or executing a query resets the **lastUsed** property of its parent connection.
- In high-load systems, setting **maxIdle** to a small value may cause new connections to close immediately. This is because active threads close connections faster than those that open connections. As a result, the number of idle connections is greater than the value of **maxIdle**. In high-load systems, the most appropriate value of **maxIdle** in a high-load system depends on the application scenario, but the default value is a good initial value.

Table 6-11 Parameters of the DBCP 2 connection pool

Parameter	Default Value	Description
driverClassName	Enter the value of **org.postgresql.Driver**.	Name of the database driver.
url	N/A	URL for connecting to the database.
username	N/A	Username.
password	N/A	Password.
connectionProperties	N/A	The connection parameters are sent to the JDBC driver when a new connection is set up. The string must be in the format of **[Parameter name=Parameter value;]**. **NOTE** The username and password attributes need to be specified. Therefore, the two parameters do not need to be included here.
defaultAutoCommit	N/A	Automatic submission. By default, the connection created through the current connection pool is in the automatic submission state. If this parameter is not set, the **setAutoCommit** method is not invoked.
defaultReadOnly	N/A	Read-only setting. By default, the connection created through the current connection pool is read-only. If the connection is not set, the **setReadOnly** method is not invoked.
defaultTransactionIsolation	N/A	Transaction isolation level. The default transaction isolation policy is used for connections created through this pool. The value can be one of the following: • **NONE** • **READ_COMMITTED**: The read operation is committed. • **READ_UNCOMMITTED**: The read operation is not committed. • **REPEATABLE_READ** • **SERIALIZABLE**
defaultCatalog	N/A	The default catalog is used for connections created through this pool.

Parameter	Default Value	Description
cacheState	true	Cache status of the connection pool. If this parameter is set to **true**, the current read-only status and auto-commit settings are cached during the first read or write operation after the resource pool connects. This eliminates the need for additional database queries on subsequent **getter** calls. If the underlying connection is accessed directly, changes to the read-only state or auto-commit settings will not update the cache. Set this parameter to **false** to disable caching in such cases.
defaultQueryTimeout	null	Query timeout interval. ● Enter an integer, which is used to specify the query timeout interval when a statement is created. ● If the value is **null**, the default driver settings are used.
enableAutoCommitOnReturn	true	When a connection is returned to the pool, the connection is automatically submitted. Setting it to **true** will return the connection to the pool with **autoCommit** set to **true** by default.
rollbackOnReturn	true	Roll back all operations when the connection is returned to the pool. Setting it to **true** will automatically execute **"rollback()"** when the connection is returned to the pool, provided that auto submission is enabled.
initialSize	0	Number of initial connections. Number of connections created during initialization when the current connection pool is started. The initial version is 1.2.
maxTotal	8	Maximum number of active connections in the pool. A negative value means there is no limit.
maxIdle	8	Maximum number of idle connections in the pool. Excess idle connections are released when returned to the pool. A negative value means there is no limit.

Parameter	Default Value	Description
minIdle	0	Minimum number of idle connections. Minimum number of idle connections to retain in the pool. If the number of idle connections falls below this value, new idle connections are created. A value of **0** means no idle connections are created. **NOTE** The value takes effect only when **timeBetweenEvictionRunsMillis** is set to a positive number.
maxWaitMillis	N/A	Maximum waiting time for obtaining a connection from the connection pool. ● If this parameter is set to **–1** and no connection is available, the connection pool waits indefinitely until a connection is obtained. ● If the parameter is set to *N*, the connection pool waits for *N* milliseconds. If the waiting time is insufficient, an exception is thrown.
validationQuery	SELECT 1	Query confirmation SQL statement, which validates the connection before it is returned to the caller by the connection pool. ● If specified, the query must be a **SELECT** statement that returns at least one row of data. ● If no value is specified, the connection is verified by invoking the **"isValid()"** method.
validationQueryTimeout	N/A	Query timeout interval for valid SQL statements, in seconds. If the parameter is set to a positive number, the value is transferred to the **"setQueryTimeOut()"** method of the JDBC driver. The setting takes effect for the SQL statement for confirming the validity of the query.
testOnCreate	false	Whether to verify the validity of a connection immediately after creation. If verification fails, the creation attempt fails.
testOnBorrow	true	Whether to verify the validity of a connection when it is leased from the pool. If verification fails, the connection is released and another is leased.

Parameter	Default Value	Description
testOnReturn	false	Whether to verify the validity of a connection before returning it to the pool.
testWhileIdle	false	Whether to verify the validity of idle connections using an evictor, if available. Invalid connections are released.
timeBetweenEvictionRunsMillis	-1	Hibernate time (in milliseconds) for the idle object eviction thread. A non-positive value disables the thread.
numTestsPerEvictionRun	3	Number of objects checked during the running of each idle object eviction thread.
minEvictableIdleTimeMillis	1000 * 60 * 30	Minimum number of milliseconds in which objects that meet the eviction conditions are idle in the pool. Minimum duration for releasing an idle connection, in milliseconds.
softMinEvictableIdleTimeMillis	-1	Minimum number of milliseconds in which objects that meet the eviction conditions are idle in the pool. Idle connections are released after at least *N* milliseconds, provided that at least the number of connections specified by minIdle is retained in the pool. If **miniEvictableIdleTimeMillis** is set to a positive number, the idle connection evictor checks **miniEvictableIdleTimeMillis** first, and then **softMinEvictableIdleTimeMillis** and the **minIdle** condition.
maxConnLifetimeMillis	-1	Maximum lifetime of a connection (in milliseconds). Connections exceeding this time fail on the next activation, passivation, or verification. A value of **0** or negative means unlimited lifetime.
logExpiredConnections	true	Whether to write logs when an expired connection is closed by the pool. If a connection's lifespan exceeds **maxConnLifetimeMillis**, it will be reclaimed by the connection pool and a log will be generated by default. If this parameter is set to **false**, no log will be written.
connectionInitSqls	N/A	This parameter executes a set of SQL statements to initialize a physical connection when it is first created. These statements run only once per connection.

Parameter	Default Value	Description
lifo	true	Last in first out. • Last in first out. If this parameter is set to **true**, the connection pool returns the last used connection first (if there are available idle connections in the pool). • If this parameter is set to **false**, the pool operates as a FIFO queue and obtains connections from the idle connection instance pool in the sequence in which they are returned.
poolPreparedStatements	false	This determines whether the preprocessing statement pool in the connection pool will be applied.

maxOpenPreparedStatements	N/A	Maximum number of statements that can be allocated in the statement pool at the same time. A negative value means no limit.
		This setting also applies to the pre-processed statement pool. When a statement pool is created for each connection, the pre-processed statements generated by the following method are included.
		`public PreparedStatement prepareStatement(String sql)`
		`public PreparedStatement prepareStatement(String sql, int resultSetType, int resultSetConcurrency)`
		NOTE
		Ensure that connections leave resources for other statements by setting **maxOpenPreparedStatements** to a value less than the maximum number of cursors.

accessToUnderlyingConnectionAllowed	false	This controls whether the PoolGuard can access underlying connections.
removeAbandonedOnMaintenance removeAbandonedOnBorrow	false	Whether to delete abandoned connections that have been abandoned for a period longer than the time specified by **removeAbandonedTimout**. If the value is **true**, connections unused for longer than **removeAbandonedTimeout** are considered abandoned and removed. Creating or executing statements resets the **lastUsed** property of the parent connection. Setting this parameter to **true** helps recover connections in applications with few write operations.

Parameter	Default Value	Description
removeAbandonedTimeout	300	Timeout interval for removing a discarded connection, in seconds.
logAbandoned	false	Whether to enable stack tracing for discarded statements or connected code in an application. When enabled, stack traces for discarded statements and connection-related logs will be overwritten each time a connection is opened or a statement is created.
abandonedUsageTracking	false	When this parameter is set to **true**, the connection pool records stack traces each time a method is called on a pooled connection, retaining the latest stack trace to aid in debugging abandoned connections. **NOTE** Setting this parameter to **true** will increase the overhead. Exercise caution when performing this operation.
fastFailValidation	false	This parameter refers to the quick failure of validation statements if a fatal exception occurs, without executing **isValid()** or the validation query. Fatal exceptions include specific **SQL_STATE** codes. ● 57P01 (ADMIN SHUTDOWN) ● 57P02 (CRASH SHUTDOWN) ● 57P03 (CANNOT CONNECT NOW) ● 01002 (SQL92 disconnect error) ● JZ0C0 (Sybase disconnect error) ● JZ0C1 (Sybase disconnect error) ● Any SQL_STATE code that starts with "08" Exception codes need to be overwritten. For details, see **disconnectionSqlCodes**.
disconnectionSqlCodes	N/A	Exception code, which is an SQL_STATE code separated by commas (,). This parameter is valid only when **fastFailValidation** is set to **true**.
jmxName	N/A	This parameter registers a DataSource as a JMX MBean with a specified name that adheres to the JMX object name syntax.
registerConnectionMBean	true	Whether to register and connect to the JMX MBean.

----End

Configuring the Hikari CP Connection Pool

Step 1 Download the JDBC driver package. For details, see **Downloading the JDBC or ODBC Driver**.

- Download the HikariCP driver package from **https://mvnrepository.com/artifact/com.zaxxer/HikariCP/4.0.3**.
- Download the SLF4J driver package from **https://www.slf4j.org/download.html**.

Step 2 Add the JDBC, HikariCP, and SLF4J driver packages to the project and configure parameters related to the database connection pool.

Table 6-12 Hikari CP connection pool parameters

Parameter	Default Value	Description
driverClassName	Enter the value of **org.postgresql.Driver**.	Name of the database driver.
jdbcUrl	N/A	URL for connecting to the database.
username	N/A	Username.
password	N/A	Password.
autoCommit	true	Whether to automatically submit transactions when the connection returns to the connection pool.
connectionTimeout	30000	Maximum timeout interval for obtaining connections from the connection pool.
idleTimeout	60000	Maximum lifetime of an idle connection. This setting takes effect only when the value of **minimumIdle** is less than that of **maximumPoolSize**. • If the number of idle connections is greater than the value of **minimumIdle** and the idle time of a connection is greater than the value of **idleTimeout**, the connection is deleted from the connection pool. • **0** indicates no timeout.
keepaliveTime	0	Interval for checking whether idle connections are available, in milliseconds. **0** indicates that the function is disabled.
maxLifetime	1800000	Maximum connection lifetime, in milliseconds. **0** indicates no limit.

Parameter	Default Value	Description
connectionTestQuery	N/A	Query statement for connection detection.
minimumIdle	10	Minimum number of idle connections. To improve performance, you are advised not to set this parameter. The size of the connection pool is fixed.
maximumPoolSize	10	Maximum number of connections.
metricRegistry	N/A	This parameter can only be accessed through programmatic configuration or the IoC container. This parameter specifies the Codahale/Dropwizard MetricRegistry instance used by the pool to record various metrics.
healthCheckRegistry	N/A	This parameter can only be accessed through programmatic configuration or the IoC container. This parameter specifies the Codahale/Dropwizard HealthCheckRegistry instance used by the pool to record health information.
poolName	N/A	Name of a connection pool.
initializationFailTimeout	1	Whether the connection pool fails to initialize quickly. ● If the value is greater than 0, the system attempts to obtain a connection within the specified duration (**connectionTimeout** + **initializationFailTimeout**). If unsuccessful, the pool is not enabled, and an exception is thrown. ● If the value is 0, the system attempts to obtain and verify the connection. If verification fails, the pool is not enabled. ● If the value is less than 0, the pool starts without attempting connection initialization.
isolateInternalQueries	false	Whether to isolate HikariCP queries in a transaction. This setting takes effect when **autoCommit** is set to **false**.

Parameter	Default Value	Description
allowPoolSuspension	false	Whether to allow the connection pool to be suspended and resumed through JMX. When the connection pool is suspended, the connection does not time out until the connection pool is restored.
readOnly	false	Whether the connection is read-only.
registerMbeans	false	Whether to enable JMX.
catalog	N/A	Default database **catalog**.
connectionInitSql	N/A	SQL statement executed after the connection pool is initialized.
transactionIsolation	N/A	Default transaction isolation level.
validationTimeout	5000	Timeout interval for connection detection. The value must be greater than the value of **connectionTimeout**. The minimum value is **250**.
leakDetectionThreshold	0	Maximum duration a connection can be lent out. The minimum value is 2000 milliseconds, used for logging connection leakage.
schema	N/A	Default database **schema**.
threadFactory	N/A	The **java.util.concurrent.ThreadFactory** instance used by the connection pool for thread creation. This parameter can only be accessed through programmatic configuration or the IoC container.
scheduledExecutor	N/A	The **java.util.concurrent.ScheduledExecutorService** instance used by the connection pool to execute scheduled tasks. This parameter can only be accessed through programmatic configuration or the IoC container.

----End

Configuring the Druid Connection Pool

Step 1 Download the JDBC driver package. For details, see **Downloading the JDBC or ODBC Driver**.

Download the Druid driver package from **https://druid.apache.org/downloads/**.

Step 2 Add the JDBC and Druid driver packages to the project and configure parameters related to the database connection pool.

Table 6-13 Druid connection pool parameters

Parameter	Default Value	Description
url	N/A	URL for connecting to the database.
username	N/A	Username.
password	N/A	Password.
driverClassName	Enter the value of **org.postgresql.Driver**.	Name of the database driver.
initialSize	0	Number of physical connections established during initialization. Initialization occurs when the **init** method is invoked explicitly or when the **getConnection** method is invoked for the first time.
maxActive	8	Maximum number of connections in the thread pool.
minIdle	0	Minimum number of idle threads in the thread pool. Druid periodically scans the number of connections. If the number exceeds the specified parameter, redundant connections are closed. If fewer connections are available, new ones are created. This parameter helps manage connections during high request volumes, though it can be time-consuming.
connectTimeout	N/A	Timeout interval for connecting to the database, in milliseconds.
socketTimeout	N/A	Timeout interval for the socket to connect to the database, in milliseconds.
maxWait	-1	Waiting time for a new request when the connections in the connection pool are used up, in milliseconds. **–1** indicates infinite waiting until timeout occurs.
poolPreparedStatements	false	Whether to cache preparedStatement, that is, PSCache. The PSCache greatly improves the performance of the database that supports cursors.

Parameter	Default Value	Description
maxOpenPreparedStatements	N/A	If PSCache is enabled, the value of this parameter must be greater than **0**. If the value is greater than **0**, **poolPreparedStatements** will be automatically set to **true**.
validationQuery	SELECT 1	SQL statement used to check whether a connection is valid. If **validationQuery** is null, the **testOnBorrow**, **testOnReturn**, and **testWhileIdle** parameters do not take effect because the three parameters are used to verify the validity of the database connection by running the SQL statement specified by **validationQuery**.
testOnBorrow	N/A	When applying for a connection, the **validationQuery** command checks its validity. This configuration may reduce performance, so use it cautiously.
testOnReturn	N/A	When a connection is returned, the **validationQuery** command checks its validity. This configuration may also impact performance, so use it cautiously.
testWhileIdle	true	Whether a connection should be checked when it is requested. It is best to set this parameter to **true** to ensure security without compromising performance. If the idle time is greater than the value of **timeBetweenEvictionRunMills**, running the **validationQuery** command to verify the connection's validity will not have any effect.

Parameter	Default Value	Description
timeBetweenEvictionRunsMillis	60s	The **validationQuery** command checks connection validity. If the number of idle connections exceeds **minIdle**, redundant connections are closed. If fewer idle connections are available, new ones are added. Connections not used within the time specified by **timeBetweenEvictionRunsMillis** are disabled. This parameter also: 1. Sets the interval for the Destroy thread to check connections. 2. Functions as a reference for checking **testWhileIdle**. For details, see the description of the **testWhileIdle** attribute.
minEvictableIdleTimeMillis	30min	Maximum lifetime of an idle connection before eviction. If the time since the last activity exceeds **minEvictableIdleTimeMillis**, the connection is closed by the Destroy thread. **NOTE** This parameter conflicts with the **timeBetweenEvictionRunsMillis** parameter. You can leave this parameter empty.
connectionInitSqls	N/A	The SQL statement is executed when the physical connection is initialized.
exceptionSorter	N/A	When the database throws some unrecoverable exceptions, the connection is discarded.
filters	N/A	This parameter configures an extension plug-in using an alias. The attribute type is string. Common plug-ins include the filters used for monitoring and statistics: • **stat**: monitoring statistics • **log4j**: log record • **wall**: SQL injection prevention
proxyFilters	N/A	The type is **List<com.alibaba.druid,filter.Filter>**. You can configure both **filter** and **proxyFilters**.

Parameter	Default Value	Description
removeAbandoned	false	Whether to reclaim leaked connections. When **getNumActive()** approaches **getMaxActive()**, the system reclaims invalid connections not used within the **removeAbandonedTimeout** period (300 seconds by default). Connections exceeding this timeout are forcibly closed.
removeAbandonedTimeout	300s	Time limit for Druid to forcibly reclaim connections, in seconds. Druid will forcibly reclaim a connection from the pool after a specified time has elapsed since the connection was established, starting from the moment the program retrieves the connection from the pool.
logAbandoned	false	Whether to print a log when reclaiming leaked connections. This parameter specifies whether to record the stack information of the current thread to logs when the **removeAbandoned** occurs.
removeAbandonedTimeoutMillis	5min	Timeout interval for reclaiming connections. If **removeAbandoned** is set to **true**, Druid periodically checks whether the thread pool overflows. If the thread pool is not in the running state and the specified time is exceeded, the thread pool is reclaimed.
maxEvictableIdleTimeMillis	7hours	Maximum idle time. The default value is 7 hours.
maxPoolPreparedStatementPerConnectionSize	20	Maximum number of SQL statements that can be cached for each connection.
keepAlive	false	Number of minIdle connections to maintain when the pool is initialized. If the number of connections falls below **minIdle** and idle time exceeds **minEvictableIdleTimeMillis**, the **keepAlive** operation is performed to maintain the **minIdle** value.

Parameter	Default Value	Description
notFullTimeoutRetryCount	0	Number of retry times when the sum of the number of lent connections in the connection pool and the number of available connections is less than the maximum allowed connections. The default value is **0**.
logSlowSql	false	Whether to print slow SQL statements.

Step 3 Use the Druid connection pool based on the following example.

Create the **db.properties** file in the **resource** directory.

```
# Database connection parameters
url=jdbc:postgresql://10.10.0.13:8000/gaussdb
username=user
password=pass
validationQuery=select 1
validationQueryTimeout=300
#driverClassName=JDBC driver name
driverClassName=org.postgresql.Driver
# Number of initialized connections
initialSize=1
# Maximum number of connections
maxActive=20
# Number of core threads. If the number of core threads is greater than this configured value, the threads are released.
minIdle=10
```

Example code:

```
import com.alibaba.druid.pool.DruidDataSource;
import com.alibaba.druid.pool.DruidDataSourceFactory;
import com.alibaba.druid.pool.DruidPooledConnection;
import java.io.IOException;
import java.io.InputStream;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Properties;

public class TestDataSource {
    private static DruidDataSource dataSource;


    public static void main(String[] args) throws Exception {
        Properties properties = loadProperties();
        dataSource = (DruidDataSource) DruidDataSourceFactory.createDataSource(properties);
        dataSource.setTimeBetweenEvictionRunsMillis(50 * 1000);
        dataSource.setRemoveAbandoned(true);
        dataSource.setRemoveAbandonedTimeout(120);
        dataSource.setSocketTimeout(5000);
        dataSource.setConnectTimeout(5000);
        dataSource.setQueryTimeout(5);
        final DruidPooledConnection connection = dataSource.getConnection();
        final Statement statement = connection.createStatement();
        execute(statement);
    }
    public static void execute(Statement statement) {
        ResultSet resultSet = null;
        try {
            resultSet = statement.executeQuery("select 1");
            while (resultSet.next()) {
                String str = resultSet.getString(1);
                System.out.println("n1 :" + str);
```



```
                return;
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    /**
     * Load the configuration file and obtain parameters from the configuration file.
     */
    public static Properties loadProperties() {
        InputStream inputStream =
TestDataSource.class.getClassLoader().getResourceAsStream("db.properties");
        Properties ps = new Properties();
        try {
            ps.load(inputStream);
        } catch (IOException e) {
            e.printStackTrace();
        }
        return ps;
    }
}
```

----End

6.5.1.7 Using ODBC to Connect to a Cluster

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_x.x.x_odbc_driver_for_xxx.zip** (for Linux) and **dws_odbc_driver_for_windows.zip** (for Windows). For details, see **Downloading the JDBC or ODBC Driver**.

  DWS also supports open source ODBC driver: PostgreSQL ODBC 09.01.0200 or later.
- You have downloaded the open-source unixODBC code file 2.3.0 from **https://sourceforge.net/projects/unixodbc/files/unixODBC/2.3.0/unixODBC-2.3.0.tar.gz/download**.
- You have downloaded the SSL certificate file. For details, see **Downloading an SSL Certificate**.

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

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

Step 2 Log in to the Linux environment.

Step 3 Prepare **unixODBC**.

1. Decompress the **unixODBC** code file.
   ```
   tar -xvf unixODBC-2.3.0.tar.gz
   ```
2. Compile the code file and install the driver.
   ```
   cd unixODBC-2.3.0
   ./configure --enable-gui=no --prefix=[your_path]
   ```



```
make
make install
```

NOTE

- In the command, **[your_path]** indicates the installation path, which can be specified as required. The path must be an absolute path.
- After the unixODBC is compiled and installed, the ***.so.2** library file will be in the installation directory. To create the ***.so.1** library file, change **LIB_VERSION** in the configure file to **1:0:0**.
  ```
  LIB_VERSION="1:0:0"
  ```
- This driver dynamically loads the **libodbcinst.so.*** library files. If one of the library files is successfully loaded, the library file is loaded. The loading priority is **libodbcinst.so** > **libodbcinst.so.1** > **libodbcinst.so.1.0.0** > **libodbcinst.so.2** > **libodbcinst.so.2.0.0**.

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

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

1. Decompress the **dws_8.1.x_odbc_driver_for_x86_redhat.zip** package.
   ```
   unzip dws_8.1.x_odbc_driver_for_x86_redhat.zip
   ```
2. Copy all files in the **lib** directory extracted from the **dws_8.1.x_odbc_driver_for_x86_redhat.zip** package to the **[your_path]/lib** directory.
3. Copy **psqlodbcw.la** and **psqlodbcw.so** in the **odbc/lib** directory to **[your_path]/lib/odbc**.

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

```
vi [your_path]/etc/odbcinst.ini
```

Copy the following content to the file:

```
[DWS]
Driver64=[your_path]/lib/odbc/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 [your_path]/etc/odbc.ini
```

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

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


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

Parameter	Description	Example Value
Sslmode	SSL certification mode. This parameter is enabled for the cluster by default. Values and meanings: • **disable**: only tries to establish a non-SSL connection. • **allow**: tries establishing a non-SSL connection first, and then an SSL connection if the attempt fails. • **prefer**: tries establishing an SSL connection first, and then a non-SSL connection if the attempt fails. • **require**: only tries establishing an SSL connection. If there is a CA file, perform the verification according to the scenario in which the parameter is set to **verify-ca**. • **verify-ca**: tries establishing an SSL connection and checks whether the server certificate is issued by a trusted CA. • **verify-full**: not supported by DWS **NOTE** The SSL mode delivers higher security than the common mode. By default, the SSL function is enabled in a cluster to allow SSL or non-SSL connections from the client. You are advised to use the SSL mode when using ODBC to connect to a DWS cluster.	Sslmode=allow

NOTE

- You can view the values of **Servername** and **Port** on the DWS console. Log in to the DWS console and click **Client Connections**. In the **Data Warehouse Connection String** area, select the target cluster and obtain **Private Network Address** or **Public Network Address**. For details, see **Obtaining the Connection Address of a DWS Cluster**.

Step 7 Configure environment variables.

```
vi ~/.bashrc
```

Add the following information to the configuration file:

```
export LD_LIBRARY_PATH=[your_path]/lib/:$LD_LIBRARY_PATH
export ODBCSYSINI=[your_path]/etc
export ODBCINI=[your_path]/etc/odbc.ini
```

NOTE

It is not recommended to add **LD_LIBRARY_PATH** in the Kylin OS, as it may cause conflicts with the **libssl.so** library. The latest cluster versions 8.2.1 and 9.1.0 now include the **rpath** mechanism, which allows the dependency to be located without using **LD_LIBRARY_PATH**.

Step 8 Import environment variables.

```
source ~/.bashrc
```

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

```
[your_path]/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 have the option to deploy the certificate either automatically or manually, depending on your requirements.

- Automatic deployment:

  Double-click the **sslcert_env.bat** file to trigger automatic deployment of the certificate 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:
  - Create a new folder named **postgresql** in the **%APPDATA%** directory.
  - 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**.
  - Copy **cacert.pem** to **%APPDATA%\postgresql** and change the name of **cacert.pem** to **root.crt**.

Step 3 Open Driver Manager.

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

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

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

  NOTE

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

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

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

  NOTE

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

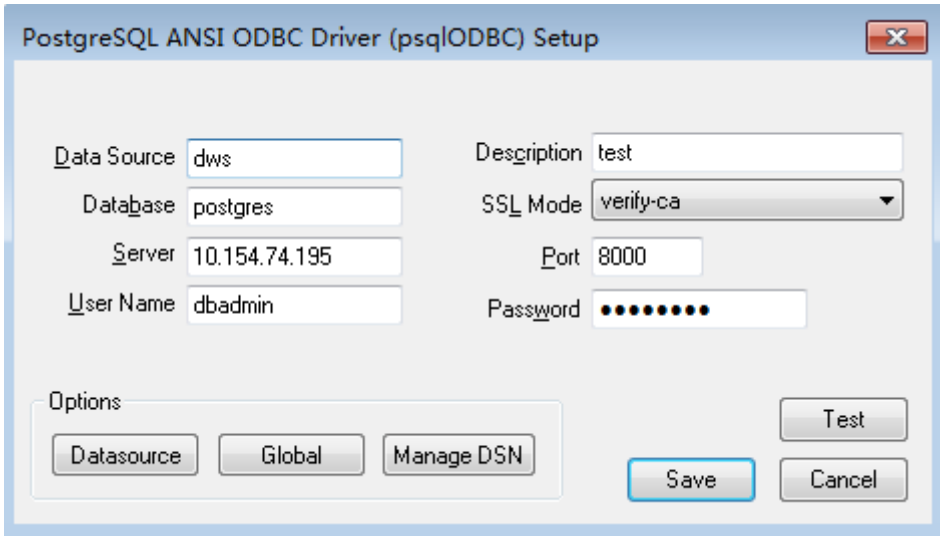
- In a 32-bit OS, open **C:\Windows\System32\odbcad32.exe**.

  Alternatively, click **Computer**, and choose **Control Panel**. Click **Administrative Tools** and click **Data Sources (ODBC)**.

Step 4 Configure a data source to be connected to.

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

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



   You can view the values of **Server** and **Port** on the DWS console. Log in to the DWS console and click **Client Connections**. In the **Data Warehouse**

Connection String area, select the target cluster and obtain **Private Network Address** or **Public Network Address**. For details, see **Obtaining the Connection Address of a DWS Cluster**.

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



```
    // 9. Prepare for execution.
    SQLPrepare(V_OD_hstmt,"insert into testtable values(?)",SQL_NTS);
    // 10. Bind parameters.
    SQLBindParameter(V_OD_hstmt,1,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_INTEGER,0,0,
                     &value,0,NULL);
    // 11. Execute the ready statement.
    SQLExecute(V_OD_hstmt);
    SQLExecDirect(V_OD_hstmt,"select id from testtable",SQL_NTS);
    // 12. Obtain the attributes of a certain column in the result set.
    SQLColAttribute(V_OD_hstmt,1,SQL_DESC_TYPE,typename,100,NULL,NULL);
    printf("SQLColAtrribute %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

6.5.2 Using the Python Library psycopg2 to Connect to a DWS Cluster

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

Preparations Before Connecting to a Cluster

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

  MD5 algorithms may by vulnerable to collision attacks and cannot be used for password verification. Currently, 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 contact the technical support to check whether the value of **password_encryption_type** is **1**. If the value is not **1**, change it. Then change the password of the database user to be used.

NOTE

- For security purposes, DWS no longer uses MD5 to store password digests by default. As a result, the open-source drives 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 DWS cluster. For details, see **Obtaining the Connection Address of a DWS Cluster**.
- You have installed the third-party database adapter 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, 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.

Version

There are many versions of DWS clusters, Python, and Psycopg2. The following table lists only the supported mainstream versions.

Table 6-14

Psycopg2 Version	Python Version	DWS Cluster Version
2.7.x	3.8.x	8.1.3 or later
	3.9.x	8.1.3 or later
2.8.x	3.8.x	8.1.3 or later
	3.9.x	8.1.3 or later
2.9.x	3.8.x	8.1.3 or later
	3.9.x	8.1.3 or later

Constraints

psycopg2 is a PostgreSQL-based client interface, and its functions are not fully supported by DWS. For details, see **Table 6-15**.

NOTE

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

Table 6-15 Psycopg2 APIs supported by DWS

Class Name	Usage	Function/Member Variable	Support	Remarks
connections	basic	cursor(*name=None*, *cursor_factory=None*, *scrollable=None*, *withhold=False*)	Y	-
		commit()	Y	-
		rollback()	Y	-
		close()	Y	-
	Two-phase commit support methods	xid(*format_id*, *gtrid*, *bqual*)	Y	-
		tpc_begin(*xid*)	Y	-
		tpc_prepare()	N	The kernel does not support explicit **PREPARE TRANSACTION**.
		tpc_commit([*xid*])	Y	-
		tpc_rollback([*xid*])	Y	-
		tpc_recover()	Y	-
		closed	Y	-
		cancel()	Y	-
		reset()	N	**DISCARD ALL** is not supported.
		dsn	Y	-

Class Name	Usage	Function/Member Variable	Support	Remarks
	Transaction control methods and attributes.	set_session(*isolation_level=None*, *readonly=None*, *deferrable=None*, *autocommit=None*)	Y	The database does not support the setting of **default_transaction_read_only** in a session.
		autocommit	Y	-
		isolation_level	Y	-
		readonly	N	The database does not support the setting of **default_transaction_read_only** in a session.
		deferrable	Y	-
		set_isolation_level(*level*)	Y	-
		encoding	Y	-
		set_client_encoding(enc)	Y	-
		notices	N	The database does not support **listen**/**notify**.
		notifies	Y	-
		cursor_factory	Y	-
		info	Y	-
		status	Y	-
		object	N	The database does not support operations related to large objects.

Class Name	Usage	Function/Member Variable	Support	Remarks
	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 **SCROLL CURSOR**.
		withhold	N	The **withhold cursor** needs to be closed before the commit operation.

Class Name	Usage	Function/Member Variable	Support	Remarks
	Commands execution methods	execute(*query*, *vars=None*)	Y	-
		executemany(*query*, *vars_list*)	Y	-
		callproc(*procname*[, *parameters*])	Y	-
		mogrify(*operation*[, *parameters*])	Y	-
		setinputsizes(*sizes*)	Y	-
		fetchone()	Y	-
		fetchmany([*size=cursor.arraysize*])	Y	-
		fetchall()	Y	-
		scroll(*value*[, *mode='relative'*])	N	The database does not support **SCROLL CURSOR**.
		arraysize	Y	-
		itersize	Y	-
		rowcount	Y	-
		rownumber	Y	-
		lastrowid	Y	-
		query	Y	-
		statusmessage	Y	-
		cast(*oid*, *s*)	Y	-
		tzinfo_factory	Y	-
		nextset()	Y	-
		setoutputsize(*size*[, *column*])	Y	-
	COPY-related methods	copy_from(*file*, *table*, *sep='||t'*, *null='||||N'*, *size=8192*, *columns=None*)	Y	-
		copy_to(*file*, *table*, *sep='||t'*, *null='||||N'*, *columns=None*)	Y	-
		copy_expert(*sql*, *file*, *size=8192*)	Y	-

Class Name	Usage	Function/Member Variable	Support	Remarks
	Interoperation with other C API modules	pgresult_ptr	Y	-

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

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

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

```
vi python_dws.py
```

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

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

from __future__ import print_function

import psycopg2


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


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


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



```
        print("Total number of rows updated :", cursor.rowcount)
        cursor.execute("select * from test order by 1;")
        rows = cursor.fetchall()
        for row in rows:
            print("id = ", row[0])
            print("name = ", row[1], "\n")
    except psycopg2.ProgrammingError as e:
        print(e)
    else:
        print("After Update, Operation done successfully")


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


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


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

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

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

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

Step 4 Connect to the cluster using the third-party database adapter 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 Psycopg to connect to the cluster:

```
python python_dws.py
```

----End

Why CN Retry Is Not Supported When Psycopg2 Is Connected to a Cluster?

With the CN retry feature, 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()
```

6.5.3 Using the Python Library PyGreSQL to Connect to a DWS Cluster

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

Preparations Before Connecting to a Cluster

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

  MD5 algorithms may by vulnerable to collision attacks and cannot be used for password verification. Currently, 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 contact the technical support to check whether the value of **password_encryption_type** is **1**. If the value is not **1**, change it. Then change the password of the database user to be used.

  NOTE

  - For security purposes, DWS no longer uses MD5 to store password digests by default. As a result, the open-source drives 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 DWS cluster. For details, see **Obtaining the Connection Address of a DWS Cluster**.
- 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 DWS. For details, see **Table 6-16**.

NOTE

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

Table 6-16 PyGreSQL APIs supported by DWS

PyGreSQL		Yes	Remarks
Module functions and constants	connect – Open a PostgreSQL connection	Y	-
	get_pqlib_version – get the version of libpq	Y	-
	get/set_defhost – default server host [DV]	Y	-
	get/set_defport – default server port [DV]	Y	-
	get/set_defopt – default connection options [DV]	Y	-
	get/set_defbase – default database name [DV]	Y	-
	get/set_defuser – default database user [DV]	Y	-

PyGreSQL		Yes	Remarks
	get/set_defpasswd – default database password [DV]	Y	-
	escape_string – escape a string for use within SQL	Y	-
	escape_bytea – escape binary data for use within SQL	Y	-
	unescape_bytea – unescape data that has been retrieved as text	Y	-
	get/set_namedresult – conversion to named tuples	Y	-
	get/set_decimal – decimal type to be used for numeric values	Y	-
	get/set_decimal_point – decimal mark used for monetary values	Y	-
	get/set_bool – whether boolean values are returned as bool objects	Y	-
	get/set_array – whether arrays are returned as list objects	Y	-
	get/set_bytea_escaped – whether bytea data is returned escaped	Y	-
	get/set_jsondecode – decoding JSON format	Y	-
	get/set_cast_hook – fallback typecast function	Y	-
	get/set_datestyle – assume a fixed date style	Y	-
	get/set_typecast – custom typecasting	Y	-
	cast_array/record – fast parsers for arrays and records	Y	-
	Type helpers	Y	-

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

PyGreSQL		Yes	Remarks
	inserttable – insert a list into a table	Y	Use double quotation marks ("") to quote **\n** in the **copy** command.
	get/set_notice_receiver – custom notice receiver	Y	-
	putline – write a line to the server socket [DA]	Y	-
	getline – get a line from server socket [DA]	Y	-
	endcopy – synchronize client and server [DA]	Y	-
	locreate – create a large object in the database [LO]	N	Operations related to large objects
	getlo – build a large object from given oid [LO]	N	Operations related to large objects
	loimport – import a file to a large object [LO]	N	Operations related to large objects
	Object attributes	Y	-
The DB wrapper class	Initialization	Y	-
	pkey – return the primary key of a table	Y	-
	get_databases – get list of databases in the system	Y	-
	get_relations – get list of relations in connected database	Y	-
	get_tables – get list of tables in connected database	Y	-
	get_attnames – get the attribute names of a table	Y	-
	has_table_privilege – check table privilege	Y	-

PyGreSQL		Yes	Remarks
	get/set_parameter – get or set run-time parameters	Y	-
	begin/commit/rollback/ savepoint/release – transaction handling	Y	-
	get – get a row from a database table or view	Y	-
	insert – insert a row into a database table	Y	-
	update – update a row in a database table	Y	-
	upsert – insert a row with conflict resolution	Y	-
	query – execute a SQL command string	Y	-
	query_formatted – execute a formatted SQL command string	Y	-
	query_prepared – execute a prepared statement	Y	-
	prepare – create a prepared statement	Y	-
	describe_prepared – describe a prepared statement	Y	-
	delete_prepared – delete a prepared statement	Y	-
	clear – clear row values in memory	Y	-
	delete – delete a row from a database table	Y	A tuple must have unique key or primary key.
	truncate – quickly empty database tables	Y	-
	get_as_list/dict – read a table as a list or dictionary	Y	-

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

PyGreSQL		Yes	Remarks
	memsize – return number of bytes allocated by query result	Y	-
LargeObject – Large Objects	open – open a large object	N	Operations related to large objects
	close – close a large object	N	Operations related to large objects
	read, write, tell, seek, unlink – file-like large object handling	N	Operations related to large objects
	size – get the large object size	N	Operations related to large objects
	export – save a large object to a file	N	Operations related to large objects
	Object attributes	N	Operations related to large objects
The Notification Handler	Instantiating the notification handler	N	The database does not support **listen**/**notify**.
	Invoking the notification handler	N	The database does not support **listen**/**notify**.
	Sending notifications	N	The database does not support **listen**/**notify**.
	Auxiliary methods	N	The database does not support **listen**/**notify**.
pgdb			
Module functions and constants	connect – Open a PostgreSQL connection	Y	-

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

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

7 Creating a DWS Database and User

The default database **gaussdb** of DWS is not used as the customer's service database. You can use multiple databases to ensure service isolation. When you first connect to **gaussdb** as the system administrator (**dbadmin**), it is important to plan the service databases, users, and roles based on the service requirements. This involves creating a service and transferring any existing upstream service data to DWS.

A role is a set of permissions. For details about the relationship between users and roles, see Permissions Management in the *Developer Guide*. You can create common roles, such as a role for database creation, before creating a user. Then, you can assign the created role to the user.

Users, roles, and permissions can be exported. For details, see **Exporting a User**, **Exporting User Permissions**, **Exporting Roles**, and **Exporting Role Permissions**.

Constraints and Limitations

- Avoid having all business operations run under a single database user. Instead, plan different database users according to the business modules.
- For better access control of different business modules, it is better to use multiple users and permissions instead of depending on the system administrator user to run business operations.
- For more information about development and design specifications, see "DWS Development and Design Proposal" in the *Data Warehouse Service (DWS) Developer Guide*.

Creating a Database

You can use the DDL syntax or SQL editor to create a table.

- DDL syntax: For details about the syntax, see "CREATE DATABASE".

Creating a Role

Step 1 Log in to the DWS console. In the navigation pane, choose **Dedicated Clusters** > **Clusters**.

Step 2 In the cluster list, click the name of the target cluster. The cluster information page is displayed.

Step 3 In the navigation pane, choose **User Management**.

Step 4 Click the **Roles** tab and click **Create Role**. The role creation page is displayed.

Step 5 Configure role information. The parameters are described as follows:

Table 7-1 Parameters for configuring role information

Parameter	Description	Example Value
Role Name	The value must start with a letter and can contain a maximum of 63 characters, including letters, digits, and underscores (_).	DWS-demo
Expires	Expiration time of the role permissions.	-
System Administrator	Whether the role has the system administrator rights.	-
Create Database	Whether the role has the permission to create databases.	-
Create Role	Whether the role has the permission to create users and roles.	-
Inherit Permissions	Whether the role inherits the permissions from its role group. By default, this function is enabled and it is best to keep it that way.	-

Step 6 Confirm the settings and click **Next**.

Step 7 Configure the permissions of the role.

Click **Add** to add a permission configuration. Select the database object type and the corresponding objects. Then, select permissions. For details about permission definitions, see "DCL Syntax" > "GRANT" in *Data Warehouse Service (DWS) SQL Syntax Reference*.

Step 8 After the authorization is complete, click **Create**.

----End

Creating a Database User

You can use the DDL syntax or create a table on the DWS console. For details about the DDL syntax, see "CREATE USER".

NOTE

- If the current console does not support this feature, contact technical support.
- After a cluster is created, the users or roles created with it cannot be modified.
- Before using this function, ensure that the cluster is available.

Step 1 Log in to the DWS console. In the navigation pane, choose **Dedicated Clusters** > **Clusters**.

Step 2 In the cluster list, click the name of the target cluster. The cluster information page is displayed.

Step 3 In the navigation pane, choose **User Management**.

Step 4 On the **Users** page, click **Create User**.

Step 5 Set the parameters on the **Configure Basic Settings** page.

Table 7-2 Parameters on the Configure Basic Settings page

Parameter	Description	Example Value
Username	The value must start with a letter and can contain a maximum of 63 characters, including letters, digits, and underscores (_).	DWS-demo
Password	Enter a value that is 12 to 32 characters long and can contain letters, digits, underscores (_), and special characters. **NOTE** Your password must contain a minimum of three of the following character types: uppercase letters, lowercase letters, digits, and special characters (~!?,.:;_(){}[]/<>@#%^&*+|\=-)	-
Maximum Connections	Maximum number of connections between the user and the database. The value **-1** indicates that the number of connections is not limited.	-1
Expires	Expiration time of the user's permissions.	-
System Administrator	Whether the user is a system administrator.	-
Create Database	Whether the user has the permission to create databases.	-
Create Role	Whether the user has the permission to create users and roles.	-

Parameter	Description	Example Value
Inherit Permissions	Whether the user inherits permissions from its user group. By default, this function is enabled and it is best to keep it that way.	-

Step 6 Confirm the settings and click **Next**.

Step 7 On the **Configure Roles** page, select the role to be assigned to the user and click **Next**.

Step 8 Configure permissions not included in the roles of the user.

Click **Add** to add a permission configuration. Select the database object type and corresponding database object, and select the permission to complete assignment. For details about permission definitions, see "DCL Syntax" > "GRANT" in *Data Warehouse Service (DWS) SQL Syntax Reference*.

Step 9 After the authorization is complete, click **Create**.

----End

Modifying a User

Step 1 Log in to the DWS console. In the navigation pane, choose **Dedicated Clusters** > **Clusters**.

Step 2 In the cluster list, click the name of the target cluster. The cluster information page is displayed.

Step 3 In the navigation pane, choose **User Management**.

Step 4 In the user list, select a user and click **Modify**. The page for modifying user details is displayed.

Step 5 Modify the user information. For details, see **Table 7-2**. After confirming that the information is correct, click **Next**.

Step 6 Select the role to be granted to the user and click **Next**.

Step 7 After selecting a permission type, you can click **Edit** in the **Operation** column and click **Modify** in the **Permission** column to add or remove a permission.

Step 8 Confirm the permissions. Click **Save**.

----End

Deleting a User

Prerequisites

To prevent any problems with deleting a user, check for dependencies between database objects (such as tables) beforehand. If there are any dependencies, delete them first before proceeding with the user deletion.

Procedure

Step 1 Log in to the DWS console. In the navigation pane, choose **Dedicated Clusters** > **Clusters**.

Step 2 In the cluster list, click the name of the target cluster. The cluster information page is displayed.

Step 3 In the navigation pane, choose **User Management**.

Step 4 Select a user from the user list and click **Delete**. A confirmation dialog box is displayed. If you select **Forcibly delete and remove dependencies**, tables and other database objects under the current user will be transferred to the administrator account.

Step 5 Click **OK**.

----End

Exporting a User

Step 1 Log in to the DWS console. In the navigation pane, choose **Dedicated Clusters** > **Clusters**.

Step 2 In the cluster list, click the name of the target cluster. The cluster information page is displayed.

Step 3 In the navigation pane, choose **User Management**.

Step 4 Click **Export** in the upper part of the user list and select the number of records to be exported to export the user list.

Step 5 Confirm the configurations and click **Export**.

----End

Exporting User Permissions

Step 1 Log in to the DWS console. In the navigation pane, choose **Dedicated Clusters** > **Clusters**.

Step 2 In the cluster list, click the name of the target cluster. The cluster information page is displayed.

Step 3 In the navigation pane, choose **User Management**.

Step 4 Select a user from the user list and click **Export Permissions** to export the user permission list.

----End

Modifying a Role

Step 1 Log in to the DWS console. In the navigation pane, choose **Dedicated Clusters** > **Clusters**.

Step 2 In the cluster list, click the name of the target cluster. The cluster information page is displayed.

Step 3 In the navigation pane, choose **User Management**.

Step 4 In the role list, select a user and click **Modify**. The page for modifying role details is displayed.

Step 5 Modify the role information. For the parameter description, see **Table 7-1**.

Step 6 Confirm the settings and click **Next**.

Step 7 Configure permissions. Select a permission type as required, click **Edit** in the **Operation** column, and click **Modify** in the **Permission** column to add or remove permissions.

Step 8 Confirm the permissions. Click **Save**.

----End

Deleting a Role

Prerequisites

To prevent any problems with deleting a role, check for dependencies such as database objects beforehand. If there are any dependencies, delete them first before proceeding with the role deletion.

Procedure

Step 1 Log in to the DWS console. In the navigation pane, choose **Dedicated Clusters** > **Clusters**.

Step 2 In the cluster list, click the name of the target cluster. The cluster information page is displayed.

Step 3 In the navigation pane, choose **User Management**.

Step 4 Select a role from the role list and click **Delete**. A confirmation dialog box is displayed.

Step 5 Click **OK** to delete the role.

----End

Exporting Roles

Step 1 Log in to the DWS console. In the navigation pane, choose **Dedicated Clusters** > **Clusters**.

Step 2 In the cluster list, click the name of the target cluster. The cluster information page is displayed.

Step 3 In the navigation pane, choose **User Management** and click **Roles** to switch to the role list page.

Step 4 Click **Export** in the upper part of the role list and select the number of roles to be exported.

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

----End

Exporting Role Permissions

Step 1 Log in to the DWS console. In the navigation pane, choose **Dedicated Clusters** > **Clusters**.

Step 2 In the cluster list, click the name of the target cluster. The cluster information page is displayed.

Step 3 In the navigation pane, choose **User Management** and click **Roles** to switch to the role list page.

Step 4 Select a user from the role list, click **Export Permissions**, and select the number of records to be exported.

----End

8 Migrating Service Data to a DWS Cluster

8.1 Data Source Management

8.1.1 MRS Data Sources

8.1.1.1 MRS Data Source Usage Overview

MRS Cluster Overview

MRS is a big data cluster running based on the open-source Hadoop ecosystem. It provides the industry's latest cutting-edge storage and analysis capabilities of massive volumes of data, satisfying your data storage and processing requirements. For details about MRS, 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 DWS, you can connect a data warehouse cluster to MRS clusters, read data from HDFS files, and write the data to DWS when the clusters are on the same network.

Operation Process

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

1. Prerequisites
   a. Create an MRS cluster. For details, see "Buying a Custom Cluster" in *MapReduce User Guide*.
   b. Create an HDFS foreign table for querying data from the MRS cluster over APIs of a foreign server.

      For details, see **Data Import > Importing Data from MRS to a Cluster** in the *Data Warehouse Service Database Development Guide*.

NOTE

- Multiple MRS data sources can exist on the same network, but one DWS cluster can connect to only one MRS cluster at a time.

2. In the DWS cluster, create an MRS data source connection according to **Creating an MRS Data Source Connection**.
3. Import data from an MRS data source to the cluster. For details, see "Data Import > Importing Data from MRS to a Data Warehouse Cluster".
4. (Optional) When the HDFS configuration of the MRS cluster changes, update the MRS data source configuration on DWS. For details, see **Updating the MRS Data Source Configuration**.

8.1.1.2 Creating an MRS Data Source Connection

Scenario

Before DWS reads data from MRS HDFS, you need to create an MRS data source connection that functions as a channel of transporting DWS cluster data and MRS cluster data.

Impact on the System

- You can create only one MRS data source connection in the DWS 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 DWS 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 DWS cluster and recorded the VPC and subnet where the cluster resides.
- An MRS cluster of the analysis type has been created.

Procedure

Step 1 Log in to the Huawei Cloud console.

Step 2 Go to the MRS console and create an MRS cluster.

Configure parameters as required. For details, see "Cluster Operation Guide > Custom Creation of a Cluster" in the *MapReduce Service User Guide*.

- The VPC of the MRS cluster must be the same as that of the DWS cluster.
- Select an MRS cluster version. The following versions are supported:
  - For clusters of version 8.1.1.300 and later, MRS clusters support versions 1.6.*, 1.7.*, 1.8.*, 1.9.*, 2.0.*, 3.0.*, 3.1.*, 3.2.*, 3.3.*, and later (*indicates a number).

  - For clusters earlier than version 8.1.1.300, MRS clusters support versions 1.6.*, 1.7.*, 1.8.*, 1.9.*, and 2.0.* (* indicates a number).
- Select the Hadoop component.

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

Step 3 Go to the DWS console.

Step 4 In the navigation pane on the left, choose **Dedicated Clusters** > **Clusters**.

Step 5 In the cluster list, click the name of a cluster. The **Cluster Information** page is displayed.

Step 6 In the navigation tree on the left, choose **Data Sources** > **MRS Data Sources**.

Step 7 Click **Create MRS Cluster Connection** and configure parameters.

Table 8-1 MRS common connection parameters

Parameter	Description
Data Source	The DWS database server name. It can contain 3 to 63 characters, including lowercase letters, numbers, and underscores (_), and must start with a lowercase letter.
Configuration Mode	The way in which the system obtains files. The options are as follows: • **MRS Account**: Configure the username and password of the Manager of the MRS cluster. The system will log in to the Manager and automatically download configuration and verification files. For more information, see **Table 8-2**. • **File upload**: Download the configuration file from the Manager of the MRS cluster and manually upload it. You can use this method for Kerberos authentication. For more information, see **Table 8-3**. **NOTE** – If you select **File upload**, ensure that MRS can communicate with the DWS cluster.
Database	Database where the data source is located.
Description	Description of the connection.

Table 8-2 Parameters of the MRS Account mode

Parameter	Description
MRS Data Source	Select an MRS cluster that can be connected to DWS from the drop-down list box. By default, the custom, hybrid, and analytical MRS clusters that are in the same VPC and subnet as the current DWS cluster and available to the current user are displayed. After you select an MRS cluster, the system automatically displays whether Kerberos authentication is enabled for the selected cluster. Click **View MRS Cluster** to view its detailed information. If the **MRS Data Source** drop-down list is empty, click **Create MRS Cluster** to create an MRS cluster.
MRS Account	Account used when a DWS cluster connects to an MRS cluster.
Password	Password of the connection user. If you change the password, you need to create a connection again. **NOTICE** Ensure the account has been used for logging in to MRS Manager. If you use a new account, you will be asked to change your password when you first log in. In this case, the MRS data source will fail to be configured.
Use a Machine-Machine Account	Creates a machine-machine account named dws in MRS and uses it for interaction with MRS. This account is in the **supergroup** group and has all permissions. If the switch is toggled off, the configured man-machine account will be used. Ensure this account has the permission to access data, or a message will be displayed during data source access, indicating the required file does not exist.

Table 8-3 Parameters of the File upload mode

Parameter	Description
Authentication Credential	Keytab file of a user A credential file downloaded from Manager of the MRS cluster. File name format: **Username_Timestamp_keytab.tar** • **For MRS 2.x or earlier**, choose **System** > **Manage User**. In the **Operation** column of a user, choose **More** > **Download authentication credential**. • **For MRS 3.x or later**, choose **System** > **Permission** > **User**. In the **Operation** column of a user, choose **More** > **Download Authentication Credential**.

Parameter	Description
Client Profile	Client configuration files of HDFS, Hive, and hosts. When downloading the client, set **Select Client Type** to **Configuration Files Only**. • **For MRS 2.x or earlier**, choose **Services** and click **Download Client**. • **For MRS 3.x or later**, choose **Homepage**. Click the **More** icon and choose **Download Client**.

Step 8 Click **OK** to save the connection.

Configuration Status turns to **Creating**. You can view the connection that is successfully created in the MRS data source list and the connection status is **Available**.

NOTE

- In the **Operation** column, you can click **Update Configurations** to update **MRS Cluster Status** and **Configuration Status**. During configuration update, you cannot create a connection. The system checks whether the security group rule is correct. If the rule is incorrect, the system rectifies the fault. For details, see **Updating the MRS Data Source Configuration**.
- In the **Operation** column, you can click **Delete** to delete the unnecessary connection. When deleting a connection, you need to manually delete the security group rule.
- If the security group rules are not deleted, nodes in the data warehouse cluster can still communicate with nodes in the MRS cluster. If you have strict requirements on network security, manually delete the rules.

----End

8.1.1.3 Updating the MRS Data Source Configuration

Scenario

For MRS, if the following parameter configurations of the HDFS cluster change, data may fail to be imported to the DWS cluster from the HDFS cluster. Before importing data using the HDFS cluster, you must update the MRS data source configuration.

Prerequisites

You have created an MRS data source connection for the DWS cluster.

Impact on the System

When you are updating an MRS data source connection, the DWS cluster will automatically restart and cannot provide services.

Procedure

Step 1 In the navigation pane on the left, choose **Dedicated Clusters** > **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. The following table describes the parameters.

Table 8-4 Parameter description

Parameter	Description
dfs.client.read.shortcircuit	Specifies whether to enable the local read function.
dfs.client.read.shortcircuit.skip.checksum	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. The value **true** indicates that the channels are encrypted. The channels are not encrypted by default. **NOTE** • This parameter is available only for clusters with Kerberos authentication enabled. • This parameter is valid only when **hadoop.rpc.protection** is set to **privacy**.
dfs.encrypt.data.transfer.algorithm	Specifies the encryption and decryption algorithm for key transmission. This parameter is valid only when **dfs.encrypt.data.transfer** is set to **true**. The default value is **3des**, indicating that the 3DES algorithm is used for encryption.
dfs.encrypt.data.transfer.cipher.suites	Specifies the encryption and decryption algorithm for the transmission of actually stored data. If this parameter is not specified, the cryptographic algorithm specified by **dfs.encrypt.data.transfer.algorithm** is used for data encryption. The default value is **AES/CTR/NoPadding**.
dfs.replication	Specifies the default number of data copies.
dfs.blocksiz	Specifies the default size of a data block.

Parameter	Description
hadoop.security.authentication	Specifies the security authentication mode.
hadoop.rpc.protection	Specifies the RPC communication protection mode. Default value: • Security mode (Kerberos authentication enabled): **privacy** • Common mode (Kerberos authentication disabled): **authentication** **NOTE** • **authentication**: indicates that only authentication is required. • **integrity**: indicates that authentication and consistency check need to be performed. • **privacy**: indicates that authentication, consistency check, and encryption need to be performed.
dfs.domain.socket.path	Specifies the locally used **Domain socket** path.

----End

8.1.2 Managing OBS Data Sources

DWS allows you to access data on OBS by using an agency. You can create a DWS agency, grant the OBS OperateAccess or OBS Administrator permission to the agency, and bind the agency to an OBS data source you created. In this way, you can access data on OBS by using OBS foreign tables.

NOTE

- This feature is supported only in 8.2.0 or later.
- For the OBS data source of a cluster, only one of the creation, modification, and deletion operations can be performed at a time.

Creating an OBS Agency

Scenario

Before creating an OBS data source, create an agency that grants DWS the OBS OperateAccess or OBS Administrator permission.

Procedure

Step 1 Click your account in the upper right corner of the page and choose **Identity and Access Management**.

Step 2 In the navigation pane on the left, choose **Agency**. In the upper right corner, click **Create Agency**.

Step 3 Select **Cloud Service** and set **Cloud Service** to **DWS**.

Step 4 Click **Next** to grant the OBS OperateAccess or OBS Administrator permission to the agency.

Step 5 Click **Next**. Select **All resources** or specific resources, confirm the information, and click **Submit**.

----End

Creating an OBS Data Source

Prerequisites

An agency has been created to grant DWS the OBS OperateAccess permission.

Procedure

Step 1 In the navigation pane on the left, choose **Dedicated Clusters** > **Clusters**.

Step 2 In the cluster list, click the name of a cluster. On the page that is displayed, choose **Data Sources** > **OBS Data Source**.

Step 3 Click **Create OBS Cluster Connection** and configure parameters.

Table 8-5 OBS data source connection parameters

Parameter	Description
Data Source Name	Name of the OBS data source connection to be created. You can assign a personalized value to this parameter. The data source name is used as the server name specified in the statement for creating an OBS foreign table.
OBS Agency	Agency with the OBS OperateAccess permission to be granted to DWS
Database	Database where the OBS data source connection is to be created
Description	Description about the OBS data source connection

Step 4 Confirm the settings and click **OK**. The creation takes about 10 seconds.

----End

Updating the OBS Data Source Configuration

Scenario

After an OBS data source connection is created, DWS periodically updates the temporary agency information used by the data source. If the automatic update fails for 24 hours, the data source connection will be unavailable. To solve this problem, manually update the information on the console.

Procedure

Step 1 In the navigation pane on the left, choose **Dedicated Clusters** > **Clusters**.

Step 2 In the cluster list, click the name of a cluster. On the page that is displayed, choose **Data Sources** > **OBS Data Source**.

Step 3 In the **Operation** column of an OBS data source, click **Update Configuration**.

Step 4 Confirm the settings and click **OK**. The update takes about 10 seconds.

----End

Changing the OBS Data Source Agency

Scenario

You can change the agency bound to the OBS data source.

Procedure

Step 1 In the navigation pane on the left, choose **Dedicated Clusters** > **Clusters**.

Step 2 In the cluster list, click the name of a cluster. On the page that is displayed, choose **Data Sources** > **OBS Data Source**.

Step 3 In the **Operation** column of a data source, click **Manage Agency**. In the dialog box that is displayed, select a new agency.

Step 4 Confirm the settings and click **OK**. The change takes about 10 seconds.

----End

Deleting an OBS Data Source

Step 1 In the navigation pane on the left, choose **Dedicated Clusters** > **Clusters**.

Step 2 In the cluster list, click the name of a cluster. On the page that is displayed, choose **Data Sources** > **OBS Data Source**.

Step 3 In the **Operation** column of an OBS data source, click **Delete**.

Step 4 Confirm the settings and click **OK**. The deletion takes about 10 seconds.

----End

Using an OBS Data Source

DWS uses foreign tables to access data on OBS. The **SERVER** parameters specified for accesses with and without an agency are different.

If you access OBS without an agency, the **SERVER** provided on the console contains parameters **access_key** and **secret_access_key**, which are the AK and SK of the OBS access protocol, respectively.

If you access OBS with an agency, the **SERVER** provided on the console contains the **access_key**, **secret_access_key**, and **security_token** parameters, which are the temporary AK, temporary SK, and the **SecurityToken** value of the temporary security credential in IAM, respectively.

After the OBS agency and OBS data source are created, you can obtain the **SERVER** information on the console. Assume that the OBS data source name is **obs_server**. The way users create and use foreign tables with an agency is the same as the way they do without an agency. For how to use the OBS data source, see "Data Migration" > "Importing Data from OBS" in the *Data Warehouse Service (DWS) Developer Guide*.

The following example shows how common user **jim** reads data from OBS through a foreign table.

1. Repeat the preceding steps to create an OBS data source named **obs_server**.
2. Connect to the database as system administrator dbadmin, create a common user, and grant the common user the permission to use OBS servers and OBS foreign tables. Replace **{Password}** with the actual password and **obs_server** with the actual OBS data source name.

   ```
   CREATE USER jim PASSWORD '{Password}';
   ALTER USER jim USEFT;
   GRANT USAGE ON FOREIGN SERVER obs_server TO jim;
   ```

3. Connect to the database as common user **jim** and create an OBS foreign table **customer_address** that does not contain partition columns.

   In the following command, replace **obs_server** with the name of the created OBS data source. Replace **/user/obs/region_orc11_64stripe1/** with the actual OBS directory for storing data files. **user** indicates the OBS bucket name.

   ```
   CREATE FOREIGN TABLE customer_address
   (
       ca_address_sk             integer               not null,
       ca_address_id             char(16)              not null,
       ca_street_number          char(10)                      ,
       ca_street_name            varchar(60)                   ,
       ca_street_type            char(15)                      ,
       ca_suite_number           char(10)                      ,
       ca_city                   varchar(60)                   ,
       ca_county                 varchar(30)                   ,
       ca_state                  char(2)                       ,
       ca_zip                    char(10)                      ,
       ca_country                varchar(20)                   ,
       ca_gmt_offset             decimal(36,33)                ,
       ca_location_type          char(20)
   )
   SERVER obs_server OPTIONS (
       FOLDERNAME '/user/obs/region_orc11_64stripe1/',
       FORMAT 'ORC',
       ENCODING 'utf8',
       TOTALROWS  '20'
   )
   DISTRIBUTE BY roundrobin;
   ```

4. Query data stored in OBS by using a foreign table.

   ```
   SELECT COUNT(*) FROM customer_address;
   count
   -------
   20
   (1row)
   ```


9 DWS Cluster Data Security and Encryption

9.1 Enabling Separation of Duties for DWS Database Users

Scenario

By default, the administrator specified when you create a 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.

To ensure cluster data security, DWS supports separation of duties for clusters. Different types of users have different permissions.

For details about the default permissions mode and the separation of permissions mode, see "Database Security Management > Managing Users and Their Permissions > Separation of Permissions" in the *Data Warehouse Service (DWS) Developer Guide*.

Impact on the System

- After you modified the security parameters and the modifications take effect, the cluster may be restarted, which makes the cluster unavailable temporarily.

Prerequisites

To modify the cluster's security configuration, ensure that the following conditions are met:

- The cluster status is **Available** or **Unbalanced**.
- The target cluster should not be undergoing any node additions, specification changes, configurations, upgrades, redistribution operations, or restarts.

Procedure

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

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.

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. **By default, this function is disabled.**

Table 9-1 Security parameters

Parameter	Description	Example Value
Security Administrator	The username must meet the following requirements: ● Consists of lowercase letters, digits, or underscores. ● Starts with a lowercase letter or an underscore. ● Contains 6 to 64 characters. ● The username cannot be a keyword of the DWS database. For details about the keywords of the DWS database, see "SQL Reference" > "Keyword" in the *Data Warehouse Service Developer Guide*.	security_admin
Password	The password complexity requirements are as follows: ● Contain 12 to 32 characters. ● Cannot be the username or the username spelled backwards. ● Contains at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters (~!?,.:;_(){}[]/<>@#%^&*+|\\=-) ● Perform the weak password check on the passwords that you have created.	-
Confirm Password	Enter the password of the security administrator again.	-

Parameter	Description	Example Value
Audit Administrator	The username must meet the following requirements: • Consists of lowercase letters, digits, or underscores. • Starts with a lowercase letter or an underscore. • Contains 6 to 64 characters. • The username cannot be a keyword of the DWS database. For details about the keywords of the DWS database, see "SQL Reference" > "Keyword" in the *Data Warehouse Service Developer Guide*.	audit_admin
Password	The password complexity requirements are as follows: • Contain 12 to 32 characters. • Cannot be the username or the username spelled backwards. • Must contain at least 3 of the following character types: uppercase letters, lowercase letters, digits, and special characters ~!@#%^&*()-_=+|[{}];:,<.>/? • Passes the weak password check.	-
Confirm Password	Enter the password of the audit administrator again.	-

Step 5 Click **Apply**.

Step 6 In the displayed **Save Configuration** dialog box, select or deselect **Restart the cluster** and click **Yes**.

- If you select **Restart the cluster**, the system saves the settings on the **Security Settings** page and restarts the cluster immediately. After the cluster is restarted, the security settings take effect immediately.
- If you do not select **Restart the cluster**, the system only saves the settings on the **Security Settings** page. Later, you need to manually restart the cluster for the security settings to take effect.

After the security settings are complete, **Configuration Status** can be one of the following on the **Security Settings** page:

- **Applying**: The system is saving the settings.
- **Synchronized**: The settings have been saved and taken effect.
- **Take effect after restart**: The settings have been saved but have not taken effect. Restart the cluster for the settings to take effect.

----End

9.2 Using KMS to Encrypt DWS Clusters

9.2.1 Overview

Encrypting DWS Databases

In 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, you must export all data from the unencrypted cluster and import it into a new cluster that has database encryption enabled. DWS encrypts data as it is written to the database, and automatically decrypts it when queried, returning the results to the user.

If encryption is required, enable it during cluster creation. Although encryption is an optional setting of 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. For details, see **Encrypting the Database**. After a normal cluster is created, you can convert it to an encrypted cluster.
- After **Encrypt DataStore** is enabled, the key cannot be disabled, deleted, or frozen when being used. Otherwise, the cluster becomes abnormal and the database becomes unavailable.
- Snapshots created after the database encryption function is enabled cannot be restored using open APIs.

Viewing Database Encryption Information

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

Step 3 Click the name of a cluster. The **Cluster Information** page is displayed.

Step 4 In the **Data Encryption Information** area on the cluster information page, view the database encryption information, as shown in **Table 9-2**.

Table 9-2 Data encryption information

Parameter	Description
Key Name	Indicates the database encryption key of the cluster when **Encrypt DataStore** is enabled.

Parameter	Description
Last Key Rotation Time	Indicates the time when the last encryption key is rotated when **Encrypt DataStore** is enabled.
Cryptographic Algorithm	Indicates the encryption algorithm of the cluster when **Encrypt DataStore** is enabled.

NOTE

If database encryption is disabled by default during cluster creation, the encryption module is not displayed on the cluster details page.

----End

Using KMS to Encrypt DWS Databases

When you choose KMS to manage 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 DWS cluster's memory, and the ciphertext is stored in DWS.
- The DEK is used to encrypt database data. The DEK plaintext is stored in the DWS cluster's memory, and the ciphertext is stored in DWS.

The procedure of using the keys is as follows:

1. You choose a CMK.
2. 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 DWS.
4. DWS uses the CEK plaintext to encrypt the DEK plaintext and saves the encrypted DEK ciphertext.
5. 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 DWS through an API. 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 DWS. On DWS, you can rotate the encrypted CEK of an encrypted cluster.

The procedure of rotating the keys is as follows:

1. The DWS cluster starts key rotation.
2. DWS decrypts the CEK ciphertext stored on DWS based on the CMK to obtain the CEK plaintext.
3. Use the obtained CEK plaintext to decrypt the DEK ciphertext in DWS to obtain the DEK plaintext.
4. DWS randomly generates new CEK plaintext.
5. 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 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 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 DWS cluster memory, and the ciphertext is stored in DWS.
- The DEK is not updated during key rotation, so data encryption and decryption are not affected.

9.2.2 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. When a normal cluster is converted to an encrypted cluster, you can rotate the encryption key for the cluster. Each key rotation will update the CEK once. During the key rotation, the cluster is still in **Available** status.

Rotating Encryption Keys for DWS Clusters

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

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

Step 4 In the **Data Encryption Information** area, click **Key Rotation**.

Step 5 In the dialog box that is displayed, click **Yes**.

----End

9.2.3 Converting an Ordinary Cluster to an Encrypted Cluster

DWS allows you to convert an unencrypted cluster to an encrypted cluster when the cluster status is **Available** on the console. To ensure data security, converting a cluster to an encrypted cluster is an **irreversible high-risk operation** and will restart the cluster. As a result, services may be unavailable for a short period of time. Exercise caution when performing this operation.

> **NOTE**
>
> If the current console does not support this function, contact technical support.

Notes and Constraints

- Only storage-compute decoupled clusters of version 9.1.0 and later support database encryption.
- If the cluster has a DR relationship, the cluster cannot be encrypted even if the cluster is in the available state. You need to cancel the DR relationship, encrypt the cluster, and then re-establish the DR relationship.
- 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.
- By default, only Huawei Cloud accounts or users with **Security Administrator** permissions can query and create agencies. IAM users under an account do not have the permission to query or create agencies by default. Contact a user with that permission and complete the authorization on the current page.

Creating a KMS Agency

Scenario

Before converting a cluster to an encrypted cluster, you need to create an agency that grants the KMS Administrator permissions to DWS.

Procedure

Step 1 Click your account in the upper right corner of the page and choose **Identity and Access Management**.

Step 2 In the navigation pane on the left, choose **Agency**. In the upper right corner, click **Create Agency**.

Step 3 Select **Cloud Service** and set **Cloud Service** to **DWS**.

Step 4 Click **Finish**. In the displayed dialog box, click **OK** to grant the **KMS Administrator** permission to the agency.

Step 5 Click **Next**. Select **All resources** or specific resources, confirm the information, and click **Submit**.

----End

Procedure

Step 1 Log in to the DWS console. In the navigation pane, choose **Dedicated Clusters** > **Clusters**.

Step 2 In the cluster list, locate the row that contains the target cluster and choose **More** > **Convert to Encrypted Cluster** in the **Operation** column.

> **NOTE**
>
> The positions of the function keys in the **Operation** column are dynamic. To ensure that there are always two function keys visible before **More**, any function keys that typically appear only when you hover over **More** will be moved to a position directly before **More**. This adjustment occurs if there are some functions whose keys are supposed to be placed before **More** but are not supported for the current site.

Step 3 In the dialog box that is displayed, select the key source, key name, and encryption algorithm to convert the cluster to an encrypted cluster.

- Method 1: Select a key name.
- Method 2: Enter the key ID. Enter the key ID used for authorizing the current tenant.

  When you grant permissions on the Creating a Grant page, the authorized object must be an account instead of a user. The authorized operations must at least contain **Querying key details**, **Encrypting data**, and **Decrypting data**.

Step 4 After the conversion, you can click the cluster name to go to the **Cluster Details** page to view the cluster details. For details, see **Viewing Database Encryption Information**.

----End

10 DWS Cluster Management

10.1 Viewing DWS Cluster Details

Log in to the DWS console. In the navigation tree on the left, click **Dedicated Clusters** > **Clusters**. In the cluster list, locate the required cluster and click its name. The **Cluster Information** page is displayed.

On the **Cluster Information** page, you can view the following information:

- **Basic Information**: Table 10-1 lists the related parameters.
- **Connection**: Table 10-2 describes the parameters.
- **Network**: Table 10-3 lists the related parameters.
- **Storage/Backup Capacity**: Table 10-4 describes the parameters.
- **Data Encryption Information**: Table 10-5 lists the related parameters.

  NOTE

  You can view this module if you enable the data encryption function when creating a cluster.

Table 10-1 Basic information

Parameter	Description
Cluster Name	Cluster name specified when a cluster is created.
Cluster Status	Cluster running status. For details, see **Cluster Status**.
Parameter Configuration Status	Parameter configuration status of a cluster.
Task Information	Cluster task status. For details, see **Cluster Task Information**.

Parameter	Description
Current Specifications	Current node specifications.
Nodes	Number of nodes in the cluster.
Cluster ID	ID of the cluster.
Cluster Version	Cluster version information.
Node Flavor	Node flavor of the cluster.
Enterprise Project	Enterprise project to which a cluster belongs. You can click the enterprise project name to view and edit it on the console of the Enterprise Project service.
Time Zone	The cluster time zone affects the node OS, log files, and data warehouse. You can change the time zone for the node OS and log files, but not for the data warehouse databases. To change the time zone of the data warehouse databases, use the GUC parameter **timezone**. For details, see **Modifying GUC Parameters of the DWS Cluster**.

Table 10-2 Connection

Parameter	Description
Private Network Domain Name	Domain name for accessing the cluster database through the internal network. The domain name corresponds to all CN IP addresses. The private network domain address is automatically generated when a cluster is created. **NOTE** • If the cluster name does not comply with the domain name standards, the prefix of the default access domain name will be adjusted accordingly. • Load balancing is not supported. You can click **Modify** to change the private network domain name. The access domain name contains 4 to 63 characters, which consists of letters, digits, and hyphens (-), and must start with a letter.

Parameter	Description
Private Network IP Address	IP address for accessing the database in the cluster over the private network. **NOTE** • A private IP address is automatically generated when you create a cluster. The IP address is fixed. • The number of private IP addresses equals the number of CNs. You can log in to any node to connect to the cluster. • If you access a fixed IP address over the internal network, all the resource pools will run on a single CN. • If IPv6 is enabled for a cluster, both IPv4 and IPv6 private addresses will be displayed. You can use either of them as needed.
Public Network Domain Name	Name of the domain for accessing the database in the cluster over the public network. **NOTE** Load balancing is not supported.
Public Network IP Address	IP address for accessing the database in the cluster over the public network. **NOTE** • If no EIP is assigned during cluster creation and **Public Network IP Address** is empty, click **Edit** to bind an EIP to the cluster. • If an EIP is bound during cluster creation, click **Edit** to unbind the EIP.
Initial Administrator	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 number for accessing the cluster database through the public network or private network. The port number is specified when the cluster is created.
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.
ELB Address	To achieve high availability and avoid single-CN failures, a new cluster needs to be bound to ELB. You are advised to use the ELB address to connect to the cluster. **NOTE** If the cluster is an IPv4 cluster, only IPv4 ELB can be manually bound. If the cluster is an IPv6 dual-stack cluster, only IPv6 dual-stack ELB can be manually bound.

Table 10-3 Network

Parameter	Description
Region	Current working zone of the cluster.

Parameter	Description
AZ	AZ selected during cluster creation
VPC	VPC selected during cluster creation. A VPC is a secure, isolated, and logical network environment. After a DWS cluster is created, its VPC cannot be changed. However, you can edit and modify the current VPC. You can click the VPC name to go to the VPC details page to configure it. For details about VPC operations, see "VPC and Subnet" in the *Virtual Private Cloud User Guide*.
Subnet	Subnet selected during cluster creation. A subnet provides dedicated network resources that are isolated from other networks, improving network security. After a DWS cluster is created, its subnet cannot be changed. However, you can edit and modify the current subnet. You can click the subnet name to go to the subnet details page to configure it. For details about subnet operations, see "VPC and Subnet" > "Modifying a Subnet" in the *Virtual Private Cloud User Guide*.
Security Group	Security group selected during cluster creation. After a DWS cluster is created, you can change the security group. You can also add, delete, or modify security group rules in the current security group. Changing the security group of a cluster may cause brief service disruption. Exercise caution when performing this operation. For better network performance, do not select more than five security groups. ● To change the security group, click **Modify** on the right of the security group name, select the security group name to be changed, and click **OK**. ● Modifying an existing security group rule: Click the security group name to go to the security group details page. For details about security group operations, see "Security" > "Security Group" in the *Virtual Private Cloud User Guide*.

Table 10-4 Storage/Backup capacity

Parameter	Description
Storage	The storage class **Ultra-high I/O** and the storage space usage are displayed. **NOTE** ● The used storage capacity does not include data on OBS foreign tables. It includes only DWS data, including files, logs, snapshots, and indexes. ● The available storage space is half of the actual disk capacity.

Parameter	Description
Backup	The space in use, free space, and charged space of the cluster are displayed.
Used Cold Partition Data Capacity	OBS capacity used by cold data. **NOTE** OBS capacity usage. It is synchronized every four hours.

Table 10-5 Data encryption information

Parameter	Description
Key Name	Indicates the database encryption key of the cluster when **Encrypt DataStore** is enabled.
Last Key Rotation Time	Indicates the time when the last encryption key is rotated when **Encrypt DataStore** is enabled.

Changing a Cluster Name

You can change the name of a created DWS cluster.

After the cluster name is changed, the names of all nodes in the current cluster are changed accordingly.

NOTE

- If the cluster name cannot be changed on the console, contact technical support to upgrade the console.
- If the cluster name fails to be changed, the cluster functions are not affected. You can contact technical support to rectify the fault if needed.

Constraints

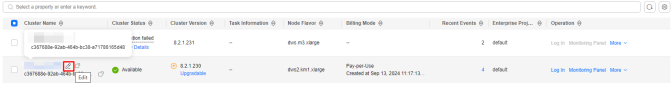
If the cluster is in the **Unavailable** status or is performing other tasks, the cluster name cannot be changed. You can change the cluster name only after the cluster status changes to **Available** or the running tasks are complete.

Method 1:

Step 1 Log in to the DWS console.

Step 2 In the cluster list, click the modification icon next to a cluster name to modify the cluster.

Figure 10-1 Changing the name of a cluster in the cluster list



Step 3 In the displayed dialog box, enter a new cluster name.

Step 4 Confirm the information and click **OK**.

----End

Method 2:

Step 1 Log in to the DWS console.

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

Step 3 On the displayed **Cluster Details** page, click the modification icon next to the cluster name in the **Basic Information** area.

Step 4 After confirming that the information is correct, click **OK** to deliver the cluster modification task. After the task is complete, the cluster name is changed.

----End

10.2 Checking the DWS Cluster Status

On the DWS console, choose **Dedicated Clusters** > **Clusters**, and you can view the general information about a cluster in the displayed cluster list, such as the cluster status, task information, recent events, and node specifications.

Querying General Information of a Cluster

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

In the upper part of the cluster list, click the search box and search for the required cluster based on the filter criteria (cluster name, cluster status, cluster version, task information, node specifications, recent events, and enterprise project). Click  to refresh the cluster list. You can also click **Search by Tag** to search for clusters based on cluster tags. 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 10-6** describes the cluster list parameters.

Table 10-6 Cluster list parameters

Parameter	Description
Cluster Name	Cluster name specified when a cluster is created. **NOTE** If the cluster name cannot be changed on the console, contact technical support.
Cluster Status	Cluster running status. For details, see **Cluster Status**.

Parameter	Description
Cluster Version	Version of the current cluster. For details, see **Table 10-8**.
Task Information	Cluster task status. For details, see **Cluster Task Information**.
Node Flavor	Node flavors of clusters.
Recent Events	Number of recent events in a cluster. You can click the number to view event details.
Enterprise Project	Enterprise project to which a cluster belongs.
Operation	• **More** – **Convert to Encryption Cluster**: For details, see **Converting an Ordinary Cluster to an Encrypted Cluster**. – **View Metric**: For details, see **Viewing DWS Cluster Monitoring Information on Cloud Eye**. – **Restart**: Click **Restart** to restart a cluster. For details, see **Restarting a cluster**. – **Start**: Click **Start** to start a stopped cluster. For details, see **Starting a Cluster**. – **Stop**: Click **Stop** to stop a running cluster. For details, see **Stopping a Cluster**. – **Scale Node**: For details, see **Scaling DWS Cluster Nodes**. – **Change Specifications**: For details, see **Changing DWS Cluster Specifications**. – **Reset Password**: For details, see **Resetting the Password the DWS Database Administrator**. – **Create Snapshot**: For details, see **Backing Up and Restoring a DWS Cluster**. – **Delete**: Click **Delete** to delete a cluster. – **Manage CN**: For details, see **Adding or Deleting a CN in a DWS Cluster**. **NOTE** The positions of the function keys in the **Operation** column are dynamic. To ensure that there are always two function keys visible before **More**, any function keys that typically appear only when you hover over **More** will be moved to a position directly before **More**. This adjustment occurs if there are some functions whose keys are supposed to be placed before **More** but are not supported for the current site.

Cluster Status

Table 10-7 Cluster status description

Status	Description
Available	Indicates that the cluster runs properly.
Read-only	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, remove the status by referring to **Removing the Read-only Status**. If the status cannot be removed, contact technical support engineers. After the read-only status is canceled for the cluster, you are advised to perform the following operations: • Use the SQL client tool to connect to the database as the administrator and run the following command to periodically clear and reclaim the storage space: `VACUUM FULL;` After you delete data stored in DWS data warehouses, dirty data may be generated possibly because the disk space is not released. This results in disk space waste. It is recommended that the storage space be cleared periodically. • You are advised to check the disk capacity and analyze whether the existing cluster specifications meet service requirements. If not, expand the cluster capacity. For details, see **Scaling Out a Cluster**.
Unbalanced	If the role of a GTM or DN in the cluster is different from the initial role, the cluster is in the **Unbalanced** state. In the **Unbalanced** state, the number of primary instances on some nodes increases. As a result, the load pressure is high. In this case, the cluster is normal, but the overall performance is not as good as that in a balanced state. You are advised to switch a cluster to the **Available** state during off-peak hours. For details, see **Performing a Primary/Standby Switchback**.
Redistributing	A cluster goes into this state when it detects that the service data on the original nodes is significantly larger than that on the new node after a new node is added to the cluster. In this case, the system automatically redistributes data on all nodes. The cluster can still work in this state.
Redistribution failed	A cluster goes into this state when data redistribution fails, but no data loss occurs. The cluster can still work in this state. You are advised to contact technical support.
Degraded	A cluster goes into this state when some nodes in the cluster are faulty, but the whole cluster runs properly. You are advised to contact technical support.

Status	Description
Unavailable	A cluster goes into this state when it cannot provide database services. You are advised to contact technical support.
Creating	A cluster goes into this state when it is being created.
Creation failed	A cluster goes into this state when it fails to be created.
Creating, restoring	A cluster goes into this state when it is being restored from a snapshot.
Deleting	A cluster goes into this state when it is being deleted.
To be restarted	This status indicates that GUC parameters have been modified in the cluster and the modification can take effect only after the cluster is restarted. Before the cluster is restarted, some O&M operations cannot be performed. After you manually restart the cluster, the GUC parameter takes effect and the cluster status changes to **Available**.
Stopped	Indicates that the cluster is stopped.

Cluster Version

Table 10-8 Cluster version

Version	Description
x.x.x	Current cluster version. To improve system stability, click **Upgradable** to go to the upgrade management page. For details, see **Upgrading a DWS Cluster**.

Cluster Task Information

Table 10-9 Task information description

Status	Description
Creating snapshot	Indicates that a snapshot is being created in the cluster.
Snapshot creation failed	Indicates that a snapshot fails to be created.
Observing	Indicates that the cluster is to be submitted after the automatic upgrade.

Status	Description
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.
Converting to encryption cluster	Indicates that the cluster is being converted to an encrypted cluster.
Encryption cluster conversion failed	Indicates that the cluster fails to be encrypted.
Scaling out	Indicates that a cluster is being scaled out.
Scale-out failed	Indicates that a cluster fails to be scaled out.
Expanding disk capacity	Indicates that disk capacity is being expanded.
Disk expansion failed	Indicates that disk capacity fails to be expanded.
Associating ELB	Indicates that ELB is being associated.
Failed to associate ELB	Indicates that ELB fails to be associated.
Disassociating ELB	Indicates that ELB is being disassociated.
Failed to disassociate ELB	Indicates that ELB fails to be disassociated.
Switching back	The primary/standby relationship of a cluster is being restored.

Status	Description
Switchback failed	The primary/standby relationship of a cluster fails to be restored. Possible causes are as follows. • Redo operations are being performed on DNs. Wait until the operations are completed and try again. • Failed to query DN redo information. Check tenant logs to identify the failure cause. • Primary/standby catchup is in progress. Wait until it is completed and try again. • Failed to query primary/standby catchup information. Check tenant logs to identify the failure cause. • Primary/standby catchup failed. Contact technical support or try again later. Check tenant logs to identify the failure cause. • The cluster is abnormal.
Changing node flavor	The cluster is being scaled.
Node flavor change failed	All specifications change failed
Maintaining	A maintenance change operation, such as cluster upgrade or plug-in upgrade, is being performed on the cluster.
Maintain_failure	A cluster fails to be restarted.
Stopping	Indicates that the cluster is being stopped.
Starting	Indicates that the cluster is being started.
Inspecting	Indicates that the cluster is being inspected before the change.
Inspection failed	Indicates that the cluster inspection fails.

10.3 Viewing the DWS Cluster Topology

Overview

A topology shows all the nodes in a cluster. You can check the node statuses, processes, and IP addresses.

NOTE

- You can check the topology structure and node processes.
- Only cluster versions 8.0.0 and later can display the topology structure. Only cluster versions 8.2.0 and later can display node processes.

Viewing the Cluster Topology

Step 1 Log in to the DWS console.

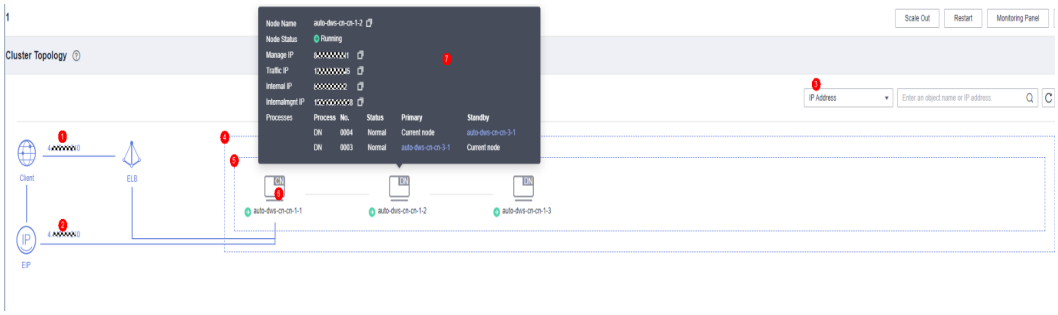
Step 2 In the cluster list, click the name of a cluster.

Step 3 On the **Cluster Details** page, click the **Cluster Topology** tab.

Step 4 In the pper right corner of the page, you can select **IP Address** or **Node Name**. After entering the IP address or node name in the search box, you can view the location of the IP address or node name in the cluster topology.

----End

Topology Overview



This figure shows a topology. The elements marked in the figure are as follows:

1. Public IP address of the ELB bound to the cluster. If no public IP addresses are bound to the ELB, the service address is displayed.
2. EIP bound to the cluster.
3. Search category. You can perform exact search by IP address or node name.
4. Rings in the cluster.
5. This box represents a ring, with each line and icon indicating a node within the ring. If there are at least three cluster rings created, you can view the distributed deployment of CNs.
6. A node. The type of the node is displayed in the upper right corner of the icon. Currently, the type can only be CN or DN. If there is a CN process on the node, **CN** is displayed. If there are no CN processes on the node, **DN** is displayed.
7. Node details, including the node name, status, IP addresses, and task process. Node details are displayed when you hover your cursor over a node icon.

Terms in the Topology View

Table 10-10 Cluster structure description

Name	Description	Usage
ELB	Elastic Load Balance (ELB) automatically distributes incoming traffic across multiple backend servers based on listening rules you configure.	If the private IP address or EIP of a CN is used to connect to a cluster, the failure of this CN will lead to cluster connection failure. If a private domain name is used for connection, the DNS service randomly selects a private IP address or EIP for each client. This cannot balance loads or avoid single-CN failures. ELB is used to solve these problems. For details, see **Binding and Unbinding Load Balancers for a DWS Cluster**.
EIP	The Elastic IP (EIP) service provides static public IP addresses and scalable bandwidths that enable your cloud resources to communicate with the Internet.	EIPs can be bound to or unbound from ECSs, virtual IP addresses, load balancers, and NAT gateways.
Ring	A security ring is used for isolating faulty servers. A fault in a ring does not affect servers outside the ring.	Data on a DN has multiple copies in a ring, and will not be lost even if the DN server is faulty. For example, if Server1 in a ring is faulty, the standby DN1 on Server2, the standby DN2 on Server3, and the standby DN3 on Server3 are still running. The loads of servers in a ring are still balanced. A cluster can run properly as long as the number of faulty servers does not exceed the number of rings. **NOTE** The ring is the minimum unit for a scale-out. When you scale out a cluster, the added nodes must be a multiple of the ring quantity.

Table 10-11 Node IP addresses

Name	Description	Usage
Manage IP	IP address used by a data warehouse node to communicate with the management plane	It is used by the management plane to deliver commands, and used by the node to report node status and monitoring information.
Traffic IP	IP address of a data warehouse node for external access.	This IP address can be bound to an EIP or ELB, or directly connect to a VPC.
Internal IP	IP address used for communication inside a data warehouse cluster.	-
Internalmgnt IP	IP address used by nodes to send internal management commands in a data warehouse cluster.	-

Table 10-12 Node processes

Name	Description	Usage
CMS	A Cluster Manager (CM) manages and monitors the running status of functional units and physical resources in the distributed system, ensuring system stability. CM Server (CMS) is a module of CM.	A CM consists of CM Agent, OM Monitor, and CM Server. • CM Agent monitors the running status of primary and standby GTMs, CNs, and primary and standby DNs on the host, and reports the status to CM Server. In addition, it executes the arbitration instruction delivered by CM Server. A CM Agent process runs on each server. • OM Monitor monitors scheduled tasks of CM Agent and restarts CM Agent when CM Agent stops. If CM Agent cannot be restarted, the server will be unavailable. In this case, you need to manually rectify this fault. **NOTE** A CM Agent restart fails probably because of lack of system resources, which rarely happens. • CM Server checks whether the current system is normal according to the instance status reported by CM Agent. In the case of exceptions, CM Server delivers recovery commands to CM Agent. CM Servers are deployed in primary/standby pairs to ensure system high availability. CM Agent

Name	Description	Usage
		connects to the primary CM Server. If the primary CM Server is faulty, the standby CM Server is promoted to primary to prevent single-CM faults.
GTM	A Global Transaction Manager (GTM) generates and maintains the globally unique information, such as the transaction ID, transaction snapshot, and timestamp.	A cluster includes only one pair of GTMs: one primary and one standby GTM.
CN	A Coordinator (CN) receives access requests from applications, and returns execution results to the client; splits tasks and allocates task fragments to different DNs for parallel processing.	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. 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.

Name	Description	Usage
CCN	Central Coordinator (CCN)	DWS handles the global resource load in a cluster using the Central Coordinator (CCN) for adaptive dynamic load management. When the cluster is started for the first time, the CM selects the CN with the smallest ID as the CCN. If the CCN is faulty, CM replaces it with a new one.
DN	A Data Node (DN) stores data in row-store, column-store, or hybrid mode, executes data query tasks, and returns execution results to CNs.	There are multiple DNs in the cluster. Each DN stores part of data. If DNs are not deployed in primary/standby mode and a DN is faulty, data on the DN will be inaccessible.

10.4 Managing DWS Cluster Connections

10.4.1 Managing DWS Cluster 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 DWS, you can access clusters using the private network domain name or the public network domain name.

Private network domain name: Name of the domain for accessing the database in the cluster through the private network. The private network domain name is automatically generated when you create a cluster.

Public network domain name: Name of the domain for accessing the database in the cluster through the public network. If a cluster is not bound to an EIP, it cannot be accessed using the public network domain name. If you bind an EIP during cluster creation, the public network domain name is automatically generated.

> NOTE
>
> Neither public nor private domain names support load balancing. To use load balancing, see "Configuring JDBC to Connect to a Cluster (Load Balancing Mode)".

After a cluster is created, you can set private and public domain names for accessing the cluster as required. The operations are as follows:

- **Modifying a Private Network Domain Name**
- **Creating a Public Network Domain Name**
- **Modifying a Public Network Domain Name**
- **Releasing a Public Network Domain Name**

Modifying a Private Network Domain Name

The private network domain name is automatically generated during cluster creation. After the cluster is created, you can modify the default domain name based on site requirements.

To modify the private network domain name, perform the following steps:

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

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

Step 4 In the **Connection** area, click **Modify** next to the automatically generated **Private Network Domain Name**.

Step 5 In the **Modify Private Network Domain Name** dialog box, enter the target domain name and click **OK**.

The private network domain name contains 4 to 63 characters, which consists of letters, digits, and hyphens (-) and must start with a letter.

After the domain name is modified, click copy button next to the private network domain name to copy it.

----End

Creating a Public Network Domain Name

A cluster is not bound to an EIP by default during cluster creation. That is, cluster access using the public network is disabled. After a cluster is created, if you want to access it over the public network, bind an EIP to the cluster and create a public network domain name.

> NOTE
>
> By default, only Huawei Cloud accounts or users with **Security Administrator** permissions can query and create agencies. By default, the IAM users in those accounts cannot query or create agencies. When the users use the EIP, the system makes the binding function unavailable. Contact a user with the **DWS Administrator** permissions to authorize the agency on the current page.

To create a public network domain name, perform the following steps:

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

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

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

Step 5 In the **Edit Elastic IP** dialog box, select an EIP from the drop-down list to bind it to a specified CN.

If no available EIPs are displayed, click **View EIP** to go to the **Elastic IP** page and create an EIP that satisfies your needs. After the new EIP is created, click the refresh icon next to the drop-down list. The newly created EIP will be displayed in the **EIP** drop-down list.

After the EIP is bound successfully, the specific public network IP address is displayed in the **Connection** area.

Step 6 In the **Connection** area, click **Create** next to **Public Network Domain Name** to create a public network domain name for the cluster.

Step 7 In the **Apply for Public Network Domain Name** dialog box, enter the target domain name and click **OK**.

The public network domain name contains 4 to 63 characters, which consists of letters, digits, and hyphens (-) and must start with a letter.

The specific public network domain name is displayed in the **Connection** area after being created. Click copy button  to copy the public network domain name.

----End

Modifying a Public Network Domain Name

If you bind an EIP during cluster creation, the public network domain name is automatically generated. After a cluster is created, you can modify the public network domain name as required.

To modify the public network domain name, perform the following steps:

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

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

Step 4 Click **Modify** next to the **Public Network Domain Name** in the **Connection** area.

Step 5 In the **Modify Public Network Domain Name** dialog box, enter the target domain name and click **OK**.

----End

Releasing a Public Network Domain Name

After a cluster is created, you can release unnecessary public network domain names.

To do so, perform the following steps:

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

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

Step 4 Click **Release** next to the **Public Network Domain Name** in the **Connection** area.

Step 5 In the **Release Domain Name** dialog box, click **Yes**.

----End

10.4.2 Binding and Unbinding Load Balancers for a DWS Cluster

Overview

If the private IP address or EIP of a CN is used to connect to a cluster, the failure of this CN will lead to cluster connection failure. If a private domain name is used for connection, the DNS service randomly selects a private IP address or EIP for each client. This cannot balance loads or avoid single-CN failures. ELB is used to solve these problems.

An ELB distributes access traffic to multiple ECSs for traffic control based on forwarding policies. It improves the fault tolerance capability of application programs. For details, see the *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.

NOTE

- This feature is supported only in cluster version 8.1.1.200 or later.
- To ensure load balancing and high availability and prevent service interruption, ensure created clusters are to an ELB.
- ELB does not support cross-database access.

Constraints and Limitations

- To bind a load balancer to a DWS cluster, ensure that the load balancer is in the same region, VPC, and enterprise project as the cluster.
- Only dedicated load balancers can be bound to DWS.

NOTICE

Load balancing is not supported in regions where the dedicated load balancer is not available. You can check whether dedicated load balancers are supported on the ELB console.

- The ELB to be associated must use TCP and has a private IP address.
- When creating an ELB instance, determine its specifications based on your service access traffic. You are advised to select the maximum specifications. On the DWS console, you can bind to a load balancer but cannot change its specifications.
- You only need to create a load balancer if you want to use ELB. DWS automatically creates the ELB listeners and backend server groups required.
- 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 bind a load balancer to a cluster, 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 unbind a load balancer from a cluster, related cluster information is cleared on DWS but the load balancer is not deleted.
- If you need to access the ELB cluster using a public IP address or domain name, bind an EIP or domain name on the ELB management console.
- If the cluster is an IPv4 cluster, only IPv4 ELBs can be bound. If the cluster is an IPv6 dual-stack cluster, only IPv6 dual-stack ELBs can be bound.

Associating ELB

Step 1 Log in to the DWS console.

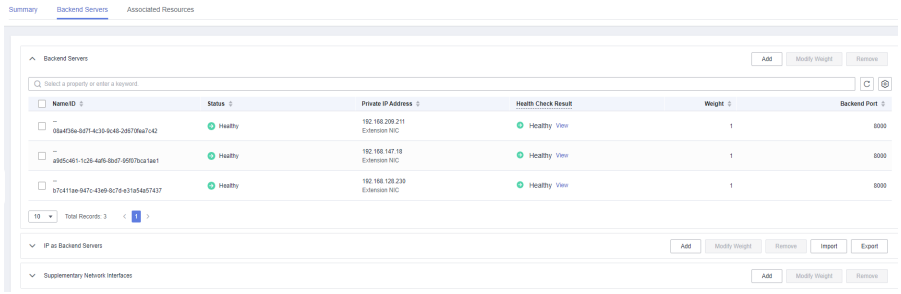
Step 2 Choose **Dedicated Clusters** > **Clusters**. All clusters are displayed by default.

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

Step 4 On the **Basic Information** page that is displayed, click **Associate ELB** and select the ELB name. If no load balancer exists, create one on the ELB management console. Then refresh the 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.

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



Step 7 In the **Basic Information** area of the **Cluster Information** page, check the **ELB Address**, which is used for connecting to the cluster.

----End

Disassociating ELB

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters**. All clusters are displayed by default.

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

Step 4 On the **Basic Information** page that is displayed, click **Disassociate ELB**.

Step 5 After the request is delivered, go back to the **Clusters** page. Task information **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

10.4.3 Adding or Deleting a CN in a DWS Cluster

Purpose

After a cluster is created, the number of required CNs varies with service requirements. The CN management function enables you to adjust the number of CNs in the cluster. The operations are as follows:

- **Adding CNs**
- **Deleting CNs**

NOTE

- This feature is supported only in cluster version 8.1.1 or later.
- Only cluster versions 8.1.3.300 and later (excluding 8.2.0) support online CN addition, deletion, and concurrent addition of multiple CNs.

Constraints and Limitations

- During resource provisioning, the default number of CNs is 3. You can adjust the number of CNs based on the number of provisioned nodes. The number of CNs ranges from 2 to 20.

- Do not perform other O&M operations when adding or deleting a CN.
- Adding CNs consumes lots of CPU and I/O resources, which will greatly impact job performance. You are advised to perform this operation during off-peak hours or after services are stopped.
- If a fault occurs when you add a CN node and the rollback fails, try adding the CN again. The deletion of a CN node cannot be rolled back.
- For a CN that fails to be added, you can only retry the addition. For a CN that fails to be deleted, you can only retry the deletion. Other O&M operations are not allowed for such CNs.
- If DDL operations, such as schema and function creation, are performed during CN deletion, an error may be reported because the deleted CN cannot be found. In this case, try again.
- If one of your CNs is abnormal, you can only delete this abnormal CN. If two or more CNs are abnormal, you can delete CNs only after the CNs are recovered from faults.

Adding CNs

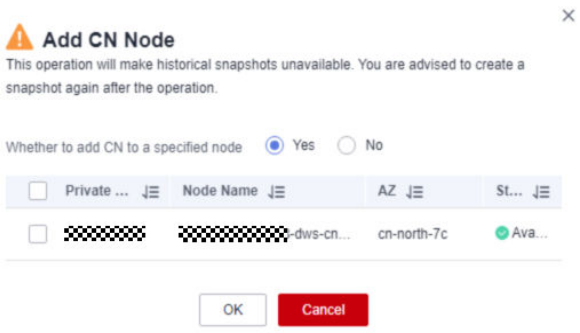
Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** and locate the cluster to which you want to add a CN.

Step 3 In the **Operation** column of the specified cluster, choose **More** > **Manage CN** > **Add CN Node**.

Step 4 On the displayed page, determine whether to add a CN to a specified node.

- If you select **No**, you can set the **CN quantity after adjustment**.
- If you select **Yes**, specify the node.



NOTICE

- Before adding a CN, ensure that the cluster is in the **Available** or **Unbalanced** state.
- The number of CNs cannot exceed the total number of nodes after adjustment.
- You cannot add more CNs than the number of CNs that have already been deployed.

Step 5 Click **OK**.

----End

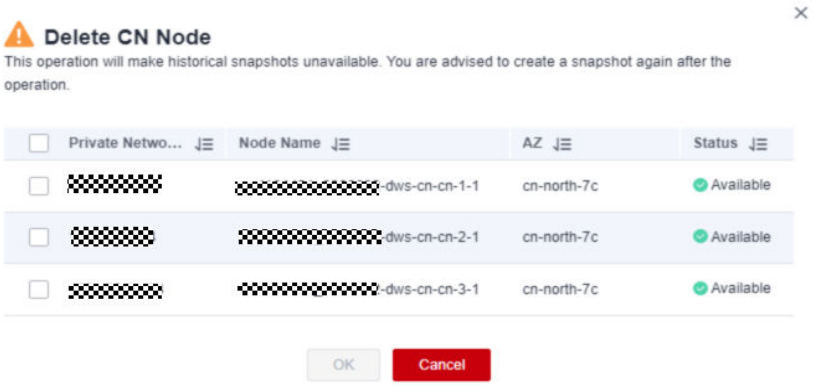
Deleting CNs

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** and locate the cluster to which you want to delete a CN.

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 node to be deleted. After confirming that the information is correct, enter **DELETE** or click **One-Click Input** and click **OK** to delete the CN node.



NOTICE

- At least two CN must be retained.
- When deleting a CN from a multi-AZ cluster, reserve a normal CN node in each AZ. Faulty CN nodes (if any) can be deleted.
- 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

10.4.4 Managing DWS 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 10-13 Number of supported connections

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

Viewing the Maximum Number of Connections

Method 1: Post cluster creation, select the cluster name on the DWS console to access the **Parameter Modification** page and view the value of **max_connections**.

Method 2: Use an SQL client tool to establish a database connection within the cluster and execute an SQL query to obtain the value of **max_connections**.

```
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)
```

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 10-14**.

NOTICE

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

Table 10-14 Viewing the number of connections

Description	Command

View the maximum number of sessions connected to a specific user.	Run the following command to view the maximum number of sessions connected to user **dbadmin**.
	`SELECT ROLNAME,ROLCONNLIMIT FROM PG_ROLES WHERE ROLNAME='dbadmin';`
	Information similar to the following is displayed. **-1** indicates that the number of sessions connected to user **dbadmin** is not limited.
	`rolname | rolconnlimit`
	`----------+--------------`
	`dwsadmin |  -1`
	`(1 row)`

View the number of session connections that have been used by a user.	Run the following command to view the number of session connections that have been used by **dbadmin**.
	`SELECT COUNT(*) FROM V$SESSION WHERE USERNAME='dbadmin';`
	Information similar to the following is displayed. **1** indicates the number of session connections used by user **dbadmin**.
	`count`
	`-------`
	`1`
	`(1 row)`

View the maximum number of sessions connected to a specific database.	View the upper limit of connections used by **gaussdb**.
	`SELECT DATNAME,DATCONNLIMIT FROM PG_DATABASE WHERE DATNAME='gaussdb';`
	Information similar to the following is displayed. **-1** indicates that the number of sessions connected to database **gaussdb** is not limited.
	`datname | datconnlimit`
	`----------+--------------`
	`gaussdb |  -1`
	`(1 row)`

Description	Command

View the number of session connections that have been used by a database.	View the number of session connections that have been used by **gaussdb**.
	`SELECT COUNT(*) FROM PG_STAT_ACTIVITY WHERE DATNAME='gaussdb';`
	Information similar to the following is displayed. **1** indicates the number of session connections used by database **gaussdb**.
	`count`
	`-------`
	`1`
	`(1 row)`

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

----End

10.5 DWS Resource Load Management

10.5.1 Overview

The system resources (CPU, memory, I/O, and storage resources) of a database are limited. When multiple types of services (such as data loading, batch analysis, and real-time query) are running at the same time, they may compete for resources and hinder operations. As a result, the throughput decreases and the overall query performance deteriorates. To avoid this problem, resources must be properly allocated.

DWS provides the resource management function. You can put resources into different resource pools, which are isolated from each other. Then, you can associate database users with these resource pools. When a user starts a SQL query, the query will be transferred to the resource pool associated with the user. You can specify the number of queries that can be concurrently executed in a resource pool, the upper limit of memory used for a single query, and the memory and CPU resources that can be used by a resource pool. In this way, you can limit and isolate the resources occupied by different workloads, properly utilizing resources to process hybrid database loads and achieve high query performance. After a cluster is converted into a logical cluster, you can create, modify, or delete a resource pool in the logical cluster.

NOTICE

- This feature is supported only by clusters of version 8.0.0 or later.
- Resources cannot be managed during offline scale-out. If a resource management plan is enabled, stop it before performing offline scale-out.

Enabling or Disabling Resource Management

You can enable or disable resource management, and configure the maximum global concurrency. **Max. Concurrent Queries** refers to the maximum concurrent queries on a single CN. If you disable **Resource Management**, all resource management functions will be unavailable.

Resource Management Functions

The resource management functions of DWS can be classified into the following types based on managed resources:

- Computing resource management. It is implemented using resource pools. Computing resources are isolated and controlled to prevent cluster-level issues caused by abnormal SQL queries. Computing resource management includes concurrency management, memory management, CPU management, and exception rules. For details, see **Resource Pool**.
- Storage space management: Storage is managed at user and schema level to prevent disk exhaustion, which makes the database read only. For details, see **Workspace Management**.
- Resource management plan: Resources are managed automatically based on a preconfigured plan, which can flexibly cope with complex scenarios. For details, see **Importing or Exporting a Resource Management Plan**.

The resource management functions of DWS can be classified into the following types based on when they are implemented:

- Management before a query

  The service checks whether there are sufficient resources for a query. If there are, the query can be executed. If there are not, the query waits in a queue, and can be executed only after resources are released by other queries. Concurrency and memory are managed in this phase.
- Management during a query

  During query execution, resources used by the query are managed and controlled to prevent cluster exceptions caused by time-consuming SQL statements. Memory, CPU, storage space, and exception rules are managed in this phase.

Simple and Complex Queries

DWS supports fine-grained resource management. Before resource management is implemented, queries are classified into complex queries (with long execution time and high resource consumption) and simple queries (with short execution time and low resource consumption). Simple and complex queries also differ in their estimated memory usage.

- The estimated memory usage of a simple query is less than 32 MB.
- The estimated memory usage of a complex query is 32 MB or higher.

In a hybrid load database, complex queries often occupy a large number of resources for a long time. A simple query queued after a complex query is time consuming, because it has to wait for the complex query to complete and resources to be freed up. To improve execution efficiency and system throughput,

DWS provides the short query acceleration function, managing simple queries separately.

- If short query acceleration is enabled, simple queries and complex queries are managed separately. Simple queries do not need to compete with complex queries for resources.
- If short query acceleration is disabled, simple and complex queries are under the same resource management rules.

To prevent a large number of simple queries from consuming too many resources during acceleration, concurrency management is performed on the queries. Resource management is not performed, because it may affect query performance and system throughput.

NOTE

Queries are categorized based on estimated memory usage, but the estimation does not equal the actual usage, nor does it reflect the query duration or CPU usage. In resource pools that are insensitive to performance and only run specific services, you can disable short query acceleration to manage resources and handle exceptions for simple queries.

10.5.2 Resource Pool

10.5.2.1 Feature Description

DWS resource pools provide concurrency management, memory management, CPU management, and exception rules.

Concurrency Management

Concurrency represents the maximum number of concurrent queries in a resource pool. Concurrency management can limit the number of concurrent queries to reduce resource contention and improve resource utilization.

In the **Short Query Configuration** area, you can enable or disable the short query acceleration function. To change the number of simple statements (**-1** by default. **0** or **-1** indicates that the concurrent short queries are not controlled), you can enable short query acceleration.

The concurrency management rules are as follows:

- If short query acceleration is enabled, complex queries are under resource pool concurrency control, and simple queries are under short query concurrency control.
- If short query acceleration is disabled, complex and simple queries are both under resource pool concurrency control. Short query concurrency control is invalid.

Memory Management

Each resource pool occupies a certain percentage of memory.

Memory management aims to prevent out of memory (OOM) in a database, isolate the memory of different resource pools, and to control memory usage. Memory is managed from the following aspects:

- Global memory management

  To prevent OOM, set the global memory upper limit (**max_process_memory**) to a proper value. Global memory management before a query controls memory usage to prevent OOM management. Global memory management during a query prevents errors during query execution.

  – Management before a query

    The service checks the estimated memory usage of a query in the slow queue, and compares it with the actual usage. The estimation will be adjusted if it is smaller than the actual usage. Before a query is executed, the service checks whether the available memory is sufficient for the query. If yes, the query can be executed directly. If no, the query needs to be queued and executed after other queries release resources.

  – Management during a query

    During a query, the service checks whether the requested memory exceeds a certain limit. If yes, an error will be reported, and memory occupied by the query will be released.

- Resource pool memory management

  Resource pool memory management puts a limit on dedicated quotas. A workload queue can only use the memory allocated to it, and cannot use idle memory in other resource pools.

  The resource pool memory is allocated in percentage. The value range is 0 to 100. The value **0** indicates that the resource pool does not perform memory management. The value **100** indicates that the resource pool performs memory management and can use all the global memory.

  The sum of memory percentages allocated to all resource pools cannot exceed 100. Resource pool memory management is performed only before a query in the slow queue starts. It works in a way similar to the global memory management before a query. Before a query in the slow queue in a resource pool is executed, its memory usage is estimated. If the estimation is greater than the resource pool memory, the query needs to be queued and can be executed only after earlier queries in the pool are complete and resources released.

CPU Management

CPU share and CPU limit can be managed.

- CPU share: If the system is heavily loaded, CPU resources are allocated to resource pools based on the specific CPU shares. If the system not busy, this configuration does not take effect.
- CPU limit: It specifies the maximum number of CPU cores used by a resource pool. The resource usage of jobs in the resource pool cannot exceed this limit no matter whether the system is busy or not.

In the Resource Configuration area, you can modify the CPU time limit and CPU usage limit.

Choose either of the preceding management methods as needed. In CPU share management, CPUs can be shared and fully utilized, but resource pools are not isolated and may affect the query performance of each other. In CPU limit management, the CPUs of different resource pools are isolated, but this may result in the waste of idle resources.

NOTE

The CPU limit is supported only by clusters of version 8.1.3 or later.

Exception Rules

To avoid query blocking or performance deterioration, you can configure exception rules to let the service automatically identify and handle abnormal queries, preventing slow SQL statements from occupying too many resources for a long time.

In the **Associated Exception Rules** area, you can view the exception rules bound to the current resource pool, bind new exception rules, and unbind existing exception rules. For more information, see **Exception parameters**.

NOTE

- The cluster version 8.2.1 and later supports downgradation of exception rules. All exception rules support downgradation behaviors. After downgradation, only network resource preemption is downgraded to a low priority. Downgraded network queries are scheduled only when there is no normal queries.
- Only clusters of version 8.2.0 or later support association and unbinding of exception rules.

Table 10-15 Exception rule parameters

Parameter	Description	Value Range (0 Means No Limit)	Operation
Blocking Time	Job blocking time. It refers to the total time spent in global and local concurrent queuing. The unit is second. For example, if the blocking time is set to 300s, a job executed by a user in the resource pool will be terminated after being blocked for 300 seconds.	An integer in the range 1 to 2,147,483,647. The value **0** indicates no limit.	**Terminated**, **Downgraded**, or **Not limited**
Execution Time	Time that has been spent in executing the job, in seconds. For example, if **Time required for execution** is set to 100s, a job executed by a user in the resource pool will be terminated after being executed for more than 100 seconds.	An integer in the range 1 to 2,147,483,647. The value **0** indicates no limit.	**Terminated**, **Downgraded**, or **Not limited**

Parameter	Description	Value Range (0 Means No Limit)	Operation
Total CPU time on all DNs.	Total CPU time spent in executing a job on all DNs, in seconds.	An integer in the range 1 to 2,147,483,647. The value **0** indicates no limit.	**Terminated**, **Downgraded**, or **Not limited**
Interval for Checking CPU Skew Rate	Interval for checking the CPU skew, in seconds. This parameter must be set together with **Total CPU Time on All DNs**.	An integer in the range 1 to 2,147,483,647. The value **0** indicates no limit.	**Terminated**, **Downgraded**, or **Not limited**
Total CPU Time Skew Rate on All DNs	CPU time skew rate of a job executed on DNs. The value depends on the setting of **Interval for Checking CPU Skew Rate**.	An integer in the range 1 to 100. The value **0** indicates no limit.	**Terminated**, **Downgraded**, or **Not limited**
Data Spilled to Disk Per DN	Allowed maximum job data spilled to disks on a DN. The unit is MB. **NOTE** This rule is supported only by clusters of version 8.2.0 or later.	An integer in the range 1 to 2,147,483,647. The value **0** indicates no limit.	**Terminated**, **Downgraded**, or **Not limited**
Average CPU Usage Per DN	Average CPU usage of a job on each DN. If **Interval for Checking CPU Skew Rate** is configured, the interval takes effect for this parameter. If the interval is not configured, the check interval is 30 seconds by default. **NOTE** This rule is supported only by clusters of version 8.2.0 or later.	An integer in the range 1 to 100. The value **0** indicates no limit.	**Terminated**, **Downgraded**, or **Not limited**
Maximum Bandwidth on a Single DN	Maximum network bandwidth (MB) for a job on a single DN. **NOTE** This rule is supported only by clusters of version 8.2.1 or later.	An integer in the range 1 to 2,147,483,647. The value **0** indicates no limit.	**Terminated**, **Downgraded**, or **Not limited**

10.5.2.2 Creating a Resource Pool

Step 1 Log in to the DWS console.

Step 2 Choose **Clusters**. Click the name of a cluster.

Step 3 Choose **Resource Management Configurations**.

Step 4 Click **Add Resource Pool**.

NOTE

Up to 63 resource pools can be created.

Step 5 Configure the resource pool. For more information, see **Table 10-16**.

Table 10-16 Resource pool parameters

Parameter	Description	Default Value
Name	Resource pool name	-

Parameter	Description	Default Value
CPU Resource (%)	• CPU share: Percentage of CPU time that can be used by users associated with the current resource pool to execute jobs. The value is an integer ranging from 1 to 99. • CPU limit: Maximum percentage of CPU cores used by a database user in a resource pool. The value is an integer ranging from 0 to 100. **0** indicates no limit. **NOTE** • The sum of the parameter values of all the resource pools cannot exceed 99%. If there is only one resource pool, the CPU share parameter does not take effect. • The CPU share parameter takes effect only when CPU contention occurs. For example, resource pools A and B are bound to CPU 1. If A and B are both running, the parameter takes effect. If there is only A running, the parameter does not take effect. • The sum of the CPU limits of all the resource pools cannot exceed 100%. The default value is 0. • The CPU limit is supported only by clusters of version 8.1.3 or later. • You are not advised to allocate multiple dedicated resource pools when the number of CPU cores is small. The minimum value of the dedicated CPU quota is **1**. If the number of CPU cores is small and the resource pool created earlier has used up the remaining CPU cores, the resource pool created later will share the CPU cores of the previous resource pool. As a result, the CPU ratio may be inconsistent with the actual CPU ratio. Assume that the cluster has two CPU cores and three resource pools are set up with CPU allocation ratios of 15%, 25%, and 60%. The first core is assigned 1 unit (minimum value), the second core is also assigned 1 unit, and the third core is assigned 1 unit as well, with no additional CPU cores available, sharing the previous CPU.	-
Memory Resource (%)	Percentage of the memory that can be used by a resource pool. You can manage memory and query concurrency separately or jointly. Under joint management, jobs can be delivered only when both the memory and concurrency conditions are met.	0 (not limited)
Storage Resource (MB)	Size of the available space for permanent tables. This parameter indicates the total tablespace of all DNs in a resource pool. Available space of a single DN = Configured value/Number of DNs.	-1 (not limited)

Parameter	Description	Default Value
Complex Statement Concurrency	Maximum number of concurrent queries in a resource pool. You can manage memory and query concurrency separately or jointly. Under joint management, jobs can be delivered only when both the memory and concurrency conditions are met.	10
Network Bandwidth Weight	Weight for network scheduling. The value is an integer ranging from 1 to 2147483647. The default value is -1. **CAUTION** Only clusters of 8.2.1 and later versions support the network bandwidth weight feature.	-1 (not limited)

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

----End

10.5.2.3 Modifying a Resource Pool

You can modify the parameters of a resource pool on the resource management page.

Step 1 Log in to the DWS console.

Step 2 Choose **Clusters**. Click the name of a cluster.

Step 3 Choose **Resource Management Configurations**.

Step 4 In the **Resource Pools** drop-down list, click the name of a resource pool, including **Short Query Configuration**, **Resource Configuration**, **Associated Exception Rules**, and **Associated User**.

Step 5 Modify the short query configuration. Set the parameters as required and click **Save** on the right.

Parameter	Description	Value
Short Query Acceleration	Whether to enable short query acceleration. This function is enabled by default.	Enable
Simple Statement Concurrency	A short query is a job whose estimated memory used for execution is less than 32 MB. The default value **-1** indicates that the job is not controlled.	10

Step 6 Modify the resource configuration.

1. Click **Edit** on the right and modify the parameters according to **Table 10-17**.

Table 10-17 Resource pool parameters

Parameter	Description	Default Value
Name	Resource pool name	-
CPU Resource (%)	– CPU share: Percentage of CPU time that can be used by users associated with the current resource pool to execute jobs. The value is an integer ranging from 1 to 99. – CPU limit: Maximum percentage of CPU cores used by a database user in a resource pool. The value is an integer ranging from 0 to 100. **0** indicates no limit. **NOTE** – The sum of the parameter values of all the resource pools cannot exceed 99%. If there is only one resource pool, the CPU share parameter does not take effect. – The CPU share parameter takes effect only when CPU contention occurs. For example, resource pools A and B are bound to CPU 1. If A and B are both running, the parameter takes effect. If there is only A running, the parameter does not take effect. – The sum of the CPU limits of all the resource pools cannot exceed 100%. The default value is 0. – The CPU limit is supported only by clusters of version 8.1.3 or later. – You are not advised to allocate multiple dedicated resource pools when the number of CPU cores is small. The minimum value of the dedicated CPU quota is **1**. If the number of CPU cores is small and the resource pool created earlier has used up the remaining CPU cores, the resource pool created later will share the CPU cores of the previous resource pool. As a result, the CPU ratio may be inconsistent with the actual CPU ratio. Assume that the cluster has two CPU cores and three resource pools are set up with CPU allocation ratios of 15%, 25%, and 60%. The first core is assigned 1 unit (minimum value), the second core is also assigned 1 unit, and the third core is assigned 1 unit as well, with no additional CPU cores available, sharing the previous CPU.	-
Memory Resource (%)	Percentage of the memory that can be used by a resource pool. You can manage memory and query concurrency separately or jointly. Under joint management, jobs can be delivered only when both the memory and concurrency conditions are met.	0 (not limited)

Parameter	Description	Default Value
Storage Resource (MB)	Size of the available space for permanent tables. This parameter indicates the total tablespace of all DNs in a resource pool. Available space of a single DN = Configured value/Number of DNs.	-1 (not limited)
Complex Statement Concurrency	Maximum number of concurrent queries in a resource pool. You can manage memory and query concurrency separately or jointly. Under joint management, jobs can be delivered only when both the memory and concurrency conditions are met.	10
Network Bandwidth Weight	Weight for network scheduling. The value is an integer ranging from 1 to 2147483647. The default value is -1. **CAUTION** Only clusters of 8.2.1 and later versions support the network bandwidth weight feature.	-1 (not limited)

NOTE

The CPU limit is supported only by clusters of version 8.1.3 or later.

2. Click **OK**.

Step 7 Associate exception rules.

1. Click **Associated Exception Rules** on the left.
2. Select the exception rules to be associated from the current exception rule list. You can select multiple exception rules at a time.
3. Click **OK**.
4. To unbind an exception rule, click **Disassociate Rule**.

NOTE

- Only clusters of version 8.2.0 or later support association and unbinding of exception rules.
- The default exception rules take effect for users not associated with any resource pools, and for users whose resource pools do not have any exception rules configured. If a user-defined rule is associated with a resource pool, this rule prevails in the pool.
  - The default exception rules are supported only by clusters of version 8.2.0 or later. After a cluster of an earlier version is upgraded to version 8.2.0 or later, the default exception rules do not take effect. You can create exception rules as needed.
  - The cluster version 8.2.1 supports downgradation of exception rules. All exception rules support downgradation behaviors. After downgradation, only network resource preemption is downgraded to a low priority. Downgraded network queries are scheduled only when there is no normal queries.
  - A resource pool can be associated with up to 16 exception rules.
- A resource pool can be associated with multiple groups of exception rules, which work in an OR way. One group of exception rules works if all its conditions are met. For example, a resource pool is associated with two groups of rules. One group specifies **elapsedtime=2400**, and the other group specifies **elapsedtime=1200** and **memsize=2000**. If the execution time of a job reaches 1,200 seconds and the memory usage reaches 2,000 MB, or if the execution time reaches 2,400 seconds, the job will be terminated.

Step 8 Associate users.

1. Click **User Association** on the left.
2. In the current user list, select the users to be associated. You can select multiple users at a time.
3. Click **OK**.
4. To disassociate a user, click **Disassociate User**.

NOTE

- The resources used by a user to run jobs can be controlled only after the user is added to a resource pool.
- A database user can be added to only one resource pool. Users removed from a resource pool can be added to another pool.
- In the user binding list, the lock status can be unlocked, locked, or unknown. In versions before 8.5.0.100, only "Unknown" is shown for user lock status. A locked user cannot be chosen for association as the selection button is disabled. An unknown user can be selected, but successful binding depends on the user's actual lock status.
- Database administrators cannot be associated.
- If no resource pools are associated with a user, the user will be associated with **default_pool** by default, and its resource usage will be restricted by **default_pool**. The **default_pool** will be automatically created after resource management is enabled.

----End

10.5.2.4 Deleting a Resource Pool

Step 1 Log in to the DWS console.

Step 2 Choose **Clusters**. Click the name of a cluster.

Step 3 Choose **Resource Management Configurations**.

Step 4 In the **Resource Pools** area on the left, click the name of a resource pool.

Step 5 Click **Delete Resource Pool**.

NOTE

Deleting a resource pool that is associated with a database user is not allowed. To delete the resource pool, you need to first disassociate it from the database user.

----End

10.5.3 Resource Management Plan

10.5.3.1 Managing Resource Management Plans

Overview

The resource management plan is an advanced resource management feature provided by DWS. You can create a resource management plan, add multiple stages to the plan, and configure different queue resource ratios for the stages. After a plan is started, it automatically changes the resource configurations in different stages as scheduled. If you need to run services in different stages with different proportions of resources, you can create a resource management plan to automatically change resource configurations in different stages.

Creating a Resource Management Plan

Step 1 Log in to the DWS console.

Step 2 Choose **Clusters**. Click the name of a cluster.

Step 3 Go to the **Basic Information** page and click the **Resource Management** tab in the navigation pane on the left.

Step 4 Click to the **Resource Management Plans** tab and click **Add**.

Step 5 Enter a plan name and click **OK**.

NOTICE

- Before creating a resource management plan, you must design and create a resource pool. For details, see **Creating a Resource Pool**.
- You can create up to 10 resource management plans.

----End

Starting or Stopping a Resource Management Plan

Step 1 Log in to the DWS console.

Step 2 Choose **Clusters**. Click the name of a cluster.

Step 3 Go to the **Basic Information** page and click the **Resource Management** tab in the navigation pane on the left.

Step 4 Switch to the **Resource Management Plans** tab page and click **Start**/**Stop**.

Step 5 After confirming that the information is correct, click **OK** in the dialog box that is displayed to start or stop the plan.

NOTICE

- Only one plan can be started for each cluster.
- A plan must have at least two stages before it can be started.

----End

Viewing the Execution Logs of a Resource Management Plan

Step 1 Log in to the DWS console.

Step 2 Choose **Clusters**. Click the name of a cluster.

Step 3 Go to the **Basic Information** page and click the **Resource Management** tab in the navigation pane on the left.

Step 4 Click **Resource Management Plans**. In the plan execution logs area, click **View** to view the plan execution logs.

----End

Deleting a Resource Management Plan

Step 1 Log in to the DWS console.

Step 2 Choose **Clusters**. Click the name of a cluster.

Step 3 Go to the **Basic Information** page and click the **Resource Management** tab in the navigation pane on the left.

Step 4 Click **Resource Management Plans** and click **Delete** to delete the current resource management plan.

NOTICE

You cannot delete a running resource management plan.

----End

10.5.3.2 Managing Resource Management Plan Stages

Prerequisites

The following conditions must be met when you add or modify a resource management plan:

- The total CPU share of all resource pools does not exceed 99%.
- The total CPU limit of all resource pools does not exceed 100%.

  NOTE

  - The CPU usage limit can be configured only in 8.1.3 and later versions.

Adding a Resource Management Plan Stage

Step 1 Log in to the DWS console.

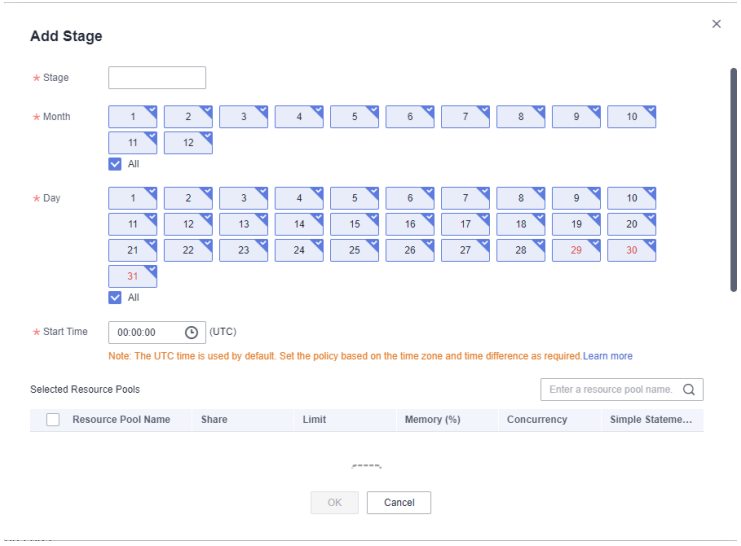
Step 2 Choose **Clusters**. Click the name of a cluster.

Step 3 Go to the **Basic Information** page and click the **Resource Management** tab in the navigation pane on the left.

Step 4 Go to the plan details page and click **Add** in the **Plan stage** area. On the **Add Stage** page, enter the stage name and configure the resource information. Confirm the configuration and click **OK**.

NOTICE

- Stages cannot be added to a running resource management plan.
- You can add a maximum of 48 stages for each plan.
- The switchover time of all phases in a plan cannot be the same.
- Configure the time, date, and month. Do not set an invalid date, for example, February 30.



----End

Modifying a Resource Management Plan Stage

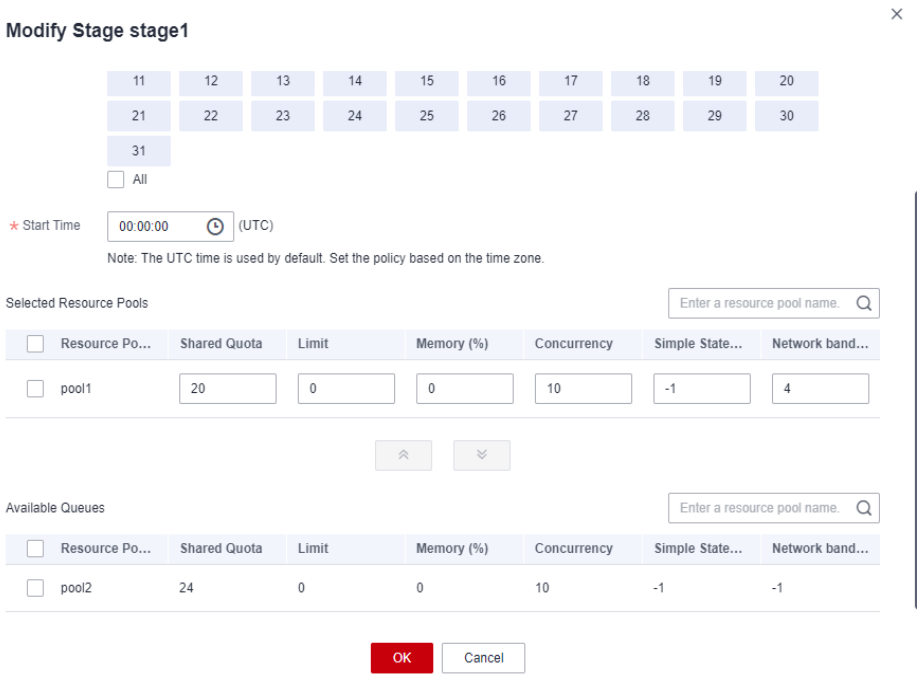
Step 1 Log in to the DWS console.

Step 2 Choose **Clusters**. Click the name of a cluster.

Step 3 Go to the **Basic Information** page and click the **Resource Management** tab in the navigation pane on the left.

Step 4 Switch to the **Resource Management Plans** tab page and click **Modify** in the **Operation** column of the plan stage.

Step 5 Modify parameters, such as the stage changing time and resource configurations.



NOTE

Only clusters of the version 8.2.1 and later support the network bandwidth weight.

----End

Manually Changing the Resource Management Plan Stage

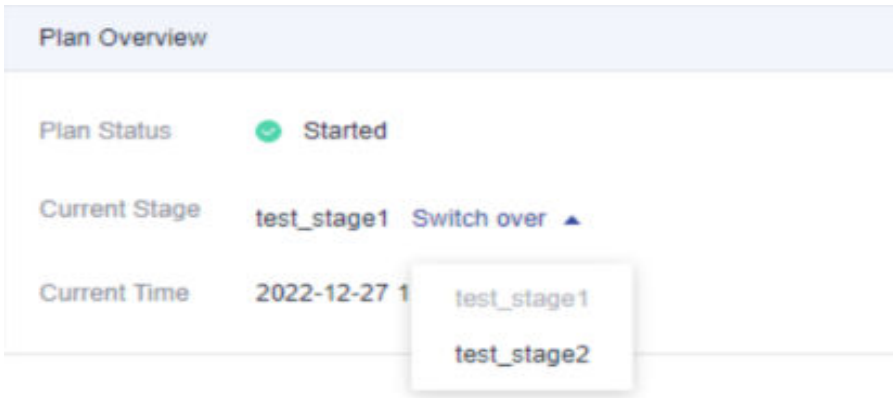
If a running plan needs to be switched to a stage in advance, you can manually do it.

Step 1 Log in to the DWS console.

Step 2 Choose **Clusters**. Click the name of a cluster.

Step 3 Go to the **Basic Information** page and click the **Resource Management** tab in the navigation pane on the left.

Step 4 Click **Resource Management Plans** and click the switch button in the plan overview area, and select the target stage.



----End

Importing/Exporting Resource Management Plan Stages

Step 1 Log in to the DWS console.

Step 2 Choose **Clusters**. Click the name of a cluster.

Step 3 Go to the **Basic Information** page and click the **Resource Management** tab in the navigation pane on the left.

Step 4 Switch to the **Resource Management Plans** tab page. In the plan stages area, click **Import**/**Export** to import or export a resource management plan stage.

NOTICE

- Configurations cannot be imported to a running resource management plan.
- Ensure there is a resource pool before import.

----End

Deleting a Resource Management Plan Stage

Step 1 Log in to the DWS console.

Step 2 Choose **Clusters**. Click the name of a cluster.

Step 3 Go to the **Basic Information** page and click the **Resource Management** tab in the navigation pane on the left.

Step 4 Switch to the **Resource Management Plans** tab page and click **Delete** in the **Operation** column of the plan stage.

----End

NOTICE

Stages in a running resource management plan cannot be deleted.

10.5.4 Workspace Management

Overview

Your cluster may run out of space if disk usage is not controlled, resulting in cluster exceptions and service interruption. Once disks are full, it takes long and huge efforts to recover workloads. Setting a database to read-only can reduce disk usage, but it also interrupts services. To solve this problem, DWS provides multi-dimensional storage management. You can limit the permanent space that can be occupied by a schema; and can limit the usage of permanent space, temporary space, and operator space for a user.

- Schema level: Schema space management allows you to query database and schema space information in a cluster and modify the total schema space.
- User level: User space management allows you to limit users' space usage, preventing task execution from being blocked due to insufficient storage space. When you create a user in DWS, you can specify the space available to the user. The following types of storage space can be managed:
  - Permanent space (**PREM SPACE**)

    Space occupied by permanent tables (non-temporary tables) created by users
  - Temporary space (**TEMP SPACE**)

    Space occupied by temporary tables created by users
  - Operator spill space (**SPILL SPACE**)

    During query execution, if the actual memory usage is greater than estimated, the query may be spilled to disks. The storage space occupied in this case is called operator spill space. You can control a user's operator spill space usage during query execution.

NOTE

- This feature is supported only in cluster version 8.1.1 or later.
- Currently, the DWS management plane only supports schema space management.

Procedure

Step 1 Log in to the DWS console.

Step 2 Choose **Clusters**. Click the name of a cluster.

Step 3 Go to the **Basic Information** page and click the **Resource Management** tab in the navigation pane on the left.

Step 4 On the **Schema Space Manage** page, select a database.

Step 5 In the row where the scheme to be edited resides, click **Edit** and modify the space limit.

Step 6 Click **OK**.

NOTE

- The space quota limits only common users but not database administrators. Therefore, when the used space is equal to the space limit, the actual used space may exceed the specified value.
- Quota for a single DN = Total quota/Number of DNs. Therefore, the configured value may be slightly different from the displayed value.

----End

10.6 Modifying GUC Parameters of the DWS Cluster

After a cluster is created, you can modify the cluster's database parameters as required. On the DWS console, you can configure common database parameters. For details, see **Modifying Parameters**. You can also view the parameter modification history. For details, see **Viewing Parameter Change History**. Click **Export** to export all parameter settings of the cluster. You can run SQL commands to view or set other database parameters. For details, see **"Configuring GUC Parameters"** in the *Data Warehouse Service (DWS) Developer Guide*.

Prerequisites

You can modify parameters only when no task is running in the cluster.

Modifying Parameters

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

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

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

To modify parameters based on filter criteria, click the drop-down list above the parameter list and search for the desired parameters. The filter criteria include **Common Configuration**, **Function Control Configuration**, **Cluster Memory Configuration**, **Cluster Disk Configuration**, **Cluster Network Configuration**, **SQL Tuning Configuration**, **SQL Compatibility Configuration**, and **All Configurations**. If you choose **Common Configuration**, you will see the first 20 frequently modified parameters in the region where the cluster is located. If there are no statistics available, you can use a customized configuration. Select **All Configurations** to view all parameters. Set the parameters based on **Function Control Parameters**.

Click the search box to search for a parameter based on the parameter name and whether to restart the cluster.

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

Step 6 You can determine whether you need to restart the cluster after parameter modification based on the **Restart Cluster** column.

NOTE

- If cluster restart is not required for a parameter, the parameter modification takes effect immediately.
- If cluster restart is required for parameter modifications to take effect, the new parameter values will be displayed on the page after the modification, but will not take effect until the cluster is restarted. Before a restart, the cluster status is **To be restarted**, and some O&M operations are disabled.

----End

Viewing Parameter Change History

Perform the following steps to view the parameter modification history and check whether the modifications have taken effect:

Procedure

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

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

Step 4 Click the **Modify Records** tab.

NOTE

- If a parameter can take effect immediately after modification, its status will change to **Synchronized** after you modify it.
- If a parameter can take effect only after a cluster restart, its status will change to **To be restarted** after you modify it. You can click the expansion icon on the left to view the parameters that have not taken effect. After the cluster is restarted, the status of the record will change to **Synchronized**.

Step 5 By default, only the change history within a specified period is displayed. To check the entire change history of a parameter, search for it in the search box in the upper right corner.

----End

Exporting the Parameter List

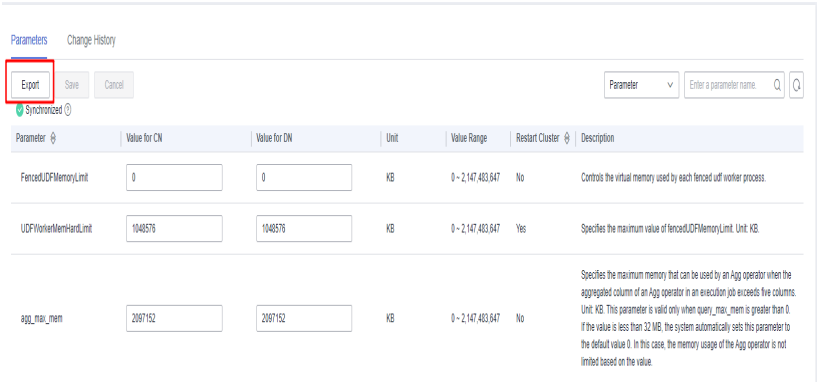
Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

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

Step 4 Click **Parameters** and click **Export**. You can export cluster configuration parameters.

Figure 10-2 Exporting parameter settings



----End

Function Control Parameters

Table 10-18 Function control parameters

Parameter	Description	Value Range
audit_enabled	Whether to enable or disable the audit process. After the audit process is enabled, it can read the auditing information written by the background process from the pipe and save it into audit files.	**on** or **off**
audit_space_limit	Maximum storage space occupied by audit files Unit: KB	1,024–1,073,741,824

Parameter	Description	Value Range
autoanalyze	Whether to allow automatic statistics collection for a table that has no statistics or a table whose amount of data modification reaches the threshold for triggering **ANALYZE** when a plan is generated. In this case, **AUTOANALYZE** cannot be triggered for foreign tables or temporary tables with the **ON COMMIT [DELETE ROWS|DROP]** option. To collect statistics, you need to manually perform the **ANALYZE** operation. If an exception occurs in the database during the execution of **AUTOANALYZE** 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 **ANALYZE** on the table to synchronize statistics.	**on** or **off**
autovacuum_max_workers	Maximum number of concurrent **autovacuum** threads. **0** indicates that **autovacuum** is disabled.	0–128
autovacuum_max_workers_hstore	Number of **automerge** threads in the **hstore** table. The value cannot be greater than the value of **autovacuum_max_workers**. To modify this parameter, add the value of **autovacuum_max_workers_hstore** to the original value of **autovacuum_max_workers**.	0–128
autovacuum_naptime	Interval between two autovacuum operations, in seconds.	1–2,147,483
autovacuum_vacuum_cost_delay	Value of the cost delay used in the **autovacuum** operation.	–1–100
behavior_compat_options	Configuration items for database compatibility. Multiple items are separated by commas (,). **strict_concat_functions** and **strict_text_concat_td** are mutually exclusive.	-
checkpoint_segments	Minimum number of WAL segment files retained in a period. The size of each log file is 16 MB.	1–2,147,483,646

Parameter	Description	Value Range
ddl_lock_timeout	Number of seconds a DDL command should wait for the locks to become available. If the time spent in waiting for a lock exceeds the specified time, an error is reported.	0–2,147,483,647
enable_resource_record	Whether to enable resource recording.	**on** or **off**
enable_resource_track	Whether to enable resource monitoring.	**on** or **off**
enable_track_record_subsql	Whether to enable the function of recording and archiving sub-statements. When this function is enabled, sub-statements in stored procedures and anonymous blocks are recorded and archived to the corresponding **INFO** table (**GS_WLM_SESSION_INFO**). This parameter is specific to a session and can be set and applied within the session connected to the CN. Only the statements executed within that session will be affected. It can also be configured on both the CN and DN and take effect globally.	**on** or **off**
enable_user_metric_persistent	Whether to dump the historical monitoring data of user resources. If this parameter is set to **on**, data in the **PG_TOTAL_USER_RESOURCE_INFO** view is periodically sampled and saved to the system catalog **PG_WLM_USER_RESOURCE_HISTORY**.	**on** or **off**
enable_view_update	Whether to enable the view update function.	**on** or **off**
extra_float_digits	Number of digits displayed for floating-point values, including float4, float8, and geometric data types. The parameter value is added to the standard number of digits (**FLT_DIG** or **DBL_DIG** as appropriate).	–15–3
failed_login_attempts	Number of consecutive incorrect password attempts after which the account is locked. **0** indicates that the number of incorrect password attempts is not limited.	0–1,000

Parameter	Description	Value Range
instr_unique_sql_count	Whether to collect unique SQL statements and how many statements can be collected.	0–2,147,483,647
job_queue_processes	Number of jobs that can be concurrently executed. This parameter is a postmaster parameter. You can set it using **gs_guc**, and you need to restart **gaussdb** to make the setting take effect.	0–1,000
lockwait_timeout	Maximum wait time for a single lock, in milliseconds. If the lock wait time exceeds the value, the system will report an error.	0–2,147,483,647
max_active_statements	Maximum number of concurrent jobs. This parameter applies to all the jobs on one CN. **–1** and **0** indicate that the number of concurrent jobs is not limited.	–1–2,147,483,647
max_files_per_node	Maximum number of files that can be opened by a single SQL statement on a single node.	–1–2,147,483,647
max_prepared_transactions	Maximum number of transactions that can stay in the **prepared** state simultaneously. If the value of this parameter is increased, DWS requires more System V shared memory than the default system setting.	0–536,870,911
max_process_memory_auto_adjust	Whether to enable automatic modification for the **max_process_memory** parameter.	**on** or **off**

Parameter	Description	Value Range
object_mtime_record_mode	Update action of the **mtime** column in the **PG_OBJECT** system catalog. • **default**: **ALTER**, **COMMENT**, **GRANT**/**REVOKE**, and **TRUNCATE** operations update the **mtime** column by default. • **none**: The **mtime** column is not updated. • **disable_acl**: **GRANT**/**REVOKE** operations do not update the **mtime** column. • **disable_truncate**: **TRUNCATE** operations do not update the **mtime** column. • **disable_partition**: The **mtime** field is not updated for **ALTER** operations in partitioned tables.	-
plog_merge_age	Output interval of performance log data.	0–2,147,483,647
random_function_version	Random function version selected by **ANALYZE** during data sampling.	0–1
resource_track_cost	Minimum execution cost for resource monitoring on statements. **–1** indicates that resource monitoring is disabled. If the value is greater than or equal to **0**, and the cost of executing statements exceeds the value and is greater than or equal to 10, resource monitoring is performed.	–1–2,147,483,647
resource_track_duration	Minimum time for archiving executed statements recorded during real-time monitoring, in seconds. **0** indicates that all the statements are archived. If the value is greater than **0**, historical information about statements whose execution time exceeds the specified value is archived.	0–2,147,483,647

Parameter	Description	Value Range
resource_track_level	Resource monitoring level of the current session. This parameter is valid only when **enable_resource_track** is set to **on**. • **none** indicates that resources are not monitored. • **query** enables the query-level resource monitoring. If this function is enabled, the plan information (similar to the output information of **explain**) of SQL statements will be recorded in top SQL statements. • **perf** enables the perf-level resource monitoring. If this function is enabled, the plan information (similar to the output information of **EXPLAIN ANALYZE**) that contains the actual execution time and the number of execution rows will be recorded in top SQL statements. • **operator** 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.	-
security_enable_options	Operations that can be unprohibited in security mode. • **on** indicates that **grant to public** can be used in security mode. • **on** indicates that **with grant option** can be used in security mode. • **foreign_table_options** allows users to perform operations on foreign tables in security mode without explicitly granting the **useft** permission to users.	-
session_timeout	Timeout interval of an idle session, in seconds. **0** indicates that the timeout limit is disabled.	0–86,400

Parameter	Description	Value Range
space_once_adjust_num	Threshold of the number of files processed each time in slow build and fine-grained calibration in space management and space statistics. **0** indicates that the slow build and fine-grained calibration functions are disabled. The number of files in the database can impact its resources. You are advised to set this parameter to a proper value. **NOTE** This parameter is supported only by clusters of version 8.1.3 or later.	0–2,147,483,647
statement_timeout	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.	0–2,147,483,647
timezone	Time zone for displaying and interpreting time stamps.	-
topsql_retention_time	Data storage retention period of the **gs_wlm_session_info** and **gs_wlm_operator_info** catalogs in historical top SQL statements, in days. Before setting this GUC parameter to enable the data storage function, clear data in the **gs_wlm_session_info** and **gs_wlm_operator_info** tables. ● If it is set to **0**, the data is stored permanently. ● If the value is greater than **0**, the data is stored for the specified number of days.	0–3,650
user_metric_retention_time	Retention time of the user historical resource monitoring data. This parameter is valid only when **enable_user_metric_persistent** is set to **on**.	0–3,650
view_independent	Whether to decouple views from tables, functions, and synonyms. After the base table is restored, automatic association and re-creation are supported.	**on** or **off**

Parameter	Description	Value Range
wlm_memory_feedback_adjust	Whether to enable memory negative feedback for dynamic load management. The available options include: • **on** indicates that memory negative feedback is enabled. • **off** indicates that memory negative feedback is disabled.	**on** or **off**

Cluster Memory Configuration Parameters

Table 10-19 Cluster memory configuration parameters

Parameter	Description	Value Range
comm_usable_memory	Maximum memory available for buffering on the TCP proxy communication library or SCTP communication library on a single DN. The unit is KB.	102,400–1,073,741,823
cstore_buffers	Size of the shared buffer used by column-store tables and column-store tables (ORC, Parquet, and CarbonData) of OBS and HDFS foreign tables. The unit is KB.	16,384–1,073,741,823
maintenance_work_mem	Maximum size of memory used for maintenance operations, involving **VACUUM**, **CREATE INDEX**, and **ALTER TABLE ADD FOREIGN KEY**. This parameter may affect the execution efficiency of **VACUUM**, **VACUUM FULL**, **CLUSTER**, and **CREATE INDEX**.	1,024–2,147,483,647
max_process_memory	Maximum physical memory available for a database node. The unit is KB. The default value is calculated by multiplying the physical memory by 0.8 and dividing it by the sum of 1 and the maximum number of primary DNs in the cluster.	2,097,152–2,147,483,647
query_max_mem	Maximum memory that can be used by a query. If the value of **query_max_mem** is greater than **0**, an error will be reported if the query's memory usage exceeds that value.	0–2,147,483,647

Parameter	Description	Value Range
session_history_memory	Memory size of historical query views, in KB.	10,240–2,147,483,647
shared_buffers	Size of the shared memory used by DWS. If the value of this parameter is increased, DWS requires more System V shared memory than the default system setting. The unit is 8 KB.	16–1,073,741,823
udf_memory_limit	Maximum physical memory that can be used when UDFs are executed on each CN and DN, in KB.	204,800–2,147,483,647
work_mem	Size of the memory used by internal sequential operations and the Hash table before data is written into temporary disk files. Sort operations are required for **ORDER BY**, **DISTINCT**, and merge joins. Hash tables are used in hash joins, hash-based aggregation, and hash-based processing of **IN** subqueries. 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 **work_mem**.	64–2,147,483,647

Cluster Disk Configuration Parameters

Table 10-20 Cluster disk configuration parameters

Parameter	Description	Value Range
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. **–1** indicates no limit.	–1–2,147,483,647

Parameter	Description	Value Range
temp_file_limit	Size of a single file spilled to disk if splitting is triggered in a session. The temporary file can be a sort or hash temporary file, or the storage file for a held cursor.	–1–2,147,483,647

Cluster Network Configuration Parameters

Table 10-21 Cluster network configuration parameters

Parameter	Description	Value Range
comm_max_stream	Maximum number of concurrent data streams supported by the TCP proxy communication library or SCTP communication library. The value of this parameter must be greater than the number of concurrent operators multiplied by the average number of stream operators per concurrent operator multiplied by the square of smp.	1–65,535
max_connections	Maximum number of allowed concurrent connections to the database. This parameter affects the concurrent processing capability of the cluster.	100–262,143
max_pool_size	Maximum number of connections between the CN connection pool and another CN/DN.	1–65,535

SQL Tuning Parameters

Table 10-22 SQL tuning parameters

Parameter	Description	Value Range
agg_redistribute_enhancement	When the aggregate operation is performed, which contains multiple **group by** columns and none of them is the distribution key, a **group by** column will be selected for redistribution. This parameter specifies the policy of selecting a redistribution column.	**on** or **off**

Parameter	Description	Value Range
best_agg_plan	Type of hashagg plan generated by the optimizer.	0–3
cost_model_version	Model used for cost estimation in the application scenario. This parameter affects the distinct estimation of the expression, HashJoin cost model, estimation of the number of rows, distribution key selection during redistribution, and estimation of the number of aggregate rows.	0–4
default_statistics_target	Default analysis ratio.	–100–10,000
enable_codegen	Whether to enable code optimization. Currently, LLVM optimization is used. The availability options are **on** and **off**. You can choose **on** to enable code optimization.	-
enable_extrapolation_stats	Whether the extrapolation logic is used for data of date type based on historical statistics. The logic can increase the accuracy of estimation for tables whose statistics are not collected in time, but will possibly provide an overlarge estimation due to incorrect extrapolation. Enable the logic only in scenarios where the data of date type is periodically inserted.	**on** or **off**

Parameter	Description	Value Range
hashjoin_spill_strategy	Select a hash join policy. • **0**: If the size of the inner table is large and cannot be partitioned after data is spilled to disks for multiple times, the system attempts to place the outer table in the available memory of the database to create a hash table. If both the inner and outer tables are large, a nested loop join is performed. • **1**: If the size of the inner table is large and cannot be partitioned after data is spilled to disks for multiple times, the system attempts to place the outer table in the available memory of the database to create a hash table. If both the inner and outer tables are large, a hash join is forcibly performed. • **2**: If the size of the inner table is large and cannot be partitioned after data is spilled to disks for multiple times, a hash join is forcibly performed. • **3**: If the size of the inner table is large and cannot be partitioned after data is spilled to disks for multiple times, the system attempts to place the outer table in the available memory of the database to create a hash table. If both the inner and outer tables are large, an error is reported. • **4**: If the size of the inner table is large and cannot be partitioned after data is spilled to disks for multiple times, an error is reported.	0–6
max_streams_per_query	Number of Stream nodes in a query plan.	–1–10,000

Parameter	Description	Value Range
qrw_inlist2join_optmode	Whether to enable inlist-to-join query rewrite. • **disable**: inlist2join disabled • **cost_base**: cost-based inlist2join query rewriting • **rule_base**: forcible rule-based inlist2join query rewriting • A positive integer: threshold of Inlist2join query rewriting. If the number of elements in the list is greater than the threshold, the rewriting is performed.	-
query_dop	User-defined symmetric multi-processing (SMP) degree. [1, 64]: Fixed SMP is enabled, and the system will use the specified degree. [–64, –1]: SMP adaptation is enabled, and the system will dynamically select a degree from the limited range.	–64–64

Parameter	Description	Value Range
rewrite_rule	Rewriting rule for enabled optional queries. Some query rewriting rules are optional. Enabling them cannot always improve query efficiency. In certain scenarios, you can set the query rewriting rules through this parameter to achieve optimal query efficiency. • **none**: No optional query rewrite rules are used. • **Lazyagg**: The Lazy Agg query rewrite rule is used to eliminate aggregate operations in subqueries. • **magicset**: The Magic Set query rewrite rule is used to push conditions from the main query down to promoted sublinks. • **uniquecheck**: uses the Unique Check rewriting rule. (The situation can be enhanced when the target column does not include the sublink expression of the aggregate function. The function can only be activated if the value of the target column becomes unique after the sublink is aggregated using the associated column. Optimization engineers are advised to utilize this function.) • **disablerep**: uses the rule for forbidding sublink pull-up for replicated tables. • **projection_pushdown**: uses the projection pushdown rewriting rule to remove the columns that are not used by the parent query in the subquery. • **or_conversion**: uses the OR conversion rewriting rule to remove inefficiently executed associated OR conditions. • **plain_lazyagg**: uses the **Plain Lazy Agg** query rewriting rule to remove aggregation operations in a single subquery. This option is supported only by clusters of version 8.1.3.100 or later. • **eager_magicset**: uses the **eager_magicset** query rewriting	-

Parameter	Description	Value Range
	rule to push down conditions from the main query to subqueries. This option is supported only by clusters of version 8.2.0 or later. • **casewhen_simplification**: This rewrite rule uses the **CASE WHEN** statement to simplify queries. When enabled, it rewrites **(case when xxx then const1 else const2)=const1**. This option is supported only by clusters of version 8.3.0 or later. • **outer_join_quality_imply**: When there is an equi-join condition between a left outer join and a right outer join, this rule pushes the expression condition on the outer table's join column down to the inner table's join column. This option is supported only by clusters of version 8.3.0 or later. • **inlist_merge**: This query rewrite rule uses the **inlist_or_inlist** method to merge **OR** statements with the same base table column. When enabled, it merges and rewrites **(where a in (list1) or a in (list2))** to support **inlist2join**. This option is supported only by clusters of version 8.3.0 or later. • **subquery_qual_pull_up**: For subqueries that cannot be promoted, if the subquery has filtering conditions on columns that are also used for joining with other tables, this rule extracts the filtering conditions from the subquery and passes them to the other side of the join condition. Currently, only **var op const** forms without type conversion, such as **a > 2**, are supported. When enabled, it is assumed that **outer_join_quality_imply** is also enabled. This option is supported only by clusters of version 9.1.0 or later.	

SQL Compatibility Parameters

Table 10-23 SQL compatibility parameters

Parameter	Description	Value Range
full_group_by_mode	Behavior after enabling **disable_full_group_by_mysql**: • **nullpadding** indicates that NULL values in non-aggregate columns are filled with the non-NULL values in that column, potentially resulting in different rows in the result set. • **notpadding** indicates that NULL values in non-aggregate columns are not processed, and the entire row data is used, resulting in a random row for non-aggregate columns in the result set.	-

10.7 Managing DWS Tags

10.7.1 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 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.

DWS supports the following tags:

- Resource tags

  Non-global tags created on 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 DWS, tags can be added to the following resources:

- Cluster

  Tags can be added to a cluster when the cluster is being created or after it is successfully created. You can search for the cluster in the cluster list using tags.

  Each cluster can have a maximum of 20 tags.

  After you add tags to a cluster and then create a snapshot for the cluster, the tags cannot be restored if you use the snapshot to restore the cluster. Instead, you need to add tags again.

  When a cluster is deleted, non-predefined tags associated with the cluster are also deleted. Predefined tags need to be deleted on TMS.

10.7.2 Managing Tags

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 Choose **Dedicated Clusters** > **Clusters**, click the name of the cluster to which a tag is to be added, and click **Tag**.

Step 2 Click **Add Tag**.

Step 3 Configure tag information in the **Add Tag** dialog box. The value of a key cannot be left blank.

Table 10-24 Tag parameters

Parameter	Description	Example Value
Tag key	You can: • Select a predefined tag key or an existing resource tag key from the drop-down list of the text box. **NOTE** To add a predefined tag, you need to create one on TMS and select it from the drop-down list of **Tag key**. You can click **View predefined tags** to enter the **Predefined Tags** page of TMS. Then, click **Create Tag** to create a predefined tag. For more information, see "Predefined Tags" > "Creating Predefined Tags" in the *Tag Management Service User Guide*. • Enter a tag key in the text box. A tag key can contain a maximum of 128 characters. It cannot be an empty string, start with **_sys_**, or start or end with a space. Only letters, digits, spaces, and the following characters are allowed: _ . : = + - @ **NOTE** A key must be unique in a given cluster.	key01

Parameter	Description	Example Value
Tag value	You can: • Select a predefined tag value or resource tag value from the drop-down list of the text box. • Enter a tag value in the text box. A tag value can contain a maximum of 255 characters, which can be an empty string. It cannot start or end with a space. Only letters, digits, spaces, and the following characters are allowed: _ . : = + - @	value01

Step 4 Click **OK**.

----End

Searching for Clusters Based on Tags

You can quickly locate a tagged cluster using tags.

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters**.

Step 3 Click the search box above the cluster list and select the **Resource Tag** filter.

Step 4 Click the tag key to be searched for and select the corresponding tag value. Click the search box again to add more tag filters.

Search by tag supports only the keys and values that exist 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 **Search**. The target cluster will be displayed in the cluster list.

----End

Modifying a Tag

Step 1 Choose **Dedicated Clusters** > **Clusters**, click the name of the cluster to which a tag is to be added, and click **Tag**.

Step 2 Locate the row that contains the tag to be modified, and click **Edit** in the **Operation** column. The **Edit Tag** dialog box is displayed.

Step 3 Enter the new key value in the **Value** text box.

Step 4 Click **OK**.

----End

Deleting a Tag

Step 1 Choose **Dedicated Clusters** > **Clusters**, click the name of the cluster from which a tag is to be deleted, and click **Tag**.

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 After confirming that the information is correct, enter **DELETE** or click **Auto Enter** and click **OK** to delete the tag.

----End

10.8 Resetting the Password the DWS Database Administrator

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 **Dedicated Clusters** > **Clusters** page. After the password is reset, the account can be automatically unlocked. You can set the maximum number of incorrect password attempts (10 by default) by configuring the **failed_login_attempts** parameter on the **Parameter** page of the cluster. For details, see **Modifying GUC Parameters of the DWS Cluster**.

Resetting a Password

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **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 12 to 32 characters.
- Cannot be the username or the username spelled backwards.
- Contains 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 database administrator account of the cluster is deleted or renamed, password resetting fails.

----End

10.9 Starting, Stopping, and Deleting a DWS Cluster

Restarting a cluster

If a cluster is in the **Unbalanced** state or cannot work properly, you may need to restart it for restoration. After modifying a cluster's configurations, such as security settings and parameters, manually restart the cluster to make the configurations take effect.

Impact on the System

- A cluster cannot provide services during the restart. Therefore, before the restart, ensure that no task is running and all data is saved.

  If the cluster is processing service data, such as importing data, querying data, creating snapshots, or restoring snapshots, cluster restarting will cause file damage or restart failure. You are advised to stop all cluster tasks before restarting the cluster.

  View the **Session Count** and **Active SQL Count** metrics to check whether the cluster has active events. For details, see **Viewing DWS Cluster Monitoring Information on 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 DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters**.

Step 3 In the **Operation** column of the cluster to be restarted, choose **More** > **Restart**.

NOTE

The positions of the function keys in the **Operation** column are dynamic. To ensure that there are always two function keys visible before **More**, any function keys that typically appear only when you hover over **More** will be moved to a position directly before **More**. This adjustment occurs if there are some functions whose keys are supposed to be placed before **More** but are not supported for the current site.

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

Stopping a Cluster

If a cluster is no longer used, you can stop the cluster to bring services offline.

NOTE

- If the current console does not support this feature, contact technical support.
- After the cluster is stopped, ECS basic resources (vCPUs and memory) are no longer reserved. When you start the service again, it may fail to be started due to insufficient resources. In this case, wait for a while and try again later.

Procedure

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

Step 3 On the **Dedicated Clusters** page, locate the row that contains the target dedicated cluster, click **More** > **Stop** in the **Operation** column.

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

The **Task Information** of the cluster changes to **Stopping**. If the **Cluster Status** changes to **Stopped**, the cluster is stopped successfully.

----End

Starting a Cluster

You can start a stopped cluster to restore cluster services.

NOTE

If the current console does not support this feature, contact technical support.

Procedure

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

Step 3 On the **Dedicated Clusters** page, locate the row that contains the target dedicated cluster, click **More** > **Start** in the **Operation** column.

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

The **Task Information** of the cluster changes to **Starting**. If the **Cluster Status** changes to **Available**, the cluster is started successfully.

----End

Deleting a Cluster

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

Impact on the System

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

Procedure

Step 1 Log in to the DWS console.

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

Step 3 Choose **Dedicated Clusters** > **Clusters** and locate the cluster to be deleted.

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

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

- Create a snapshot for the cluster.

  If the cluster status is normal, click **Create Snapshot**. On the snapshot list page, click **Create Snapshot** to create a snapshot for the cluster to be deleted. For details, see **Manual Snapshots**. In the row of a cluster, choose **More** > **Delete**.

- Delete associated resources.
  - Release the EIP bound to a cluster.

    If an EIP is bound to the cluster, you are advised to select **EIP** to release the EIP of the cluster to be deleted.
  - Delete automated snapshots.
  - Delete manual snapshots.

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

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

If 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

10.10 Managing Enterprise Projects

An enterprise project is a cloud resource management mode. Enterprise Management provides users with comprehensive management in cloud-based . The Enterprise Management console differs from typical management consoles as it focuses on resource management rather than independent control and configuration of cloud products. It assists enterprises in managing within the hierarchy of companies, departments, and projects.

Binding an Enterprise Project

You can select an enterprise project during cluster creation to associate it with the cluster. For details, see **Creating a DWS Storage-Compute Coupled Cluster**. The **Enterprise Project** drop-down list displays the projects you created. In addition, the system has a built-in enterprise project (**default**). If you do not select an enterprise project for the cluster, the default project is used.

During cluster creation, if the cluster is successfully bound to an enterprise project, the cluster will be successfully created. If the binding fails, the system sends an alarm and the cluster fails to be created.

Snapshots of a DWS cluster retain the association between the cluster and its enterprise project. When the cluster is restored, the association is also restored.

When you delete a DWS cluster, the association between the cluster and its enterprise project is automatically deleted.

Viewing Enterprise Projects

After a cluster is created, you can view the associated enterprise project in the cluster list and **Cluster Information** page. You can query only the cluster resources of the project on which you have the access permission.

- In the cluster list on the **Clusters** page, view the enterprise project to which the cluster belongs.
- In the cluster list, find the target cluster and click the cluster name. The **Cluster Information** page is displayed, on which you can view the enterprise project associated with the cluster. Click the enterprise project name to view and edit it on the Enterprise Management console.
- When querying the resource list of a specified project on the Enterprise Management console, you can also query the DWS resources.

Searching for Clusters by Enterprise Project

Log in to the DWS console and choose **Dedicated Clusters** > **Clusters**. Click the search box above the cluster list and select **Enterprise Project**. Enter the project name and click the search button to view all clusters associated with the project.

Migrating a Cluster to or Out of an Enterprise Project

A DWS cluster can be associated with only one enterprise project. After a cluster is created, you can migrate it from its current enterprise project to another one on the Enterprise Management console, or migrate the cluster from another enterprise project to a specified enterprise project. After the migration, the cluster is associated with the new enterprise project. The association between the cluster and the original enterprise project is automatically released.

11 DWS Cluster O&M

11.1 Viewing DWS Cluster Monitoring Information on 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 DWS. For details, see the *User Guide* and *API Reference* of Cloud Eye.

Namespace

SYS.DWS

Cluster Monitoring Metrics

With the 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 11-1 describes DWS monitoring metrics.

Table 11-1 DWS monitoring metrics

Metric ID	Name	Description	Value Range	Unit	Conversion Rule	Monitored Object & Dimension	Monitoring Period (Raw Data)
dws001_shared_buffer_hit_ratio	Cache Hit Ratio	Ratio of requested data that already exists in the cache. It is the ratio of the amount of data that already exists in the cache to the total amount of requested data. A higher cache hit ratio means higher cache usage of the system, fewer times that data needs to be read from the disk or network, and faster system response speed.	0% to 100%	%	N/A	Data warehouse cluster	4 minutes

Metric ID	Name	Description	Value Range	Unit	Conversion Rule	Monitored Object & Dimension	Monitoring Period (Raw Data)
dws002_in_memory_sort_ratio	In-memory Sort Ratio	Ratio of the extra memory space used by the sorting algorithm to the memory space occupied by the sorted data. In a merge sort, for example, the size of the merge buffer is often proportional to the size of the sorted data, so the in-memory ratio is usually between 10% and 50%.	0% to 100%	%	N/A	Data warehouse cluster	4 minutes
dws003_physical_reads	File Reads	Total number of database file reads	> 0	count	N/A	Data warehouse cluster	4 minutes
dws004_physical_writes	File Writes	Total number of database file writes	> 0	count	N/A	Data warehouse cluster	4 minutes
dws005_physical_reads_per_second	File Reads per Second	Number of database file reads per second	≥ 0	count/s	N/A	Data warehouse cluster	4 minutes

Metric ID	Name	Description	Value Range	Unit	Conversion Rule	Monitored Object & Dimension	Monitoring Period (Raw Data)
dws006_physical_writes_per_second	File Writes per Second	Number of database file writes per second	≥ 0	count/s	N/A	Data warehouse cluster	4 minutes
dws007_db_size	Data Volume	Total data volume of the database	≥ 0 MB	MB	1024(IEC)	Data warehouse cluster	4 minutes
dws008_active_sql_count	Active SQL Count	Number of active SQLs in the database	≥ 0	count	N/A	Data warehouse cluster	4 minutes
dws009_session_count	Session Count	Number of sessions that access the database	≥ 0	count	N/A	Data warehouse cluster	4 minutes
dws010_cpu_usage	CPU Usage	CPU usage of each node in the cluster	0% to 100%	%	N/A	Data warehouse node	1 minute

Metric ID	Name	Description	Value Range	Unit	Conversion Rule	Monitored Object & Dimension	Monitoring Period (Raw Data)
dws011_mem_usage	Memory Usage	Memory usage of each node in a cluster, in percentage **NOTE** After the console is upgraded to 8.3.0.202, the memory usage includes the memory occupied by the cache. Therefore, the value of this metric increases compared with that before the upgrade.	0% to 100%	%	N/A	Data warehouse node	1 minute
dws012_iops	IOPS	Number of I/O requests processed by each node in the cluster per second	≥ 0	count/s	N/A	Data warehouse node	1 minute
dws013_bytes_in	Network Input Throughput	Data input to each node in the cluster per second over the network	≥ 0 bytes/s	byte/s	1024(IEC)	Data warehouse node	1 minute
dws014_bytes_out	Network Output Throughput	Data sent to the network per second from each node in the cluster	≥ 0 bytes/s	byte/s	1024(IEC)	Data warehouse node	1 minute

Metric ID	Name	Description	Value Range	Unit	Conversion Rule	Monitored Object & Dimension	Monitoring Period (Raw Data)
dws015_disk_usage	Disk Usage	Disk usage of each node in the cluster	0% to 100%	%	N/A	Data warehouse node	1 minute
dws016_disk_total_size	Total Disk Size	Total disk space of each node in the cluster	100 to 2,000 GB	GB	1024(IEC)	Data warehouse node	1 minute
dws017_disk_used_size	Used Disk Space	Used disk space of each node in the cluster	0 to 3,600 GB	GB	1024(IEC)	Data warehouse node	1 minute
dws018_disk_read_throughput	Disk Read Throughput	Data volume read from each disk in the cluster per second	≥ 0 bytes/s	byte/s	1024(IEC)	Data warehouse node	1 minute
dws019_disk_write_throughput	Disk Write Throughput	Data volume written to each disk in the cluster per second	≥ 0 bytes/s	byte/s	1024(IEC)	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	> 0s	Second	N/A	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	> 0s	Second	N/A	Data warehouse node	1 minute

Metric ID	Name	Description	Value Range	Unit	Conversion Rule	Monitored Object & Dimension	Monitoring Period (Raw Data)
dws022_avg_disk_queue_length	Average Disk Queue Length	Average I/O queue length of a disk	≥ 0	count	N/A	Data warehouse node	1 minute
dws_024_dn_diskio_util	DN I/O usage	Average disk I/O usage of DNs in a cluster	0% to 100%	%	N/A	Data warehouse instance	1 minute

Dimensions

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

Cluster and Node Monitoring Information

Step 1 Log in to the DWS console and choose **Dedicated Clusters** > **Clusters**.

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

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

NOTE

The positions of the function keys in the **Operation** column are dynamic. To ensure that there are always two function keys visible before **More**, any function keys that typically appear only when you hover over **More** will be moved to a position directly before **More**. This adjustment occurs if there are some functions whose keys are supposed to be placed before **More** but are not supported for the current site.

----End

Comparing the Monitoring Metrics of Multiple Nodes

Step 1 In the navigation pane of the Cloud Eye management console, choose **Dashboards** > **My Dashboards**. Click the name of the dashboard for which you want to add a graph. On the **My Dashboards** page that is displayed, click **Add Graph**.

Step 2 On the **Add Graph** page, you can select **Line Chart** or **Bar Chart** to display the graph. After confirming that the information is correct, click **OK**.

For example, select **Line Chart** and **One View for Multiple Metrics** to compare the CPU usage of three DWS nodes. The following table describes the parameters.

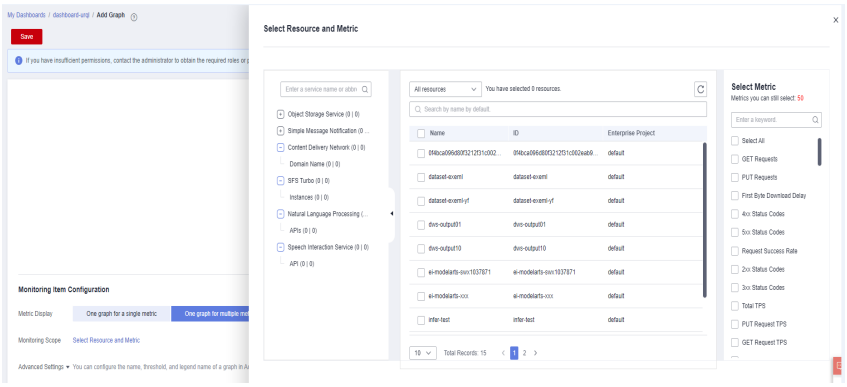


Table 11-2 Configuration example

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

Step 3 Click **OK**.

On the selected **My Dashboards** page, you can view the metric trend on the newly added monitoring graph. You can click the zoom in button to zoom in and view detailed metric comparison data.

----End

Creating Alarm Rules

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

The DWS alarm rules include alarm rule name, instance, metric, threshold, monitoring interval and whether to send notification. This section describes how to set DWS alarm rules.

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

Step 3 Locate the row containing the target cluster, click **View Metric** in the **Operation** column to enter the Cloud Eye management console and view the 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 11-3 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 **Data Warehouse Nodes** or **Data Warehouses**.	Data Warehouse Node
Monitoring Scope	Resource scope to which an alarm rule applies. Select **Specific resources** and select one or more monitoring objects. For DWS, select the cluster ID or node ID in the dialog box that is displayed.	Specific resources

Parameter	Description	Example Value
Trigger Rule	You can select an associated template, use an existing template or create a custom template as required.	Create manually
Template	This parameter is valid only when **Use template** is selected. Select the template to be imported. If no alarm template is available, click **Create Custom Template** to create one that meets your requirements.	-
Alarm Policy	This parameter is valid only when **Create manually** is selected. 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. **Table 11-1** lists the DWS monitoring metrics.	-
Alarm Severity	Severity of an alarm. Valid values are **Critical**, **Major**, **Minor**, and **Informational**.	Major

3. Configure the alarm notification parameters as prompted.

Table 11-4 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. You can enable (recommended) or disable **Alarm Notification**.	Enable
Validity Period	Cloud Eye sends notifications only within the validity period specified in the alarm rule. For example, if **Validity Period** is set to **00:00-8:00**, Cloud Eye sends notifications only within 00:00-8:00.	-

Parameter	Description	Example Value
Notification Object	Name of the topic to which the alarm notification is sent. If you enable **Alarm Notification**, 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 *Simple Message Notification User Guide*. For details about how to create a topic, see the *Simple Message Notification User Guide*.	-
Trigger Condition	Condition for triggering the alarm. You can select **Generated alarm**, **Cleared alarm**, 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

11.2 Viewing and Subscribing to DWS Cluster Alarms

11.2.1 Alarm Management

Overview

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

NOTE

- This feature is supported only in cluster version 8.1.1.200 and later.
- Currently, alarms cannot be categorized and managed by enterprise project.

Visiting the Alarms Page

Step 1 Log in to the DWS console.

Step 2 In the navigation tree on the left, choose **Monitoring** > **Alarm**.

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, alarm source, 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

NOTE

The alarm policy is triggered based on the current configuration.

Table 11-5 Threshold alarms of DMS alarm sources

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

Type	Name	Severity	Description
Default	Node Data Disk I/O Usage Exceeds the Threshold	Urgent	This alarm is generated if the threshold of data disk (**/var/chroot/DWS/data***[n]*) I/O usage (**util**) 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 usage (**util**) is lower than the threshold and the constraint is not met.
Default	Node Data Disk Latency Exceeds the Threshold	Important	This alarm is generated if the threshold of data disk (**/var/chroot/DWS/data***[n]*) I/O latency (**await**) 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 (**await**) is lower than the threshold and the constraint is not met.
Default	Data Spilled to Disks of the Query Statement Exceeds the Threshold	Urgent	This alarm is generated if the threshold of data flushed to disks of the SQL statement in the cluster is exceeded within the specified period and the constraint is not met. The alarm can be cleared only after you handle the SQL statement.
Default	Number of Queuing Query Statements Exceeds the Threshold	Urgent	This alarm is generated if the threshold of the number of queuing SQL statements is exceeded within the specified period. The alarm will be cleared when the number of queuing SQL statements is less than the threshold.
Default	Queue Congestion in the Default Cluster Resource Pool	Urgent	This alarm is generated if the queue in the default resource pool of a cluster is congested and no alarm suppression conditions are met. This alarm will be cleared if the queue is not congested.

Type	Name	Severity	Description
Default	Long SQL Probe Execution Duration in a Cluster	Urgent	This alarm is generated if the DMS alarm module detects a SQL probe execution duration on a server and no alarm suppression conditions are met. If no execution duration exceeds the threshold, the alarm will be automatically cleared. **NOTE** The alarm is supported only in 8.1.1.300 and later cluster versions. For earlier versions, contact technical support.
Default	A Vacuum Full Operation That Holds a Table Lock for A Long Time Exists in the Cluster	Important	In a specified period, the DMS alarm module detects that VACUUM FULL has been running for a long time in the cluster and blocks other operations. This alarm is generated if there are other SQL statements in the lock wait state and no suppression conditions are met. This alarm will be cleared if VACUUM FULL in the cluster did not cause lock wait. **NOTE** If this alarm is generated, contact technical support engineers.
Default	Instance Memory Usage of a Cluster Node Exceeds the Threshold	Urgent	This alarm is generated if the DMS alarm module detects the instance memory usage on a node in a cluster exceeds the threshold and no alarm suppression conditions are met. If the usage decreases, the alarm will be automatically cleared. **NOTE** If this alarm is generated, contact technical support engineers.
Default	Dynamic Memory Usage of a Cluster Node Exceeds the Threshold	Urgent	This alarm is generated if the DMS alarm module detects the dynamic memory usage on a node in a cluster exceeds the threshold and no alarm suppression conditions are met. If the usage decreases, the alarm will be automatically cleared. **NOTE** If this alarm is generated, contact technical support engineers.

Type	Name	Severity	Description
Default	Disk Usage of a DWS Cluster Resource Pool Exceeds the Threshold	Urgent	The DMS alarm module generates an alarm if the disk usage of the cluster resource pool exceeds the set threshold within a specific time frame and the suppression conditions are not met. The alarm is cleared when the DMS alarm module detects that the disk usage of the cluster resource pool is below the threshold. **NOTE** If this alarm is generated, contact technical support engineers.
Default	Session Usage in a DWS Cluster Exceeds the Threshold	Urgent	The DMS alarm module generates an alarm if the session usage in the cluster exceeds the set threshold within a specific time frame and the suppression conditions are not met. The alarm is cleared when the DMS alarm module detects that the session usage in the cluster is below the threshold. **NOTE** If this alarm is generated, contact technical support engineers.
Default	Active Session Usage in a DWS Cluster Exceeds the Threshold	Urgent	The DMS alarm module generates an alarm if the active session usage in the cluster exceeds the set threshold within a specific time frame and the suppression conditions are not met. The alarm is cleared when the DMS alarm module detects that the active session usage in the cluster is below the threshold. **NOTE** If this alarm is generated, contact technical support engineers.
Default	Number of Database Deadlocks in a DWS Cluster Exceeds the Threshold	Urgent	If the number of deadlocks in the cluster database exceeds the threshold within a specific time frame and the suppression conditions are not met, the DMS alarm module will generate an alarm. The alarm will be cleared once the DMS alarm module detects that the number of deadlocks in the cluster database is below the threshold. **NOTE** If this alarm is generated, contact technical support engineers.

Type	Name	Severity	Description
Default	Database Session Usage of the DWS Cluster Exceeds the Threshold	Urgent	The DMS alarm module will generate an alarm if the session usage of the cluster database goes over the threshold within a specific time frame and the suppression conditions are not met. The alarm will be resolved by the DMS alarm module once it detects that the session usage of the cluster database is below the threshold. **NOTE** If this alarm is generated, contact technical support engineers.

11.2.2 Alarm Subscriptions

You can subscribe to 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 DWS console.

Step 2 In the navigation pane on the left, choose **Management** > **Alarm** and click **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.

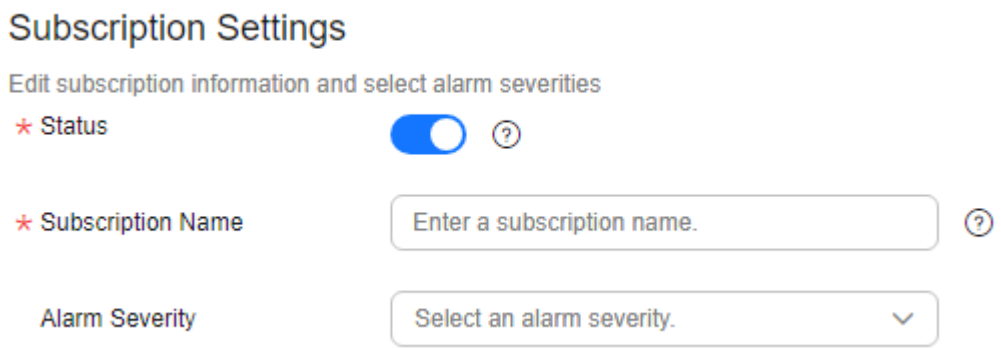


Table 11-6 Subscription parameters

Parameter	Description
Status	Whether to enable the alarm subscription. 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: • Contains only letters, digits, hyphens (-), and underscores (_), and must start with a letter or digit. • Contains 1 to 256 characters.
Cluster	Select the cluster to subscribe to. Note that only one cluster can be subscribed to multiple alarms.

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**. The SMN console is displayed.

NOTE

The selected topic must have granted 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** for services 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 DWS console.

Step 2 In the navigation pane on the left, choose **Management** > **Alarm** and click **Subscriptions**.

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

Step 4 On the **Edit Subscription** page displayed, modify the parameters. For details, see **Step 4** to **5**.

Step 5 Click **OK**.

----End

Deleting a Subscription

Step 1 Log in to the DWS console.

Step 2 In the navigation pane on the left, choose **Management** > **Alarm** and click **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

11.3 Viewing and Subscribing to DWS Cluster Events

11.3.1 Event Notifications Overview

Overview

DWS uses the Simple Message Notification (SMN) service to send notifications of DWS events. The SMN function is only available by subscription. In a subscription, you need to specify one or more event filtering conditions. When an event that matches all filtering conditions occurs, DWS sends a notification based on the subscription. The filter conditions include the **Event Type** (for example, **Management**, **Monitoring**, or **Security**), **Event Severity** (for example, **Normal** or **Warning**), and **Event Source Category** (for example, **Cluster** or **Snapshot**).

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 DWS.

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

Table 11-7 Events whose **Event Source Category** is **Cluster**

Event Type	Event Name	Event Severity	Event
Management	createClusterFail	Warning	Cluster creation failed.
Management	createClusterSuccess	Normal	The cluster is created.
Management	createCluster	Normal	Cluster creation started.
Management	extendCluster	Normal	Cluster scale-out started.
Management	extendClusterSuccess	Normal	A cluster is scaled out.
Management	extendClusterFail	Warning	Cluster scale-out failed.

Event Type	Event Name	Event Severity	Event
Management	deleteClusterFail	Warning	Failed to delete the cluster.
Management	deleteClusterSuccess	Normal	The cluster is deleted.
Management	deleteCluster	Normal	Cluster deletion started.
Management	restoreClusterFail	Warning	Cluster restoration failed.
Management	restoreClusterSuccess	Normal	The cluster is restored.
Management	restoreCluster	Normal	Cluster restoration started.
Management	restartClusterFail	Warning	Cluster restart failed.
Management	restartClusterSuccess	Normal	The cluster is restarted.
Management	restartCluster	Normal	Cluster restarted.
Management	configureMRSExtDataSources	Normal	Configuration of MRS external data source for the cluster started.
Management	configureMRSExtDataSourcesFail	Warning	The cluster's MRS external data source configurations failed.
Management	configureMRSExtDataSourcesSuccess	Normal	The MRS external data source of the cluster is configured.
Management	deleteMRSExtDataSources	Normal	Deletion of MRS external data source for the cluster started.
Management	deleteMRSExtDataSourcesFail	Warning	The deletion of the MRS external data source for the cluster failed.
Management	deletedMRSExtDataSourcesSuccess	Normal	MRS external data source is deleted.
Management	bindEipToCluster	Normal	An EIP is bound to the cluster.

Event Type	Event Name	Event Severity	Event
Management	bindEipToClusterFail	Warning	Cluster EIP binding failed.
Management	unbindEipToCluster	Normal	The EIP is unbound from the cluster.
Management	unbindEipToCluster-Fail	Warning	Cluster EIP unbinding failed
Management	refreshEipToCluster	Normal	The cluster's EIP is refreshed.
Management	refreshEipToCluster-Fail	Warning	Cluster EIP refreshing failed.
Security	resetPasswordFail	Warning	The password reset attempt was unsuccessful.
Security	resetPasswordSuccess	Normal	The cluster password has been reset.
Security	updateConfiguration	Normal	Start to update cluster security parameters.
Security	updateConfiguration-Fail	Warning	Cluster security parameter update failed.
Security	updateConfiguration-Success	Normal	Cluster security parameters were updated.
Monitoring	repairCluster	Normal	The node is faulty and the cluster starts to be repaired.
Monitoring	repairClusterFail	Warning	Cluster repairing failed.
Monitoring	repairClusterSuccess	Normal	The cluster is repaired.

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

Table 11-8 Events whose **Event Source Category** is **Snapshot**

Event Type	Event Name	Event Severity	Event
Management	deleteBackup	Normal	The snapshot is deleted.
Management	deleteBackupFail	Warning	Snapshot deletion failed.
Management	createBackup	Normal	The snapshot is being created.
Management	createBackupSuccess	Normal	The snapshot is created.
Management	createBackupFail	Warning	Snapshot creation failed.

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

Table 11-9 Events whose **Event Source Category** is **DR**.

Event Type	Event Name	Event Severity	Event
Management	beginCreateDisasterRecovery	Normal	The DR task is being created.
Management	createDisasterRecoverySuccess	Normal	The DR task is created.
Management	createDisasterRecoveryFail	Warning	DR task creation failed.
Management	beginStartDisasterRecovery	Normal	The DR task starts.
Management	startDisasterRecoverySuccess	Normal	The DR task started successfully.

Event Type	Event Name	Event Severity	Event
Management	startDisasterRecoveryFail	Warning	The DR task was unable to start.
Management	beginStopDisasterRecovery	Normal	The DR task is being stopped.
Management	stopDisasterRecoverySuccess	Normal	DR stopped successfully.
Management	stopDisasterRecoveryFail	Warning	The DR task could not be stopped.
Management	beginSwitchoverDisasterRecovery	Normal	The DR switchover starts.
Management	switchoverDisasterRecoverySuccess	Normal	The DR switchover is successful.
Management	switchoverDisasterRecoveryFail	Warning	The DR switchover failed.
Management	beginDeleteDisasterRecovery	Normal	The DR task is being deleted.
Management	deleteDisasterRecoverySuccess	Normal	The DR task is deleted.
Management	deleteDisasterRecoveryFail	Warning	The DR task deletion failed.
Management	disasterRecoveryAbnormal	Warning	The DR task runs abnormally.
Management	beginFailoverDisasterRecovery	Normal	The abnormal switchover starts.
Management	failoverDisasterRecoverySuccess	Normal	The abnormal switchover is successful.

Event Type	Event Name	Event Severity	Event
Management	failoverDisasterRecoveryFail	Warning	The abnormal switchover fails.
Management	beginRecoveryDisaster	Normal	The disaster recovery starts.
Management	recoveryDisasterSuccess	Normal	The disaster recovery is successful.
Management	recoveryDisasterFail	Warning	The disaster recovery fails.
Management	emptyDisasterRecovery	Warning	No DR table exists in the current DR object.
Management	switchoverContinueAsFailoverDisasterRecovery	Warning	The DR switchover is degraded to an abnormal switchover.

- The following table lists the events whose event source type is data migration.

  Table 11-10 Events whose event source type is data migration

Event Type	Event Name	Event Severity	Event
Data migration	dataMigrationApplicationDetectedAbnormal	Warning	The job task status is abnormal.
Data migration	dataMigrationApplicationReturnNormal	Normal	The job task is restored.
Data migration	dataMigrationCreateApplication	Normal	Create a job task.
Data migration	dataMigrationCreateCluster	Normal	Start to create a data migration instance.

Event Type	Event Name	Event Severity	Event
Data migration	dataMigrationCreateClusterFailed	Warning	Failed to create the data migration instance.
Data migration	dataMigrationCreateClusterSuccess	Normal	The data migration instance is created.
Data migration	dataMigrationCreateConnection	Normal	Create a connection.
Data migration	dataMigrationCreateMapping	Normal	Create a table mapping configuration.
Data migration	dataMigrationDeleteApplication	Normal	Start to delete the job task.
Data migration	dataMigrationDeleteApplicationFailed	Warning	Failed to delete the job.
Data migration	dataMigrationDeleteApplicationSuccess	Normal	The job is deleted.
Data migration	dataMigrationDeleteCluster	Normal	Start to delete the data migration instance.
Data migration	dataMigrationDeleteClusterApplication	Normal	Start to delete the job task.
Data migration	dataMigrationDeleteClusterApplicationFailed	Warning	Failed to delete the job.
Data migration	dataMigrationDeleteClusterApplicationSuccess	Normal	The job is deleted.
Data migration	dataMigrationDeleteClusterFailed	Warning	Failed to delete the data migration instance.
Data migration	dataMigrationDeleteClusterSuccess	Normal	The data migration instance is deleted.

Event Type	Event Name	Event Severity	Event
Data migration	dataMigrationDeleteConnection	Normal	Delete the connection configuration.
Data migration	dataMigrationDeleteMapping	Normal	Delete the table mapping configuration.
Data migration	dataMigrationDialsConnection	Normal	Test the connection configuration.
Data migration	dataMigrationModifyConnection	Normal	Modify the connection configuration.
Data migration	dataMigrationModifyMapping	Normal	Modify the table mapping configuration.
Data migration	dataMigrationStartApplication	Normal	Start the job task.
Data migration	dataMigrationStartApplicationFailed	Warning	Failed to start the job.
Data migration	dataMigrationStartApplicationSuccess	Normal	The job task is started.
Data migration	dataMigrationStopApplication	Normal	Start to stop the job.
Data migration	dataMigrationStopApplicationFailed	Warning	Failed to stop the job.
Data migration	dataMigrationStopApplicationSuccess	Normal	The job task is stopped.

11.3.2 Subscribing to Event Notifications

After subscribing to 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 DWS console.

Step 2 In the navigation tree on the left, choose **Management** > **Events**.

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 11-11 Subscription parameters

Parameter	Description
Notification	Enable or disable event subscription. After notification is disabled, the system stops sending notifications of subscribed events but does not delete the subscription.
Subscription Name	Enter the name of a subscription. • The name can contain letters (upper or lower case), digits, hyphens (-), and underscores (_) and must start with a letter or digit. • The name must be between 1 and 256 characters in length.
Event Type	Select the type of the event to be subscribed. Possible values are **Management**, **Monitoring**, and **Security**.
Event Severity	Select the alarm severity of the event. Possible values are **Normal** and **Warning**.
Event Source Category	Select the event source category: cluster or snapshot.

Step 5 Select a message notification topic from the **Message Notification Topic** drop-down list.

- The selected topic must have granted DWS the permission for publishing messages to the topic.

  To grant permissions, configure topic policies on the SMN management console. For details, see **Topic Management** > **Configuring Topic Policies** in the *Simple Message Notification User Guide*. When configuring the topic policy, select **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 Choose **Management** > **Events** and click **Subscriptions**.

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 Choose **Management** > **Events** and click **Subscriptions**.

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

11.3.3 Viewing Events

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

Step 1 Log in to the DWS console.

Step 2 In the navigation tree on the left, choose **Management** > **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 search for events by time, event, event level, event source, event source type, or event type using the search box at the top of the event list.

----End

11.4 Backing Up and Restoring a DWS Cluster

11.4.1 Overview

A snapshot is a full or incremental backup of a DWS cluster at a specific point in time. It records the current database data and cluster information, including the number of nodes, node specifications, and database administrator name. Snapshots can be created manually or automatically. For details, see **Manual Snapshots** and **Automated Snapshots**.

If you restore a snapshot to a new cluster, DWS creates a cluster based on the cluster information recorded in the snapshot, and then restores data from the snapshot. For more information, see **Restoring a Snapshot to a New Cluster**.

If you restore a snapshot to the original cluster, DWS clears the existing data in the cluster, and then restores the database information from the snapshot to the cluster. For more information, see **Restoring a Snapshot to the Current Cluster**.

The snapshot backup and restoration rates are listed below. The rates are obtained from the test environment with local SSDs as the backup media. The rates are for reference only. The actual rate depends on your disk, network, and bandwidth resources.

- Backup rate: 200 MB/s/DN
- Restoration rate: 125 MB/s/DN

Constraints and Limitations

- **Backing up the cluster is essential for maintaining data reliability, especially when the service provider cannot restore data through upstream re-import. This helps prevent data loss caused by human or other factors.**
- The cluster versions that support schema-level snapshots are listed below. If the current console interface does not support this feature, contact technical support.
  - 9.1.0.100 or later
  - 8.3.0.110 or later 8.3.0.*xxx* cluster versions
  - 8.2.1.230 or later 8.2.1.2*xx* versions
- OBS snapshot storage space
  - The cluster storage is provided by DWS free of charge. Cluster storage = Storage space per node x Number of nodes
- The dependency of the snapshot service is as follows:
  - The snapshot management function depends on OBS or NFS.
  - If the backup device is an NFS backup media, the NFS backup media must be mounted to the high-performance SFS Turbo. For details, see **11.1.3.2 Automatic Snapshot Policy**.
  - Only the snapshots stored in OBS can be used to restore data to a new cluster.
- The new DWS cluster created based on the snapshot must have the same configurations as the original cluster. That is, the number and specifications of nodes, memory, and disks in the new cluster must be the same as those in the original cluster.
- If you create a new cluster based on a snapshot without modifying parameters, the parameters of the new cluster will be the same as those of the snapshot.
- During snapshot creation, do not perform the **VACUUM FULL** operation, or the cluster may become read-only.
- Snapshot creation affects disk I/O performance. You are advised to create snapshots during off-peak hours.
- During the snapshot creation, some intermediate files are retained, which occupy extra disk space. Therefore, create snapshots in off-peak hours and ensure that the disk capacity usage is less than 70%.

11.4.2 Manual Snapshots

11.4.2.1 Creating a Manual Snapshot of a Cluster

Prerequisites

A cluster snapshot is a complete backup that records point-in-time configuration data and service data of a 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 DWS console. Manual snapshots are full backup data, which takes a long time to create.

NOTE

- Manual cluster snapshots can be backed up to OBS or NFS.
- To create a manual snapshot of a cluster, the cluster state must be **Available**, **To be restarted**, or **Unbalanced**. In cluster versions earlier than 8.1.3.101, you can also create a snapshot of a cluster in the **Read-only** state.

Impact on the System

If a snapshot is being created for a cluster, the cluster cannot be restarted, scaled, its password cannot be reset, and its configurations cannot be modified.

NOTE

To ensure the integrity of snapshot data, do not write data during snapshot creation.

Procedure

Step 1 Log in to the DWS console.

Step 2 Choose **Management** > **Snapshots**. Alternatively, in the cluster list, click the name of the target cluster to switch to the **Cluster Information** page. Then, click **Snapshots**.

Step 3 Click **Create Snapshot** in the upper right corner. Alternatively, choose **More** > **Create Snapshot** in the **Operation** column.

Step 4 Configure the following snapshot information:

- **Cluster Name**: Select a DWS cluster from the drop-down list. The drop-down list only displays clusters that are in the **Available** state.
- **Snapshot Name**: Enter a snapshot name. The snapshot name must be 4 to 64 characters in length and start with a letter. It is case-insensitive and contains only letters, digits, hyphens (-), and underscores (_).
- **Snapshot Level**: Select **cluster**.
- **Snapshot Description**: Enter the snapshot information. This parameter is optional. Snapshot information contains 0 to 256 characters and does not support the following special characters: !<>'=&"

Step 5 Click **Create**.

Task status of the cluster for which you are creating a snapshot is **Creating snapshot**. The status of the snapshot that is being created is **Creating**. After the snapshot is created, its status changes to **Available**.

NOTE

If the snapshot size is much greater than that of the data stored in the cluster, the data is possibly labeled with a deletion tag, but is not cleared and reclaimed. In this case, clear the data and recreate a snapshot. For details, see **How Can I Clear and Reclaim the DWS Storage Space?**

----End

11.4.2.2 Creating a Manual Snapshot of a Schema

Overview

A schema snapshot is a backup of specific schemas in a DWS cluster at a specific point in time. This section describes how to create a schema snapshot on the **Snapshots** page.

A manual fine-grained snapshot can be created at any time. It will be retained until it is deleted from the DWS console.

NOTE

- If the current console does not support this feature, contact technical support.
- Manual schema snapshots can be backed up to OBS or NFS.
- Schema snapshots can be created only for clusters in **Available** or **Unbalanced** state.

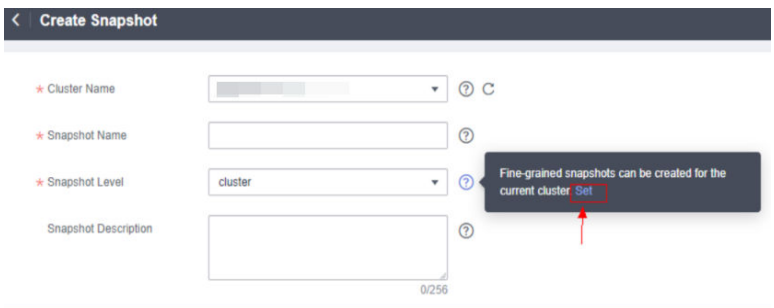
Prerequisites

Manually enable the fine-grained snapshot.

Step 1 Choose **Management** > **Snapshots**. Alternatively, in the cluster list, click the name of the target cluster to switch to the **Cluster Information** page. Then, click **Snapshots**.

Step 2 Click **Create Snapshot** in the upper right corner. Alternatively, choose **More** > **Create Snapshot** in the **Operation** column.

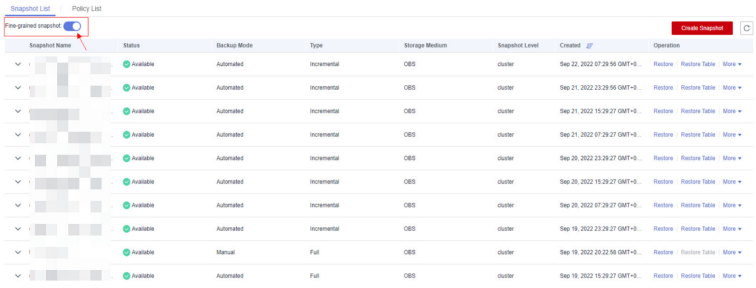
Step 3 Click next to **Snapshot Level** and click **Set**.



Step 4 On the **Snapshot List** page, toggle the fine-grained snapshot switch.

: enabled

: disabled



NOTE

- If the fine-grained snapshot is enabled, you can restore specific tables from automatic or manual snapshots.

----End

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 DWS console.

Step 2 Choose **Management** > **Snapshots**. Alternatively, in the cluster list, click the name of the target cluster to switch to the **Cluster Information** page. Then, click **Snapshots**.

Step 3 Click **Create Snapshot** in the upper right corner. Alternatively, choose **More** > **Create Snapshot** in the **Operation** column.

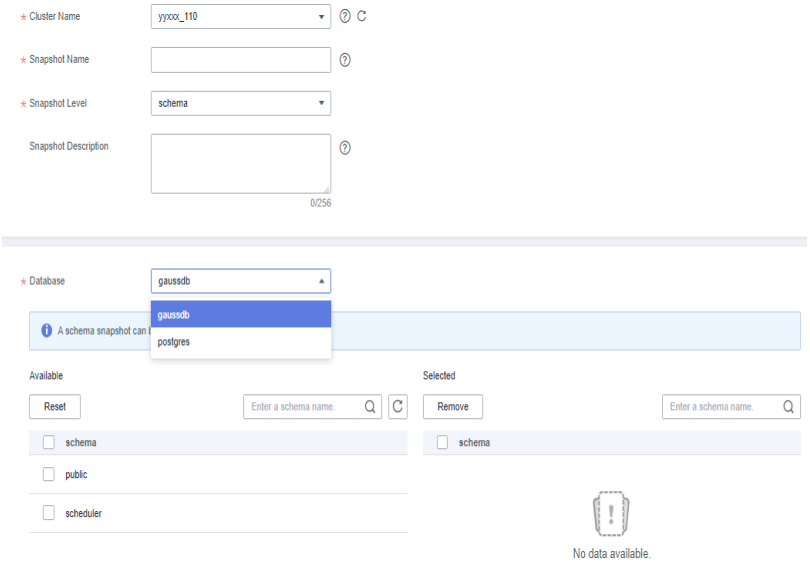
Step 4 Configure the following snapshot information:

- **Cluster Name**: Select a DWS cluster from the drop-down list. The drop-down list only displays clusters that are in the **Available** state.
- **Snapshot Name**: Enter a snapshot name. The snapshot name must be 4 to 64 characters in length and start with a letter. It is case-insensitive and contains only letters, digits, hyphens (-), and underscores (_).
- **Snapshot Level**: Select **schema**.
- **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: !<>'=&"

Step 5 Specify the snapshots to be backed up.

- Select a database from the **Database** drop-down list.

- In the schema list, select the schemas to be backed up. To search for a schema, enter its name in the search box in the upper right corner of the list, and click . Fuzzy search is supported.



NOTE

- Schemas in different databases cannot be backed up at a time.
- By default, a maximum of 50 schemas can be backed up at a time.

Step 6 Click **Create**.

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 a snapshot is larger than the available storage space in the cluster, check whether there is data that has been marked as deleted but actually still exists in the cluster. In this case, delete such data and create a snapshot again. For details, see **How Can I Clear and Reclaim the DWS Storage Space?**

----End

11.4.2.3 Deleting a Manual Snapshot

On the **Snapshot Management** page of the DWS console, you can delete an unwanted snapshot in the **Unavailable** state or delete an available snapshot to release the storage space.

CAUTION

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

Procedure

Step 1 Log in to the DWS console.

Step 2 Choose **Management** > **Snapshots**. Alternatively, in the cluster list, click the name of the target cluster to switch to the **Cluster Information** page. Then, click **Snapshots**. All snapshots are displayed by default.

Step 3 In the **Operation** column of the snapshot that you want to delete, choose **More** > **Delete**.

NOTE

You can only delete snapshots that were manually created.

Step 4 If the information is correct, enter **DELETE** and click **OK** to delete the snapshot.

----End

11.4.3 Automated Snapshots

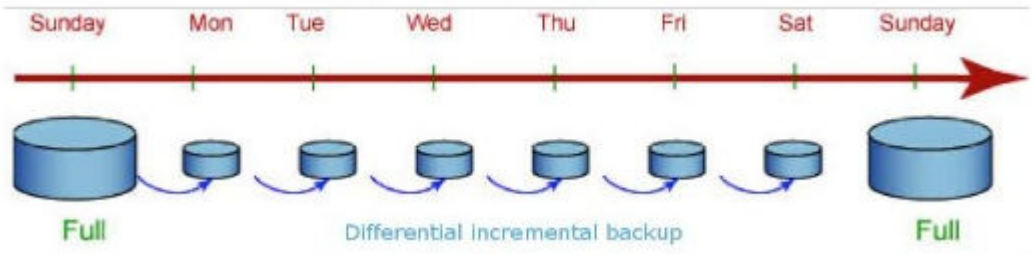
11.4.3.1 Automated Snapshot Overview

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.

For snapshot restoration, DWS uses all backups from the latest full backup to the current one, ensuring no data loss.

If the retention period of an incremental snapshot exceeds the maximum retention period, DWS does not delete the snapshot immediately. Instead, DWS retains it until the next full snapshot is completed, when the deletion of the snapshot will not hinder incremental data backup and restoration.

Figure 11-1 Snapshot backup process



Automated snapshots are enabled by default when you create a cluster. If automated snapshots are enabled for a cluster, 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. If no full backup policy is configured, a full backup is performed every 15 incremental backups. For how to configure an automated snapshot policy for a cluster, 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 7 days. The system deletes the snapshot at the end of the retention period. The retention period sets the duration for which users can access snapshots. If an incremental snapshot does not expire, both the incremental and full snapshots will be kept available instead of being deleted right away. Expired snapshots are hidden and can no longer be viewed by users. After all incremental snapshots expire, the hidden snapshots are physically deleted. If you want to keep an automated snapshot for a longer period, you can create a copy of it as a manual snapshot. The automated snapshot is retained until the end of the retention period, whereas the corresponding manual snapshot is retained until you manually delete it. For details about how to copy an automated snapshot, see **Copying Automated Snapshots**.

11.4.3.2 Configuring an Automated Snapshot Policy

You can choose the type of snapshot you need and configure either one or three automated snapshot policies at the cluster level, as well as one or 20 snapshot policies at the schema level for a cluster.

After an automated snapshot policy is enabled, the system automatically creates snapshots based on the time, period, and snapshot type you configured.

Procedure

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

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

Step 4 Click **Snapshots** and click **Policy List**. All policies of the current cluster are displayed on the **Policy List** page. Toggle on **Snapshot Policy**.

Step 5 (Optional) Click **Automated Snapshot** to enable the snapshot policy.

-  indicates that the policy is enabled (default). The default retention period is seven days.
-  indicates that the policy is disabled. Once disabled, snapshots will not be automatically created.

Step 6 Set the retention mode and the backup device used by the current cluster for automated snapshots. For more information, see **Table 11-12**.

Table 11-12 Automated snapshot parameters

Parameter	Description
Retention Days	Retention days of the snapshots that are automatically created. The value ranges from 1 to 31 days. **NOTE** Snapshots that are automatically created cannot be deleted manually. The system automatically deletes these snapshots when their retention duration exceeds the threshold.
Backup Device	Select **OBS** or **NFS** from the drop-down list.
NFS Backup File System Address (NFS)	NFS shared IP address. Enter the IP address of the SFS shared path. After the mounting is successful, a mount directory is created in the **/var/chroot/nfsbackup** directory of the cluster instance by default.

Step 7 After automated snapshot is enabled, you can configure its parameters. For more information, see **Table 11-13**.

- If you choose a cluster-level or schema-level full snapshot, you can either create a one-time snapshot or set up periodic snapshots. For schema-level full snapshots, you need to select the database associated with the schema. You can back up one or up to 20 schemas at a time.
  - Set a cluster-level or schema-level full periodic snapshot policy. You can specify a week or date and select a triggering time point.

    WARNING

    - Exercise caution when selecting the 29th, 30th, or 31st as end dates for a month, as this may result in missing data. The specific policy and execution are subject to the actual month and date.
    - The snapshot policy time is the local time.

  - Set a cluster-level or schema-level one-time full snapshot policy. Specify the desired date and time for the snapshot to be triggered.
- Incremental cluster-level or schema-level snapshots can be set only to **Periodic**.

  You can configure an incremental periodic snapshot policy at the cluster or schema level. Specify a target week or date, along with the trigger time, start time, and interval for the snapshot.

Table 11-13 Snapshot policy parameters

Parameter	Description
Snapshot Policy 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 (_).
Snapshot Level	The available options are **Cluster** and **Schema**.
Snapshot Type	You can choose either full or incremental snapshots. **NOTE** • A full snapshot is created after every fifteen incremental snapshots are created. • Incremental snapshot restoration is based on full snapshots. Incremental snapshots are used to restore all data to the time point when they were created. • An incremental snapshot records the changes made after the previous snapshot was created. A full snapshot backs up the data of an entire cluster. It takes a short time to create an incremental snapshot, and a long time to create a full snapshot. When restoring a snapshot to a new cluster, DWS uses all snapshots between the latest full backup and the current snapshot. • The following versions support automated incremental policies at the schema level: – 9.1.0.100 or later – 8.3.0.110 or later 8.3.0.*xxx* cluster versions – 8.2.1.230 or later 8.2.1.2*xx* versions • Schema-level policies, the policy level, database, and schema attributes cannot be modified after configuration.
Policy	You can choose either periodic or one-time snapshots. **NOTE** **One-time** can be selected only for full snapshots.
One-time	You can create a full snapshot at a specified time in the future. The local time is used.

Parameter	Description
Periodic Policy Configurations	You can create automated snapshots on a daily, weekly, or monthly basis: • **Days**: Specify days for every week or every month. **Weekly** and **Monthly** cannot be selected at the same time. For **Monthly**, the specified days are applicable only to months that contain the dates. For example, if you select **29**, no automated snapshot will be created on February, 2022. • **Time**: Specify the exact time on the selected days. For incremental snapshots, you can specify the start time and interval. The interval for cluster-level snapshots ranges from 4 to 24 hours, while for schema-level snapshots, it ranges from 1 to 24 hours. This setting determines the frequency at which snapshots will be taken. **NOTICE** Incremental snapshots can be set only to **Periodic**, as shown in the first figure below.

Step 8 Click **OK**.

Step 9 (Optional) To modify an automated snapshot policy, click **Modify** in the **Operation** column.

Step 10 (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 15 incremental snapshots.

NOTICE

Implementation of the same policy varies according to operations in the cluster. For example:

- If the automated snapshot function is disabled, the configured snapshot policy will not appear on the snapshot preview page.
- If the fine-grained snapshot function is disabled in the snapshot list, the schema-level automatic policy will not be shown on the preview page.
- The policy preview time is for your reference only. The cluster triggers a snapshot within one hour before and after the preset time.
- The next automated snapshots after cluster scale-out, upgrade, resize, and media modification are full snapshots by default.
- If a periodic policy is used for a cluster, no automatic backup is allowed within 4 hours after the last automated snapshot is complete.
- In the event of conflicting triggering times between multiple policies, the following priority order applies: cluster-level > schema-level, one-time > periodic, and full > incremental.
- You can use any backup, full or incremental, to restore the full data of a resource.
- When both the schema-level automatic incremental policy and the cluster-level full or incremental policies are in use, incremental schema snapshots will automatically be converted to full snapshots when certain conditions are met.

----End

11.4.3.3 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 DWS console.

Step 2 In the navigation pane, choose **Management** > **Snapshot**.

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

Step 3 In the **Operation** column of the snapshot that you want to copy, choose **More** > **Copy**.

- **New Snapshot Name**: Enter a new snapshot name.

  The snapshot name must be 4 to 64 characters in length and start with a letter. It is case-insensitive and contains only letters, digits, hyphens (-), and underscores (_).

- **Snapshot Description**: Enter the snapshot information.

  This parameter is optional. Snapshot information contains 0 to 256 characters and does not support the following special characters: !<>'=&"

Step 4 Click **OK**. The system starts to copy the snapshot for the cluster.

The system displays a message indicating that the snapshot is successfully copied and delivered. After the snapshot is copied, the status of the copied snapshot is **Available**.

NOTE

If the snapshot size is much greater than that of the data stored in the cluster, the data is possibly labeled with a deletion tag, but is not cleared and reclaimed. In this case, clear the data and recreate a snapshot. For details, see **How Can I Clear and Reclaim the DWS Storage Space?**

----End

11.4.3.4 Deleting an Automated Snapshot

Only DWS can delete automated snapshots; you cannot delete them manually.

DWS deletes an automated snapshot if:

- The retention period of the snapshot ends.
- The cluster is deleted.

CAUTION

To help users restore a cluster deleted by mistake, DWS provides the following policies (supported only in clusters of version 8.2.0 and later):

- If the latest snapshot is an automated snapshot, it will be retained for one day.
- If the latest snapshot is a manual snapshot, the automated snapshot of the cluster will be deleted.

11.4.4 Viewing Snapshot Information

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

Viewing Snapshot Information

Step 1 Log in to the DWS console.

Step 2 In the navigation pane, choose **Management** > **Snapshot**.

In the snapshot list, all snapshots are displayed by default.

Step 3 You can view the **Snapshot Name**, **Snapshot Status**, **Cluster Name**, **Backup Mode**, **Snapshot Type**, **Storage Media**, **Snapshot Level**, and creation time of snapshots.

You can also enter a snapshot name or cluster name in the upper right corner of the snapshot list and click to search for the specified snapshot. DWS supports fuzzy search.

Table 11-14 describes snapshot status.

Table 11-14 Snapshot status

Status	Description
Available	Indicates that the existing snapshot works properly.
Creating	Indicates that a snapshot is being created.
Unavailable	Indicates that the existing snapshot cannot provide services.

Table 11-15 lists the backup modes.

Table 11-15 Backup modes

Type	Description
Manual	Snapshot that you manually create through the DWS console or using APIs. You can delete the snapshots that are manually created.
Automated	Indicates the snapshot that is automatically created after the automated snapshot backup policy is enabled. You cannot delete the snapshots that are automatically created. The system automatically deletes the snapshots whose retention duration expires.

The following table describes the snapshot types.

Table 11-16 Type

Type	Description
Full	The snapshot is a full backup.
Incremental	The snapshot is an incremental backup.

The following table describes the snapshot media.

Table 11-17 Storage media

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

The following table describes the snapshot levels.

Table 11-18 Snapshot levels

Snapshot Level	Description
cluster	A backup of all the configurations and service data of a cluster at a specific point in time.
schema	A backup of all the service data of a schema at a specific point in time.

----End

Querying Snapshot Information by Table Name

Prerequisites

Only the snapshots that are created after the fine-grained snapshot function is enabled support fine-grained search.

NOTE

Only cluster versions 8.2.1.230 and later support snapshot query by table name.

Procedure

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

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

Step 4 In the navigation pane, choose **Snapshots**. Click the **Snapshots** tab and enable the fine-grained snapshot function. Click the advanced search button on the right.

You can set a triplet consisting of a database, a schema, and a table in a fine-grained query on the snapshot information in a specified database schema table. For details about the snapshot information, see **Step 3**.

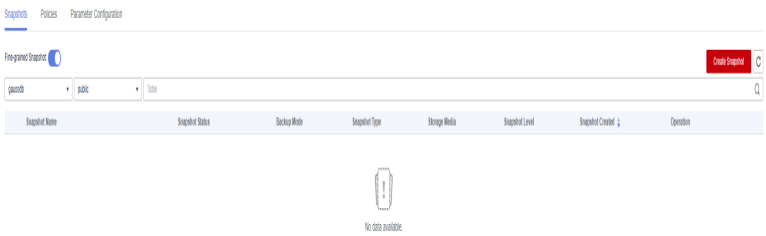


Table 11-19 Triplet description

Tuple Name	Description
database	Database name.

Tuple Name	Description
schema	Schema name.
table	Table name.

----End

11.4.5 Restoration Using a Snapshot

11.4.5.1 Constraints on Restoring a Snapshot

Cluster-Level Snapshot Restoration

Cluster-level restoration consists of two steps:

1. Data restoration: Restores data in the backup set to the data directory of each primary DN/CN instance in parallel.
2. Rebuilding the standby DN: After the primary DN is restored, standby DNs are rebuilt with full data in parallel.

NOTE

- The restoration process takes 1.5 to 2 times longer than the backup process.
- After a cluster-level restoration, the parameters will be identical to those during the backup. The new cluster must have the same specifications as the original cluster. If any changes were made to the specifications of the original cluster, the new cluster must still match the specifications prior to the changes. If the specifications of the new cluster are smaller, the restoration may fail.

11.4.5.2 Restoring a Snapshot to a New Cluster

Scenario

This section describes how to restore a snapshot to a new cluster when you want to check point-in-time snapshot data of the cluster.

When a snapshot is restored to a new cluster, the restoration time is determined by the amount of data backed up by the snapshot. If a snapshot contains a large amount of data, the restoration will be slow. A small snapshot can be quickly restored.

Automatic snapshots are incremental backups. When restoring a snapshot to a new cluster, DWS uses all snapshots between the latest full backup and the current snapshot. You can set the backup frequency. If snapshots are backed up only once a week, the backup will be slow if the incremental data volume is large. You are advised to increase the backup frequency.

NOTICE

- Currently, you can only use the snapshots stored in OBS to restore data to a new cluster.
- By default, the new cluster created during restoration has the same specifications and node quantity as the original cluster.
- Restoring data to a new cluster does not affect the services running in the original cluster.
- Fine-grained restoration does not support tables in absolute or relative tablespace.
- Logical clusters and resource pools cannot be restored 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 DWS console.

Step 2 Choose **Management** > **Snapshots**. Alternatively, in the cluster list, click the name of the target cluster to switch to the **Cluster Information** page. Then, click **Snapshots**. All snapshots are displayed by default.

Step 3 In the **Operation** column of a snapshot, click **Restore**.

Step 4 On the **Restore Snapshot** page, configure the parameters of the new cluster, as shown in the following figure.

- Restore to a single-AZ cluster.
- Restore to a multi-AZ cluster.

  NOTE

  - Only clusters later than 8.2.0.100 can be restored to a multi-AZ cluster.
  - This feature is only available for storage-compute coupled clusters.
  - The number of AZs in the current region is greater than or equal to 3.
  - The number of nodes and CNs must be a multiple of 3.
  - DNs in the multi-AZ cluster must be less than or equal to 2.

You can modify cluster parameters. For details, see **Table 11-20**. By default, other parameters are the same as those in the snapshot. For details, see **Table 11-13**.

Table 11-20 Parameters for the new cluster

Category	Operation
Basic settings	Configure the AZ, node flavor, cluster name, database port, VPC, subnet, security group, public access, and enterprise project.

Category	Operation
Advanced settings	If **Custom** is selected, configure the following parameters: • Backup devices: Select OBS or NFS from the drop-down list. • Label: a key-value pair used to identify a cluster. For details about labels, see **Overview**.

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

Viewing Restoration Details

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters**. By default, all clusters of the user are displayed.

Step 3 In the cluster list, if the cluster status is **Restoring**, click **View Details**.

Step 4 You can view the snapshot restoration progress of the cluster on the task details page.

NOTE

- The estimated duration in the task details is for reference only. The actual duration depends on the current data volume.
- In the restore phase, click **View** to view the kernel restoration process. Note that there may be a time gap between the task time displayed in the task details area and the actual kernel execution time due to task scheduling and restart.

----End

11.4.5.3 Restoring a Snapshot to the Current Cluster

Scenario

You can use a snapshot to restore data to the original cluster. This function is used when a cluster is faulty or data needs to be rolled back to a specified snapshot version.

NOTICE

- This function is supported only by clusters of version 8.1.3.200 or later.
- Snapshots whose backup device is OBS can be backed up.
- Only a snapshot in the **Available** state can be used for restoration.
- Logical clusters and resource pools cannot be restored to the current cluster.

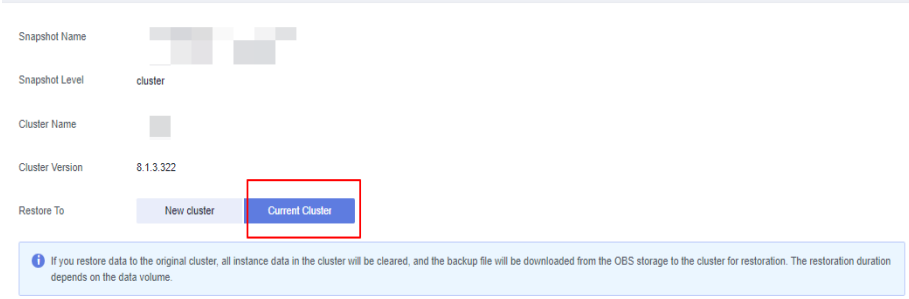
Procedure

Step 1 Log in to the DWS console.

Step 2 Choose **Management** > **Snapshots**. Alternatively, in the cluster list, click the name of the target cluster to switch to the **Cluster Information** page. Then, click **Snapshots**. All snapshots are displayed by default.

Step 3 In the **Operation** column of a snapshot, click **Restore**.

Step 4 Restore the snapshot to the current cluster.



NOTE

If you use a snapshot to restore data to the original cluster, the cluster will be unavailable during the restoration.

----End

11.4.5.4 Restoring a Table to the Original Cluster

Scenario

You can create a table from a cluster or schema snapshot to the original cluster if the table was modified or deleted by mistake.

NOTICE

- If the current console does not support this feature, contact technical support.
- Only the tables stored in OBS can be used to restore data to the original cluster.
- Currently, only cluster- and schema-level snapshots can be used for such restoration.
- Restoration can be performed only if the snapshot and the cluster are both in the **Available** state.
- A table in a read-only cluster cannot be restored.
- Fine-grained restoration does not support tables in an absolute or relative tablespace.

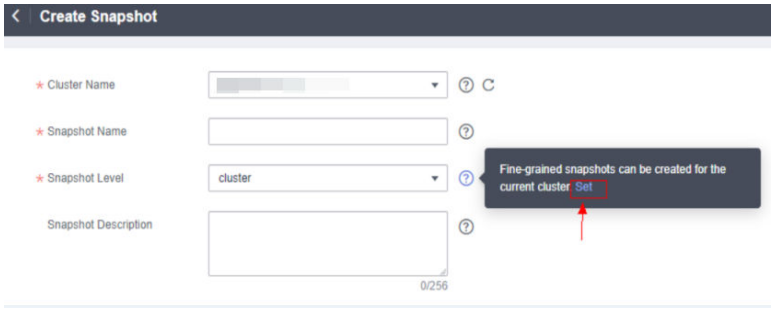
Prerequisites

Manually enable the fine-grained snapshot.

Step 1 Choose **Management** > **Snapshots**. Alternatively, in the cluster list, click the name of the target cluster to switch to the **Cluster Information** page. Then, click **Snapshots**.

Step 2 Click **Create Snapshot** in the upper right corner. Alternatively, choose **More** > **Create Snapshot** in the **Operation** column.

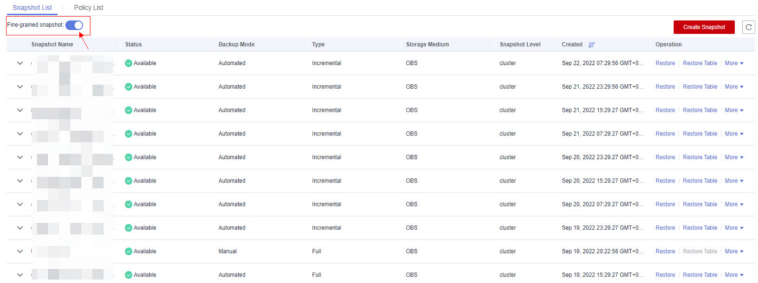
Step 3 Click  next to **Snapshot Level** and click **Set**.



Step 4 On the **Snapshot List** page, toggle the fine-grained snapshot switch.

 : enabled

 : disabled



NOTE

- If the fine-grained snapshot is enabled, you can restore specific tables from automatic or manual snapshots.

----End

Procedure

Step 1 Log in to the DWS console.

Step 2 In the navigation pane, choose **Management** > **Snapshot**.

Step 3 In the **Operation** column of a snapshot, click **Restore**.

Step 4 On the **Restore Table** page, configure the following parameters:

- **Database:** To restore a cluster snapshot, select a database. To restore a schema snapshot, select the database specified during backup. For details, see **Creating a Manual Snapshot of a Cluster** and **Creating a Manual Snapshot of a Schema**.
- **Source Schema**: Specify the schema of the table to be restored.
- **Source Table**: Specify the name of the table to be restored.
- **Destination Schema**: Specify the schema where the table is to be restored to.
- **Destination Table**: Specify the name of the new table.

CAUTION

- The table name can contain up to 63 characters and must start with a letter or underscore (_). Only letters (case-sensitive), digits underscores (_) are allowed.
- The source table to be restored must be a table in the backup set, or the restoration will fail.
- If the target table already exists in the database, this table will be overwritten during restoration. Check the table name before starting restoration.

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

----End

11.4.5.5 Restoring a Table or Multiple Tables to a New Cluster

Scenario

You can create a table from a cluster or schema snapshot to the original cluster. In case of accidental deletion or operation on data in a table during service operations, you can use this function to find the latest snapshot that contains the table data and restore it to a new cluster. Compare the data between the old and new clusters without affecting the original table data, and then restore the data as needed.

NOTICE

- This function is supported only by clusters of version 9.1.0 or later. The system supports OBS.
- You can restore fine-grained snapshots of clusters from an earlier version to a new cluster of version 9.1.0, even if the versions are different.
- You can restore the fine-grained snapshot of the 9.1.0 cluster to a new heterogeneous cluster of version 9.1.0, even if the number of nodes and specifications of the old and new clusters are different.
- Only fine-grained single-table or multi-table snapshots can be restored to a new cluster.

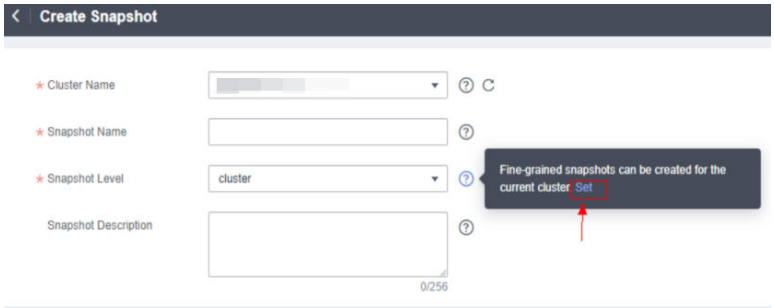
Prerequisites

Manually enable the fine-grained snapshot.

Step 1 Choose **Management** > **Snapshots**. Alternatively, in the cluster list, click the name of the target cluster to switch to the **Cluster Information** page. Then, click **Snapshots**.

Step 2 Click **Create Snapshot** in the upper right corner. Alternatively, choose **More** > **Create Snapshot** in the **Operation** column.

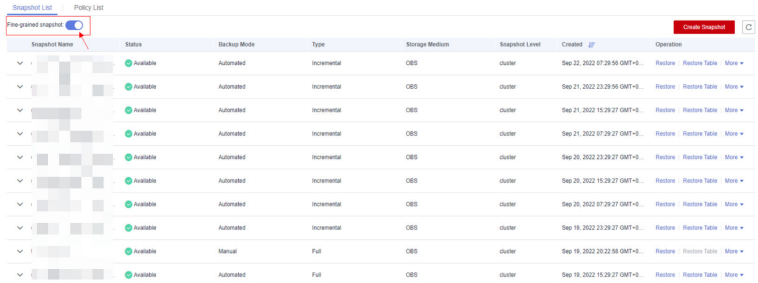
Step 3 Click  next to **Snapshot Level** and click **Set**.



Step 4 On the **Snapshot List** page, toggle the fine-grained snapshot switch.

 : enabled

 : disabled



NOTE

- If the fine-grained snapshot is enabled, you can restore specific tables from automatic or manual snapshots.

----End

Procedure

Step 1 Log in to the DWS console.

Step 2 Choose **Management** > **Snapshots**. Alternatively, in the cluster list, click the name of the target cluster to switch to the **Cluster Information** page. Then, click **Snapshots**. All snapshots are displayed by default.

Step 3 In the **Operation** column of a snapshot, click **Restore**.

Step 4 Set the recovery level to the table level.

Step 5 Select the basic information about the new cluster to be restored. For details, see **Creating a DWS Storage-Compute Coupled Cluster**.

NOTE

- If fine-grained heterogeneous recovery is available, you can select different node specifications and quantities for the new cluster, regardless of whether they match those of the original cluster.
- You need a cluster version of 9.1.0 or later to restore one or more tables to a new cluster.

Step 6 Select a single table or multiple tables. Select a database name from the drop-down list. If you select custom database configuration, you can adjust the following configuration parameters. If you select the default configuration, the parameters will use their default values. Once you've finished configuring, choose one or more tables in the table list to restore.

NOTE

When you restore data to a new cluster, a new database is created. If the configuration of the new database is not the same as the snapshot database, the restoration process may fail. Before restoring, make sure to review the configuration of the original database. If it differs from the default configuration, adjust it accordingly.

Table 11-21 Custom database parameters

Parameter	Description	Value Range	Default Value
Template Name	Name of the template from which the database is created. DWS creates a database by copying a database template. DWS has two initial template databases **template0** and **template1** and a default user database **gaussdb**.	Names of existing databases, **template0**, and **template1**	template0

Parameter	Description	Value Range	Default Value
Character Encoding	• Encoding format used by the new database. The value can be a string (for example, **SQL_ASCII**) or an integer. • By default, the encoding format of the template database is used. The encoding formats of the template databases **template0** and **template1** vary based on OS environments by default. – The **template1** database does not allow encoding customization. To specify encoding for a database when creating it, use **template0**. – To specify encoding, set **template** to **template0**.	Value range: **GBK**, **UTF8**, **Latin1**, and **SQL_ASCII**	SQL_ASCII
Character Set Support	Character set used by the new database. For example, this parameter can be set using **lc_collate = 'zh_CN.gbk'**. The use of this parameter affects the sort order applied to strings, for example, in queries with **ORDER BY**, as well as the order used in indexes on text columns. The default is to use the collation order of the template database.	Valid collation order	C
Character Classification	Character classification to use in the new database. For example, this parameter can be set using **lc_ctype = 'zh_CN.gbk'**. The use of this parameter affects the categorization of characters, for example, lower, upper and digit. The default is to use the character classification of the template database.	Valid character classification	C
Type	Compatible database type.	ORA, TD, and MySQL	ORA

Step 7 Click **Next: Confirm**.

Step 8 Confirm the information and click **Restore**.

----End

11.4.6 Configuring a Snapshot

You can configure the parameters for creating and restoring a snapshot.

NOTE

- This feature applies only to clusters of 8.2.0 or later. (For clusters of versions earlier than 8.2.0, only some parameters can be configured.)
- The parameters take effect on all the snapshot creation and restoration tasks.
- The thread-count configuration is available in cluster 9.1.0.210 and later versions. When Roach initiates a multi-threaded backup of cold and hot tables, and V3 tables, the default number of concurrent threads is set to twice the number of CPU cores.

Procedure

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

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

Step 4 Click the **Snapshots** tab page and click **Configure Parameters**. All the configurable parameters of the current cluster will be displayed.

Step 5 Configure parameters as required. For details, see **Table 11-22**.

Step 6 Click **Save**.

----End

Snapshot parameters

Table 11-22 Snapshot information

Parameter	Type	Description	Default Value
parallel-process	Backup parameter	Number of concurrent processes on each node during Roach backup. **NOTE** This parameter can be configured for clusters earlier than 8.2.0.	The value is the number of DNs on the current node.
compression-type	Backup parameter	Compression algorithm. • zlib • LZ4 **NOTE** This parameter can be configured for clusters earlier than 8.2.0.	LZ4

Parameter	Type	Description	Default Value
compression-level	Backup parameter	Compression level. The value range is 0 to 9. • **0**: fast backup and no compression • **9**: slow backup and maximum compression **NOTE** This parameter can be configured for clusters earlier than 8.2.0.	6
buffer-size	Backup parameter	Buffer size of the Roach upload media. The value range is 256 to 16,384, in MB.	256
buffer-block-size	Backup parameter	Data block size of the data file to be read by Roach. The value range is 5,242,880 to 268,435,456, in bytes.	67108864
cpu-cores	Backup parameter	Number of CPU cores that can be used when Roach starts multiple threads concurrently	1/2 of the total number of logical CPU cores on the node
master-timeout	Backup parameter	Timeout period for the communication between the Roach master and agent nodes. The value range is 600 to 3600, in seconds.	3600
max-backup-io-speed	Backup parameter	I/O flow control during Roach backup. The value range is 0 to 2048, in MB/s. The value must be greater than the value of **buffer-block-size**. The value **0** indicates no limit.	0
backup-mode	Backup parameter	Full backup mode. • **0**: phase-1 backup • **1**: phase-2 backup	0

Parameter	Type	Description	Default Value
cbm-parse-mode	Backup parameter	Incremental backup mode. • **0**: one-time CBM scan (high memory usage and high performance) • **1**: multiple CBM scans (stable memory usage and low performance)	0
thread-count	Backup parameter	Number of concurrent threads used when Roach starts multi-thread backup of cold and hot tables and V3 tables. The maximum value is 16 times the number of CPU cores. **NOTE** This parameter is supported only by clusters of version 9.1.0.210 or later.	The default value is twice the number of CPU cores.
dump-options	Backup parameter	Backup options supported during fine-grained backup. DWS support permission and comment backup. • Permission: **enable_handle_acl** • Comment: **enable_handle_comment** **NOTE** This parameter is supported only by clusters of version 9.1.0 or later.	Do not back up permissions and comments.
parallel-process	Restoration parameter	Number of concurrent processes on each node during Roach backup. By default, the value is the number of primary DNs on the current node plus 1.	1
cpu-cores	Restoration parameter	Number of CPU cores that can be used when Roach starts multiple threads concurrently	The default value is 1/2 of the number of CPU cores.

Parameter	Type	Description	Default Value
logging-level	Restoration parameter	Log levels: • **FATAL**: Unrecoverable faults that cause the system suspension. This is the most severe level. • **ERROR**: Major errors. • **WARNING**: Exceptions. In this case, the system may continue to process tasks. • **INFO**: Notes. • **DEBUG**: Debugging details. • **DEBUG2**: Detailed debugging information, which is generally not displayed. This is the least severe level.	INFO
thread-count	Restoration parameter	Number of concurrent threads used when Roach starts multi-thread restoration of cold and hot tables and V3 tables. The maximum value is 16 times the number of CPU cores. **NOTE** This parameter is supported only by clusters of version 9.1.0.210 or later.	The default value is twice the number of CPU cores.

11.4.7 Stopping Snapshot Creation

You can stop snapshot creation on the **Snapshots** page.

NOTE

- This feature is supported only in version 8.1.3.200 and later.
- If the snapshot is ready to complete, the command for stopping the snapshot will not take effect and the snapshot will end normally.

Precautions

Only the snapshots in the **Creating** state can be stopped. A snapshot creation task that just started or is about to complete cannot be stopped.

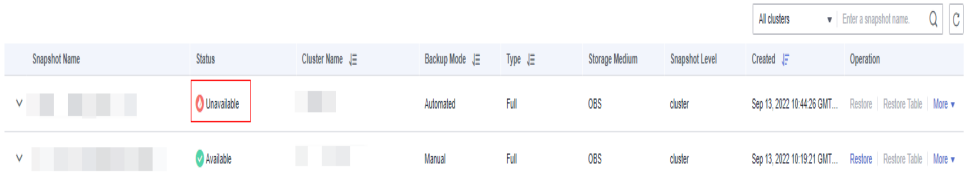
Procedure

Step 1 Log in to the DWS console.

Step 2 Choose **Management** > **Snapshots**. Alternatively, in the cluster list, click the name of the target cluster to switch to the **Cluster Information** page. Then, click **Snapshots**. All snapshots are displayed by default.

Step 3 In the **Operation** column of a snapshot that is being created, and click **Cancel Creation**.

Step 4 In the dialog box that is displayed, click **Yes** to stop the snapshot. The snapshot state will change to **Unavailable**.



----End

11.5 Scaling DWS Cluster Nodes

11.5.1 Viewing Inspection Results

Context

DWS allows you to inspect the cluster before making any changes like scaling, changing specifications, or upgrading. Simply click **Inspect** on the relevant page, and the system will check if the cluster's health status and metrics meet the requirements for the change. Once the inspection is passed, you can proceed with the change. If the inspection fails, you can view the inspection details page to see which items did not pass the inspection. From there, you can handle the inspection items based on the details provided. For details about the inspection standards, see **Table 11-23**.

NOTE

- This feature is supported only in cluster version 8.1.1 or later.
- If you cannot handle the failed inspection items, contact technical support engineers.

Precautions

- The inspection plug-in 8.3.1.100 or later has been installed in the cluster.
- The inspection result is valid for 24 hours, during which you can make the change operation. Once the 24-hour validity period expires, you will need to perform the inspection again.
- If the cluster has not been inspected within 24 hours before the change, inspect it before making any changes like scaling, changing specifications, or upgrading. Ensure that the inspection is passed before proceeding with the change.

Viewing Inspection Details

Step 1 Log in to the DWS console.

Step 2 In the cluster list, click the name of the target cluster.

Step 3 On the cluster details page, click **Inspection Management**.

Step 4 Click the drop-down button next to its name to check the inspection status, execution progress, inspection result, and pass rate. For more details about these inspection items, click **View Details** in the task's row.

NOTE

After creating an inspection task on the configuration change page, you can keep track of its progress and view details. You can even stop the inspection from that same page.

----End

Stopping an Inspection Task

Step 1 Log in to the DWS console.

Step 2 In the cluster list, click the name of the target cluster.

Step 3 On the cluster details page, click **Inspection Management**.

Step 4 Locate the row that contains the inspection task and click **Stop** in the **Operation** column to stop the inspection task.

----End

Inspection Criteria

Table 11-23 Inspection criteria

Change Operation	Item	Check Criteria
Scaling operations and typical modifications	CheckTimeZone	The inspection passes if all nodes in the cluster use the same time zone, and fails if they do not.
	CheckSpaceUsage	When usage goes beyond the warning threshold (set at 70% by default), a warning is issued. If it goes beyond the NG threshold (set at 90% by default), the inspection fails. The inspection fails if the available space in the **GAUSSHOME/PGHOST/GPHOME/GAUSSLOG/tmp/data** directory is less than the threshold.

Change Operation	Item	Check Criteria
	CheckClusterState	It verifies the CM process, fenced UDF, and cluster status. If the CM process is missing, the inspection fails. A warning is issued if the fenced UDF status is **down**. The inspection passes for a normal cluster status, but fails otherwise.
	CheckEnvProfile	It checks the environment variables (**$GAUSSHOME**, **$LD_LIBRARY_PATH**, and **$PATH**) of a node. If these variables exist and are properly configured, the inspection passes. Otherwise, the inspection fails.
	CheckReadonlyMode	It checks the value of **default_transaction_read_only** on all nodes that contain CNs in the cluster. If the value is **off**, the inspection passes. Otherwise, the inspection fails.
	CheckCatchup	It checks whether the **CatchupMain** function can be found in the **gaussdb** process stack. If it cannot be found, the inspection passes. Otherwise, the inspection fails.
	CheckCollector	It checks whether the information collection is successful. If it is, the inspection passes. Otherwise, the inspection fails.
	CheckTrust	If any node is not trusted, the inspection fails. Otherwise, the inspection passes.
	CheckBalanceState	It checks the **Balanced** attribute of the cluster. If it is **Yes**, the inspection passes. A warning is displayed if it is not. If the query fails, the inspection also fails.
	CheckCnNumberSame	It checks if the number of CNs queried by running the **/opt/dws/xml/cluster.xml** command is different from that queried by running the **cm_ctl query -Cv** command. If they are different, the inspection passes. If not, the inspection fails.

Change Operation	Item	Check Criteria
	CheckCMParam	It checks if the value of **enable_transaction_read_only** is **on** and if the value of **coordinator_heartbeat_timeout** is consistent on each node. If both are true, the inspection passes.
	CheckUtilslib	It checks for the existence of the **$GAUSSHOME/utilslib** directory. If it exists, the inspection fails. If it does not exist, the inspection passes.
	CheckPgxcgroup	It checks the number of records in the **pgxc_group** table where **in_redistribution** is **Y**. If the number is **0**, the inspection passes. If the number is greater than 0, the inspection fails.
	CheckLockState	It checks whether the cluster is locked. If the cluster is not locked, the inspection passes. If it is locked, the inspection fails.
	CheckDBConnection	It checks whether the database can be connected. If it can, the inspection passes. Otherwise, the inspection fails.
	CheckGUCConsistent	It checks whether the GUC parameters of CNs and DNs are consistent. If they are consistent, the inspection passes. Otherwise, the inspection fails.
	CheckTDDate	The inspection fails if the ORC table in the TD database exists and has columns of the date type.
	CheckPgxcRedistb	If any temporary table remains in the database after data redistribution, the inspection fails.
	CheckMetaData	If metadata in the system table is consistent, the inspection passes. Otherwise, the inspection fails.
	CheckGUCSetting	If the GUC parameters in **postgresql.conf** are consistent with those in **pg_settings**, the inspection passes. Otherwise, the inspection fails.
	CheckProacl	The inspection fails if **proacl** in the **pg_proc** system table has usernames with only digits. Otherwise, it passes.

Change Operation	Item	Check Criteria
	CheckMetaDataConsistency	If the data in the system table between CN and DN is consistent, the inspection passes. Otherwise, the inspection fails.
	CheckReturnType	If the return value is invalid, the inspection fails. Otherwise, the inspection passes.
	CheckUltraWideTable	If a table with more than 996 columns exists, the inspection fails. Otherwise, the inspection passes.
	CheckDataRedisSchema	If a **data_redis** schema exists in the database and the owner name is not **redisuser**, the inspection fails. Otherwise, the inspection passes.
	CheckDiskSpaceLimited	If the disk space of the user is limited, the inspection fails. Otherwise, the inspection passes.
	CheckTableCollate	The inspection fails if the database has a PCK table or a column-store partitioned table with a **collate** field. Otherwise, it passes.
	CheckDefaultOrientation	It checks the GUC parameter. If both the database and **default_orientation** are set to row storage, the inspection passes. Otherwise, the inspection fails.
	CheckReplicatio-nUuid	The inspection passes if there is no replicated table that uses the default UUID. Otherwise, it fails.
	CheckUserStatementTimeout	If **statement_timeout** is not set or is set to **0**, the inspection passes. Otherwise, the inspection fails.
	CheckJsonb	Run the **select attrelid::regclass from pg_attribute a join pg_type t on a.atttypid = t.oid and t.typname = 'jsonb' group by 1** SQL statement. If the **jsonb** type is used, the inspection fails. Otherwise, the inspection passes.
	CheckLengthOfIndex	Run the **SELECT length(pg_get_indexdef(indexrelid)) FROM pg_index order by 1 desc limit 1** SQL statement. If the result is greater than 192 x 1024, the inspection fails. Otherwise, the inspection passes.

Change Operation	Item	Check Criteria
	CheckLengthOfTable	Run the **select c.oid from pg_class c,pg_namespace n where c.relnamespace=n.oid and relkind='r' and n.nspname not in ('cstore') and length(n.nspname||'.'||c.relname)>=64;** SQL statement. If the result is not empty, the inspection fails. Otherwise, the inspection passes.
	CheckUseWorkloadManager	Run the **show use_workload_manager** SQL statement. If the result is **on**, the inspection passes. Otherwise, the inspection fails. This inspection item is not included in version 8.1.3.320 and beyond. Meaning, it has been verified and passed in later versions.
	CheckNecessarySchema	It checks whether the necessary **schema(public)** exists. If **schema(public)** does not exist, the check fails.
	CheckCMParamConsistency	To pass this check, ensure that the parameter settings in the **cm.conf** file obtained from both the active and standby CM nodes are consistent. The check will pass if they are consistent and fail if they are not.
	CheckSQLCompatibility	In MySQL compatibility mode, redistribution of temporary tables with indexes is slow. If the **SHOW sql_compatibility** value is set to **mysql** in the service database and the **disable_including_all_mysql** enumerated value is not present in the **behavior_compat_options**, this item will not pass the check. However, if the enumerated value is included, the item will pass the check.
	CheckBinaryUpgrade	Check whether the corresponding backup file exists in the **/DWS/manager/upgrade_backup/**directory. If the backup file exists, the check fails. Otherwise, the check is passed.
	CheckColdTableSpace	Checks whether cold and hot tables exist in all databases. If yes, this item fails the check. If no, this item passes the check.

Change Operation	Item	Check Criteria
	CheckXFS	View the **/etc/os-release** file to obtain the version information. If EulerOS is used and the version is 4.19.87 or earlier, XFS bugs are involved and the check is not passed. Otherwise, the check is passed.
	CheckGTMConfigConsistency	It obtains the parameters in the configuration files of the active and standby GTMs. If the parameter settings are consistent, this item passes the check.
	CheckColversion	If there are column-store tables that are not marked as 1.0 and the current default column-store table is 2.0, this item fails the check. Otherwise, this item passes the check.
	CheckTopSqlSize	It checks the size of the **topsql** table. If the size exceeds 50 GB, the check fails.
	CheckDeltaTable	It checks the existence of the **delta** table. If the table exists, this item fails the check.
	CheckMaxDatanode	It checks the value of **comm_max_datanode**. If the value is not equal to the actual number of primary DataNodes, this item fails the check.
	CheckSSHIP	To pass this check, ensure that the parameter settings in the **cm.conf** file obtained from both the active and standby CM nodes are consistent. The check will pass if they are consistent and fail if they are not.
	CheckTimeZoneLink	Run the **ll /etc/localtime** command in the sandbox. If the file path to which the link points contains **/var/chroot**, the check is not passed.
	CheckSpecialFile	Checks whether files in the program directory **(GAUSSHOME)** contain special characters and files of non-Ruby users. If no, this item passes the check.
	CheckSysSchemaTable	If a user-created table exists in the system schema, the check fails. Otherwise, the check is successful.

Change Operation	Item	Check Criteria
Pre-upgrade health check	CheckClusterParams	The cluster configuration parameters (IP address, port, and path parameters) specified in **postgresql.conf** or **pgxc_node** should match those in the static configuration file. Otherwise, the inspection fails.
	CheckCNNum	It checks the number of CNs in the cluster. If the number is greater than 2 and no more 10, the inspection passes. Otherwise, the inspection fails.
	CheckDDL	Start a transaction to delete schemas and tables. If the transaction can be committed, the inspection passes. Otherwise, the inspection fails.
	CheckTimeZone	The inspection passes if all nodes in the cluster use the same time zone, and fails if they do not.
	CheckXidEpoch	It checks the XID consumption. If the value is greater than or equal to 2 to the power of 32, the inspection fails.
	CheckCnNumberSame	It checks if the number of CNs queried by running the **/opt/dws/xml/cluster.xml** command is different from that queried by running the **cm_ctl query -Cv** command. If they are different, the inspection passes. If not, the inspection fails.
	CheckGaussVer	The inspection passes if the binary files in the **$GAUSSHOME/bin** directory on each node have identical versions.
	CheckPsort	The inspection fails if the **psort** index exists.
	CheckCatchup	It checks whether the **CatchupMain** function can be found in the **gaussdb** process stack. If it cannot be found, the inspection passes. Otherwise, the inspection fails.
	CheckClusterState	It verifies the CM process, fenced UDF, and cluster status. If the CM process is missing, the inspection fails. A warning is issued if the fenced UDF status is **down**. The inspection passes for a normal cluster status, but fails otherwise.

Change Operation	Item	Check Criteria
	CheckMetaDataConsistency	If the data in the system table between CN and DN is consistent, the inspection passes. Otherwise, the inspection fails.
	CheckDependSystemObj	If self-created objects depend on system objects, the inspection fails. Otherwise, the inspection passes.
	CheckPgKeyWords	If names of tables, columns, functions, or data types are new reserved keywords, the inspection fails. Otherwise, the inspection passes.
	CheckReadonlyMode	It checks the value of **default_transaction_read_only** on all nodes that contain CNs in the cluster. If the value is **off**, the inspection passes. Otherwise, the inspection fails.
	CheckMetaData	If metadata in the system table is consistent, the inspection passes. Otherwise, the inspection fails.
	CheckGUCSetting	If the GUC parameters in **postgresql.conf** are consistent with those in **pg_settings**, the inspection passes. Otherwise, the inspection fails.
	CheckPgxcgroup	It checks the number of records in the **pgxc_group** table where **in_redistribution** is **Y**. If the number is **0**, the inspection passes. If the number is greater than 0, the inspection fails.
	CheckCmserverStandby	If the value of **cm_server** in the cluster is **standby**, the inspection passes. Otherwise, a warning is displayed.
	CheckSpaceUsage	When usage goes beyond the warning threshold (set at 70% by default), a warning is issued. If it goes beyond the NG threshold (set at 90% by default), the inspection fails. The inspection fails if the available space in the **GAUSSHOME/PGHOST/GPHOME/GAUSSLOG/tmp/data** directory is less than the threshold.

Change Operation	Item	Check Criteria
	CheckEnvProfile	It checks the environment variables (**$GAUSSHOME**, **$LD_LIBRARY_PATH**, and **$PATH**) of a node. If these variables exist and are properly configured, the inspection passes. Otherwise, the inspection fails.
	CheckBalanceState	It checks the **Balanced** attribute of the cluster. If it is **Yes**, the inspection passes. A warning is displayed if it is not. If the query fails, the inspection also fails.
	CheckTDDate	The inspection fails if the ORC table in the TD database exists and has columns of the date type.
	CheckCatalog	If there is a custom database object in **pg_catalog**, the inspection fails. Otherwise, the inspection passes.
	CheckPgauthid	If the most significant bit of **oid** in **pg_authid** is **1**, the inspection fails. Otherwise, the inspection passes.
	CheckSysdate	If the **sysdate** view is used in tables, views, and stored procedures, the inspection fails. Otherwise, the inspection passes.
	CheckFilesNumber	If the number of **tmp** files in the **GAUSSHOME/PGHOST/GPHOME** directory is greater than 10,000, the inspection fails. Otherwise, the inspection passes.
	CheckKeyFilesExist	If the **upgrade_version**, **conf**, **control**, and **data** files exist in key directories, the inspection passes. Otherwise, the inspection fails.
	CheckReturnType	If the return value is invalid, the inspection fails. Otherwise, the inspection passes.
	CheckTrust	If any node is not trusted, the inspection fails. Otherwise, the inspection passes.
	CheckEnumGUCValue	It checks whether some parameters in **pg_postgres.conf** contain quotation marks. If single quotation marks are missing, this item fails the check.

Change Operation	Item	Check Criteria
	CheckSpecialFile	It checks whether files in the program directory **(GAUSSHOME)** contain special characters and files of non-Ruby users. If no, this item passes the check.
	CheckNecessarySchema	It checks the existence of the necessary **schemas (public)**.
	CheckUserDefinedDataType	It connects to all databases. Run the **select count(*) from pg_type t,pg_namespace n,PG_ATTRIBUTE a where t.typnamespace=n.oid and t.oid=a.atttypid and t.typname ='time_stamp' and n.nspname='information_schema' and a.atttypid> 16384;** SQL statement . If the result is empty, the check is passed.
	CheckCMParamConsistency	To pass this check, ensure that the parameter settings in the **cm.conf** file obtained from both the active and standby CM nodes are consistent. The check will pass if they are consistent and fail if they are not.
	CheckLightProxy	It checks the **enable_light_proxy** parameter. If the value is **off** and **behavior_compat_options** does not contain the enumerated value **enable_force_add_batch**, the item fails the check.
	CheckSSHIP	To pass this check, ensure that the parameter settings in the **cm.conf** file obtained from both the active and standby CM nodes are consistent. The check will pass if they are consistent and fail if they are not.
	CheckSysSchemaTable	If a user-created table exists in the system schema, the check fails. Otherwise, the check is successful.
	CheckTimeZoneLink	Run the **ll /etc/localtime** command in the sandbox. If the file path to which the link points contains **/var/chroot**, the check is not passed.

11.5.2 Managing Nodes

Overview

On the **Nodes** tab page, you can view the node list of the current cluster, add new nodes to or remove nodes from it, and view the node usage, status, flavor, and AZ.

In addition, you can click the  icon next to the text in the **Node Alias Name** column of a specified node to modify the alias of the node. If the node does not have an alias, you can add an alias for the node.

NOTE

- This feature is supported only in 8.1.1.200 or later cluster versions.

Adding Nodes

This function is more suited for large-scale scale-out. Nodes can be added in batches in advance without interrupting services. To add 180 nodes, add them in three batches of 60 nodes each. If any nodes fail to be added, retry adding them. Once all 180 nodes are added, use them for scaling out.

Precautions

- Nodes can be added only when no other task is running on the management side.
- The storage size of a new node must be the same as that of each of the existing nodes in the cluster.
- An added node, typically for scaling out, is referred to as an idle node. You are advised to only add nodes when needed and promptly use them for scaling out.
- The anti-affinity rule dictates that the number of nodes to be added at a time must be an integer multiple of the cluster ring size. For example, if the cluster ring size is 3, the number of nodes to be added must be an integer multiple of 3.
- In the anti-affinity deployment mode, when a node is idle and fails due to power-off or other causes, it makes other nodes in its server group unavailable. In this case, you should remove and re-add the failed node.
- The anti-affinity rule dictates that, if a node fails to be added and is rolled back, other nodes that are being added in the same server group will also be rolled back.

Procedure

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters**. All clusters are displayed by default.

Step 3 Click the name of the target cluster. On the **Cluster Information** page that is displayed, choose **Nodes**.

Step 4 Click **Add Node**, enter the number of idle nodes to be added, and click **Next: Confirm**. If there are not enough IP addresses in the original subnet, you can add idle nodes from other subnets.

Step 5 After confirming that the information is correct, click **Submit**. The **Nodes** page is displayed. On this page, you can start adding nodes. Nodes that fail to be added are automatically rolled back and recorded in the failure list.

----End

Removing Nodes

Precautions

- Nodes can be removed only when no other task is running on the management side.
- Only nodes whose resource status is **Idle** can be removed. Nodes that are in use cannot be removed.
- In anti-affinity deployment, nodes are removed by cluster ring. For example, when you remove a node, other nodes in the same ring will be automatically selected and displayed.

Procedure

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters**. All clusters are displayed by default.

Step 3 Click the name of the target cluster. On the **Cluster Information** page that is displayed, choose **Nodes**.

Step 4 On the **Nodes** page, select the node to be deleted and click **Delete**.

Step 5 Confirm the information and click **OK**. After the deletion is successful, the node is no longer displayed on the **Nodes** page.

----End

11.5.3 Scaling Nodes

11.5.3.1 Scaling Out a Cluster

When you need more compute and storage resources, add more nodes for cluster scale-out on the management console.

NOTE

- When you scale out a storage-compute coupled data warehouse cluster, use the same storage specifications as the cluster.
- If the number of subnet IP addresses is insufficient, cross-subnet scale-out is allowed.

After the data in a DWS data warehouse is deleted, the occupied disk space may not be released, resulting in dirty data and disk waste. Therefore, if you need to scale out your cluster due to insufficient storage capacity, run the **VACUUM** command to reclaim the storage space first. If the used storage capacity is still high after you run the **VACUUM** command, you can scale out your cluster. For details about the VACUUM syntax, see "SQL Syntax Reference" > "DDL Syntax" > "VACUUM" in the *Data Warehouse Service (DWS) SQL Syntax Reference*.

Impact on the System

- Before the scale-out, disable the client connections that have created temporary tables because temporary tables created before or during the scale-out will become invalid and operations performed on these temporary tables will fail. Temporary tables created after the scale-out will not be affected.
- Certain cluster functions, including restarting, stopping, and starting, modifying specifications, adding or removing CNs, creating snapshots, and resetting the database administrator's password, cannot be performed while scaling out the cluster.
- During an offline scale-out, the cluster automatically restarts. Therefore, the cluster changes to **Unavailable** for a period of time. After the cluster is restarted, the status becomes **Available**. At the end of the scale-out, if you select automatic redistribution, the system dynamically redistributes user data in the cluster to all nodes. Otherwise, you need to start data redistribution.
- During offline scale-out, stop all services or run only a few query statements. During table redistribution, a shared lock is added to tables. All insert, update, and delete operations as well as DDL operations on the tables are blocked for a long time, which may cause a lock wait timeout. After a table is redistributed, you can access the table. Do not perform queries that take more than 20 minutes during the redistribution (the default time for applying for the write lock during redistribution is 20 minutes). Otherwise, data redistribution may fail due to lock wait timeout.
- If a new snapshot is created for the cluster after the scale-out, the new snapshot contains data on the newly added nodes.
- If the cluster scale-out fails, the database automatically performs the rollback operation in the background so that the number of nodes in the cluster can be restored to that before the scale-out.
  - If the rollback is successful and the cluster can be normally used, you can perform **Scale Out** again. If the scale-out still fails, contact the technical support.
  - If the rollback fails due to some exceptions, the cluster may become **Unavailable**. In this case, you cannot perform **Scale Out** or restart the cluster. Contact the technical support.

Prerequisites

- The cluster to be scaled out is in the **Available**, **Read-only**, 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.
- The cluster will be intermittently disconnected during scale-out. Exercise caution when performing this operation.
- To ensure data security, you are advised to create a snapshot before the scale-out. For details about how to create a snapshot, see **Manual Snapshots**.

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters**.

All clusters are displayed by default.

Step 3 In the **Operation** column of the target cluster, choose **More** > **Scale Node** > **Scale Out**. The scale-out page is displayed.

Before scaling out the cluster, it is crucial to verify if it meets the inspection conditions. Click **Immediate Inspection** to complete the inspection and proceed to the next step only if it passes. For more information, see **Viewing Inspection Results**.

- If the IP addresses of the original subnet are insufficient, you can expand the capacity across subnets.

Step 4 Specify the number of nodes to be added.

- DNs are added during scale-out. For details about how to add CNs, see **Adding or Deleting a CN in a DWS Cluster**.
- 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 and security group of the cluster with new nodes added are the same as those of the original cluster.

Step 5 Configure advanced parameters.

- If you choose **Default**, **Auto Redistribution** will be enabled and **Redistribution Mode** will be **Offline** by default.
- If you choose **Custom**, you can configure the following advanced configuration parameters for scale-out:
  - **Auto Redistribution**: Automatic redistribution can be enabled. If automatic redistribution is enabled, data will be redistributed immediately after the scale-out is complete. If this function is disabled, only the scale-out is performed. In this case, to redistribute data, select a cluster and choose **More** > **Scale Node** > **Redistribute**.
  - **Redistribution Concurrency**: If automatic redistribution is enabled, you can set the number of concurrent redistribution tasks. The value range is 1 to 200. The default value is **4**.

  - **Auto Redistribution**: Select **Offline**.

Step 6 Confirm the settings, select the confirmation check box, and 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**.
- After the scale-out is complete, 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

Scaling Out with Idle Nodes

To ensure reliability, prepare ECS first by referring to **Adding Nodes** for a large-scale cluster, and scale out the cluster using idle nodes.

NOTE

- Disable automatic redistribution when you scale out a large-scale cluster to facilitate retries upon failures for improved reliability.
- After the scale-out is complete, manually perform **redistribution** to ensure that multiple retries can be performed in this phase.

Precautions

- A number of available nodes must be added to the cluster in advance so that idle nodes can be created and added for scale-out.
- The anti-affinity rule dictates that the number of idle nodes to be added must be an integer multiple of the cluster ring size.
- Make sure to configure the scale-out task before submitting it. This involves completing the scale-out preparation. Once done, wait for a moment.

Procedure

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters**. All clusters are displayed by default.

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

Before scaling out the cluster, it is crucial to verify if it meets the inspection conditions. Click **Immediate Inspection** to complete the inspection and proceed to the next step only if it passes. For more information, see **Viewing Inspection Results**.

If there are idle nodes in the cluster, the system displays a message asking you whether to add nodes.

Step 4 Click the corresponding button to make scale-out preparations and wait until the preparation is complete.

Step 5 Configure the parameters as required. For details, see **Scaling Out a Cluster**.

After setting the scale-out and redistribution parameters, select the confirmation check box, and click **Next: Confirm**.

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

----End

Viewing Scaling Details

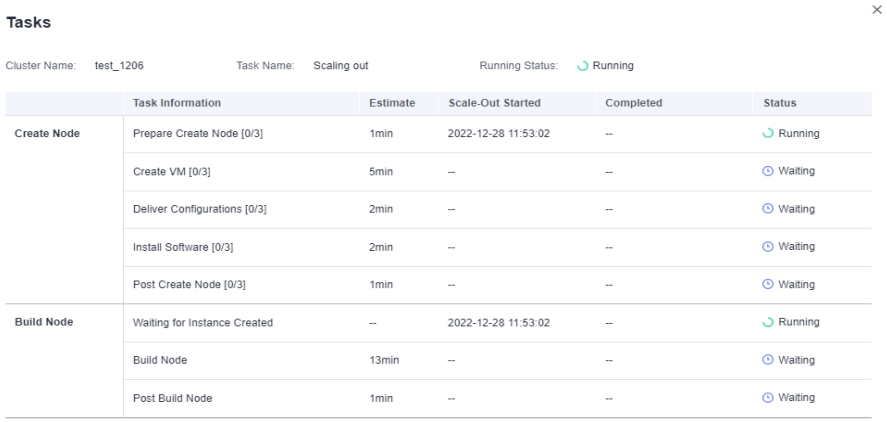
Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters**. By default, all clusters of the user are displayed.

Step 3 In the **Task Information** column of a cluster, click **View Details**.

Step 4 Check the scale-out status of the cluster on the scaling details page.

Figure 11-2 Viewing scale-out details



----End

11.5.3.2 Cluster Redistribution

11.5.3.2.1 Redistributing Data

Data redistribution, where data in existing nodes is evenly allocated to the new nodes after you scale out a cluster, is a time-consuming yet crucial task that accelerates service response.

By default, redistribution is automatically started after cluster scale-out. For enhanced reliability, disable the automatic redistribution function and manually start a redistribution task after the scale-out is successful. In this way, both scale-out and redistribution can be retried upon failures.

DWS supports **offline redistribution**.

Before redistribution starts or when redistribution is paused, you can set redistribution priorities for the tables that have not been redistributed by schema or table.

NOTICE

- The cluster redistribution function is supported in 8.1.1.200 or later cluster versions.
- This function can be manually enabled only when the cluster task information displays **To be redistributed** after scale-out.
- You can also select the redistribution mode when you configure cluster scale-out (see **Configure advanced parameters**).
- Redistribution queues are sorted based on the relpage size of tables. To ensure that the relpage size is correct, you are advised to perform the **ANALYZE** operation on the tables to be redistributed.

Offline Redistribution

Precautions

- In offline redistribution mode, the database does not support DDL and DCL operations. Tables that are being redistributed support only simple DQL operations.
- During table redistribution, a shared lock is added to tables. All insert, update, and delete operations as well as DDL operations on the tables are blocked for a long time, which may cause a lock wait timeout. Do not perform queries that take more than 20 minutes during the redistribution (the default time for applying for the write lock during redistribution is 20 minutes). Otherwise, data redistribution may fail due to lock wait timeout.

Procedure

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters**. All clusters are displayed by default.

Step 3 In the **Operation** column of the target cluster, choose **More** > **Scale Node** > **Redistribute**, as shown in the following figure.

The **Redistribution** page is displayed.

Step 4 On the **Redistribute** page that is displayed, keep the default **offline** redistribution mode and click **Next: Confirm** to submit the task.

----End

11.5.3.2.2 Viewing Redistribution Details

On the **View Redistribution Details** page, you can check the monitoring information, including the redistribution mode, redistribution progress, and table redistribution details of the current cluster. You can pause and resume redistribution, set the redistribution priority, and change the number of concurrent redistribution tasks.

> **NOTE**
>
> The function of viewing redistribution details is supported by 8.1.1.200 and later cluster versions. Details about the data table redistribution progress are supported only by 8.2.1 and later cluster versions.

Precautions

You can check redistribution details only if the cluster is being redistributed, failed to be redistributed, or is suspended. There may be a delay in the statistics update.

Procedure

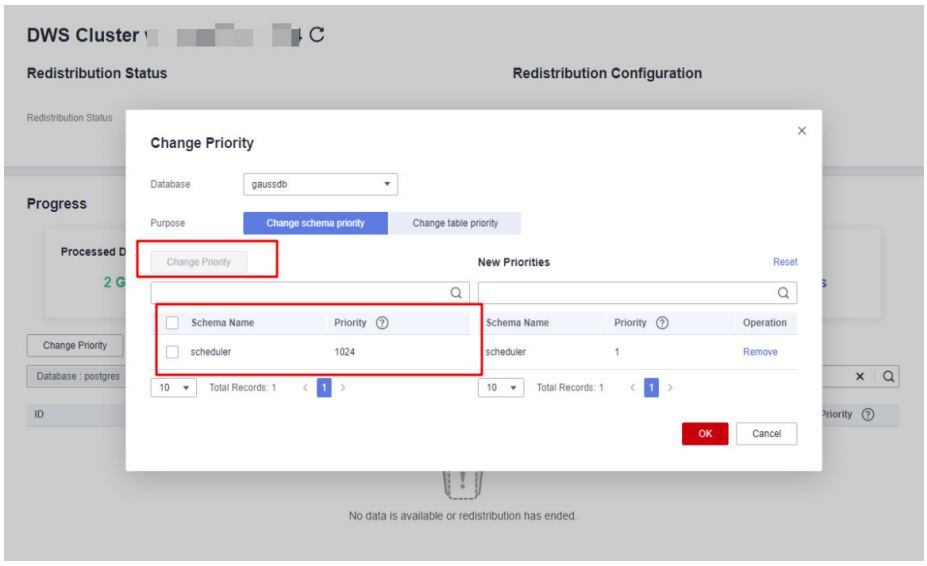
Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters**. All clusters are displayed by default.

Step 3 In the **Task Information** column of a cluster, click **View Details**.

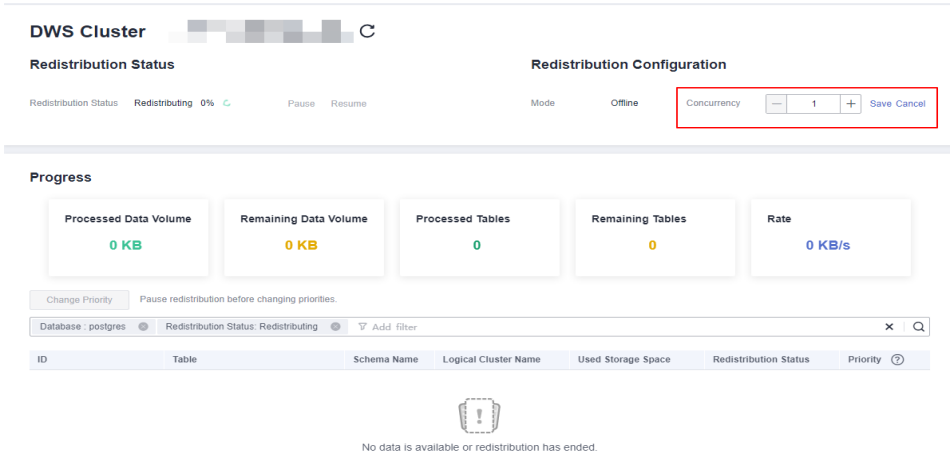
Step 4 Check the redistribution status, configuration, progress, and redistribution details of all the tables in a specified database. Specify a database that and can be searched by table redistribution status and table name. If all the tables in a database have completed redistribution, no data will be displayed for the database.

Step 5 When redistribution is paused, you can set the redistribution priority (in schema or table dimension), and redistribution will be performed based on the configured redistribution sequence. You can also set the redistribution priority before the redistribution starts.

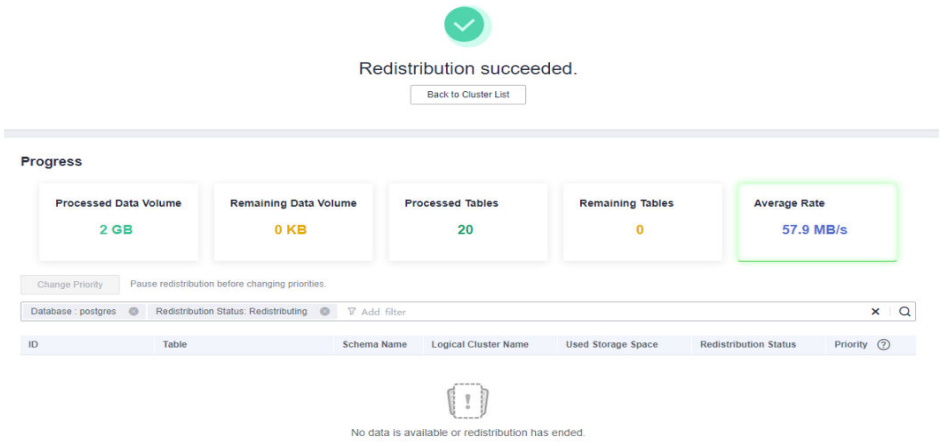


Step 6 The number of concurrent redistribution tasks can be adjusted during redistribution.

> **NOTE**
>
> Cluster 8.1.0 and earlier versions do not support dynamic adjustment. To change redistribution concurrency, suspend redistribution first.



Step 7 Check the redistribution progress. After the redistribution is complete, the amount of completed data, amount of remaining data, number of completed tables, number of remaining tables, and average rate during redistribution are displayed.



----End

11.6 Changing DWS Cluster Specifications

11.6.1 Using the Elastic Specification Change

Overview

Heavy service traffic requires additional resources (such as CPU, memory, and disk resources) to support it. If the current cluster resources are insufficient, creating a new cluster with more resources may be necessary. However, this can be costly and time-consuming. Moreover, creating a cluster with many resources but low service volume can result in resource redundancy and high costs.

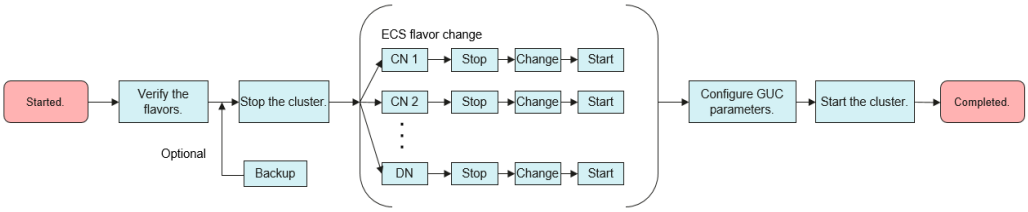
The elastic flavor change function is introduced to tackle this problem. It is ideal for scenarios where computing capabilities (CPU and memory) need to be adjusted during peak hours or when only computing capabilities need to be changed. By using elastic flavor change before peak hours, the cluster's computing

capability can be quickly increased. After peak hours, the cluster configuration can be reduced to minimize costs.

You can modify the CPU and memory configurations of the VM nodes in the target cluster by utilizing the underlying ECS capabilities. The following figure illustrates this process.

- To prevent service disruptions, it is crucial to schedule the elastic flavor change time window properly since the cluster must be stopped during the entire process.
- Changing all nodes concurrently ensures that the process will not take longer due to the number of nodes. Typically, the entire process takes around 5 to 10 minutes.

Figure 11-3 Principle of elastic flavor change



NOTE

- Only cluster versions 8.1.1.300 and later support elastic flavor change. For an earlier version, contact technical support to upgrade it first.

Precautions

- Choosing a lower target flavor when decreasing a cluster's flavor can impact its performance, so it is crucial to assess the potential impact on services before proceeding with the operation.
- Make sure to check if there are enough ECS resources and tenant CPU quotas in the current region before modifying the flavors.
- You can change the flavors again if needed. In case the flavors of some nodes fail to change, you can resubmit the change task to execute the process.

Constraints and Limitations

- You can upgrade or downgrade ECS flavors of the same type. For instance, you can change from **dwsx2.2xlarge.m7** to **dwsx2.4xlarge.m7**, but not to **dwsx2.4xlarge.m6**.
- Stop the VM before changing the flavor. The flavor change can only be done offline and it takes 5 to 10 minutes.

Procedure

Step 1 Log in to the DWS console.

Step 2 Click **Dedicated Clusters** > **Clusters**. All clusters are displayed by default.

Step 3 In the row of a cluster, choose **More** > **Change Flavor** in the **Operation** column and click **Change Node Flavor**.

Step 4 Configure the flavor. Enable automatic backup as needed.

Step 5 Confirm the settings, select the confirmation check box, and click **Next: Confirm**.

Step 6 Click **Submit**.

Step 7 Return to the cluster list. The cluster status will change to **Changing node flavor**. Wait for about 5 to 10 minutes.

----End

11.6.2 Disk Capacity Expansion of an EVS Cluster

Context

As customer services evolve, disk space often becomes the initial bottleneck. In scenarios where other resources are ample, the conventional scale-out process is not only time-consuming but also resource-inefficient. Disk capacity expansion can quickly increase storage without service interruption. You can expand the disk capacity when no other services are running. If the disk space is insufficient after the expansion, you can continue to expand the disk capacity. If the expansion fails, you can expand the disk capacity again.

NOTE

- Disk capacity expansion can be performed only for storage-compute coupled data warehouses with cloud SSDs. Only version 8.1.1.203 and later are supported.
- Disk capacity can be expanded only if the cluster is in **Available**, **To be restarted**, **Read-only**, or **Node fault**, **Unbalanced** state.

Precautions

- Hot storage disks cannot be scaled down.
- Scale up hot data storage during off-peak hours.
- If the cluster is in the read-only state, a message will be displayed after you click **Expand Disk Capacity**. After you start expansion, wait until it is completed and the cluster changes to the available state.

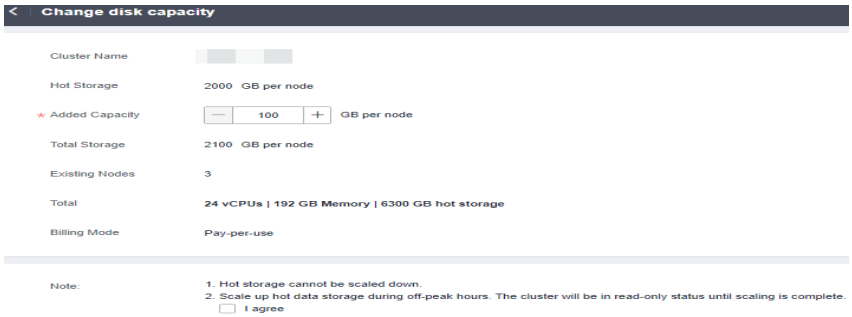
Procedure

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters**. All clusters are displayed by default.

Step 3 In the **Operation** column of the target cluster, choose **More** > **Change Specifications** and click **Change disk capacity**. The **Expand Disk Capacity** page is displayed.

Step 4 Select the appropriate storage space based on the storage step of the corresponding flavor on the **Change disk capacity** page. The step refers to the interval for increasing or decreasing storage space. Click **Resize Cluster Now**.



Step 5 Confirm the settings and click **Submit**.

Step 6 Return to the cluster list and check the disk capacity expansion progress.

----End

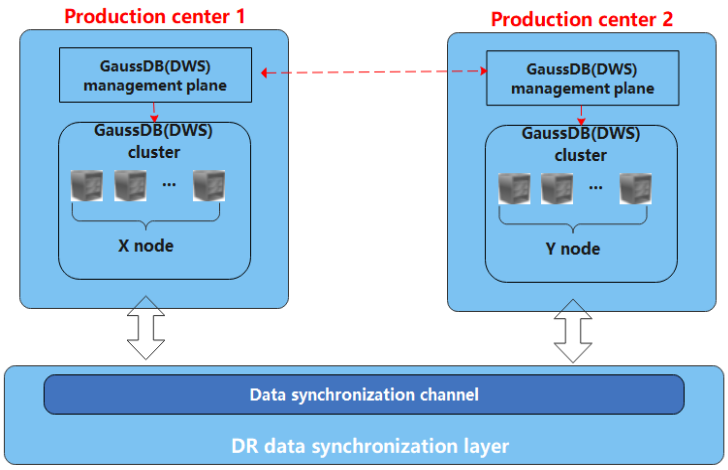
11.7 DWS Cluster DR Management

11.7.1 DWS Cluster DR Scenarios

Overview

A homogeneous DWS disaster recovery (DR) cluster is deployed in the same region. If the production cluster fails to provide read and write services due to natural disasters in the specified region or cluster internal faults, the DR cluster becomes the production cluster to ensure service continuity. The following figure shows the architecture.

Figure 11-4 DR architecture



NOTE

- Intra-region DR is supported only in cluster version 8.1.1 and later.

DR Features

- Multi-form DR
  - Intra-region DR
  - Multiple data synchronization modes: synchronization layer based on mutual trust
- Low TCO
  - Heterogeneous deployment (logical homogeneity)
  - Cluster-level DR
- Visual console

  Automatic and one-click DR drills

Constraints and Limitations

- Encrypted clusters do not support DR.
- During data synchronization, a non-fine-grained DR cluster cannot provide read or write services.
- When the DR task is stopped or abnormal but the DR cluster is normal, the DR cluster can provide the read service. After the DR switchover is successful, the DR cluster can provide the read and write services.
- When the DR task is created, the snapshot function of the production cluster is normal, but that of the DR cluster is disabled. Besides, snapshot restoration of both clusters is disabled.
- The synchronization resource pool supports cluster-level DR only in cluster version 8.3.0 or later.
- If cold and hot tables are used, cold data is synchronized using OBS.
- DR does not synchronize data from external sources.
- DR management refers to dual-cluster DR under the same tenant.
- The DR cluster and the production cluster must be logically homogeneous and in the same type and version.
- The production cluster and DR cluster used for intra-region DR must be in the same VPC.
- In intra-region DR, after services are switched over from the production cluster to the DR cluster, the bound ELB is automatically switched to the new production cluster. During the switchover, the connection is interrupted for a short period of time. Do not run service statements to write data during the switchover.
- During intra-region DR, the EIP, intranet domain name, and connection IP address of the original production cluster are not automatically switched with the cluster switchover. The EIP, domain name, or IP address used for connection in the service system need to be switched to the new cluster.

11.7.2 Creating and Starting DR for a DWS Cluster

Creating an Intra-Region Cluster-Level DR Task

Prerequisites

You can create a DR task only when the cluster is in the **Available** or **Unbalanced** state.

Procedure

Step 1 Log in to the DWS console.

Step 2 In the navigation pane, choose **Management** > **DR Tasks**.

Step 3 On the displayed page, click **Create**.

Step 4 Select the type and enter the name of the DR task to be created.

- **Type**: **Intra-region DR**
- **Name**: Enter 4 to 64 case-insensitive characters, starting with a letter. Only letters, digits, hyphens (-), and underscores (_) are allowed.

Step 5 Configure the production cluster.

- Select a created production cluster from the drop-down list.
- After a production cluster is selected, the system automatically displays its AZ.

Step 6 Configure the DR cluster.

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

  NOTE

  The AZ of the DR cluster can be the same as that of the production cluster. In a 3-AZ cluster, any of the three AZs can be selected for DR.

- **Cluster Name**: Upon selecting an AZ for the DR cluster, the system will auto-filter DR clusters that fulfill the logical homogeneity criteria. Should there be no qualifying DR clusters, you can click **Create DR Cluster** to create a DR cluster with the same configuration as the production cluster.

Step 7 Configure advanced parameters. Select **Default** to keep the default values of the advanced parameters. You can also select **Custom** to modify the values.

- The DR synchronization period indicates the interval for synchronizing incremental data from the production cluster to the DR cluster. Set this parameter based on the actual service data volume.

  NOTE

  The default DR synchronization period is 30 minutes.

Step 8 Click **OK**.

The DR status will then change to **Creating**. Wait until the creation is complete, and the DR status will change to **Not Started**.

----End

Starting a DR Task

Step 1 Log in to the DWS console.

Step 2 In the navigation pane, choose **Management** > **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 such as restoration, scale-out, upgrade, restart, node replacement, and password update, on the production cluster or DR cluster, and backup is also not allowed on the DR cluster. Exercise caution when performing this operation.

----End

Viewing DR Information

Step 1 Log in to the DWS console.

Step 2 In the navigation pane, choose **Management** > **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 type, DR creation time, DR start time, and DR status.
- **Production Cluster Information**: You can view the production cluster ID, cluster name, AZ, used storage capacity, cluster DR status, and the time of the latest successful DR task.
- **DR Cluster Information**: You can view the DR cluster ID, cluster name, AZ, used storage capacity, cluster DR status, and the time of the latest successful DR task.
- **DR Configuration**: You can view and modify the DR synchronization period.

----End

Updating DR Configurations

Step 1 Log in to the DWS console.

Step 2 In the navigation pane, choose **Management** > **DR Tasks**.

Step 3 In the DR list, click the name of a DR task.

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

Case 1: How Do I Scale out a Cluster in the DR State?

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

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 (Optional) If the **Task Information** is other than **DR not started**, delete the DR task. For details, see **Deleting a DR Task**.

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 and Starting DR for a DWS Cluster**.

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

11.7.3 Performing a DR Switchover for the DWS Cluster

Switching to the DR Cluster

Step 1 Log in to the DWS console.

Step 2 In the navigation pane, choose **Management** > **DR Tasks**.

Step 3 Click **Switch to DR Cluster** in the **Operation** column of the target DR task.

Step 4 In the dialog box that is displayed, click **OK**.

The DR status will change to **DR switching**.

After the switchover is successful, the DR status will change to the original status.

NOTE

- To perform a switchover when the DR cluster is running properly, click **Switch to DR Cluster**.
- You can perform a DR switchover when the DR task is in the **Running** state.
- During a switchover, the original production cluster is not available.
- The Recovery Point Object (RPO, time point to which the system and data must be recovered after a disaster occurs) in different DR switchover scenarios is described as follows:
  - Production cluster in the **Available** state: RPO = 0
  - Production cluster in the **Unavailable** state: A zero RPO may not be achieved, but data can at least be restored to that of the latest successful DR synchronization (**Last DR Succeeded**). For details, see **Viewing DR Information**.

----End

Exception Switchover

Scenario

The production cluster is unavailable, the DR cluster is normal, and the DR status is **Abnormal**.

Procedure

Step 1 Log in to the DWS console.

Step 2 In the navigation pane, choose **Management** > **DR Tasks**.

Step 3 Choose **More** > **Exception Switchover** in the **Operation** column of the target DR task.

Step 4 In the dialog box that is displayed, click **OK**.

The **Status** will change to **Switchover in progress**.

After the switchover is successful, the DR status will change to the original status. In this procedure, the DR status will change back to **Abnormal**.

NOTE

- To perform a switchover when the DR cluster is abnormal or the production cluster is faulty, click **Exception Switchover**.
- DR exception switchover is supported only by clusters of version 8.1.2 or later.
- Before a switchover, check the latest synchronization time in the DR cluster. The DR cluster will serve as a production cluster after an abnormal switchover, but the data that failed to be synchronized from the original production cluster to the DR cluster will not exist in the DR cluster.

----End

Performing a DR Switchback

Scenario

After abnormal switchover, if you have confirmed that the original production cluster was recovered, you can perform a switchback.

Procedure

Step 1 Log in to the DWS console.

Step 2 In the navigation pane, choose **Management** > **DR Tasks**.

Step 3 Click **DR Recovery** in the **Operation** column of a DR task.

Step 4 In the displayed dialog box, set **Synchronization Mode** to **Incremental** or **Full**.

NOTE

You are advised to set **Synchronization Mode** to **Incremental** when updating a DR creation task.

Step 5 Click **OK**.

The **Status** will change to **Recovering**.

After the DR recovery is successful, the **Status** will change to **Running**.

NOTE

- DR is supported only by clusters of 8.1.2 or later.
- During DR recovery, data in the DR cluster will be deleted, and the DR relationship will be re-established with the new production cluster.

----End

11.7.4 Stopping and Deleting DR for a DWS Cluster

Stopping the DR Task

Step 1 Log in to the DWS console.

Step 2 In the navigation pane, choose **Management** > **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

Deleting a DR Task

Step 1 Log in to the DWS console.

Step 2 In the navigation pane, choose **Management** > **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

11.8 Upgrading a DWS Cluster

DWS allows you to upgrade clusters on the console. For details, see **Upgrading a Cluster**.

During cluster O&M operations, DWS will send SMS notifications to keep you informed. It is important to be careful when performing any operations on the cluster during this time.

During the upgrade, the cluster will be restarted and cannot provide services for a short period of time.

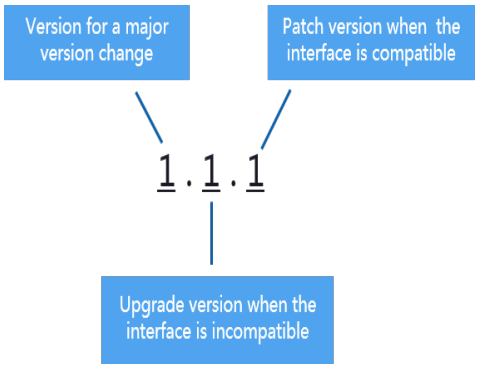
NOTE

- After a cluster is upgraded to 8.1.3 or later, it enters the observation period. During this period, you can check service status and roll back to the earlier version if necessary.
- Upgrading the cluster from 9.0.3 to 9.1.0 changes the **disk_cache_base_paths** value for the cache path, which brings the performance back to normal.
- Upgrading the cluster does not affect the original cluster data or specifications.

Upgrade Version Description

The following figure shows the cluster version.

Figure 11-5 Version description



- **Service patch upgrade**: The last digit of cluster version *X.X.X* is changed. For example, the cluster is upgraded from 1.1.0 to 1.1.1.
  - Duration: The whole process will take less than 10 minutes.
  - Impact on services: During this period, if the source version is upgraded to 8.1.3 or later, online patching is supported. During the patch upgrade, you do not have to stop services, but the services will be intermittently interrupted for seconds. If the target version is earlier than 8.1.3, services will be interrupted for 1 to 3 minutes. Therefore, you are advised to perform the upgrade during off-peak hours.
- **Service upgrade**: The first two digits of cluster version *X.X.X* are changed. For example, the cluster is upgraded from 1.1.0 to 1.2.0.
  - Duration: The whole process will take less than 30 minutes.
  - Impact on services: Online upgrade is supported for update to 8.1.1 or later. During the upgrade, you are not required to stop services, but services are intermittently interrupted for seconds. You are advised to perform the upgrade during off-peak hours.
- Hot patch upgrade: A hot patch upgrade involves adding a one-digit version number (in the format of **0001-9999**) to the current cluster version.
  - Duration: The upgrade of a single hot patch takes less than 10 minutes.

- Impact on services: The hot patch upgrade will not affect services, but there is a chance that the issues resolved by the current hot patch may come back after it is uninstalled.

Upgrading a Cluster

Prerequisites

Cluster 8.1.1 and later versions allow users to deliver cluster upgrade operations on the console.

Procedure

Step 1 Log in to the DWS console.

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

Step 3 In the navigation pane, choose **Upgrade Management**.

Step 4 Choose either **Upgrade** or **Hot patch** from the **Type** drop-down list depending on the type of upgrade you want to perform.

Step 5 On the **Upgrade Management** page, select a version from the **Target Version** drop-down list.

NOTE

- Before the upgrade, it is crucial to verify if it meets the inspection conditions. Click **Immediate Inspection** to complete the inspection and proceed to the next step only if it passes. For more information, see **Viewing Inspection Results**.
- DR cannot be established after a hot patch is installed in a cluster.

Step 6 Click **Upgrade**. Click **OK** in the displayed dialog box.

Step 7 Check whether the cluster is successfully upgraded.

- If the cluster version is 8.1.3 or later, the cluster enters the service observation period after the upgrade is complete. If you have verified your services, click **Submit** on the **Upgrade Management** page to complete the cluster upgrade. If you find your cluster performance affected by the upgrade, you can click **Rollback** to roll back the upgrade.

  NOTE

  - If you are using a version of 8.1.3 or earlier, you will not be able to roll back or submit upgrade tasks until the cluster upgrade is finished.
  - After an upgrade task is successfully delivered, if the upgrade task is not submitted, the **wlm** thread occupies the system storage space and affects the system performance.
- If the cluster upgrade fails, click **Rollback** to roll back to the original cluster version, or click **Retry** to deliver the upgrade again.

----End

11.9 DWS Cluster Log Management

11.9.1 Log Types Supported by DWS Clusters

DWS provides database audit logs, management console audit logs, and other logs for users to query service logs, analyze problems, and learn product security and performance.

Database Audit Logs

If the **Security** function is enabled, DWS records any DML and DDL operations performed by the database. You can locate and analyze faults based on the database audit logs, and perform behavior analysis and security auditing on historical database operations to improve DWS security.

For how to enable and view database audit logs, see **Viewing DWS Database Audit Logs**.

Management Console Audit Logs

DWS uses Cloud Trace Service (CTS) to record mission-critical operations performed on the DWS console, such as cluster creation, snapshot creation, cluster scale-out, and cluster restart. The logs can be used in purposes such as security analysis, compliance audit, resource tracing, and fault locating.

For how to enable and view management console audit logs, see **Viewing Operation Logs on the DWS Console**.

Other Logs

DWS interconnects with Log Tank Service (LTS). You can view collected cluster logs or dump logs on the LTS console.

The following log types are supported: CN logs, DN logs, OS messages logs, audit logs, cms logs, gtm logs, Roach client logs, Roach server logs, upgrade logs, and scale-out logs.

11.9.2 Dumping DWS Database Audit Logs

DWS records information (audit logs) about connections and user activities in your database. The audit logs help you monitor the database to ensure security, rectify faults, and locate historical operation records. DWS saves audit logs in the database, but they can be viewed outside of it by dumping them to OBS. It is worth mentioning that the audit log dump and kernel audit log dump functions can be enabled or disabled independently. With the kernel audit log dump feature, audit logs stored in the database can be dumped directly to OBS.

NOTE

- This function cannot be used if OBS is not available.
- Only 9.1.0.100 and later versions support kernel log dump.
- Data may during cluster specifications change, CN addition, or CN deletion. You are advised to disable audit log dump during these operations.
- If a CN node is faulty, data on the CN node may be lost.
- After audit log dumping is enabled, audit logs will be dumped if the size of saved audit logs exceeds 1 GB. This may cause abnormal query results. Exercise caution when performing this operation.
- Version support for the audit log dump directory partition is as follows:
  - For version 8.1.3.*x* clusters, it is only supported by version 8.1.3.322 or later clusters.
  - For version 8.2.0.*x* clusters, it is only supported by version 8.2.0.106 or later clusters.
  - It is supported by version 8.2.1 or later clusters.
  - To use this feature in earlier versions, contact technical support to upgrade your cluster first. Manually enable this feature after the upgrade.

Prerequisites

After a DWS cluster is created, you can enable log dump for it to dump audit logs to OBS. **Before enabling audit log dump, ensure the following conditions are met:**

Enabling Audit Log Dumps

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

Step 3 In the cluster list, click the name of the cluster for which you want to enable audit log dump. In the navigation pane, choose **Security Settings**.

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

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, DWS can dump audit logs to OBS.

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

- **OBS Foreign Table**: Audit logs can be read using OBS foreign tables during dumping. Audit logs are stored in CSV format and compressed in GZ format.
- **OBS Bucket**: Name of the OBS bucket used to store the audit data. If no OBS bucket is available, click **View OBS Bucket** to access the OBS console and create one. For details, see "Console Operation Guide" > "Managing Buckets" > "Creating a Bucket" in the *Object Storage Service User Guide*.
- **OBS Path**: User-defined directory on OBS for storing audit files. Different directory levels are separated by forward slashes (/). The value is a string containing 1 to 50 characters, which cannot start with a forward slash (/). If

the entered OBS path does not exist, the system creates one and dumps data to it.

- **Dump Interval (Minute)**: Interval based on which DWS periodically dumps data to OBS. The value range is 5 to 43200. The unit is minute.

Step 5 Click **Apply**.

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

When the status changes to **Synchronized**, the configurations are saved and take effect.

----End

Enabling Kernel Audit Log Dump

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

Step 3 Click the name of the cluster for which you want to enable kernel log dump. Choose **Security**.

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

When you enable the kernel audit log dump feature 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, DWS can dump audit logs to OBS.

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

- **OBS Bucket**: name of the OBS bucket for storing kernel 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*.
- **Kernel OBS Path**: user-defined directory for storing kernel logs on OBS. 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.

Step 5 Click **Apply**.

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

When the status changes to **Synchronized**, the configurations are saved and take effect.

Step 6 After the kernel audit log dump function is enabled, you can use the **pg_query_audit** function to view the dumped logs. For details, see **Using Functions to View Database Audit Logs**.

Alternatively, select the OBS bucket and folder where logs are stored to view the log files. For details, see **Step 6**.

----End

Modifying Audit Log Dump Configurations

After audit log dump is enabled, you can modify the dump configuration. For example, you can modify the OBS bucket and path for storing logs and the dump period.

The procedure is as follows:

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

Step 3 In the cluster list, click the name of the cluster for which you want to modify the audit log dump configurations. In the navigation pane, choose **Security**.

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

Step 5 Click **Apply**.

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

When the status changes to **Synchronized**, the configurations are saved and take effect.

----End

Viewing Dumped Audit Logs

After audit log dump is enabled, you can view the dumped audit logs on OBS.

To view dumped audit logs, perform the following steps:

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

Step 3 In the cluster list, click the name of the target cluster for which you want to view the log dump history. In the navigation pane, choose **Security**.

Step 4 In the **Audit Settings** area, click **View Dump Record**.

Step 5 In the **Audit Log Dump Records** dialog box, click **View OBS Bucket**. The OBS console page is displayed.

Step 6 Select the OBS bucket and folder where the logs are stored to view the log files.

You can download and decompress the files to view. The fields of audit log files are described as follows:

Table 11-24 Log file fields

Field	Type	Description
begintime	timestamp with time zone	Operation start time.
endtime	timestamp with time zone	Operation end time.
operation_type	text	Operation type. For details, see **Table 11-25**.
audit_type	text	Audit type. For details, see **Table 11-26**.
result	text	Operation result.
username	text	Name of the user who performs the operation.
database	text	Database name.
client_conninfo	text	Client connection information, that is, gsql, JDBC, or ODBC.
object_name	text	Object name.
object_details	text	Operation object details.
command_text	text	Command used to perform the operation.
detail_info	text	Operation details.
transaction_xid	text	Transaction ID.
query_id	text	Query ID.
node_name	text	Node name.
thread_id	text	Thread ID.
local_port	text	Local port.
remote_port	text	Remote port.
result_rows	text	Number of rows in the operation result.
error_code	text	Error code.

Table 11-25 operation_type: operation types

Operation Type	Description
audit_switch	Indicates that the operations of enabling and disabling the audit log function are audited.
login_logout	Indicates that user login and log-out operations are audited.
system	Indicates that the system startup, shutdown, and instance switchover operations are audited.
sql_parse	Indicates that SQL statement parsing operations are audited.
user_lock	Indicates that user locking and unlocking operations are audited.
grant_revoke	Indicates that user permission granting and revoking operations are audited.
violation	Indicates that user's access violation operations are audited.
ddl	Indicates that DDL operations are audited. DDL operations are controlled at a fine granularity based on operation objects. Therefore, **audit_system_object** is used to control the objects whose DDL operations are to be audited. (The audit function takes effect as long as **audit_system_object** is configured, no matter whether **ddl** is set.)
dml	Indicates that the DML operations are audited.
select	Indicates that the **SELECT** operations are audited.
internal_event	Indicates that internal incident operations are audited.
user_func	Indicates that operations related to user-defined functions, stored procedures, and anonymous blocks are audited.
special_func	Indicates that special function invoking operations are audited. Special functions include **pg_terminate_backend** and **pg_cancel_backend**.
copy	Indicates that the **COPY** operations are audited.
set	Indicates that the **SET** operations are audited.
transaction	Indicates that transaction operations are audited.
vacuum	Indicates that the **VACUUM** operations are audited.
analyze	Indicates that the **ANALYZE** operations are audited.
cursor	Indicates that cursor operations are audited.

Operation Type	Description
anonymous_block	Indicates that the anonymous block operations are audited.
explain	Indicates that the **EXPLAIN** operations are audited.
show	Indicates that the **SHOW** operations are audited.
lock_table	Indicates that table lock operations are audited.
comment	Indicates that the **COMMENT** operations are audited.
preparestmt	Indicates that the **PREPARE, EXECUTE**, and **DEALLOCATE** operations are audited.
cluster	Indicates that the **CLUSTER** operations are audited.
constraints	Indicates that the **CONSTRAINTS** operations are audited.
checkpoint	Indicates that the **CHECKPOINT** operations are audited.
barrier	Indicates that the **BARRIER** operations are audited.
cleanconn	Indicates that the **CLEAN CONNECTION** operations are audited.
seclabel	Indicates that security label operations are audited.
notify	Indicates that the notification operations are audited.
load	Indicates that the loading operations are audited.

Table 11-26 audit_type parameters

Parameter	Description
audit_open/audit_close	Indicates that the audit type is operations enabling or disabling audit logs.
user_login/user_logout	Indicates that the audit type is operations and users with successful login/logout.
system_start/system_stop/ system_recover/system_switch	Indicates that the audit type is system startup, shutdown, and instance switchover.
sql_wait/sql_parse	Indicates that the audit type is SQL statement parsing.
lock_user/unlock_user	Indicates that the audit type is successful user locking and unlocking.
grant_role/revoke__role	Indicates that the audit type is user permission granting and revoking.

Parameter	Description
user_violation	Indicates that the audit type is unauthorized user access operations.
ddl_*database_object*	Indicates that successful DDL operations are audited. DDL operations are controlled at a fine granularity based on operation objects. So, **audit_system_object** is used to control the objects whose DDL operations are to be audited. (The audit function takes effect as long as **audit_system_object** is configured, no matter whether **ddl** is set.) For example, **ddl_sequence** indicates that the audit type is sequence-related operations.
dml_action_insert/ dml_action_delete/ dml_action_update/ dml_action_merge/ dml_action_select	Indicates that the audit type is DML operations such as **INSERT**, **DELETE**, **UPDATE**, and **MERGE**.
internal_event	Indicates that the audit type is internal events.
user_func	Indicates that the audit type is user-defined functions, stored procedures, or anonymous block operations.
special_func	Indicates that the audit type is special function invocation. Special functions include **pg_terminate_backend** and **pg_cancel_backend**.
copy_to/copy_from	Indicates that the audit type is **COPY** operations.
set_parameter	Indicates that the audit type is **SET** operations.
trans_begin/trans_commit/ trans_prepare/ trans_rollback_to/ trans_release/trans_savepoint/ trans_commit_prepare/ trans_rollback_prepare/ trans_rollback	Indicates that the audit type is transaction-related operations.
vacuum/vacuum_full/ vacuum_merge	Indicates that the audit type is **VACUUM** operations.
analyze/analyze_verify	Indicates that the audit type is **ANALYZE** operations.
cursor_declare/cursor_move/ cursor_fetch/cursor_close	Indicates that the audit type is cursor-related operations.

Parameter	Description
codeblock_execute	Indicates that the audit type is anonymous blocks.
explain	Indicates that the audit type is **EXPLAIN** operations.
show	Indicates that the audit type is **SHOW** operations.
lock_table	Indicates that the audit type is table locking operations.
comment	Indicates that the audit type is **COMMENT** operations.
prepare/execute/deallocate	Indicates that the audit type is **PREPARE**, **EXECUTE**, or **DEALLOCATE** operations.
cluster	Indicates that the audit type is **CLUSTER** operations.
constraints	Indicates that the audit type is **CONSTRAINTS** operations.
checkpoint	Indicates that the audit type is **CHECKPOINT** operations.
barrier	Indicates that the audit type is **BARRIER** operations.
cleanconn	Indicates that the audit type is **CLEAN CONNECTION** operations.
seclabel	Indicates that the audit type is security label operations.
notify	Indicates that the audit type is notification operations.
load	Indicates that the audit type is loading operations.

----End

Disabling Audit Log Dump/Kernel Audit Log Dump

After the audit log dump or kernel audit log dump is enabled, you can disable it if you no longer need to dump audit logs or kernel audit logs to OBS.

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** in the navigation pane.

Step 3 Click the name of the cluster for which you want to disable **Audit Log Dump** or **Kernel Audit Log Dump**. In the navigation pane, choose **Security Settings**.

Step 4 In the audit configuration area, toggle the audit log dump/kernel audit log dump function off.

Step 5 Click **Apply**.

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

When the status changes to **Synchronized**, the configurations are saved and take effect.

----End

11.9.3 Viewing DWS Database Audit Logs

Database audit logs can be set on the **Security Settings** page of the cluster. Security configurations can be modified only for clusters in the **Available** or **Unbalanced** state. Furthermore, the target cluster should not be undergoing any node additions, specification changes, configurations, upgrades, redistribution operations, or restarts.

Prerequisites

- The audit function has been enabled by setting **audit_enabled**. The default value of **audit_enabled** is **ON**. To disable audit, set it to **OFF** by referring to **Modifying GUC Parameters of the DWS Cluster**.
- The audit items have been configured. For how to enable audit items, see **Configuring the Database Audit Logs**.
- The database is running properly and a series of addition, modification, deletion, and query operations have been executed in the database. Otherwise, no audit result is generated.
- The audit logs of each database node are recorded separately.
- Only users with the **AUDITADMIN** permission can view audit records.

Configuring the Database Audit Logs

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters**.

Step 3 In the cluster list, click the name of a cluster. Choose **Security**.

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

Step 4 In the **Audit Settings** area, set the audit items:

NOTE

The default audit log retention policy is space-first, which means audit logs will be automatically deleted when the size of audit logs on a single node exceeds 1 GB. This function prevents node faults or low performance caused by high disk space occupied by audit logs.

Table 11-27 describes the detailed information about the audit items.

Table 11-27 Audit items

Audit Item	Description
Unauthorized access	Whether to record unauthorized operations. This parameter is disabled by default.
DQL operations	**SELECT** operations can be selected. **NOTE** Clusters of 8.1.1.100 and later versions support the **DQL operations** audit item.
DML operations	Whether to record **INSERT**, **UPDATE**, and **DELETE** operations on tables. This parameter is disabled by default. **NOTE** 8.1.1.100 and later versions support fine-grained splitting of audit items, and the **COPY** and **MERGE** options are added.
DDL operations	Whether to record the **CREATE**, **DROP**, and **ALTER** operations of specified database objects. **DATABASE**, **SCHEMA**, and **USER** are selected by default. **NOTE** 8.1.1.100 and later versions support **TABLE**, **DATA SOURCE**, and **NODE GROUP** operations. These operations are enabled by default.
Other operations	Whether to record other operations. Only the **TRANSACTION** and **CURSOR** operations are selected by default. **NOTE** • 8.1.1.100 and later versions support the **Other operations** audit item. • You are advised to select **TRANSACTION**. Otherwise, statements in a transaction will not be audited. • You are advised to select **CURSOR**. Otherwise, **SELECT** statements in a cursor will not be audited. The Data Studio client automatically encapsulates **SELECT** statements using **CURSOR**.

Except the audit items listed in **Table 11-27**, key audit items in **Table 11-28** are enabled by default on DWS.

Table 11-28 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.

Parameter	Description
	Records the grants and reclaims of user permissions.
	Records the audit function of the **SET** operation.

Step 5 Enable or disable audit log dumps.

For more information, see **Enabling Audit Log Dumps**.

Step 6 Click **Apply**.

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

When the status changes to **Synchronized**, the configurations are saved and take effect.

You can click  to refresh the configuration information.

----End

Viewing Database Audit Logs

Method 1: Audit logs will occupy disk space. To prevent excessive disk usage, DWS supports audit log dumping. You can enable the **Log Dump** function to dump audit logs to OBS (you need to create an OBS bucket for storing audit logs first). For details about how to view the dumped logs, see **Enabling Audit Log Dumps**.

Method 2: Use the **Log** function of LTS to view or download the collected database audit logs. For details, see **Checking Cluster Logs**.

Method 3: Database audit logs are stored in the database by default. After connecting to the cluster, you can use the **pg_query_audit** function to view the logs. For details, see **Using Functions to View Database Audit Logs**.

Using Functions to View Database Audit Logs

Step 1 Use the SQL client tool to connect to the database cluster. For details, see **Connecting to a DWS Cluster**.

Step 2 Use the **pg_query_audit** function to query the audit logs of the current CN. The syntax is as follows:

```
pg_query_audit(timestamptz starttime,timestamptz endtime,audit_log)
```

starttime and **endtime** indicate the start time and end time of the audit record, respectively. **audit_log** indicates the physical file path of the queried audit logs. If **audit_log** is not specified, the audit log information of the current instance is queried.

For example, view the audit records of the current CN node in a specified period.

```
SELECT * FROM pg_query_audit('2021-02-23 21:49:00','2021-02-23 21:50:00');
```

The query result is as follows:

```
      begintime       |       endtime       | operation_type | audit_type | result |  username  | database | client_conninfo | object_name | command_text |                detail_info                | transaction_xid | query_id |  node_name   |          session_id          | local_port | remote_port
```



```
----------------------------+----------------------------+----------------+------------+--------+------------+----------
+-----------------+-------------+-----------------+-------------------------------------------------------------------------
+----------------+----------+-------------+---------------------------+------------+-------------
 2021-02-23 21:49:57.76+08 | 2021-02-23 21:49:57.82+08 | login_logout   | user_login | ok     | dbadmin |
gaussdb | gsql@[local]    | gaussdb    | login db     | login db(gaussdb) successfully, the current user is:
dbadmin | 0               | 0        | coordinator1 | 140324035360512.667403397820909.coordinator1 | 27777
|
```

This record indicates that user **dbadmin** logged in to the **gaussdb** database at 2021-02-23 21:49:57.82 (GMT+08:00). After the host specified by **log_hostname** is started and a client is connected to its IP address, the host name found by reverse DNS resolution is displayed following the at sign (@) in the value of **client_conninfo**.

Step 3 Use the **pgxc_query_audit** function to query audit logs of all CNs. The syntax is as follows:

```
pgxc_query_audit(timestamptz starttime,timestamptz endtime)
```

For example, view the audit records of all CN nodes in a specified period.

```
SELECT * FROM pgxc_query_audit('2021-02-23 22:05:00','2021-02-23 22:07:00') where audit_type =
'user_login' and username = 'user1';
```

The query result is as follows:

```
        begintime          |          endtime           | operation_type | audit_type | result | username | database |
client_conninfo | object_name | command_text |                detail_info                | transaction_xid
| query_id |  node_name   |              session_id                | local_port | remote_port
----------------------------+----------------------------+----------------+------------+--------+----------+----------
+-----------------+-------------+--------------+-------------------------------------------------------------------
+----------------+----------+-------------+---------------------------+------------+-------------
 2021-02-23 22:06:22.219+08 | 2021-02-23 22:06:22.271+08 | login_logout    | user_login | ok     | user1    |
gaussdb  | gsql@[local]    | gaussdb     | login db     | login db(gaussdb) successfully, the current user is:
user1 | 0               | 0        | coordinator2 | 140689577342720.667404382271356.coordinator  | 27782      |
 2021-02-23 22:05:51.697+08 | 2021-02-23 22:05:51.749+08 | login_logout    | user_login | ok     | user1    |
gaussdb  | gsql@[local]    | gaussdb     | login db     | login db(gaussdb) successfully, the current user is:
user1 | 0               | 0        | coordinator1 | 140525048424192.667404351749143.coordinator1 | 27777      |
```

The query result shows the successful login records of **user1** in to CN1 and CN2.

Step 4 Query the audit records of multiple objects.

```
SET audit_object_name_format TO 'all';
SELECT object_name,result,operation_type,command_text FROM pgxc_query_audit('2022-08-26
8:00:00','2022-08-26 22:55:00') where command_text like '%student%';
```

The query result is as follows:

```
                        object_name                         | result | operation_type
|                                                  command_text

-------------------------------------------------------------------+--------+----------------
+-----------------------------------------------------------------------------------------------------------------------
-------------------------------------------
 student                                                  | ok     | ddl            | CREATE TABLE student(stuNo int,
stuName TEXT);
 studentscore                                               | ok     | ddl            | CREATE TABLE studentscore(stuNo
int, stuscore int);
 ["public.student_view01","public.studentscore","public.student"] | ok     | ddl            | CREATE OR REPLACE
VIEW student_view01 AS SELECT * FROM student t1 where t1.stuNo in (select stuNo from studentscore t2
where t1.stuNo = t2.stuNo);
 ["public.student_view01","public.student","public.studentscore"] | ok     | dml            | SELECT * FROM
student_view01;
```

In the **object_name** column, the table, view, and base table associated with the view are displayed.

----End

11.9.4 Viewing Operation Logs on the DWS Console

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 DWS for later query, audit, and backtrack operations.

NOTE

- The creation and deletion of automated snapshots are not performed by users, therefore not recorded in audit logs.
- There are many DWS cluster operation events, but the table below only includes some frequently audited operations.

Table 11-29 DWS operations that can be recorded by CTS

Operation	Resource	Event Name
Creating a cluster	cluster	createCluster
Deleting a cluster	cluster	deleteCluster
Performing a cluster inspection	cluster	createInspection
Stopping an inspection	cluster	AbortInspection
Scaling out a cluster	cluster	growCluster
Increasing the capacity of idle nodes	cluster	resizeWithFreeNodes
Performing cluster redistribution	cluster	redistributeCluster
Querying redistribution details	cluster	queryRedisInfo
Adding disk capacity	cluster	executeDiskExpand
Changing cluster flavors	cluster	flavorResize
Restarting a cluster	cluster	restartCluster
Performing a cluster switchover	cluster	activeStandySwitchover
Resetting passwords	cluster	resetPassword
Restoring a cluster	cluster	repairCluster
Creating a cluster connection	cluster	createClusterConnection
Modifying a cluster connection	cluster	modifyClusterConnection
Deleting a cluster connection	cluster	deleteClusterConnection
Resizing a cluster	cluster	resizeCluster
Binding or unbinding an EIP	cluster	bindOrUnbindEIP

Operation	Resource	Event Name
Creating or binding an ELB	cluster	createOrBindElb
Unbinding an ELB	cluster	unbindElb
Adding a CN	cluster	addCN
Deleting a CN	cluster	deleteCN
Upgrading a cluster	cluster	clusterUpdateMgr
Scaling in a cluster	cluster	shrinkCluster
Creating a resource management plan	cluster	addWorkloadPlan
Deleting a Resource Pool	cluster	deleteWorkloadQueueInfo
Creating a resource pool	cluster	addWorkloadQueueInfo
Modifying cluster GUC parameters	cluster	updateClusterConfigurations
Removing the read-only status	cluster	cancelReadonly
Modifying a maintenance window	cluster	modifyMaintenanceWindow
Adding CN nodes in batches	cluster	batchCreateCn
Deleting CN nodes in batches	cluster	batchDeleteCn
Adding tags in batches	cluster	batchCreateResourceTag
Deleting tags in batches	cluster	batchDeleteResourceTag
Creating a logical cluster	cluster	createLogicalCluster
Deleting logical clusters	cluster	deleteLogicalCluster
Editing a logical cluster	cluster	editLogicalCluster

Operation	Resource	Event Name
Restarting logical clusters	cluster	restartLogicalCluster
Converting to a logical cluster	cluster	switchLogicalCluster
Starting a cluster	cluster	startCluster
Stopping a cluster	cluster	stopCluster
Modifying the security group of a cluster	cluster	changeSecurityGroup
Changing the cluster time zone	cluster	modifyClusterTimezone
Creating a snapshot	backup	createBackup
Deleting a snapshot	backup	deleteBackup
Restoring a cluster	backup	restoreCluster
Copying snapshots	backup	copySnapshot
Deleting a snapshot policy	backup	deleteBackupPolicy
Updating a snapshot policy	backup	updateClustersBackupPolicy
Creating a DR task	disasterRecovery	createDisasterRecovery
Deleting a DR task	disasterRecovery	deleteDisasterRecovery
Starting a DR task	disasterRecovery	startDisasterRecoveryAction
Stopping a DR task	disasterRecovery	stopDisasterRecoveryAction
Switching to the DR cluster	disasterRecovery	switchoverDisasterRecoveryAction
Performing an exception switchover	disasterRecovery	failoverDisasterRecoveryAction
Performing DR	disasterRecovery	recoveryDisaster
Updating DR configurations	disasterRecovery	updateRecoveryDisaster
Querying DR details	disasterRecovery	disasterRecoveryOperate

Operation	Resource	Event Name
Setting security parameters	configurations	updateConfigurations
Creating an extended data source	dataSource	createExtDataSource
Deleting an extended data source	dataSource	deleteExtDataSource
Updating an extended data source	dataSource	updateExtDataSource
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 Click the search box above the trace list and set the search criteria.

The following filters are available:

- **Trace Name**: If you select this option, you also need to select a specific trace name.
- **Cloud Service**: Select **DWS**.
- **Resource Type**: Select **All resource types** or specify a resource type.
- **Resource Name**: If you select this option, you also need to select or enter a specific resource name.
- **Resource ID**: If you select this option, you also need to select or enter a specific resource ID.
- **Operator**: Select a specific operator (at user level rather than tenant level).
- **Event ID**: If you select this option, you also need to select or enter an event ID.
- **Trace Status**: Available options include **All trace statuses**, **normal**, **warning**, and **incident**. You can only select one of them.
- **Enterprise Project ID**: If you select this option, you also need to select or enter a specific enterprise project ID.

- **Access Key ID**: If you select this option, you also need to select or enter a specific access key ID.

Step 4 Click **Query**.

Step 5 Click the name of the trace to be viewed. A window is displayed, showing the trace details.

For details about key fields of a CTS trace, see "Trace References" > "Trace Structure" and "Trace References" > "Example Traces" in *Cloud Trace Service User Guide*.

----End

11.9.5 Viewing Other Logs of the DWS Cluster

Overview

Cluster logs are collected and sent to Log Tank Service (LTS). You can check or dump the collected cluster logs on LTS.

The following log types are supported: CN logs, DN logs, OS messages logs, audit logs, cms logs, gtm logs, Roach client logs, Roach server logs, upgrade logs, and scale-out logs.

NOTE

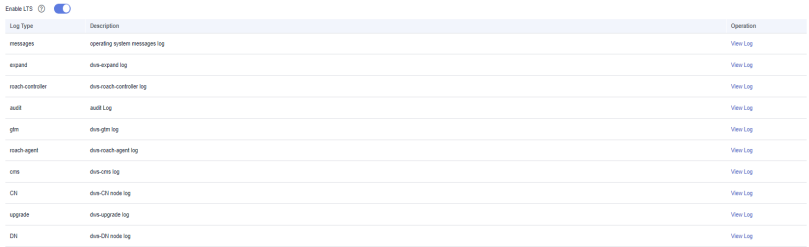
- Only 8.1.1.300 and later versions support cluster log management.
- Only 8.3.0 and later versions support CMS logs, GTM logs, Roach client logs, Roach server logs, scaling logs, and upgrade logs.

Enabling LTS

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters**. All clusters are displayed by default.

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



Step 4 On the **Logs** tab, enable LTS. If LTS is enabled for the first time, the following dialog box will be displayed. Confirm the information and click **Yes**.

NOTE

- If LTS has been enabled and authorized to create an agency, no authorization is required when LTS is enabled again.
- By default, only Huawei Cloud accounts or users with **Security Administrator** permissions can query and create agencies. IAM users under an account do not have the permission to query or create agencies by default. Contact a user with that permission and complete the authorization on the current page.
- When interconnecting with LTS, you need to grant LTS-related permission policies **(LTS Admin**, **LTS Administrator**, **LTS FullAccess**, and **LTS ReadOnlyAccess**) to users.

----End

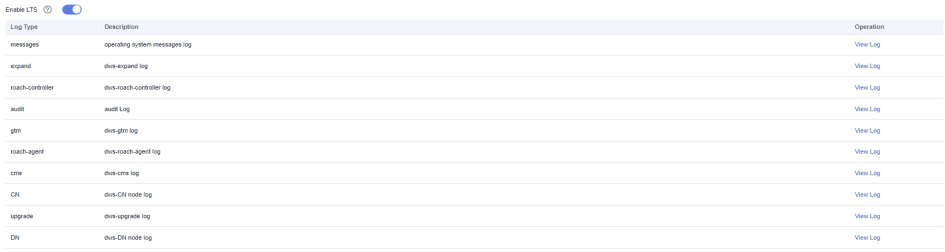
Checking Cluster Logs

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters**. All clusters are displayed by default.

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

Step 4 On the **Logs** tab, click **View Log** in the **Operation** column of a log type to go to the Log Tank Service (LTS) page and view logs.



----End

Disabling LTS

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters**. All clusters are displayed by default.

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

Step 4 Toggle off the LTS switch.

Step 5 Click **OK** in the dialog box.

----End

11.10 Handling Abnormal DWS Clusters

Removing the Read-only Status

A cluster in read-only status does not allow write operations. You can remove this status on the management console. A cluster becomes read-only probably because of high disk usage. For how to solve this problem, see "Solution to High Disk

Usage and Cluster Read-Only" in *Data Warehouse Service (DWS) Troubleshooting Guide*.

NOTE

- Clusters of version 1.7.2 or later support removal of the read-only state.
- In 8.2.0 and later versions, you can free up disk space by using **DROP/TRUNCATE TABLE** in a read-only cluster.

Impacts on the System

- You can cancel the read-only status only when a cluster is read-only.
- When a cluster is in read-only status, stop the write tasks to prevent data loss caused by used up disk space.
- After the read-only status is canceled, clear the data as soon as possible to prevent the cluster from entering the read-only status again after a period of time.

Procedure

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters**.

All clusters are displayed by default.

Step 3 In the row containing the cluster whose cluster status is **Read-only**, click **Cancel Read-only**.

Step 4 In the dialog box that is displayed, click **OK** to confirm and remove the read-only status for the cluster.

----End

Performing a Primary/Standby Switchback

In the **Unbalanced** state, the number of primary instances on some nodes increases. As a result, the load pressure is high. In this case, the cluster is normal, but the overall performance is not as good as that in a balanced state. Restore the primary-standby relationship to recover the cluster to the available state.

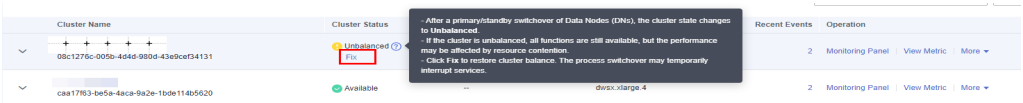
NOTE

- Only 8.1.1.202 and later versions support primary/standby cluster restoration.
- Cluster restoration interrupts services for a short period of time. The interruption duration depends on the service volume. You are advised to perform this operation during off-peak hours.

Step 1 Log in to the DWS console.

Step 2 Choose **Dedicated Clusters** > **Clusters** and locate the cluster whose load is unbalanced.

Step 3 In the **Cluster Status** column of the cluster, click **Fix** under **Unbalanced**.



Step 4 In the dialog box that is displayed, confirm that the service is in off-peak hours, and click **Yes**. A message will be displayed in the upper right corner, indicating that the 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

11.11 Reclaiming DWS Space Using Vacuum

11.11.1 Overview

DWS provides the intelligent O&M feature to help you quickly and efficiently run O&M tasks. Intelligent O&M selects a proper time window and concurrency to complete specified tasks based on the cluster load. During O&M tasks, intelligent O&M monitors user service changes and promptly adapts task execution policies to minimize the impact on user services. Periodic tasks and one-off tasks are supported, and you can configure the time window as required.

Intelligent O&M ensures high availability. When the cluster is abnormal, failed O&M tasks will be retried. If some steps of an O&M task cannot be completed due to an abnormal cluster, the failed steps will be skipped for cost saving.

The intelligent O&M page consists of the following parts:

- Setting the common configurations of O&M tasks
  - **Maximum number of concurrent O&M tasks in the VacuumFull user table**: applies to VacuumFull O&M tasks for each user table. You are advised to set this parameter based on the available disk space and I/O load within a specific time window. The value ranges from 1 to 24. The recommended value is **5**.
  - Small CU threshold: helps identify small CU tables. A table is classified as small if its CU value is lower than the threshold. When the CU value is at or below the threshold, it triggers Vacuum. A higher threshold value increases the trigger sensitivity. The recommended default value for this parameter is **1000**.
  - Small CU ratio: indicates the ratio of small CU tables to all CU tables. When the ratio is at or above the parameter value, Vacuum is triggered. A lower value increases the trigger sensitivity. The recommended default value for this parameter is **50%**.
- Information about ongoing O&M tasks. (Currently, only VACUUM tasks are displayed. If disk space is insufficient because of table bloating, you can vacuum tables.).
  - Frequent table creation and deletion can lead to table bloating. To free up space, you can run the **VACUUM** command on system catalogs.
  - Frequently update and delete operations can lead to table bloating. To free up space, you can run the **VACUUM** or **VACUUM FULL** command on system catalogs.

- O&M details: **O&M Plan** and **O&M Status**. **O&M Plan** displays the basic information about all O&M tasks, and **O&M Status** displays the running status.

NOTE

- This feature is supported only in 8.1.3 or later. The small CU threshold and small CU ratio are displayed only for version 9.1.0.200 or later clusters.
- After completing the **VACUUM FULL** O&M task, the system automatically performs the **ANALYZE** operation.
- Only cluster 8.1.3 and later versions support the common configuration module for O&M tasks. For earlier versions, contact technical support to upgrade them.

11.11.2 Managing O&M Plans

Prerequisites

- Avoid setting the time window for automatic Vacuum O&M tasks during peak hours to prevent conflicts with user services.
- For user tables, the maximum concurrency for Vacuum Full O&M tasks is 24, and the minimum is 0. For system tables, the maximum concurrency is 1, and the minimum is 0. The concurrency value adjusts automatically based on **io_util**.
  - Two intervals for 0% to 60%
    - 0% to 30%: The concurrency value increases by 2 each time the value of **io_util** decreases by 15%.
    - 30% to 60%: The concurrency value is incremented by 1 each time the value of **io_util** decreases by 15%.
  - 60% to 70%: The concurrency value remains unchanged.
  - Above 70%: The concurrency value decreases by 1 until it reaches 0.
- The scheduler scans the expansion of column-store compression units (CUs) within the time window. If the average number of CU records in a column-store table is less than 1000, the scheduler scans the table first. The scanning of column-store CUs is not limited by table bloat or table reclaimable space.
- A maximum of 100 tables can be added to the priority list.
- The scheduler autovacuum function depends on the statistics. If the statistics are inaccurate, the execution sequence and results may be affected.
- The scheduler does not support names containing spaces or single quotation marks, including database names, schema names, and table names. Otherwise, the tables will be skipped. Tables with spaces or single quotes in the priority list are skipped automatically.

Setting the Common Configurations of O&M Tasks

Precautions

- The configuration applies to the VacuumFull O&M task for each user table. Running tasks are unaffected. The new periodic task settings take effect on the next run.

Procedure

Step 1 Log in to the DWS console.

Step 2 Click the name of the target cluster.

Step 3 In the navigation pane, choose **Intelligent O&M**.

Step 4 In the **Common O&M Task Configurations** area on the upper part of the page, modify the following common configuration items:

- **Maximum number of concurrent O&M tasks in the VacuumFull user table**: applies to VacuumFull O&M tasks for each user table. You are advised to set this parameter based on the available disk space and I/O load within a specific time window. The value range is 1 to 24. Configure it based on the remaining disk space and I/O load. You are advised to set it to 5. Running tasks are not affected. For periodic tasks, the new limit for concurrent tasks applies the next time the O&M task runs.
- Small CU threshold: helps identify small CU tables. A table is classified as small if its CU value is lower than the threshold. When the CU value is at or below the threshold, it triggers Vacuum. A higher threshold value increases the trigger sensitivity. The recommended default value for this parameter is **1000**.
- Small CU ratio: indicates the ratio of small CU tables to all CU tables. When the ratio is at or above the parameter value, Vacuum is triggered. A lower value increases the trigger sensitivity. The recommended default value for this parameter is **50%**.

Step 5 Confirm the settings and click **Save**.

----End

Adding an O&M Plan

Step 1 Log in to the DWS console.

Step 2 Click the name of the target cluster.

Step 3 In the navigation pane, choose **Intelligent O&M**.

Step 4 In the **O&M Plan** area, click **Add O&M Task**.

Step 5 In the displayed **Add O&M Task** dialog box, configure basic information about the O&M task.

Table 11-30 Basic configuration items of an O&M task

Configuration Item	Description	Example
O&M Task	Vacuum (Currently, only Vacuum O&M tasks are supported.)	Vacuum

Configuration Item	Description	Example
Description	Brief description of the intelligent O&M task.	This intelligent O&M task helps users periodically invoke Vacuum commands to reclaim space.
Remarks	Supplementary information.	-
Scheduling Mode	The supports the following scheduling modes: ● **Auto**: Intelligent O&M scans the database in a specified time window, and automatically delivers table-level vacuum tasks by service load and reclaimable space of user tables. ● **Specify**: You need to specify a vacuum target. Intelligent O&M will automatically deliver a table-level vacuum task in a specified time window. ● **Priority**: You can specify the preferential vacuum targets. During the remaining time window (if any), Intelligent O&M will automatically scan other tables that can be vacuumed and deliver table-level vacuum tasks. **NOTE** You are advised to select **Specify** for **VACUUM** and **VACUUM FULL** operations. Do not perform **VACUUM FULL** on wide column-store tables. Otherwise, memory bloat may occur.	Specify

Configuration Item	Description	Example
Autovacuum	Supported: system catalog Vacuum or user table VacuumFull. • A system catalog VACUUM transaction holds a level-4 lock (share update exclusive lock), which does not affect user services. Only the transactions on the DDL process of the system catalog are blocked. • A user table VACUUM FULL transaction holds a level-8 lock (access exclusive lock). All the other transactions on the table are blocked until VACUUM FULL is complete. To avoid affecting services, you are advised to perform VACUUM FULL during off-peak hours. **CAUTION** During **VACUUM FULL**, the space usage will first increase and then decrease, because this operation requires the same space as the table to be vacuumed. The actual table size is calculated as the total table size multiplied by (1 - dirty page rate). Ensure you have sufficient space before doing **VACUUM FULL**.	User table (VACUUM FULL)
Vacuum First	You can configure the preferred Vacuum target. Each row corresponds to a table. Each table is represented by the database name, schema name, and table name, which are separated by spaces.	-

Configuration Item	Description	Example
Advanced settings	If you select **Custom**, you can configure the autovacuum triggers, including the table bloat and table reclaimable space. If you select **Default**, the default configuration is used. **NOTE** VACUUM bloat rate: After frequent UPDATE and DELETE operations are performed in a database, the deleted or updated rows are logically deleted from the database, but actually still exist in tables. Before VACUUM is complete, such data is still stored in disks, causing table bloat. If the bloat rate reaches the percentage threshold set in an O&M task, VACUUM will be automatically triggered.	Default configuration (table bloat rate 80% or reclaimable space 100 GB.)

Step 6 Click **Next** > **Schedule** to configure scheduling for O&M tasks.

Select an O&M type.

- **One-off**: Set the start time and end time of the task.
- **Periodic**: Select a time window type, which includes **Daily**, **Weekly**, and **Monthly**, and select a time segment. Intelligent O&M will automatically analyze the time window and deliver O&M tasks accordingly.

Step 7 Click **Next: Finish**. After you confirm the information, click **Finish** to submit the request.

----End

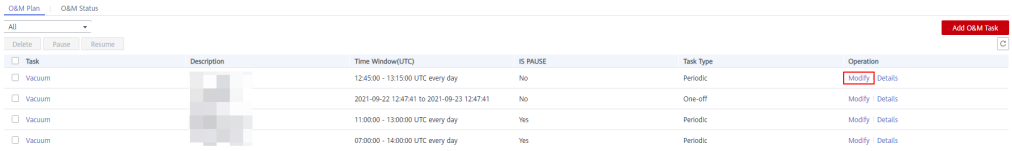
Modifying an O&M Plan

Step 1 Log in to the DWS console.

Step 2 Click the name of the target cluster.

Step 3 In the navigation pane, choose **Intelligent O&M**.

Step 4 In the **O&M Plan** area, click **Modify** in the **Operation** column of the target task.



Step 5 The **Modify O&M Task** page is displayed. The parameters for modifying an O&M task are identical to those for adding one. For details, see **Adding an O&M Plan**. The modification takes effect upon the next running.

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

----End

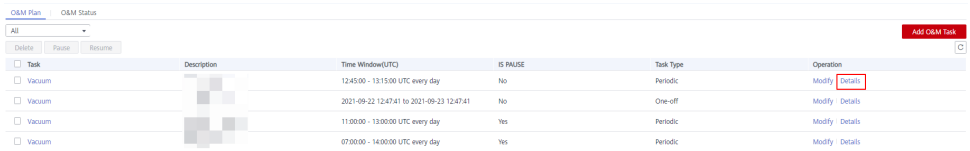
Viewing O&M Task Details

Step 1 Log in to the DWS console.

Step 2 Click the name of the target cluster.

Step 3 In the navigation pane, choose **Intelligent O&M**.

Step 4 In the **O&M Plan** area, click **Details** in the **Operation** column of the target task.



Step 5 The **O&M Task Details** panel is displayed for you to check the information.

----End

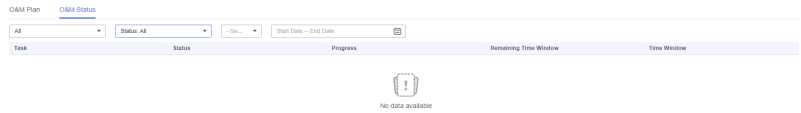
11.11.3 Viewing O&M Tasks

Step 1 Log in to the DWS console.

Step 2 Click the name of the target cluster.

Step 3 In the navigation pane, choose **Intelligent O&M**.

Step 4 Switch to the **O&M Status** area.



Step 5 Click the arrow next to the specified O&M task name to view the task details.

- **O&M Task**: **Vacuum**
- Status: **Waiting**, **Running**, **Completed**, or **Failed**.
- **Progress**
- **Remaining Time Window**
- **Time Window** (Local Time)
- Tables being vacuumed
- Tables to be vacuumed
- Vacuumed tables
- Number of tables failed to be vacuumed
- Total number of vacuumed tables
- Amount of space freed up during the vacuum process
- Vacuum details list

NOTE

- You can view up to 100 tables that are currently being vacuumed, have completed the vacuuming process, or have failed to be vacuumed.
- If the cluster is read-only, the INSERT statement cannot be executed for intelligent O&M tasks. There may be tasks remaining in the **Running** status. The **Running** status in this case is a historical status, and it indicates that the task is not completed within the specified time. If you manually pause the task and the task is not scheduled, the task may remain in the **Waiting** status. In this case, cancel the cluster read-only state and contact technical support to update the task status.

----End

12 FAQs

12.1 Product Consulting

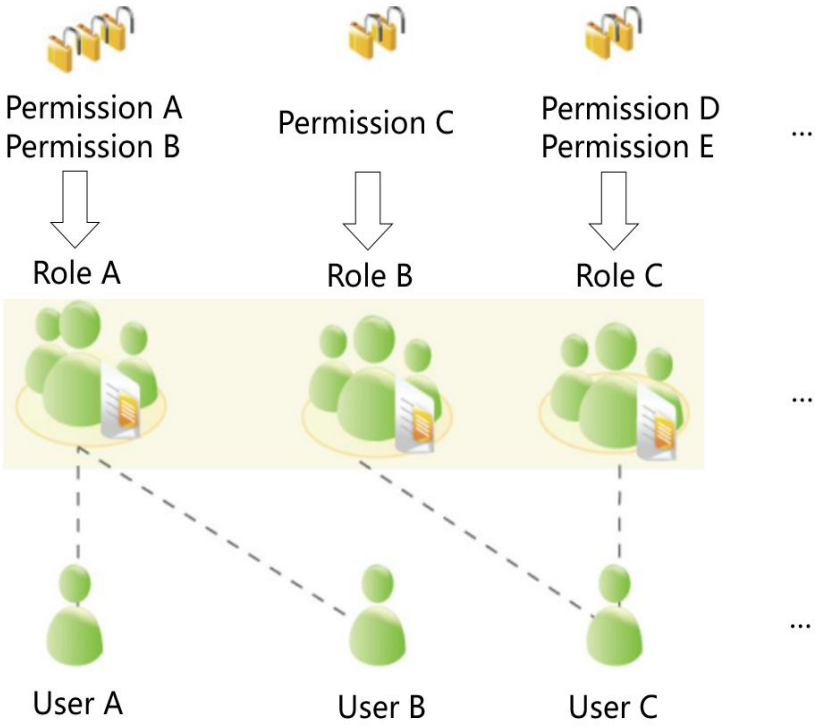
12.1.1 What Are the Differences Between DWS Users and Roles?

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 DWS include the O&M permissions for components on the management plane. You can grant 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.



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.

12.1.2 How Do I Check the Creation Time of a DWS Database User?

Method 1:

When you create a 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;
 usename | usesysid | usecreatedb | usesuper | usecatupd | userepl |  passwd  |        valbegin         | valuntil |
respool    | parent | spacelimit | useconfig | nodegroup | tempspacelimit |
 spillspacelimit
---------+----------+-------------+----------+-----------+---------+----------+-------------------------+----------
+-------------+--------+------------+-----------+-----------+----------------+
-----------------
 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)
```

12.1.3 How Do I Select a DWS Region and AZ?

Concepts

A region and availability zone (AZ) identify the location of a data center. You can create resources in regions and AZs.

- A region is a physical data center location. Each region is completely isolated to ensure high fault tolerance and stability. After creating resources in a region, you cannot change the region.
- An AZ is a physical location with independent power supplies and network in a region. A region contains one or more AZs that are physically isolated but

interconnected through high-speed internal connections. Faults that occur in one AZ will not affect other AZs. The inter-AZ connections are low-latency and inexpensive.

Selecting a Region

You are advised to select a region close to you or your target users. This reduces network latency and improves speed.

How Do I Select an AZ?

Consider your requirements for DR and network latency when selecting an AZ:

- Deploy resources in different AZs in the same region to improve DR.
- For an application that requires an extremely low latency, deploy all its resources in the same AZ.

Regions and Endpoints

When you use resources with API calls, you must specify the regional endpoint. Obtain the regions and endpoints from the enterprise administrator.

12.1.4 Is Data Secure in DWS?

Yes. In the big data era, data has become a core asset. will adhere to the commitment made over the years that we do not touch your applications or data, helping you protect your core assets. This is our commitment to users and the society, laying the foundation for the business success of and their partners.

DWS is a data warehousing system with telecom-class security to safeguard your data and privacy. Moreover, DWS delivers carrier-class quality, which can satisfy data security and privacy requirements of governments, financial organizations, and carriers. Therefore, it is widely used by various industries. DWS won the following security authentication:

- Internal Cyber Security Lab (ICSL) in compliance with cyber security standards issued by the UK authorities.
- Privacy and Security Assessment (PSA) to meet EU requirements of data security and privacy.

Service Data Security

DWS is built on software infrastructure, including ECS and OBS.

Service data of DWS users is stored in the ECSs in the cluster. Neither users nor O&M administrators can log in to these ECSs.

ECSs have their operating systems hardened through various measures such as kernel hardening, patch updates, access controls, port management, and protocol and port attack defense.

DWS provides comprehensive security measures, such as password policies, authentication, session management, user permissions management, and database auditing.

Snapshot Data Security

DWS stores its backup data in OBS as snapshots. OBS supports access permission control, key access, and data encryption features. DWS snapshots can be used for data backup and restoration only and cannot be accessed by any user. The DWS system administrator can view the OBS storage space occupied by snapshots on the DWS console and through public cloud bills.

Network Access Security

The L2 and L3 networks of DWS can be fully isolated to meet the security requirements of government and financial customers.

- DWS is deployed in a dedicated ECS environment, which is not shared with any other tenant. This eliminates the possibility of data leakage caused by compute resource sharing.
- The VMs of DWS clusters are isolated using VPCs, preventing other tenants from discovering and intruding the VMs.
- 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 DWS over SSL.
- Data imported from OBS is encrypted.

12.1.5 Can I Modify the Security Group of a DWS Cluster?

After a DWS cluster is created, you can add, delete, or modify rules in its current security group.

Modify an existing security group rule:

1. Log in to the DWS console.
2. In the navigation pane on the left, choose **Cluster** > **Cluster List**.
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.

12.1.6 How Are Dirty Pages Generated in DWS?

Causes

DWS employs the multi-version concurrency control (MVCC) mechanism to guarantee consistency and concurrency when multiple transactions access the database. Its advantages include unblocked read-write operations, while its disadvantages include disk bloating issues. The MVCC mechanism is the primary cause of dirty pages.

The scenarios are as follows:

- When the DELETE operation is performed on a table, data is logically deleted but not physically deleted from the disk.

- When an UPDATE operation is performed on a table, DWS logically marks the original data to be updated for deletion while inserting new data.

For the DELETE and UPDATE operations in a table, the data marked as deleted is called discarded tuples. The proportion of discarded tuples in the entire table is the dirty page rate. Therefore, when the dirty page rate of a table is high, the proportion of data marked as deleted in the table is high.

Solution:

With DWS, you can easily query the dirty page rate through a system view. For details, see "PGXC_STAT_TABLE_DIRTY" in *Data Warehouse Service (DWS) Development Guide*.

DWS offers the **VACUUM** function to address disk space bloat resulting from high dirty page rates. This function clears data marked for DELETE and UPDATE. For details, see "VACUUM" in the *Data Warehouse Service (DWS) SQL Syntax Reference*.

VACUUM does not release the allocated space. To completely reclaim the cleared space, run **VACUUM FULL**.

NOTE

- **VACUUM FULL** clears and releases the space of deleted data, improving database performance and efficiency. However, running **VACUUM FULL** consumes more time and resources, and may cause some tables to be locked. Therefore, run **VACUUM FULL** only when the database load is light.
- To reduce the impact of disk bloat on database performance, you are advised to do **VACUUM FULL** on non-system catalogs whose dirty page rate exceeds 80%. You can determine whether to do **VACUUM FULL** based on service scenarios.

12.2 Database Connections

12.2.1 How Applications Communicate with DWS?

To communicate applications with DWS, make sure the networks between them are connected. The following table lists common connection scenarios.

Table 12-1 Communication between applications and DWS

Scenario		Description	Supported Connection Type
Cloud	**Application and DWS Are in the Same VPC in the Same Region**	Two private IP addresses in the same VPC can directly communicate with each other.	• gsql • Data Studio • JDBC/ODBC

Scenario		Description	Supported Connection Type
	Applications and DWS Are in Different VPCs in the Same Region	After a **VPC peering** connection is created between two VPCs, the two private IP addresses can directly communicate with each other.	
	Applications and DWS Are in Different Regions	After a **cloud connection** (CC) is established between two regions, the two regions communicate with each other through private IP addresses.	
On-premises and on-cloud	Applications are deployed in on-premise data centers and need to communicate with DWS.	• Use the **public IP address** of DWS for communication. • Use **Direct Connect** (DC) is for communication.	

Application and DWS Are in the Same VPC in the Same Region

To ensure low latency, you are advised to deploy applications and DWS in the same region. For example, if an application is deployed on an ECS, you are advised to deploy the DWS cluster in the same VPC as the ECS. In this way, the application can directly communicate with DWS through a private 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 DWS cluster and ensure that the 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 DWS is **192.168.120.2**. Therefore, they can communicate with each other through private IP addresses.

Perform the following operations to check the ECS outbound rules and DWS inbound rules:

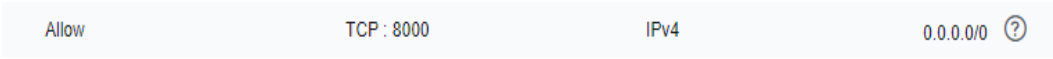
Step 1 Check the ECS outbound rules:

Ensure that the outbound rule of the ECS security group allows access. If access is not allowed, see .

Action	Protocol & Port	Type	Destination
Allow	All	IPv4	0.0.0.0/0

Step 2 Check the DWS inbound rules:

If no security group is configured when 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



Step 3 Log in to the ECS. If the private IP address of 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**

Applications and DWS Are in Different VPCs in the Same Region

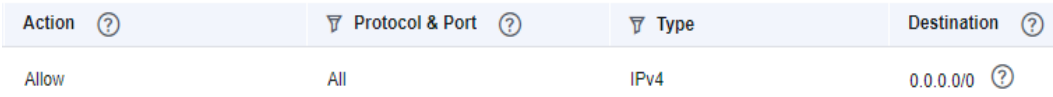
To ensure low latency, you are advised to deploy applications and DWS in the same region. For example, if applications are deployed on an ECS, you are advised to deploy the DWS cluster in the same VPC as the ECS. If a different VPC is selected for the DWS cluster, the ECS cannot directly connect to DWS.

For example, both ECS and DWS are deployed in , but ECS is in **VPC1** and DWS is in **VPC2**. In this case, you need to create a between **VPC1** and **VPC2** so that ECS can access DWS using the private IP address of DWS.

Perform the following operations to check ECS outbound rules, DWS inbound rules, and VPC peering connection between the two VPCs.

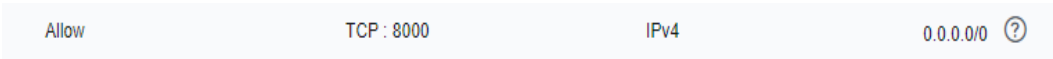
Step 1 Check the ECS outbound rules:

Ensure that the outbound rule of the ECS security group allows access. If access is not allowed, see .



Step 2 Check the DWS inbound rules:

If no security group is configured when 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



Step 3 Create a between **VPC1** where the ECS is and **VPC2** where DWS is.

Step 4 Log in to the ECS. If the private IP address of 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**

Applications and DWS Are in Different Regions

If the application and DWS are in different regions, for example, ECS is in and DWS is in , you need to establish a between the two regions for communication.

Applications are deployed in on-premise data centers and need to communicate with DWS.

If applications are not on the cloud but in the local data center, they need to communicate with DWS on the cloud.

- **Scenario 1**: On-premises applications communicate with DWS through 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 applications cannot access the external network. In this case, use .

12.2.2 Does DWS Support Third-Party Clients and JDBC and ODBC Drivers?

Yes, but DWS clients and drivers are recommended. Unlike open-source PostgreSQL clients and drivers, DWS clients and drivers have two key advantages:

- **Security hardening**: PostgreSQL drivers only support MD5 authentication, but DWS drivers support both SHA256 and MD5.
- **Data type enhancement**: DWS drivers support new data types smalldatetime and tinyint.

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 how to use JDBC/ODBC to connect to DWS, see .

NOTICE

- You are advised to use the officially recommended method for connecting to the database. For details, see .
- Compatibility with other clients cannot be guaranteed, so it may be necessary to verify it.
- If an error occurs due to incompatibility with another client and the client cannot be replaced, try replacing the libpq driver on the client. Download and decompress the gsql client package, obtain the **libpg.so** file in the **gsql** directory, and replace the **libpg.so** file in the specified directory on the client. For details, see .

12.2.3 How Do I Do If I Cannot Connect to a DWS Cluster?

Possible Causes

Check the following:

- Whether the cluster status is normal.
- Whether the connection command, username, password, IP address, and port number are incorrect.
- Whether the operating system type and version of the client are correct.
- Whether the client is incorrectly installed.

If cluster connection failed on the , check for the following as well:

- The ECSs are not in the same AZ, VPC, subnet, and security group as the cluster.
- Some of the inbound and outbound rules of the security group are incorrect.

If cluster connection failed through the Internet, confirm the following:

- Whether your network is connected to the Internet.
- Whether the firewall blocked the access.
- Whether you need to access the Internet through a proxy.

Contacting Customer Service

If the fault cannot be identified, submit a service ticket to report the problem. Log in to the management console and choose .

12.2.4 Why Was I Not Notified of Failure After Unbinding the EIP When DWS Is Connected Over the Internet?

The network is disconnected when the EIP is unbound. However, the TCP layer does not detect a faulty physical connection in time due to keepalive settings. As a result, the gsql, ODBC, and JDBC clients also cannot identify the network fault in time.

The time for the client to wait for the database to return is related to the setting of the **keepalive** parameter, and may be specifically expressed as: **keepalive_time** + **keepalive_probes***keepalive_intvl**.

Keepalive values affect network communication stability. Adjust them to service pressure and network conditions.

On Linux, run the **sysctl** command to modify the following parameters:

- net.ipv4.tcp_keepalive_time
- net.ipv4.tcp_keeaplive_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.

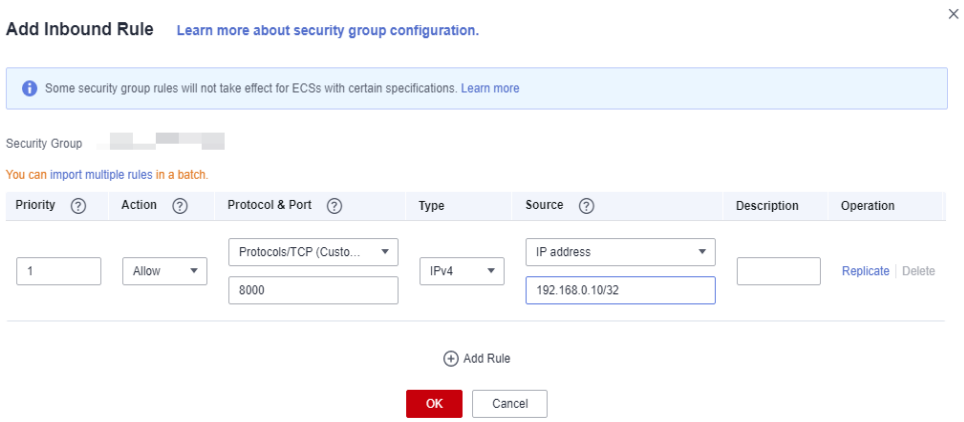
12.2.5 How Do I Configure a Whitelist If I Want to Connect to a DWS Cluster Using EIP?

You can log in to the VPC management console to manually create a security group. Then, return to the DWS cluster creation page, click  next to the **Security Group** drop-down list to refresh the page, and select the newly created security group.

To enable the DWS client to connect to the cluster, add an inbound rule to the new security group to allow access to the DWS cluster's database port.

- **Protocol**: **TCP**
- **Port**: **8000** Use the database port set when creating the DWS cluster. This port receives client connections to DWS.
- **Source**: Select **IP address** and use the host IP address of the client host, for example, **192.168.0.10/32**.

Figure 12-1 Adding an inbound rule



The whitelist will be added.

12.2.6 What Are the Differences Between API Access and Direct Database Connection?

You can compare the two connection methods from the perspectives of security and performance.

Security

Table 12-2 Security comparison

Dimension	API Access	Direct Database Access
Communication encryption	HTTPS and TLS encryption are provided.	SSL must be manually configured.
Identity authentication	AK/SK or token authentication is supported, which is more flexible.	Database account password authorization is used. The permission granularity is coarse (for example, schema or table level).
Access control	The IP address, frequency, and blacklist can be restricted at the API gateway layer.	Whitelist is used for managing database networks (such as VPCs and security groups).
SQL injection prevention	Parameter-based APIs can naturally prevent SQL injection.	Code is used to prevent SQL injection.

Dimension	API Access	Direct Database Access
Audit and logging	API calling logs (request sources and parameters) are recorded in a centralized manner.	There are database logs (SQL statements and source IP addresses).

API access is more secure especially when your database is exposed to external systems. Direct connection to your database requires additional security configurations.

Performance

Table 12-3 Performance comparison

Dimension	API Access	Direct Database Access
Network overhead	Additional HTTP protocol headers are required, and the serialization and deserialization costs are high.	Efficient binary protocols (such as libpq) are used.
Latency	Higher delay (To access a database, the API gateway and application server need to be accessed first).	Lower latency (direct communication with a database)
Throughput	The throughput is limited by the API server performance.	The high-performance database engine is directly used to deploy high throughput.
Long connection reuse	Short connections are usually used, because connections need to be frequently established.	Using connection pools helps to alleviate connection management overhead.
Complex queries	Complex SQL statements cannot be directly transmitted.	Native SQL is supported, and the optimizer can maximize the database performance.

Direct database connection is more efficient, especially in complex query or high-throughput scenarios. API access is suitable for lightweight, high-frequency simple requests.

Scenarios

The two methods are suitable for different scenarios.

- Direct database connection is better for operations on a large amount of data. The security is not high. You need to provide the database name, account, and password.
- API access is an ideal choice for low-frequency access and fixed operations. It features high security, requires a service layer, and supports field control.

12.3 Data Migration

12.3.1 What Are the Differences Between Data Formats Supported by OBS and GDS Foreign Tables in DWS?

The file formats supported by OBS and GDS foreign tables are as follows:

OBS supports ORC, text, JSON, CSV, CarbonData, and Parquet file formats for data import and ORC, CSV, Text, and Parquet file formats for data export. The default format is text.

GDS supports the following file formats: TEXT, CSV, and FIXED. The default format is TEXT.

12.3.2 How Is Data Stored in DWS?

DWS efficiently imports data from various sources through several modes. For details, see section "Importing Data" in the *Data Warehouse Service (DWS) Developer Guide*.

- Importing data from the OBS

  Upload data to OBS and then export it to DWS clusters. Data formats such as CSV and TEXT are supported.
- Inserting data with **INSERT** statements

  Use the gsql client tool provided by DWS or the JDBC/ODBC driver to write data to DWS from upper-layer applications. DWS supports complete database transaction-level CRUD operations. This is the simplest method and is applicable to scenarios with small data volume and low concurrency.
- Importing data with the **COPY FROM STDIN** command

  Run the **COPY FROM STDIN** command to write data to a table.

12.3.3 How Much Data Can Be Stored in DWS?

Each node in a DWS cluster has the storage of . A cluster can house 3 to 256 nodes and the total storage 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 DWS system creates backups, indexes, temporary cache files, and run logs that take up storage. Each node stores data that accounts for approximately half of the total storage capacity.

12.3.4 How Do I Import and Export Data in DWS Using \copy?

DWS is a fully managed cloud service. As a result, users cannot directly access the background to import or export data using **COPY**, which is therefore disabled. You are advised to store data files on OBS and use OBS foreign tables to import data. If you want to use **COPY** to import and export data, perform the following operations:

1. Place the data file on the client.
2. Use gsql to connect to the target cluster.
3. Run the following command to import data. Enter the directory name and file name of the data file on the client and specify the import option in **with**. The command is almost the same as the common **COPY** command. You only need to add a backslash (\) before the command. When the data is successfully imported, no notification will be displayed.
   ```
   \copy tb_name from '/directory_name/file_name' with(...);
   ```
4. Run the following command to export data to a local file. Retain the default settings of parameters.
   ```
   \copy table_name to '/directory_name/file_name';
   ```
5. Specify the **copy_option** parameter to export data to a CSV file.
   ```
   \copy table_name to '/directory_name/file_name' CSV;
   ```
6. Use **with** to specify parameters, exporting data as CSV files that use vertical bars (|) as delimiters.
   ```
   \copy table_name to '/directory_name/file_name' with(format 'csv',delimiter '|') ;
   ```

12.3.5 How Do I Implement Fault Tolerance Import Between Different DWS Encoding Libraries?

To import data from database A (UTF8) to database B (GBK), there may be a character set mismatch error which causes the data import to fail.

To import a small amount of data, run the **\COPY** command. The procedure is as follows:

Step 1 Create databases A and B. The encoding format of database A is UTF8, and that of database B is GBK.

```
postgres=> CREATE DATABASE A ENCODING 'UTF8' template = template0;
postgres=> CREATE DATABASE B ENCODING 'GBK' template = template0;
```

Step 2 View the database list. You can view the created databases A and B.

```
postgres=> \l
                                  List of databases
   Name    |  Owner  | Encoding  | Collate | Ctype | Access privileges
-----------+---------+-----------+---------+-------+-------------------
 a         | dbadmin | UTF8      | C       | C     |
 b         | dbadmin | GBK       | C       | C     |
 gaussdb   | Ruby    | SQL_ASCII | C       | C     |
 postgres  | Ruby    | SQL_ASCII | C       | C     |
 template0 | Ruby    | SQL_ASCII | C       | C     | =c/Ruby          +
           |         |           |         |       | Ruby=CTc/Ruby
 template1 | Ruby    | SQL_ASCII | C       | C     | =c/Ruby          +
           |         |           |         |       | Ruby=CTc/Ruby
```



```
 xiaodi    | dbadmin | UTF8      | C       | C     |
(7 rows)
```

Step 3 Switch to database A and enter the user password. Create a table named **test01** and insert data into the table.

```
postgres=> \c a
Password for user dbadmin:
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_128_GCM_SHA256, bits: 128)
You are now connected to database "a" as user "dbadmin".

a=> CREATE TABLE test01
 (
    c_customer_sk             integer,
    c_customer_id             char(5),
    c_first_name              char(6),
    c_last_name               char(8)
 )
 with (orientation = column,compression=middle)
 distribute by hash (c_last_name);
CREATE TABLE
a=> INSERT INTO test01(c_customer_sk, c_customer_id, c_first_name) VALUES (3769, 'hello', 'Grace');
INSERT 0 1
a=> INSERT INTO test01 VALUES (456, 'good');
INSERT 0 1
```

Step 4 Run the **\COPY** command to export data from the UTF8 library in Unicode format to the **test01.dat** file.

```
\copy test01 to '/opt/test01.dat' with (ENCODING 'Unicode');
```

Step 5 Switch to database B and create a table with the same name **test01**.

```
a=> \c b
Password for user dbadmin:
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_128_GCM_SHA256, bits: 128)
You are now connected to database "b" as user "dbadmin".

b=> CREATE TABLE test01
 (
    c_customer_sk             integer,
    c_customer_id             char(5),
    c_first_name              char(6),
    c_last_name               char(8)
 )
 with (orientation = column,compression=middle)
 distribute by hash (c_last_name);
```

Step 6 Run the **\COPY** command to import the **test01.dat** file to database B.

```
\copy test01 from '/opt/test01.dat' with (ENCODING 'Unicode' ,COMPATIBLE_ILLEGAL_CHARS 'true');
```

NOTE

- The error tolerance parameter **COMPATIBLE_ILLEGAL_CHARS** specifies that invalid characters are tolerated during data import. Invalid characters are converted and then imported to the database. No error message is displayed. The import is not interrupted.
- The BINARY format is not supported. When data of such format is imported, error "cannot specify bulkload compatibility options in BINARY mode" will occur.
- The parameter is valid only for data importing using the **COPY FROM** option.

Step 7 View data in the **test01** table in database B.

```
b=> select * from test01;
 c_customer_sk | c_customer_id | c_first_name | c_last_name
---------------+---------------+--------------+-------------
          3769 | hello         | Grace        |
           456 | good          |              |
(2 rows)
```


Step 8 After the preceding operations are performed, data is imported from database A (UTF8) to database B (GBK).

----End

12.3.6 Which Factors Are Related to DWS Import Performance?

The 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

12.4 Database Use

12.4.1 How Do I Adjust DWS 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 DNs will be poor, affecting the overall query performance. Proper selection and adjustment of distribution columns are critical to table query performance.

If the hash distribution policy is used, you need to check tables to ensure their data is evenly distributed on each DN. Generally, over 5% difference between the amount of data on different DNs 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, specify the distribution column when rebuilding a table.**

Step 1 Use Data Studio or gsql in Linux to access the database.

Step 2 Create a table.

NOTE

In the following statements, **table1** is the original table name and **table1_new** is the new table name. **column1** and **column2** are distribution column names.

```
CREATE TABLE IF NOT EXISTS table1_new
( LIKE table1 INCLUDING ALL EXCLUDING DISTRIBUTION)
DISTRIBUTE BY
HASH (column1, column2);
```

Step 3 Migrate data to the new table.

```
START TRANSACTION;
LOCK TABLE table1 IN ACCESS EXCLUSIVE MODE;
INSERT INTO table1_new SELECT * FROM table1;
COMMIT;
```

Step 4 Verify that the table data has been migrated. Delete the original table.

```
SELECT COUNT(*) FROM table1_new;
DROP TABLE table1;
```

Step 5 Replace the original table.

```
ALTER TABLE table1_new RENAME TO table1;
```

----End

- In version 8.1.0 or later, 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 Check the error reported when data in the distribution column is updated.

```
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

12.4.2 How Do I View and Set the Character Set Encoding Format of a DWS Database?

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

Once a database is created, its encoding format cannot be modified on DWS.

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. DWS creates a database by copying data from a database template. DWS initially has two database templates: **template0** and **template1**, as well as a default user database .

  Value range: an existing database name. If this is not specified, the system copies **template1** by default. Its value cannot be .

  NOTICE

  Currently, database templates cannot contain sequences. If sequences exist in the template library, database creation will fail.

- **ENCODING [=] encoding**

  Character encoding used by the database. The value can be a character string (for example, **SQL_ASCII'**) or an integer number.

  By default, the encoding format of the template database is used. 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;
```

12.4.3 How Do I Do If a Field of the Date Type Is Automatically Converted to a Timestamp Type During Table Creation in DWS?

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). The date type is only supported in the MySQL compatibility mode.

To solve the problem, you need to change the compatibility mode to MySQL. DWS does not allow you to modify the compatibility mode of an existing database. You can only specify the compatibility mode when creating a database. DWS supports the MySQL compatibility mode in clusters of version 8.1.1 or 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
```

If the problem cannot be solved by changing the compatibility, you can try to change the column type. For example, insert data of the date type as strings into a table. Example:

```
gaussdb=> CREATE TABLE mytable (a date,b int);
CREATE TABLE
gaussdb=> INSERT INTO mytable VALUES(date '12-08-2023',01);
INSERT 0 1
gaussdb=> SELECT * FROM mytable;
         a          | b
--------------------+---
 2023-12-08 00:00:00 | 1
(1 row)
gaussdb=> ALTER TABLE mytable MODIFY a VARCHAR(20);
ALTER TABLE
gaussdb=> INSERT INTO mytable VALUES('2023-12-10',02);
INSERT 0 1
gaussdb=> SELECT * FROM mytable;
         a          | b
--------------------+---
 2023-12-08 00:00:00 | 1
 2023-12-10          | 2
(2 rows)
```

12.4.4 Do I Need to Run VACUUM FULL and ANALYZE on Common Tables Periodically in DWS?

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;
```

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 performed on the management plane. For details, see "Reclaiming DWS Space Using Vacuum" in *Data Warehouse Service (DWS) User Guide*.

VACUUM and VACUUM FULL

In DWS, **VACUUM** is essentially a vacuum cleaner used to absorb dust. Here, "dust" means old data. If the data is not cleared in a timely manner, more database space will be used to store such data, causing performance downgrade or even a system breakdown.

Purposes of VACUUM:

- Solve space bloat: Clear obsolete tuples and corresponding indexes, which include the tuple (and index) of a committed **DELETE** transaction, the old version (and index) of an **UPDATE** transaction, the inserted tuple (and index) of a rolled back **INSERT** transaction, the new version (and index) of an **UPDATE** transaction, and the tuple (and index) of a **COPY** transaction.
- VACUUM FREEZE: Prevents system breakdown caused by transaction ID wraparound. It converts transaction IDs smaller than OldestXmin to freeze xids, update relfrozenxids in a table, and update relfrozenxids and truncate clogs in a database.
- Update statistics: **VACUUM ANALYZE** updates statistics, enabling the optimizer to select a better way to execute SQL statements.

The VACUUM statement includes **VACUUM** and **VACUUM FULL**. Currently, **VACUUM** can only work on row-store tables. **VACUUM FULL** can be used to release space of column-store tables. For details, see the following table.

Table 12-4 VACUUM and VACUUM FULL

Item	VACUUM	VACUUM FULL
Clearing space	If the deleted record is at the end of a table, the space occupied by the deleted record is physically released and returned to the operating system. If the data is not at the end of a table, the space occupied by dead tuples in the table or index is set to be available for reuse.	Despite the position of the deleted data, the space occupied by the data is physically released and returned to the operating system. When data is inserted, a new disk page is allocated.
Lock type	Shared lock. The **VACUUM** operation can be performed in parallel with other operations.	Exclusive lock. All operations based on the table are suspended during execution.
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.

12.4.5 How Do I Export a DWS Table Schema?

Using SQL Editor

You are advised to use SQL Editor to export table data. Log in to a data source, select the corresponding database and schema, enter the SQL statement for querying table data, and click **Export**. The following export methods are supported:

- Local export: Export all SQL query results to an XLSX or CSV file. You can open the file on your local PC. A maximum of 20,000 records can be exported.
- Full export: Export all query SQL results to a specified path in an OBS bucket. By default, the results are exported to a CSV file.

12.4.6 Does DWS Provide an Efficient Way to Delete Table Data?

Yes. **TRUNCATE** is more efficient than **DELETE** for deleting massive data.

Function

TRUNCATE quickly removes all rows from a table. It has the same effect as an unqualified DELETE but since it does not actually scan the table it is faster. This is most effective on large tables.

Functions

- **TRUNCATE TABLE** works like a **DELETE** statement with no **WHERE** clause, that is, emptying a table.
- **TRUNCATE TABLE** uses less system and transaction log resources.
  - **DELETE** deletes a row each time, and records each deletion in the transaction log.
  - **TRUNCATE TABLE** deletes all rows in a table by releasing the data page, and only records each releasing of the data page in the transaction log.
- **TRUNCATE**, **DELETE**, and **DROP** are different in that:
  - **TRUNCATE TABLE** deletes content, releases space, but does not delete definitions.
  - **DELETE TABLE** deletes content, but does not delete definitions or release space.
  - **DROP TABLE** deletes content and definitions, and releases space.

Examples

- Create a table.

```
CREATE TABLE tpcds.reason_t1 AS TABLE tpcds.reason;
```

  Truncate the table.

```
TRUNCATE TABLE tpcds.reason_t1;
```

  Delete the table.

```
DROP TABLE tpcds.reason_t1;
```

- Create a partitioned table.

```
CREATE TABLE tpcds.reason_p
(
  r_reason_sk integer,
  r_reason_id character(16),
  r_reason_desc character(100)
)PARTITION BY RANGE (r_reason_sk)
(
  partition p_05_before values less than (05),
  partition p_15 values less than (15),
  partition p_25 values less than (25),
  partition p_35 values less than (35),
  partition p_45_after values less than (MAXVALUE)
);
```

  Insert data.

```
INSERT INTO tpcds.reason_p SELECT * FROM tpcds.reason;
```

  Truncate the **p_05_before** partition.

```
ALTER TABLE tpcds.reason_p TRUNCATE PARTITION p_05_before;
```

  Truncate the partition **p_15** where 13 is located.

```
ALTER TABLE tpcds.reason_p TRUNCATE PARTITION for (13);
```

  Truncate the partitioned table.

```
TRUNCATE TABLE tpcds.reason_p;
```


Delete the table.

```
DROP TABLE tpcds.reason_p;
```

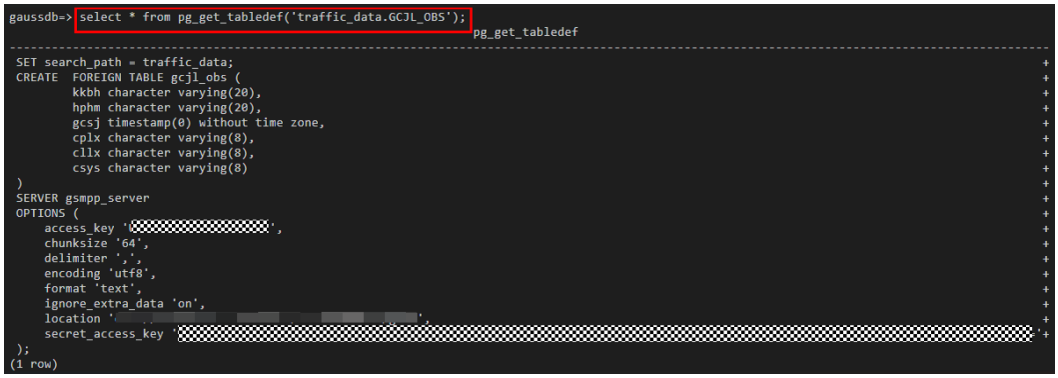
12.4.7 How Do I View DWS Foreign Table Information?

To query information about OBS/GDS foreign tables such as OBS paths, run the following statement:

```
SELECT * FROM pg_get_tabledef ('foreign_table_name')
```

The following uses table **traffic_data.GCJL_OBS** as an example:

```
SELECT * FROM pg_get_tabledef('traffic_data.GCJL_OBS');
```



12.4.8 How Will Data Be Stored in a DWS Table If No Distribution Column Is Specified During Its Creation?

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)
```

12.4.9 How Do I Replace the Null Results with 0 in a DWS Join Query?

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)
```

12.4.10 How Do I Check Whether a DWS 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)
```


12.4.11 How Do I Query DWS Column-Store Table Information?

The following SQL statements are used to query common information about column-store tables:

Create a column-store table named my_table, and insert data into the table.

```
CREATE TABLE my_table
(
    product_id INT,
    product_name VARCHAR2(40),
    product_quantity INT
 )
WITH (ORIENTATION = COLUMN)
PARTITION BY range(product_quantity)
(
partition my_table_p1 values less than(600),
partition my_table_p2 values less than(800),
partition my_table_p3 values less than(950),
partition my_table_p4 values less than(1000));

INSERT INTO my_table VALUES(1011, 'tents', 720);
INSERT INTO my_table VALUES(1012, 'hammock', 890);
INSERT INTO my_table VALUES(1013, 'compass', 210);
INSERT INTO my_table VALUES(1014, 'telescope', 490);
INSERT INTO my_table VALUES(1015, 'flashlight', 990);
INSERT INTO my_table VALUES(1016, 'ropes', 890);
```

Run the following command to view the created column-store partitioned table:

```
SELECT * FROM my_table;
 product_id | product_name | product_quantity
------------+--------------+------------------
       1013 | compass      |              210
       1014 | telescope    |              490
       1011 | tents        |              720
       1015 | flashlight   |              990
       1012 | hammock      |              890
       1016 | ropes        |              890
(6 rows)
```

Querying the Boundary of a Partition

```
SELECT relname, partstrategy, boundaries FROM pg_partition where parentid=(select parentid from
pg_partition where relname='my_table');
   relname   | partstrategy | boundaries
-------------+--------------+------------
 my_table    | r            |
 my_table_p1 | r            | {600}
 my_table_p2 | r            | {800}
 my_table_p3 | r            | {950}
 my_table_p4 | r            | {1000}
(5 rows)
```

Querying the Number of Columns in a Column-Store Table

```
SELECT count(*) FROM ALL_TAB_COLUMNS where table_name='my_table';
 count
-------
     3
(1 row)
```

Querying Data Distribution on DNs

```
SELECT table_skewness('my_table');
          table_skewness
```



```
-------------------------------------
 ("dn_6007_6008        ",3,50.000%)
 ("dn_6009_6010        ",2,33.333%)
 ("dn_6003_6004        ",1,16.667%)
 ("dn_6001_6002        ",0,0.000%)
 ("dn_6005_6006        ",0,0.000%)
 ("dn_6011_6012        ",0,0.000%)
(6 rows)
```

Querying the Names of the Cudesc and Delta Tables in Partition P1 on a DN

```
EXECUTE DIRECT ON (dn_6003_6004) 'select a.relname from pg_class a, pg_partition b where
(a.oid=b.reldeltarelid or a.oid=b.relcudescrelid) and b.relname=''my_table_p1''';
      relname
----------------------
 pg_delta_part_60317
 pg_cudesc_part_60317
(2 rows)
```

12.4.12 Why Is the Index Invalid During DWS Query?

Creating indexes for tables can improve database query performance. However, sometimes indexes cannot be used in a query plan. This section describes several common reasons and optimization methods.

Reason 1: The Returned Result Sets Are Large.

The following uses Seq Scan and Index Scan on a row-store table as an example:

- Seq Scan: searches table records in sequence. All records are retrieved during each scan. This is the simplest and most basic table scanning method, and its cost is high.
- Index Scan: searches the index first, find the target location (pointer) in the index, and then retrieve data on the target page.

Index scan is faster than sequence scan in most cases. However, if the obtained result sets account for a large proportion (more than 70%) of all data, Index Scan needs to scan indexes before reading table data. This makes it slower table scan.

Reason 2: ANALYZE Is Not Performed In a Timely Manner.

ANALYZE is used to update table statistics. If **ANALYZE** is not executed on a table or a large amount of data is added to or deleted from a table after **ANALYZE** is executed, the statistics may be inaccurate, which may cause a query to skip the index.

Optimization method: Run the **ANALYZE** statement on the table to update statistics.

Reason 3: Filtering Conditions Contains Functions or Implicit Data Type Conversion

If calculation, function, or implicit data type conversion is contained in filter criteria, indexes may fail to be selected.

For example, when a table is created, indexes are created in columns **a**, **b**, and **c**.

```
CREATE TABLE test(a int, b text, c date);
```


- Perform calculation on the indexed columns.

  The following command output indicates that both **where a = 101** and **where a = 102 - 1** use the index in column a, but **where a + 1 = 102** does not use the index.

```
explain verbose select * from test where a  = 101;
                                      QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 id |                 operation                 | E-rows | E-distinct | E-memory | E-width | E-costs
----+-------------------------------------------+--------+------------+----------+---------+---------
  1 | ->  Streaming (type: GATHER)              |      1 |            |          |      44 | 16.27
  2 |    ->  Index Scan using index_a on public.test |      1 |            | 1MB      |      44 | 8.27

Predicate Information (identified by plan id)
-----------------------------------------------
  2 --Index Scan using index_a on public.test
        Index Cond: (test.a = 101)

Targetlist Information (identified by plan id)
------------------------------------------------
  1 --Streaming (type: GATHER)
        Output: a, b, c
        Node/s: dn_6005_6006
  2 --Index Scan using index_a on public.test
        Output: a, b, c
        Distribute Key: a

  ====== Query Summary =====
--------------------------------
System available mem: 3358720KB
Query Max mem: 3358720KB
Query estimated mem: 1024KB
(24 rows)
 explain verbose select * from test where a  = 102 - 1;
                                      QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 id |                 operation                 | E-rows | E-distinct | E-memory | E-width | E-costs
----+-------------------------------------------+--------+------------+----------+---------+---------
  1 | ->  Streaming (type: GATHER)              |      1 |            |          |      44 | 16.27
  2 |    ->  Index Scan using index_a on public.test |      1 |            | 1MB      |      44 | 8.27

Predicate Information (identified by plan id)
-----------------------------------------------
  2 --Index Scan using index_a on public.test
        Index Cond: (test.a = 101)

Targetlist Information (identified by plan id)
------------------------------------------------
  1 --Streaming (type: GATHER)
        Output: a, b, c
        Node/s: dn_6005_6006
  2 --Index Scan using index_a on public.test
        Output: a, b, c
        Distribute Key: a

  ====== Query Summary =====
--------------------------------
System available mem: 3358720KB
Query Max mem: 3358720KB
Query estimated mem: 1024KB
(24 rows)
explain verbose select * from test where a + 1 = 102;
                                    QUERY PLAN
-----------------------------------------------------------------------------------------------
 id |          operation           | E-rows | E-distinct | E-memory | E-width | E-costs
----+------------------------------+--------+------------+----------+---------+---------
  1 | ->  Streaming (type: GATHER) |      1 |            |          |      44 | 22.21
  2 |    ->  Seq Scan on public.test |      1 |            | 1MB      |      44 | 14.21
```



```
 Predicate Information (identified by plan id)
 ---------------------------------------------
   2 --Seq Scan on public.test
         Filter: ((test.a + 1) = 102)

 Targetlist Information (identified by plan id)
 ----------------------------------------------
   1 --Streaming (type: GATHER)
         Output: a, b, c
         Node/s: All datanodes
   2 --Seq Scan on public.test
         Output: a, b, c
         Distribute Key: a

   ====== Query Summary =====
 -------------------------------
 System available mem: 3358720KB
 Query Max mem: 3358720KB
 Query estimated mem: 1024KB
(24 rows)
```

Optimization method: Use constants instead of expressions, or put constant calculation on the right of the equal sign (=).

- Use functions on indexed columns.

  According to the following execution result, if a function is used on an indexed column, the index fails to be selected.

```
explain verbose select * from test where to_char(c, 'yyyyMMdd') =
to_char(CURRENT_DATE,'yyyyMMdd');
                                                               QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
---------------------
 id |           operation            | E-rows | E-distinct | E-memory | E-width | E-costs
 ----+-------------------------------+--------+------------+----------+---------+---------
   1 | ->  Streaming (type: GATHER)   |      1 |            |          |      44 | 22.28
   2 |    ->  Seq Scan on public.test |      1 |            | 1MB      |      44 | 14.28

                                             Predicate Information (identified by plan id)

-----------------------------------------------------------------------------------------------------------------------
-------------------
   2 --Seq Scan on public.test
         Filter: (to_char(test.c, 'yyyyMMdd'::text) = to_char(('2022-11-30'::pg_catalog.date)::timestamp
with time zone, 'yyyyMMdd'::text))

 Targetlist Information (identified by plan id)
 ----------------------------------------------
   1 --Streaming (type: GATHER)
         Output: a, b, c
         Node/s: All datanodes
   2 --Seq Scan on public.test
         Output: a, b, c
         Distribute Key: a

   ====== Query Summary =====
 -------------------------------
 System available mem: 3358720KB
 Query Max mem: 3358720KB
 Query estimated mem: 1024KB
(24 rows)
explain verbose select * from test where c = current_date;
                                                 QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 id |                  operation                   | E-rows | E-distinct | E-memory | E-width | E-costs
 ----+----------------------------------------------+--------+------------+----------+---------+---------
   1 | ->  Streaming (type: GATHER)                  |      1 |            |          |      44 | 16.27
   2 |    ->  Index Scan using index_c on public.test |      1 |            | 1MB      |      44 | 8.27

        Predicate Information (identified by plan id)
```

```
-------------------------------------------------------------
  2 --Index Scan using index_c on public.test
        Index Cond: (test.c = '2022-11-30'::pg_catalog.date)

 Targetlist Information (identified by plan id)
-----------------------------------------------
  1 --Streaming (type: GATHER)
        Output: a, b, c
        Node/s: All datanodes
  2 --Index Scan using index_c on public.test
        Output: a, b, c
        Distribute Key: a

  ====== Query Summary =====
------------------------------
 System available mem: 3358720KB
 Query Max mem: 3358720KB
 Query estimated mem: 1024KB
(24 rows)
```

Optimization method: Do not use unnecessary functions on indexed columns.

- Implicit conversion of data types.

  This scenario is common. For example, the type of column **b** is Text, and the filtering condition is **where b = 2**. During plan generation, the Text type is implicitly converted to the Bigint type, and the actual filtering condition changes to **where b::bigint = 2**. As a result, the index in column **b** becomes invalid.

```
explain verbose select * from test where b = 2;
                                  QUERY PLAN
-------------------------------------------------------------------------------------------
 id |           operation            | E-rows | E-distinct | E-memory | E-width | E-costs
----+--------------------------------+--------+------------+----------+---------+---------
  1 | ->  Streaming (type: GATHER)   |      1 |            |          |      44 | 22.21
  2 |    ->  Seq Scan on public.test |      1 |            | 1MB      |      44 | 14.21

 Predicate Information (identified by plan id)
-----------------------------------------------
  2 --Seq Scan on public.test
        Filter: ((test.b)::bigint = 2)

 Targetlist Information (identified by plan id)
-----------------------------------------------
  1 --Streaming (type: GATHER)
        Output: a, b, c
        Node/s: All datanodes
  2 --Seq Scan on public.test
        Output: a, b, c
        Distribute Key: a

  ====== Query Summary =====
------------------------------
 System available mem: 3358720KB
 Query Max mem: 3358720KB
 Query estimated mem: 1024KB
(24 rows)
explain verbose select * from test where b = '2';
                                       QUERY PLAN
-----------------------------------------------------------------------------------------------------
 id |                  operation                   | E-rows | E-distinct | E-memory | E-width | E-costs
----+----------------------------------------------+--------+------------+----------+---------+---------
  1 | ->  Streaming (type: GATHER)                 |      1 |            |          |      44 | 16.27
  2 |    ->  Index Scan using index_b on public.test |      1 |            | 1MB      |      44 | 8.27

 Predicate Information (identified by plan id)
-----------------------------------------------
  2 --Index Scan using index_b on public.test
        Index Cond: (test.b = '2'::text)
```



```
Targetlist Information (identified by plan id)
------------------------------------------------
  1 --Streaming (type: GATHER)
        Output: a, b, c
        Node/s: All datanodes
  2 --Index Scan using index_b on public.test
        Output: a, b, c
        Distribute Key: a

  ====== Query Summary =====
-------------------------------
System available mem: 3358720KB
Query Max mem: 3358720KB
Query estimated mem: 1024KB
(24 rows)
```

Optimization method: Use constants of the same type as the indexed column to avoid implicit type conversion.

Scenario 4: Hashjoin Is Replaced with Nestloop + Indexscan.

When two tables are joined, the number of rows in the result set filtered by the WHERE condition in one table is small, thus the number of rows in the final result set is also small. In this case, the effect of nestloop+indexscan is better than that of hashjoin. The better execution plan is as follows:

You can see that the Index Cond: (t1.b = t2.b) at layer 5 has pushed the join condition down to the base table scanning.

```
explain verbose select t1.a,t1.b from t1,t2 where t1.b=t2.b and t2.a=4;
 id |                    operation                    | E-rows | E-distinct | E-memory | E-width | E-costs
----+-------------------------------------------------+--------+------------+----------+---------+---------
  1 | ->  Streaming (type: GATHER)                    |     26 |            |          |       8 | 17.97
  2 |    ->  Nested Loop (3,5)                        |     26 |            | 1MB      |       8 | 11.97
  3 |       ->  Streaming(type: BROADCAST)            |      2 |            | 2MB      |       4 | 2.78
  4 |          ->  Seq Scan on public.t2              |      1 |            | 1MB      |       4 | 2.62
  5 |       ->  Index Scan using t1_b_idx on public.t1 |     26 |            | 1MB      |       8 | 9.05
(5 rows)

 Predicate Information (identified by plan id)
-----------------------------------------------
   4 --Seq Scan on public.t2
         Filter: (t2.a = 4)
   5 --Index Scan using t1_b_idx on public.t1
         Index Cond: (t1.b = t2.b)
(4 rows)

 Targetlist Information (identified by plan id)
-------------------------------------------------
   1 --Streaming (type: GATHER)
         Output: t1.a, t1.b
         Node/s: All datanodes
   2 --Nested Loop (3,5)
         Output: t1.a, t1.b
   3 --Streaming(type: BROADCAST)
         Output: t2.b
         Spawn on: datanode2
         Consumer Nodes: All datanodes
   4 --Seq Scan on public.t2
         Output: t2.b
         Distribute Key: t2.a
   5 --Index Scan using t1_b_idx on public.t1
         Output: t1.a, t1.b
         Distribute Key: t1.a
(15 rows)
```



```
  ====== Query Summary =====
---------------------------------
 System available mem: 9262694KB
 Query Max mem: 9471590KB
 Query estimated mem: 5144KB
(3 rows)
```

If the optimizer does not select such an execution plan, you can optimize it as follows:

```
set enable_index_nestloop = on;
set enable_hashjoin = off;
set enable_seqscan = off;
```

Reason 5: The Scan Method Is Incorrectly Specified by Hints.

DWS plan hints can specify three scan methods: table scan, index scan, and index-only scan.

- Table scan: scans full tables. There are 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.

12.4.13 How Do I Use a User-Defined DWS Function to Rewrite the CRC32() Function?

DWS currently does not have a built-in **CRC32()** function. However, if you need to implement the **CRC32()** function in MySQL, you can rewrite it using a user-defined DWS 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 user-defined DWS function statements:

```
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.

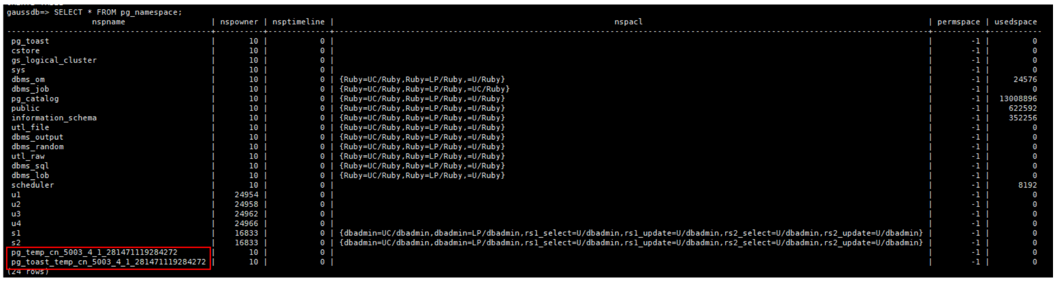
```
select crc32(null),crc32(''),crc32('1');
 crc32 | crc32 |   crc32
-------+-------+------------
       |     0 | 2212294583
(1 row)
```

For how to use user-defined functions, see "CREATE FUNCTION" in *Database Repository Service (DWS) SQL Syntax References*.

12.4.14 What Is a DWS Schema 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;
```



```
gaussdb=> SELECT * FROM pg_namespace;
               nspname                | nspowner | nsptimeline |                                                    nspacl                                                     | permspace | usedspace
--------------------------------------+----------+-------------+---------------------------------------------------------------------------------------------------------------+-----------+-----------
 pg_toast                             |       10 |           0 |                                                                                                               |        -1 |         0
 cstore                               |       10 |           0 |                                                                                                               |        -1 |         0
 gs_logical_cluster                   |       10 |           0 |                                                                                                               |        -1 |         0
 sys                                  |       10 |           0 |                                                                                                               |        -1 |         0
 dbms_om                              |       10 |           0 | {Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}                                                                           |        -1 |     24576
 dbms_job                             |       10 |           0 | {Ruby=UC/Ruby,Ruby=LP/Ruby,=UC/Ruby}                                                                          |        -1 |         0
 pg_catalog                           |       10 |           0 | {Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}                                                                           |        -1 |  13008896
 public                               |       10 |           0 | {Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}                                                                           |        -1 |    622592
 information_schema                   |       10 |           0 | {Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}                                                                           |        -1 |    352256
 utl_file                             |       10 |           0 | {Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}                                                                           |        -1 |         0
 dbms_output                          |       10 |           0 | {Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}                                                                           |        -1 |         0
 dbms_random                          |       10 |           0 | {Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}                                                                           |        -1 |         0
 utl_raw                              |       10 |           0 | {Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}                                                                           |        -1 |         0
 dbms_sql                             |       10 |           0 | {Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}                                                                           |        -1 |         0
 dbms_lob                             |       10 |           0 | {Ruby=UC/Ruby,Ruby=LP/Ruby,=U/Ruby}                                                                           |        -1 |         0
 scheduler                            |       10 |           0 |                                                                                                               |        -1 |      8192
 u1                                   |    24954 |           0 |                                                                                                               |        -1 |         0
 u2                                   |    24958 |           0 |                                                                                                               |        -1 |         0
 u3                                   |    24962 |           0 |                                                                                                               |        -1 |         0
 u4                                   |    24966 |           0 |                                                                                                               |        -1 |         0
 s1                                   |    16833 |           0 | {dbadmin=UC/dbadmin,dbadmin=LP/dbadmin,rs1_select=U/dbadmin,rs1_update=U/dbadmin,rs2_select=U/dbadmin,rs2_update=U/dbadmin} |        -1 |         0
 s2                                   |    16833 |           0 | {dbadmin=UC/dbadmin,dbadmin=LP/dbadmin,rs1_select=U/dbadmin,rs1_update=U/dbadmin,rs2_select=U/dbadmin,rs2_update=U/dbadmin} |        -1 |         0
 pg_temp_cn_5003_4_1_281471119284272  |       10 |           0 |                                                                                                               |        -1 |         0
 pg_toast_temp_cn_5003_4_1_281471119284272 |  10 |           0 |                                                                                                               |        -1 |         0
(24 rows)
```

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.

12.4.15 Solutions to Inconsistent DWS Query Results

In DWS, inconsistencies in the returned results of the same query statement when using SQL queries occur. Such issues are primarily due to improper syntax or usage. Proper service usage can prevent these problems. The following are some examples of query results inconsistency along with the solutions.

Window Function Results Are Incompletely Sorted

Scenario:

In the window function **row_number()**, column **c** of table **t3** is queried after sorting. The two query results are different.

```
SELECT * FROM t3 order by 1,2,3;
 a | b | c
---+---+---
 1 | 2 | 1
 1 | 2 | 2
 1 | 2 | 3
(3 rows)

SELECT c,rn FROM (select c,row_number() over(order by a,b) as rn from t3) where rn = 1;
 c | rn
---+----
1  |  1
(1 row)
SELECT c,rn FROM (select c,row_number() over(order by a,b) as rn from t3) where rn = 1;
 c | rn
---+----
 3 |  1
(1 row)
```

Analysis:

As shown above, run **select c,rn from (select c,row_number() over(order by a,b) as rn from t3) where rn = 1;** twice, the results are different. That is because duplicate values **1** and **2** exist in the sorting columns **a** and **b** of the window function while their values in column **c** are different. As a result, when the first record is obtained based on the sorting result in columns **a** and **b**, the obtained data in column **c** is random, as a result, the result sets are inconsistent.

Solution:

The values in column **c** need to be added to the sorting.

```
SELECT c,rn FROM (select c,row_number() over(order by a,b,c) as rn from t3) where rn = 1;
 c | rn
---+----
 1 |  1
(1 row)
```

Using Sorting in Subviews/Subqueries

Scenario

After table **test** and view **v** are created, the query results are inconsistent when sorting is used to query table **test** in a subquery.

```
CREATE TABLE test(a serial ,b int);
INSERT INTO test(b) VALUES(1);
INSERT INTO test(b) SELECT b FROM test;
...
INSERT INTO test(b) SELECT b FROM test;
CREATE VIEW v as SELECT * FROM test ORDER BY a;
```

Problem SQL:

```
SELECT * FROM v limit 1;
 a | b
---+---
 3 | 1
(1 row)

SELECT * FROM (select * from test order by a) limit 10;
 a  | b
----+---
 14 | 1
(1 row)

SELECT * FROM test order by a limit 10;
 a | b
---+---
 1 | 1
(1 row)
```

Analysis:

ORDER BY is invalid in 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.

DWS currently supports three database compatibility modes: Oracle, TD, and MySQL. The syntax and behavior vary depending on the compatibility mode. For details about the compatibility differences, see "Syntax Compatibility Differences Among Oracle, Teradata, and MySQL" in *DWS Developer Guide*.

Databases in different compatibility modes have different compatibility issues. You can run **select datname, datcompatibility from pg_database;** to check the database compatibility.

Solution:

The problem is solved when the compatibility modes of the databases in the two environments are set to the same. The **DBCOMPATIBILITY** attribute of a database does not support **ALTER**. You can only specify the same **DBCOMPATIBILITY** attribute when creating a database.

The configuration item behavior_compat_options for database compatibility behaviors is configured inconsistently.

Scenario: The calculation results of the **add_months** function are inconsistent.

database1:

```
SELECT add_months('2018-02-28',3) from dual;
add_months
---------------------
2018-05-28 00:00:00
(1 row)
```

database2:

```
SELECT add_months('2018-02-28',3) from dual;
add_months
---------------------
2018-05-31 00:00:00
(1 row)
```

Analysis:

Some behaviors may vary depending on the settings of the database compatibility configuration item **behavior_compat_options**. For details about the options of this item, see "GUC Parameters" > "Miscellaneous Parameters" > "behavior_compat_options" in *DWS Developer Guide*..

The **end_month_calculate** in **behavior_compat_options** controls the calculation logic of the **add_months** function. If this parameter is specified, and the **Day** of **param1** indicates the last day of a month shorter than **result**, the **Day** in the calculation result will equal that in **result**.

Solution:

The **behavior_compat_options** parameter must be configured consistently. This parameter is of the **USERSET** type and can be set at the session level or modified at the cluster level.

The attributes of the user-defined function are not properly set.

Scenario: When the customized function **get_count()** is invoked, the results are inconsistent.

```
CREATE FUNCTION get_count() returns int
SHIPPABLE
as $$
declare
    result int;
begin
result = (select count(*) from test); --test table is a hash table.
    return result;
end;
$$
language plpgsql;
```

Call this function.

```
SELECT get_count();
 get_count
-----------
      2106
(1 row)

SELECT get_count() FROM t_src;
 get_count
-----------
      1032
(1 row)
```

Analysis:

This function specifies the **SHIPPABLE** attribute. When a plan is generated, the function pushes it down to DNs for execution. The test table defined in the function is a hash table. Therefore, each DN has only part of the data in the table, the result returned by **select count(*) from test;** is not the result of full data in the test table. The expected result changes after **from** is added.

Solution:

Use either of the following methods (the first method is recommended):

1. Change the function to not push down: **ALTER FUNCTION get_count() not shippable;**
2. Change the table used in the function to a replication table. In this way, the full data of the table is stored on each DN. Even if the plan is pushed down to DNs for execution, the result set will be as expected.

Using the Unlogged Table

Scenario:

After an unlogged table is used and the cluster is restarted, the associated query result set is abnormal, and some data is missing in the unlogged table.

Analysis:

If **max_query_retry_times** is set to **0** and the keyword **UNLOGGED** is specified during table creation, the created table will be an unlogged table. Data written to unlogged tables is not written to the write-ahead log, which makes them considerably faster than ordinary tables. However, an unlogged table is automatically truncated after a crash or unclean shutdown, incurring data loss risks. The contents of an unlogged table are also not replicated to standby servers. Any indexes created on an unlogged table are not automatically logged as well. If the cluster restarts unexpectedly (process restart, node fault, or cluster restart), some data in the memory is not flushed to disks in a timely manner, and some data is lost, causing the result set to be abnormal.

Solution:

The security of unlogged tables cannot be ensured if the cluster goes faulty. In most cases, unlogged tables are only used as temporary tables. If a cluster is faulty, you need to rebuild the unlogged table or back up the data and import it to the database again to ensure that the data is normal.

12.4.16 Which System Catalogs in DWS Cannot Undergo the VACUUM FULL Operation?

From a functional perspective, **VACUUM FULL** can be performed on all DWS system catalogs, which involves using eight levels of locks, thereby blocking services related to these system catalogs.

The suggestions are based on database versions:

Version 8.1.3 or Later

- 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.

Version 8.1.1 or Earlier

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 the system catalogs in the following statement every week. If the space must be reclaimed, process key system catalogs first.

   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;
```

12.4.17 In Which Scenarios Will a DWS Statement Be in the "idle in transaction" State?

When user SQL information is queried in the **PGXC_STAT_ACTIVITY** view, the **state** column in the query result sometimes shows **idle in transaction**. **idle in transaction** indicates that the backend is in a transaction, but no statement is being executed. This status indicates that a statement has been executed. Therefore, the value of query_id is 0, but the transaction has not been committed or rolled back. Statements in this state have been executed and do not occupy CPU and I/O resources, but they occupy connection resources such as connections and concurrent connections.

If a statement is in the **idle in transaction** state, rectify the fault by referring to the following common scenarios and solutions:

Scenario 1: A Transaction Is Started But Not Committed, and the Statement Is in the "idle in transaction" State

BEGIN/START TRANSACTION is manually executed to start a transaction. After statements are executed, **COMMIT/ROLLBACK** is not executed. View the **PGXC_STAT_ACTIVITY**:

```
SELECT state, query, query_id FROM pgxc_stat_activity;
```

The result shows that the statement is in the **idle in transaction** state.


```
       state        |                                query                                |      query_id
--------------------+---------------------------------------------------------------------+--------------------
 active             |                                                                     |                  0
 idle               |                                                                     |                  0
 idle               |                                                                     |                  0
 active             | WLM fetch collect info from data nodes                              |  73464968921613282
 active             | WLM calculate space info process                                    |                  0
 active             | WLM monitor update and verify local info                            |  73464968921613276
 active             | WLM arbiter sync info by CCN and CNs                                |                  0
 idle in transaction | select count(1) from t group by a order by 1 desc limit 1;         |                  0
 idle               |                                                                     |                  0
 active             | select state,query,query_id from pgxc_stat_activity;                |  73464968921613283
 active             |                                                                     |                  0
 idle               |                                                                     |                  0
 idle               |                                                                     |                  0
 active             | WLM fetch collect info from data nodes                              | 145522562959541153
 active             | WLM calculate space info process                                    |                  0
 active             | WLM monitor update and verify local info                            | 145522562959541123
 active             | WLM arbiter sync info by CCN and CNs                                |                  0
 active             | SELECT * FROM pg_stat_activity                                      |  73464968921613283
 idle               |                                                                     |                  0
(19 rows)
```

Solution: Manually execute **COMMIT/ROLLBACK** on the started transaction.

Scenario 2: After a DDL Statement in a Stored Procedure Is Executed, Other Nodes of the Stored Procedure Is In the "idle in transaction" State

Create a stored procedure:

```
CREATE OR REPLACE FUNCTION public.test_sleep()
RETURNS void
LANGUAGE plpgsql
AS $$

BEGIN
   truncate t1;
   truncate t2;
   EXECUTE IMMEDIATE 'select pg_sleep(6)';
   RETURN;
END$$;
```

View the **PGXC_STAT_ACTIVITY** view:

```
SELECT coorname,pid,query_id,state,query,usename FROM pgxc_stat_activity WHERE usename='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         | usename
--------------+-----------------+-------------------+---------------------+----------------------+---------
 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,usename FROM pgxc_stat_activity WHERE usename='jack';
```

The result shows that the SAVEPOINT/RELEASE statement is in the **idle in transaction** state.


```
  coorname   |       pid       |     query_id      |        state        |          query          | usename
-------------+-----------------+-------------------+---------------------+-------------------------+---------
 coordinator1 | 140127877723904 | 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:

The **SAVEPOINT** and **RELEASE** statements are automatically generated by the system when the stored procedure with **EXCEPTION** is executed. (In clusters of version later than 8.1.0, **SAVEPOINT** is not delivered to CNs.) DWS stored procedures with **EXCEPTION** are implemented based on subtransactions, and 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 and exception during the execution, the current transaction is rolled back and the exception is handled; if there is no exception, the subtransaction is committed.

This problem may occur when there are many such stored procedures and the stored procedures are nested. Similar to scenario 2, you only have to wait after the entire stored procedure is executed. If there are a large number of **RELEASE** messages, the stored procedure triggered multiple exceptions. In this case, you must re-examine the logic of the stored procedure.

12.4.18 How Does DWS Implement Row-to-Column and Column-to-Row Conversion?

This section describes how to use SQL statements to convert rows to columns and convert columns to rows in DWS.

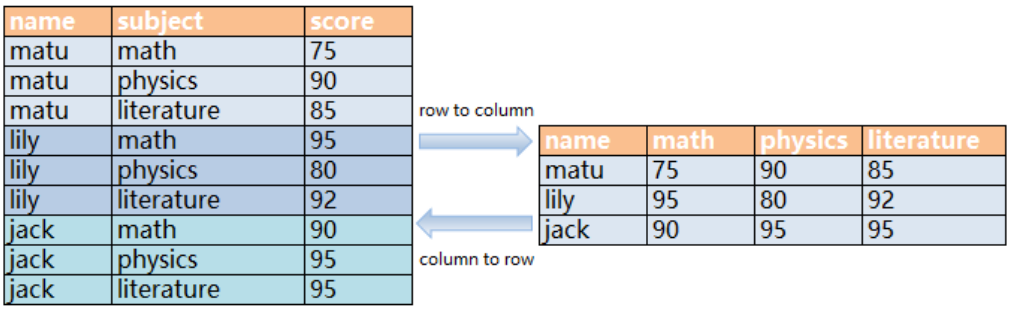
Scenario

Use a student score table as an example:

Teachers record the score of each subject of each student in a table, but students care only bout their own scores. A student needs to use row-to-column conversion to view their scores of all subjects. If the teacher of a subject wants to view the scores of all students of that subject, the teacher needs to use the column-to-row conversion.

The following figure shows the row-to-column and column-to-row conversion.

Figure 12-2 Diagram



- Rows-to-column conversion

  Convert multiple rows of data into one row, or convert one column of data into multiple columns.

- Column-to-row conversion

  Convert a row of data into multiple rows, or convert multiple columns of data into one column.

Example

- Create a row-store table **students_info**, and insert data into the table.

```
CREATE TABLE students_info(name varchar(20),subject varchar(100),score bigint) distribute by hash(name);
INSERT INTO students_info VALUES('lily','math',95);
INSERT INTO students_info VALUES('lily','physics',80);
INSERT INTO students_info VALUES('lily','literature',92);
INSERT INTO students_info VALUES('matu','math',75);
INSERT INTO students_info VALUES('matu','physics',90);
INSERT INTO students_info VALUES('matu','literature',85);
INSERT INTO students_info VALUES('jack','math',90);
INSERT INTO students_info VALUES('jack','physics',95);
INSERT INTO students_info VALUES('jack','literature',95);
```

  View information about the **students_info** table.

```
SELECT * FROM students_info;
 name |  subject   | score
------+------------+-------
 matu | math       |    75
 matu | physics    |    90
 matu | literature |    85
 lily | math       |    95
 lily | physics    |    80
 lily | literature |    92
 jack | math       |    90
 jack | physics    |    95
 jack | literature |    95
```

- Create a column-store table **students_info1**, and insert data into the table.

```
CREATE TABLE students_info1(name varchar(20), math bigint, physics bigint, literature bigint) with (orientation = column) distribute by hash(name);
INSERT INTO students_info1 VALUES('lily',95,80,92);
INSERT INTO students_info1 VALUES('matu',75,90,85);
INSERT INTO students_info1 VALUES('jack',90,95,95);
```

  View information about table **students_info1**.

```
SELECT * FROM students_info1;
 name | math | physics | literature
------+------+---------+------------
 matu |   75 |      90 |         85
```



```
 lily |   95 |      80 |         92
 jack |   90 |      95 |         95
(3 rows)
```

Static row-to-column conversion

Static row-to-column conversion requires you to manually specify the column names using the given values. If no value is given to a column, the default value **0** is assigned to the column.

```
SELECT name,
sum(case when subject='math' then score else 0 end) as math,
sum(case when subject='physics' then score else 0 end) as physics,
sum(case when subject='literature' then score else 0 end) as literature FROM students_info GROUP BY
name;
 name | math | physics | literature
------+------+---------+------------
 matu |   75 |      90 |         85
 lily |   95 |      80 |         92
 jack |   90 |      95 |         95
(3 rows)
```

Dynamic row-to-column conversion

For clusters of 8.1.2 or later, you can use **GROUP_CONCAT** to generate column-store statements.

```
SELECT group_concat(concat('sum(IF(subject = ''', subject, ''', score, 0)) AS "', name, '"'))FROM students_info;
      group_concat

-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------
 sum(IF(subject = 'literature', score, 0)) AS "jack",sum(IF(subject = 'literature', score, 0)) AS
"lily",sum(IF(subject = 'literature', score, 0)) AS "matu",sum(IF(subject = 'math', score, 0)) AS "jack",sum(IF
(subject = 'math', score, 0)) AS "lily",sum(IF(subject = 'math', score, 0)) AS "matu",sum(IF(subject =
'physics', score, 0)) AS "jack",sum(IF(subject = 'physics', score, 0)) AS "lily",sum(IF(subject = 'physics
', score, 0)) AS "matu"
(1 row)
```

In 8.1.1 and earlier versions, you can use **LISTAGG** to generate column-store statements.

```
SELECT listagg(concat('sum(case when subject = ''', subject, ''' then score else 0 end) AS "', subject, '"'),',')
within GROUP(ORDER BY 1)FROM (select distinct subject from students_info);
                                                                                                   listagg

-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------
--
 sum(case when subject = 'literature' then score else 0 end) AS "literature",sum(case when subject =
'physics' then score else 0 end) AS "physics",sum(case when subject = 'math' then score else 0 end) AS
"math
"
(1 row)
```

Dynamically rebuild the view:

```
CREATE OR REPLACE FUNCTION build_view()
RETURNS VOID
LANGUAGE plpgsql
AS $$ DECLARE
sql text;
rec record;
```



```
BEGIN
sql := 'select LISTAGG(
    CONCAT( ''sum(case when subject = ''''', subject, ''''' then score else 0 end) AS ''', subject, ''''' )
    ,'','' ) within group(order by 1) from (select distinct subject from students_info);';
    EXECUTE sql INTO rec;
    sql := 'drop view if exists get_score';
    EXECUTE sql;
    sql := 'create view get_score as select name, ' || rec.LISTAGG || ' from students_info group by name';
    EXECUTE sql;
END$$;
```

Rebuild the database:

```
CALL build_view();
```

Query view:

```
SELECT * FROM get_score;
 name | literature | physics | math
------+------------+---------+------
 matu |         85 |      90 |   75
 lily |         92 |      80 |   95
 jack |         95 |      95 |   90
(3 rows)
```

Column-to-Row Conversion

Use **UNION ALL** to merge subjects (math, physics, and literature) into one column. The following is an example:

```
SELECT * FROM
(
SELECT name, 'math' AS subject, math AS score FROM students_info1
union all
SELECT name, 'physics' AS subject, physics AS score FROM students_info1
union all
SELECT name, 'literature' AS subject, literature AS score FROM students_info1
)
order by name;
 name |  subject   | score
------+------------+-------
 jack | math       |    90
 jack | physics    |    95
 jack | literature |    95
 lily | math       |    95
 lily | physics    |    80
 lily | literature |    92
 matu | math       |    75
 matu | physics    |    90
 matu | literature |    85
(9 rows)
```

12.4.19 What Are the Differences Between DWS Unique Constraints and Unique Indexes?

- The concepts of a unique constraint and a unique index are different.

  A unique constraint specifies that the values in a column or a group of columns are all unique. If **DISTRIBUTE BY REPLICATION** is not specified, the column table that contains only unique values must contain distribution columns.

  A unique index is used to ensure the uniqueness of a field value or the value combination of multiple fields. **CREATE UNIQUE INDEX** creates a unique index.

- The functions of a unique constraint and a unique index are different.

  Constraints are used to ensure data integrity, and indexes are used to facilitate query.

- The usages of a unique constraint and a unique index are different.

  a. Both unique constraints and unique indexes can be used to ensure the uniqueness of column values which can be NULL.

  b. When a unique constraint is created, a unique index with the same name is automatically created. The index cannot be deleted separately. When the constraint is deleted, the index is automatically deleted. A unique constraint uses a unique index to ensure data uniqueness. DWS row-store tables support unique constraints, but column-store tables do not.

  c. A created unique index is independent and can be deleted separately. Currently in DWS, unique indexes can only be created using B-tree indexes.

  d. If you want to have both a unique constraint and a unique index on a column, and they can be deleted separately, you can create a unique index and then a unique constraint with the same name.

  e. If a field in a table is to be used as a foreign key of another table, the field must have a unique constraint (or it is a primary key). If the field has only a unique index, an error is reported.

Example: Create a composite index for two columns, which is not required to be a unique index.

```
CREATE TABLE t (n1 number,n2 number,n3 number,PRIMARY KEY (n3));
CREATE INDEX t_idx ON t(n1,n2);
```

DWS supports multiple unique indexes for a table.

```
CREATE UNIQUE INDEX u_index ON t(n3);
CREATE UNIQUE INDEX u_index1 ON t(n3);
```

You can use the index **t_idx** created in the example above to create a unique constraint **t_uk**, which is unique only on column **n1**. A unique constraint is stricter than a unique index.

```
ALTER TABLE t ADD CONSTRAINT t_uk UNIQUE USING INDEX u_index;
```

12.4.20 What Are the Differences Between DWS Functions and Stored Procedures?

Functions and stored procedures are two common objects in database management systems. They have similarities and differences in implementing specific functions. Understanding their characteristics and application scenarios is important for properly designing the database structure and improving database performance.

Table 12-5 Differences between functions and stored procedures

Function	Stored procedures
Both can be used to implement specific functions. Both functions and stored procedures can encapsulate a series of SQL statements to complete certain specific operations.	
Both can receive input parameters and perform corresponding operations based on the parameters.	
The identifier of a function is **FUNCTION**.	The identifier of the stored procedure is **PROCEDURE**.
A function must return a specific value of the specified numeric type.	A stored procedure can have no return value, one return value, or multiple return values. You can use output parameters to return results or directly use the SELECT statement in a stored procedure to return result sets.
Functions are used to return single values, for example, a number calculation result, a string processing result, or a table.	Stored procedures are used for DML operations, for example, inserting, updating, and deleting data in batches.

- **Creating and Invoking a Function**

  Create the **emp** table and insert data into the table. The table data is as follows:

  ```
  SELECT * FROM emp;
   empno | ename |   job    | mgr |      hiredate       |   sal   |  comm  | deptno
  -------+-------+----------+-----+---------------------+---------+--------+--------
    7369 | SMITH | CLERK    | 7902 | 1980-12-17 00:00:00 |  800.00 |        |     20
    7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 00:00:00 | 1600.00 | 300.00 |     30
    7566 | JONES | MANAGER  | 7839 | 1981-04-02 00:00:00 | 2975.00 |        |     20
    7521 | WARD  | SALESMAN | 7698 | 1981-02-22 00:00:00 | 1250.00 | 500.00 |     30
  (4 rows)
  ```

  Create the **emp_comp** function to accept two numbers as input and return the calculated value.

  ```
  CREATE OR REPLACE FUNCTION emp_comp (
      p_sal           NUMBER,
      p_comm          NUMBER
  ) RETURN NUMBER
  IS
  BEGIN
      RETURN (p_sal + NVL(p_comm, 0)) * 24;
  END;
  /
  ```

  Run the **SELECT** command to invoke the function:

  ```
  SELECT ename "Name", sal "Salary", comm "Commission", emp_comp(sal, comm) "Total
  Compensation" FROM emp;
   Name  | Salary  | Commission | Total Compensation
  -------+---------+------------+--------------------
   SMITH |  800.00 |            |           19200.00
   ALLEN | 1600.00 |     300.00 |           45600.00
   JONES | 2975.00 |            |           71400.00
  ```



```
 WARD  | 1250.00 |     500.00 |           42000.00
(4 rows)
```

- **Creating and Invoking a Stored Procedure**

  Create the **MATCHES** table and insert data into the table. The table data is as follows:

  ```
  SELECT * FROM MATCHES;
   matchno | teamno | playerno | won | lost
  ---------+--------+----------+-----+------
        1 |      1 |        6 |   3 |    1
        7 |      1 |       57 |   3 |    0
        8 |      1 |        8 |   0 |    3
        9 |      2 |       27 |   3 |    2
       11 |      2 |      112 |   2 |    3
  (5 rows)
  ```

  Create the stored procedure **delete_matches** to delete all matches that a specified player participates in.

  ```
  CREATE PROCEDURE delete_matches(IN p_playerno INTEGER)
  AS
  BEGIN
    DELETE FROM MATCHES WHERE playerno = p_playerno;
  END;
  /
  ```

  Invoke the stored procedure **delete_matches**.

  ```
  CALL delete_matches(57);
  ```

  Query the **MATCHES** table again. The returned result indicates that the data of the player whose **playerno** is **57** has been deleted.

  ```
  SELECT * FROM MATCHES;
   matchno | teamno | playerno | won | lost
  ---------+--------+----------+-----+------
       11 |      2 |      112 |   2 |    3
        8 |      1 |        8 |   0 |    3
        1 |      1 |        6 |   3 |    1
        9 |      2 |       27 |   3 |    2
  (4 rows)
  ```

12.4.21 How Do I Delete Duplicate Table Data from DWS?

When clearing dirty data in the database, you may retain only one piece of duplicate data. In this scenario, you can use the aggregate function or window function.

Constructing Table Data

Step 1 Create a table t_customer and insert data that contains duplicate records into the table.

```
CREATE TABLE t_customer (
  id int NOT NULL,
cust_name varchar(32) NOT NULL COMMENT' Name',
gender varchar(10) NOT NULL COMMENT' Gender',
  email varchar(32) NOT NULL COMMENT 'email',
  PRIMARY KEY (id)
) ;

INSERT INTO t_customer VALUES ('1', 'Tom', 'Male', 'high_salary@sample.com');
INSERT INTO t_customer VALUES ('2', 'Jennifer', 'Female', 'good_job@sample.com');
INSERT INTO t_customer VALUES ('3', 'Tom', 'Male', 'high_salary@sample.com');
INSERT INTO t_customer VALUES ('4', 'John', 'Male', 'good_job@sample.com');
INSERT INTO t_customer VALUES ('5', 'Jennifer', 'Female', 'good_job@sample.com');
INSERT INTO t_customer VALUES ('6', 'Tom', 'Male', 'high_salary@sample.com');
```


Step 2 Query the t_customer table.

```
SELECT * FROM t_customer ORDER BY id;
```

```
 id | cust_name | gender |          email
----+-----------+--------+-------------------------
  1 | Tom       | Male   | high_salary@sample.com
  2 | Jennifer  | Female | good_job@sample.com
  3 | Tom       | Male   | high_salary@sample.com
  4 | John      | Male   | good_job@sample.com
  5 | Jennifer  | Female | good_job@sample.com
  6 | Tom       | Male   | high_salary@sample.com
```

----End

If the name, gender, and email of a customer are the same, the customer is regarded as a duplicate record. In the t_customer table, data whose IDs are 1, 3, and 6 is duplicate, and data whose IDs are 2 and 5 is also duplicate. Delete redundant data and retain one of them.

Method 1: Use the aggregate function **min(expr)**.

Use aggregate functions to obtain non-duplicate rows with the smallest ID through subqueries, and then use NOT IN to delete duplicate data.

Step 1 Run the following command to query the unique row with the smallest ID:

```
SELECT
   min(id) id,
   cust_name,
   gender,
   COUNT( cust_name ) count
FROM t_customer
GROUP BY cust_name,gender
ORDER BY id;
```

```
 id | cust_name | gender | count
----+-----------+--------+-------
  1 | Tom       | Male   |     3
  2 | Jennifer  | Female |     2
  4 | John      | Male   |     1
```

According to the query result, duplicate data rows whose IDs are 3, 5, and 6 are filtered out.

Step 2 Use NOT IN to filter out duplicate data rows and delete them.

```
DELETE from t_customer where id not in (
   SELECT
   min(id) id
   FROM t_customer
   GROUP BY cust_name,gender
);
```

Step 3 Query the t_customer table after duplicate data is deleted.

```
SELECT * FROM t_customer ORDER BY id;
```

```
 id | cust_name | gender |          email
----+-----------+--------+-------------------------
  1 | Tom       | Male   | high_salary@sample.com
  2 | Jennifer  | Female | good_job@sample.com
  4 | John      | Male   | good_job@sample.com
```


The command output indicates that duplicate data has been deleted.

----End

Method 2: Use the window function row_number().

Use PARTITION BY to partition and sort columns, generate sequence number columns, and delete rows whose sequence numbers are greater than 1.

Step 1 Partition query. Sort columns by partition and generate sequence number columns.

```
SELECT
   id,
   cust_name,
   gender,
   ROW_NUMBER() OVER (PARTITION BY cust_name,gender ORDER BY id) num
FROM t_customer;
```

```
 id | cust_name | gender | num
----+-----------+--------+-----
  4 | John      | Male   |   1
  1 | Tom       | Male   |   1
  3 | Tom       | Male   |   2
  6 | Tom       | Male   |   3
  2 | Jennifer  | Female |   1
  5 | Jennifer  | Female |   2
```

According to the command output, the data in num>1 is duplicate.

Step 2 Delete the data of num>1.

```
DELETE FROM t_customer WHERE id in (
  SELECT id FROM(
   SELECT * FROM (
      SELECT ROW_NUMBER() OVER w AS row_num,id
   FROM t_customer
   WINDOW w AS (PARTITION BY cust_name,gender ORDER BY id) )
  WHERE row_num >1 )
);
```

Step 3 Query the t_customer table after duplicate data is deleted.

```
SELECT * FROM t_customer ORDER BY id;
```

```
 id | cust_name | gender |          email
----+-----------+--------+-------------------------
  1 | Tom       | Male   | high_salary@sample.com
  2 | Jennifer  | Female | good_job@sample.com
  4 | John      | Male   | good_job@sample.com
```

----End

12.5 Cluster Management

12.5.1 How Can I Clear and Reclaim the DWS Storage Space?

After you delete data stored in 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.

Periodically clear dirty data to clean up and reclaim storage space. The procedure varies depending on the cluster version. The details are as follows:

Version 8.1.3 or Later: Automatically Cleaning Using the Intelligent O&M Function on the Management Console

Step 1 Log in to the DWS console.

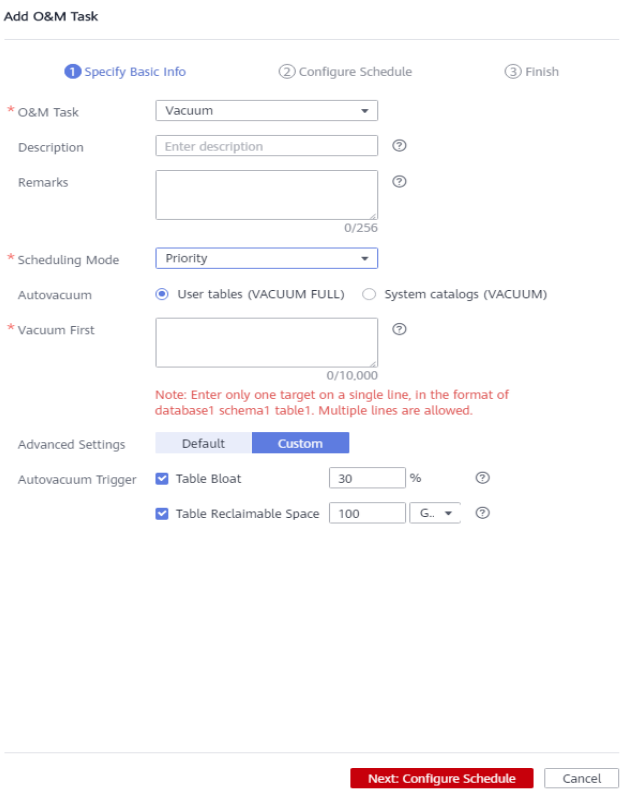
Step 2 Click the name of the target cluster.

Step 3 In the navigation pane, choose **Intelligent O&M**.

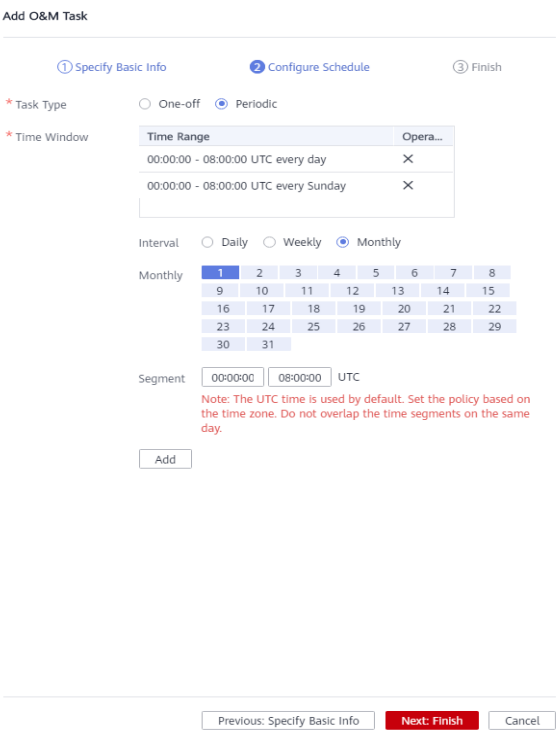
Step 4 Click the **O&M Plan** tab. Click **Add O&M Task**.

Step 5 The **Add O&M Task** page is displayed.

- Select **Vacuum** for **O&M Task**.
- Set **Scheduling Mode** to **Auto**. DWS automatically scans tables that require **VACUUM** operation.
- Select **System catalogs** or **User tables** for **Autovacuum**.
  - If there are a large number of **UPDATE** and **DELETE** operations, select the **User tables**.
  - If there are a large number of **CREATE** and **DELETE** operations, select **System catalogs**.



Step 6 Click **Next: Configure Schedule** to configure the schedule and Vacuum type. You are advised to select **Periodic** for **Task Type**. The DWS automatically executes VACUUM in your selected time windows.



> **NOTE**
>
> For automatic Vacuum O&M tasks, the system uses the **VACUUM FULL** operation to process user tables. VACUUM FULL holds a level 8 lock, which blocks other transactions. Other transactions will be in lock waiting during **VACUUM FULL** execution. After 20 minutes, a timeout error is reported. Therefore, do not perform other transactions in the configured time window.

Step 7 After confirming that the information is correct, click **Next** to complete the configuration.

----End

Version 8.1.2 or Earlier: Manually Cleaning by Running the VACUUM FULL Command

NOTICE

1. The **VACUUM FULL** operation locks a table, blocking all access to it during this period and waiting for its completion. To avoid impacting services due to table locking, schedule appropriately.
2. **VACUUM FULL** extracts valid data from the current table, reorganizes it, and removes dirty data. This process temporarily requires **additional space**, which is released after the reorganization is complete. As a result, space initially increases before decreasing. Calculate the necessary space for **VACUUM FULL** in advance using the formula: **Additional reorganization space** = **Table size** x (**1** – **Dirty page rate**).

Step 1 Connect to the database, run the following SQL statements to query large tables whose dirty page rate exceeds 30%, and sort the tables by size in descending order:

```
SELECT schemaname AS schema, relname AS table_name, n_live_tup AS analyze_count,
pg_size_pretty(pg_table_size(relid)) as table_size, dirty_page_rate
FROM PGXC_GET_STAT_ALL_TABLES
WHERE schemaName NOT IN ('pg_toast', 'pg_catalog', 'information_schema', 'cstore', 'pmk')
AND dirty_page_rate > 30
ORDER BY table_size DESC, dirty_page_rate DESC;
```

Step 2 Check whether any command output is displayed.

- If yes, perform **Step 3** for tables larger than 10 GB.
- If no, no further action is required.

Step 3 Run the **VACUUM FULL** command to clear the top 5 tables with the most dirty pages. If the maximum disk space is greater than 70%, clear the tables one by one.

```
VACUUM FULL ANALYZE schema.table_name;
```

----End

12.5.2 Why Does the Used Storage of DWS Decrease Significantly After Scale-Out?

Cause Analysis

If you do not run the **VACUUM** command to clear and reclaim the storage space before the scale-out, the data deleted from the data warehouse may not release the occupied disk space, causing dirty data and disk waste.

During the scale-out, the system redistributes the data because the service data volume on the original nodes is significantly larger than that on the newly added nodes. When the redistribution starts, the system automatically performs **VACUUM** to free up the storage space. In this way, the used storage is reduced.

Handling Procedure

You are advised to periodically clear and reclaim the storage space by running **VACUUM FULL** to prevent data expansion.

If the used storage space is still large after you run **VACUUM FULL**, analyze whether the existing cluster flavor meets service requirements. If no, scale out the cluster.

12.5.3 How Is the Disk Space or Capacity of DWS Calculated?

1. Total disk capacity of a DWS cluster: Taking three data nodes in the cluster as an example, assume each node has a capacity of 320 GB, resulting in a total capacity of 960 GB. When 1 GB of data is stored, DWS employs its replication mechanism to maintain copies of the data across two nodes, thereby utilizing 2 GB of storage space. With additional metadata and indexing considerations, although the original dataset remains at 1 GB, the actual storage requirements exceed 2 GB upon ingestion into DWS. Therefore, a three-node cluster with a total capacity of 960 GB can store 480 GB data. This mechanism ensures data security.

   When you create a cluster on the DWS console, the actual capacity of a node is displayed on the page. For example, the disk size of the **dwsx2.xlarge** node is 160 GB on the creation page. However, the actual disk size of the node is 320 GB, which is displayed as 160 GB. In this way, you can create a node based on the actual disk data.

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 DWS console and choose **Monitoring** > **Node Monitoring** > **Overview** to view the usage of disks and other resources on each node.

NOTE

- The disk space shown on the node management page represents the combined capacity of all disks in the DWS cluster, including system disks and data disks. On the overview page, the displayed disk space only refers to the available space for storing table data in the cluster. Additionally, the DWS cluster has backup copies of tables, which also occupy disk storage.
- If the cluster is read-only and an alarm for insufficient disk space is generated, scale out the cluster by following the instructions provided in "Scaling Out a Cluster".

12.5.4 How Do I Set the Session Threshold When Creating Alarm Rules for DWS in Cloud Eye?

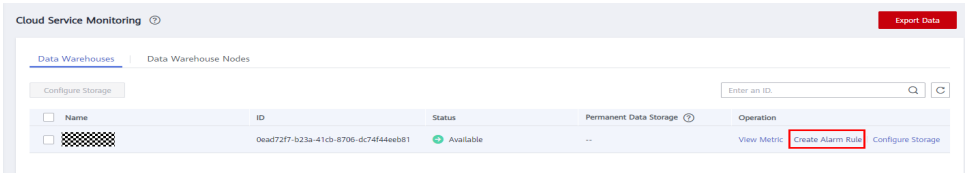
After connecting to a database, run the following SQL statement to check the maximum number of concurrent sessions globally:

```
show max_active_statements;
```

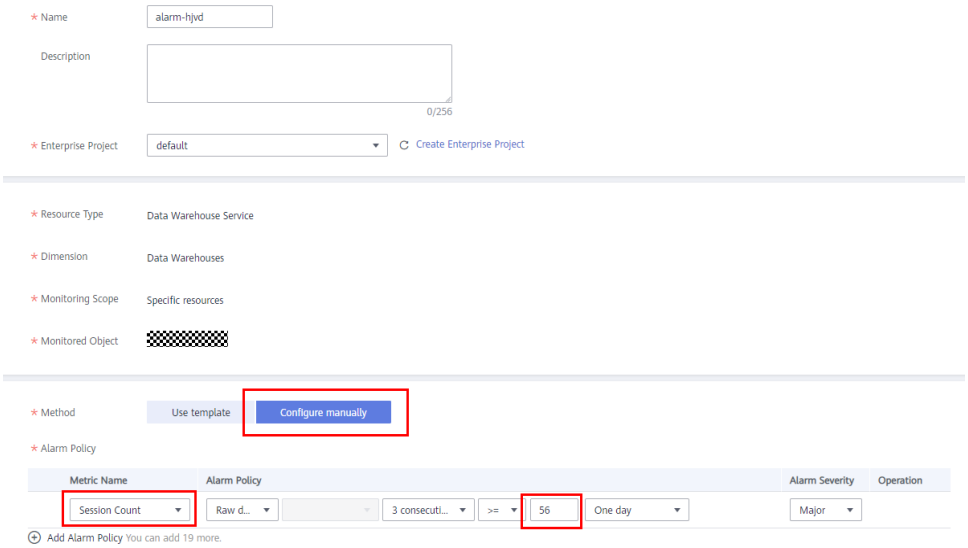
Go to the Cloud Eye console and set the threshold to 70% to 80% of the obtained value. For example, if the value of **max_active_statements** is **80**, set the threshold to **56** (80 x 70%).

Procedure:

1. Log in to the DWS console and choose **Cluster** > **Cluster List**.
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**.



12.5.5 How Do I Determine Whether to Add CNs to a DWS Cluster or Scale Out the Cluster?

Introduction to CN Concurrency

Coordinator Node (CN) is an important component in DWS that is most closely related to users. It provides external application interfaces, optimizes global execution plans, distributes the execution plans to DataNodes, and summarizes and processes execution results. A CN is an interface to external applications. The concurrency capability of the CN determines the service concurrency.

CN concurrency is determined by the following parameters:

- **max_connections**: specifies the maximum number of concurrent connections to the database. This parameter affects the concurrent processing capability of the cluster. The default value depends on the cluster specifications. For details, see "Managing Database Connections".
- **max_active_statements**: specifies the maximum number of concurrent jobs. This parameter applies to all the jobs on one CN. The default value is **60**, which indicates a maximum of 60 jobs can run at the same time. Other jobs will be queued.

Add CNs or Scale out a Cluster?

- Insufficient connections: When a cluster is created for the first time, the default number of CNs in the cluster is 3, which can meet the customer's basic connection requirements. If the cluster has a large number of concurrent requests and the number of connections to each CN is large, or the CPU usage of a CN exceeds its capacity, you are advised to add CNs. For details, see "CNs".
- Insufficient storage capacity and performance: If your business grows and you have higher requirements on storage capacity and performance, or the CPU of

your cluster is insufficient, you are advised to scale out your cluster. For details, see "Scaling Out a Cluster".

More CNs are needed to meet the distribution requirements of DWS. In short, adding CNs does not necessarily require cluster scale-out. However, after cluster scale-out, CNs may need to be added.

12.5.6 How Do I Choose Between a Small-Specification Multi-Node DWS Cluster and a Large-Specification Three-Node DWS Cluster with Identical CPU Cores and Memory?

- Small-flavor many-node:

  If your data volume is small and you have requirement for node scaling, but you have limited budget, you can select a small-flavor many-node cluster.

  For example, a small-flavor cluster (dwsx2.h.2xlarge.4.c6) with 8 cores and 32 GB memory can provide strong computing capabilities. The cluster has a large number of nodes and can process high concurrent requests. In this case, you only need to ensure that the network speed between nodes is normal to avoid cluster performance limitation.

- Large-flavor three-node:

  If you have a large amount of data to be processed, have high requirement on computing, and have a high budget, you can select a large-flavor three-node cluster.

  For example, a large-flavor cluster (dws2.m6.8xlarge.8) with 32 cores and 256 GB memory has faster CPU processing capability and larger memory, and can process data more quickly. However, the cluster has limited nodes, which may cause low performance in high-concurrency scenarios.

12.5.7 What Are the Differences Between Hot Data Storage and Cold Data Storage in DWS?

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 12-6**.

Table 12-6 Differences between hot and cold data storage

Storage	Read and Write	Cost	Capacity	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.

12.5.8 How Do I Do If the DWS Scale-In Button Is Unavailable?

Symptom

When a user performs a scale-in operation, the **Scale In** button is unavailable and the user cannot proceed to the next scale-in operation.

Possible Causes

The system verifies the cluster's eligibility for scaling in before each operation. The **Scale In** button is dimmed if the cluster does not qualify.

Solution

Check the cluster configuration information and check whether the scale-in meets the following conditions:

- The cluster consists of rings of four or five hosts each, with primary, standby, and secondary DNs deployed on them. A cluster ring is the smallest unit for scaling in, which requires at least two rings. The system removes nodes from the last ring to the first when scaling in.
- The removed nodes cannot contain the GTM, CM Server, or CN component.
- The cluster status is **Normal**, and no other task information is displayed.
- The cluster tenant account cannot be in the read-only, frozen, or restricted state.
- The cluster is not in logical cluster mode.
- The cluster cannot have idle nodes.

12.6 Account Permissions

12.6.1 How Does DWS Isolate Workloads?

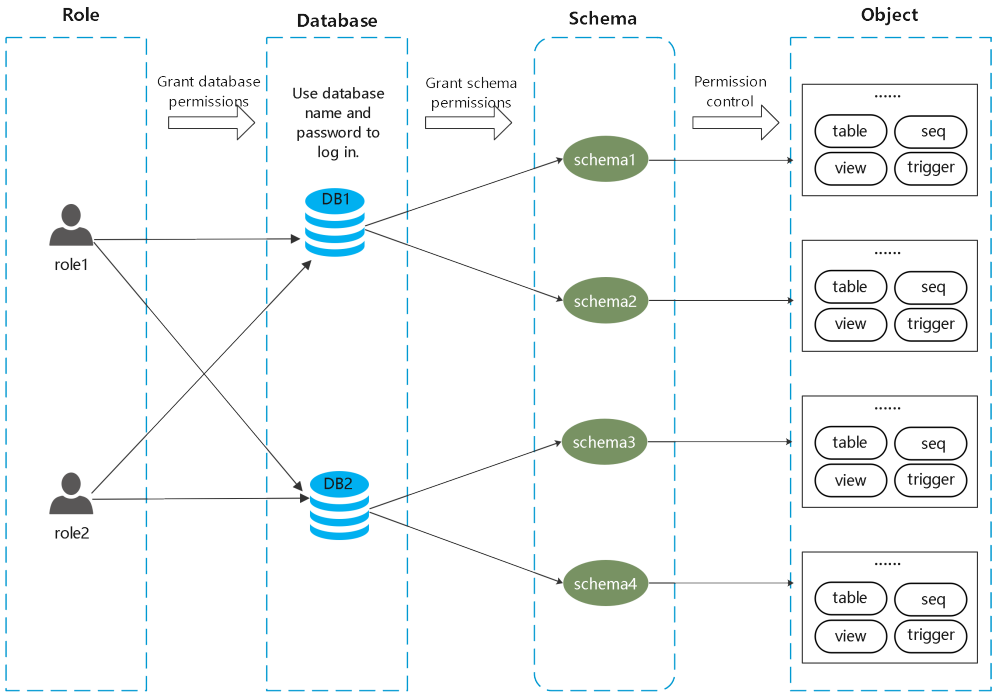
Workload Isolation

In DWS, you can isolate workloads through database and schema configurations. The differences are:

- 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 12-3 Used for permission control.



DATABASE

A database is a physical collection of database objects. Resources of different databases are completely isolated (except some shared objects). Databases are used to isolate workloads. Objects in different databases cannot access each other. For example, objects in Database B cannot be accessed in Database A. Therefore, when logging in to a cluster, you must connect to the specified database.

SCHEMA

In a database, database objects are logically divided and isolated based on schemas.

With permission management, you can access and operate objects in different schemas in the same session. Schemas contain objects that applications may access, such as tables, indexes, data in various types, functions, and operators.

Database objects with the same name cannot exist in the same schema, but object names in different schemas can be the same.

```
gaussdb=> CREATE SCHEMA myschema;
CREATE SCHEMA
gaussdb=> CREATE SCHEMA myschema_1;
CREATE SCHEMA

gaussdb=> CREATE TABLE myschema.t1(a int, b int) DISTRIBUTE BY HASH(b);
CREATE TABLE
gaussdb=> CREATE TABLE myschema.t1(a int, b int) DISTRIBUTE BY HASH(b);
ERROR:  relation "t1" already exists
gaussdb=> CREATE TABLE myschema_1.t1(a int, b int) DISTRIBUTE BY HASH(b);
CREATE TABLE
```

Schemas logically divide workloads. These workloads are interdependent with the schemas. Therefore, if a schema contains objects, deleting it will cause errors with dependency information displayed.

```
gaussdb=> DROP SCHEMA myschema_1;
ERROR: cannot drop schema myschema_1 because other objects depend on it
Detail: table myschema_1.t1 depends on schema myschema_1
Hint: Use DROP ... CASCADE to drop the dependent objects too.
```

When a schema is deleted, the **CASCADE** option is used to delete the objects that depend on the schema.

```
gaussdb=> DROP SCHEMA myschema_1 CASCADE;
NOTICE:  drop cascades to table myschema_1.t1
gaussdb=> DROP SCHEMA
```

USER/ROLE

Users and roles are used to implement permission control on the database server (cluster). They are the owners and executors of cluster workloads and manage all object permissions in clusters. A role is not confined in a specific database. However, when it logs in to the cluster, it must explicitly specify a user name to ensure the transparency of the operation. A user's permissions to a database can be specified through permission management.

A user is the subject of permissions. Permission management is actually the process of deciding whether a user is allowed to perform operations on database objects.

Permissions Management

Permission management in 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. You are advised to strictly control the creation of databases and schemas. Create databases and schemas as an administrator and assign related permissions to users.

Step 1 Assign the permission to create schemas in the **testdb** database to user **user_1** as user **dbadmin**.

```
testdb=> GRANT CREATE ON DATABASE testdb to user_1;
GRANT
```

Step 2 Switch to user **user_1**.

```
testdb=> SET SESSION AUTHORIZATION user_1 PASSWORD '********';
SET
```

Create a schema named **myschema_2** in the **testdb** database as **user_1**.

```
testdb=> CREATE SCHEMA myschema_2;
CREATE SCHEMA
```

Step 3 Switch to the administrator **dbadmin**.

```
testdb=> RESET SESSION AUTHORIZATION;
RESET
```

Create **table t1** in schema **myschema_2** as the administrator **dbadmin**.

```
testdb=> CREATE TABLE myschema_2.t1(a int, b int) DISTRIBUTE BY HASH(b);
CREATE TABLE
```

Step 4 Switch to user **user_1**.

```
testdb=> SET SESSION AUTHORIZATION user_1 PASSWORD '********';
SET
```

Delete table **t1** created by administrator **dbadmin** in schema **myschema_2** as user **user_1**.


```
testdb=> drop table myschema_2.t1;
DROP TABLE
```

----End

Example 2:

Due to the logical isolation of schemas, database objects need to be verified at both the schema level and the object level.

Step 1 Grant the permission on the **myschema.t1** table to **user_1**.

```
gaussdb=> GRANT SELECT ON TABLE myschema.t1 TO user_1;
GRANT
```

Step 2 Switch to user **user_1**.

```
SET SESSION AUTHORIZATION user_1 PASSWORD '********';
SET
```

Query the table **myschema.t1**.

```
gaussdb=> SELECT * FROM myschema.t1;
ERROR:  permission denied for schema myschema
LINE 1: SELECT * FROM myschema.t1;
```

Step 3 Switch to the administrator **dbadmin**.

```
gaussdb=> RESET SESSION AUTHORIZATION;
RESET
```

Grant the permission on the **myschema.t1** table to user **user_1**.

```
gaussdb=> GRANT USAGE ON SCHEMA myschema TO user_1;
GRANT
```

Step 4 Switch to user **user_1**.

```
gaussdb=> SET SESSION AUTHORIZATION user_1 PASSWORD '********';
SET
```

Query the table **myschema.t1**.

```
gaussdb=> SELECT * FROM myschema.t1;
 a | b
---+---
(0 rows)
```

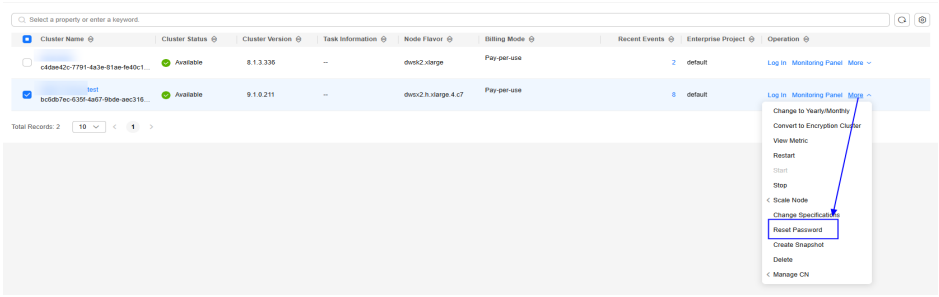
----End

12.6.2 How Do I Change the Password of a DWS Database Account When It Expires?

Below are the methods for changing the password of a database account:

- To change the password of user **dbadmin**, log in to the DWS console, locate the target cluster and choose **More** > **Reset Password** in the **Operation** column.

Figure 12-4 Resetting the password of user dbadmin



- 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;
```

12.6.3 How Do I Grant Table Permissions to a Specified DWS 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 |
ANALYSE } [, ...]
    | 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 INSERT 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**.

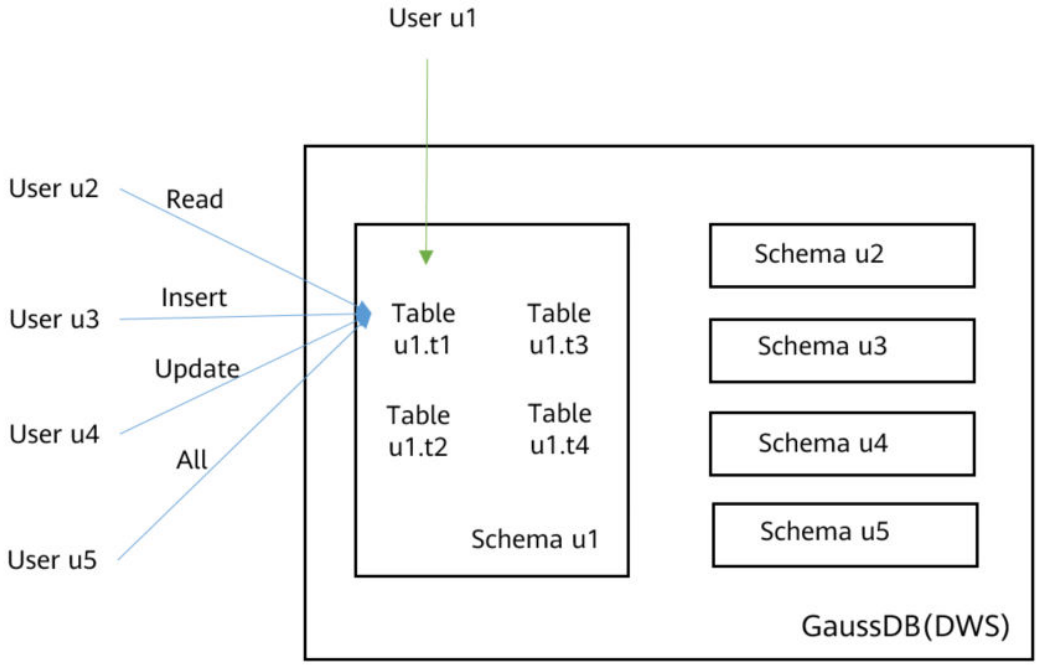


Table 12-7 Permissions of the u1.t1 table

User	Type	GRANT Statement	Query	Insert	Update	Delete
u1	Owner	-	√	√	√	√
u2	Read-only user	GRANT SELECT ON u1.t1 TO u2;	√	x	x	x
u3	INSERT user	GRANT INSERT ON u1.t1 TO u3;	x	√	x	x
u4	UPDATE user	GRANT SELECT,UPDATE ON u1.t1 TO u4; **NOTICE** The UPDATE permission must be granted together with the SELECT permission, or information leakage may occur.	√	x	√	x

User	Type	GRANT Statement	Query	Insert	Update	Delete
u5	Users with all permissions	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;
```

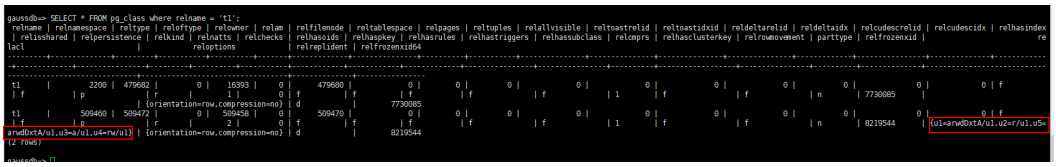
```
gaussdb=> SELECT * FROM u1.t1;
 c1 | c2
----+----
  1 |  3
  1 |  3
  1 |  3
(3 rows)

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

12.6.4 How Do I Grant the Permissions of a Schema to a Specified DWS User?

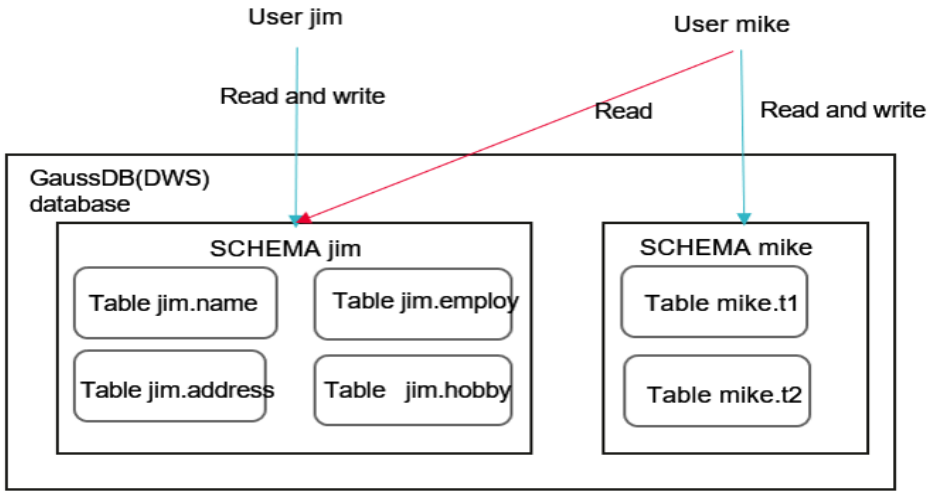
This section explains how to give query permission for schema-level permissions. If you need other permissions, see "How Do I Grant Table Permissions to a User?" in FAQ.

- Permission for a table in a schema
- Permission for all the tables in a schema
- Permission for tables to be created in the schema

> **CAUTION**
>
> The VACUUM, DROP, and ALTER permissions on foreign tables cannot be granted to users.

Assume that there are users **jim** and **mike**, and two schemas named after them. User **mike** needs to access tables in schema **jim**, as shown in **Figure 12-5**.

Figure 12-5 User **mike** accesses a table in SCHEMA **jim**.



Step 1 Connect to your database as **dbadmin**. Run the following statements to create users **jim** and **mike**. Two schemas will be created and named after the users by default.

```
CREATE USER jim PASSWORD '{password}';
CREATE USER mike PASSWORD '{password}';
```

Step 2 Create tables **jim.name** and **jim.address** in schema **jim**.

```
CREATE TABLE jim.name (c1 int, c2 int);
CREATE TABLE jim.address (c1 int, c2 int);
```

Step 3 Grant the access permission of schema **jim** to user **mike**.

```
GRANT USAGE ON SCHEMA jim TO mike;
```

Step 4 Grant user **mike** the permission to query table **jim.name** in schema **jim**.

```
GRANT SELECT ON jim.name TO mike;
```

Step 5 Start a new session and connect to the database as user **mike**. Verify that user **mike** can query the **jim.name** table but not the **jim.address** table.

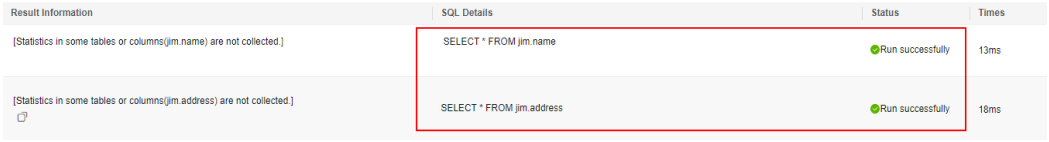
```
SELECT * FROM jim.name;
SELECT * FROM jim.address;
```

Step 6 In the session started by user **dbadmin**, grant user **mike** the permission to query all the tables in schema **jim**.

```
GRANT SELECT ON ALL TABLES IN SCHEMA jim TO mike;
```

Step 7 In the session started by user **mike**, verify that **mike** can query all tables.

```
SELECT * FROM jim.name;
SELECT * FROM jim.address;
```

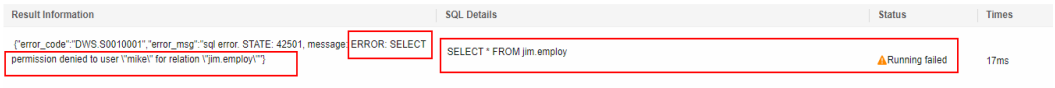



Step 8 In the session started by user **dbadmin**, create table **jim.employ**.

```
CREATE TABLE jim.employ (c1 int, c2 int);
```

Step 9 In the session started by user **mike**, verify that user **mike** does not have the query permission for **jim.employ**. It indicates that user **mike** has the permission to access all the existing tables in schema **jim**, but not the tables to be created in the future.

```
SELECT * FROM jim.employ;
```

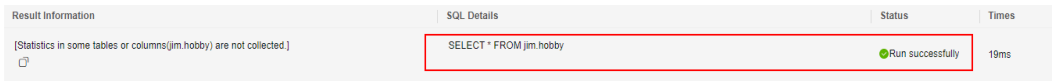


Step 10 In the session started by user **dbadmin**, grant user **mike** the permission to query the tables to be created in schema **jim**. Create table **jim.hobby**.

```
ALTER DEFAULT PRIVILEGES FOR ROLE jim IN SCHEMA jim GRANT SELECT ON TABLES TO mike;
CREATE TABLE jim.hobby (c1 int, c2 int);
```

Step 11 In the session started by user **mike**, verify that user **mike** can access table **jim.hobby**, but does not have the permission to access **jim.employ**. To let the user access table **jim.employ**, you can grant permissions by performing **Step 4**.

```
SELECT * FROM jim.hobby;
```



----End

12.6.5 How Do I Create a DWS 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 Grant read-only access to user1 for a foreign table in schema **s1**.

```
ALTER USER user1 USEFT;
```

Without proper permissions, querying the foreign table as a read-only user will result in the error message "ERROR: permission denied to select from foreign table in security mode."

Step 6 Read all table data in schema **s1** as read-only user **user1**.

----End

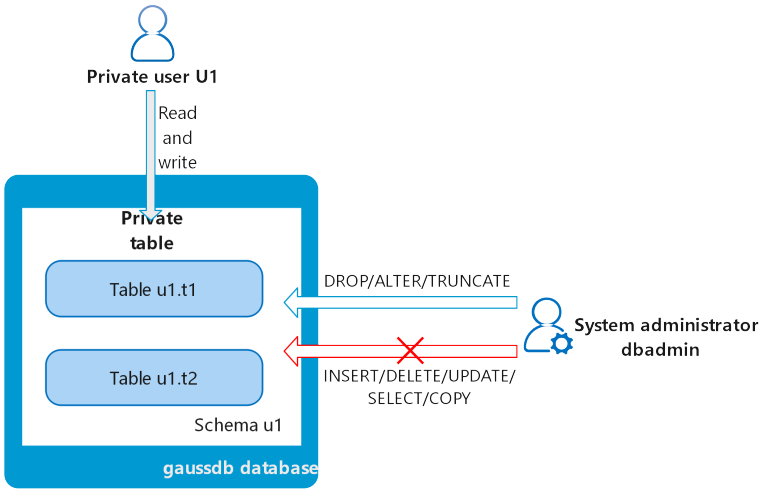
12.6.6 How Do I Create Private Users and Tables in a DWS Database?

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 12-6 Private users



Principles

This function is implemented by creating a user with the INDEPENDENT attribute.

INDEPENDENT | NOINDEPENDENT defines private and independent roles. For a role with the **INDEPENDENT** attribute, administrators' rights to control and access this role are separated. Specific rules are as follows:

- Administrators have no rights to add, delete, query, modify, copy, or authorize the corresponding table objects without the authorization from the INDEPENDENT role.
- Administrators have no rights to modify the inheritance relationship of the INDEPENDENT role without the authorization from this role.
- Administrators have no rights to modify the owner of the table objects for the INDEPENDENT role.
- Administrators have no rights to change the database password of the INDEPENDENT role. The INDEPENDENT role must manage its own password, which cannot be reset if lost.
- The **SYSADMIN** attribute of a user cannot be changed to the **INDEPENDENT** attribute.

Procedure

Step 1 Connect to the DWS database as user **dbadmin**.

Step 2 Run the following SQL statement to create private user **u1**:

```
CREATE USER u1 WITH INDEPENDENT IDENTIFIED BY 'password';
```

Step 3 Switch to user **u1**, create the table **test**, and insert data into the table.

```
CREATE TABLE test (id INT, name VARCHAR(20));
INSERT INTO test VALUES (1, 'joe');
INSERT INTO test VALUES (2, 'jim');
```

Step 4 Switch to user **dbadmin** and run the following SQL statement to check whether user **dbadmin** can access the private table **test** created by private user **u1**:

```
SELECT * FROM u1.test;
```


The query result indicates that the user **dbadmin** does not have the access permission. This means the private user and private table are created successfully.

```
gaussdb=> SELECT * FROM u1.test;
ERROR:  SELECT permission denied to user "dbadmin" for relation "u1.test"
```

Step 5 Run the **DROP** statement as user **dbadmin** to delete the table **test**.

```
DROP TABLE u1.test;
```

```
gaussdb=> drop table u1.test;
DROP TABLE
```

----End

12.6.7 How Do I Revoke the CONNECT ON DATABASE Permission of a User on DWS?

Scenario

In a service, the permission of user **u1** to connect to a database needs to be revoked. After the **REVOKE CONNECT ON DATABASE *gaussdb* FROM u1;** command is executed successfully, user **u1** can still connect to the database. This means the revocation does not take effect.

Cause Analysis

If you run the **REVOKE CONNECT ON DATABASE gaussdb from u1** command to revoke the permissions of user **u1**, the revocation does not take effect because the **CONNECT** permission of the database is granted to **PUBLIC**. Therefore, you need to specify **PUBLIC**.

- 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.
- DWS grants the permissions on certain types of objects to **PUBLIC**. By default, permissions on tables, columns, sequences, foreign data sources, foreign servers, schemas, and tablespaces are not granted to **PUBLIC**, but the following permissions are granted to **PUBLIC**;
  - **CONNECT** permission of a database
  - **CREATE TEMP TABLE** permission of a database
  - **EXECUTE** permission of a function
  - **USAGE** permission for languages and data types (including domains)
- An object owner can revoke the default permissions granted to **PUBLIC** and grant permissions to other users as needed.

Example Operations

Run the following command to revoke the permission for user **u1** to access database **gaussdb**:

Step 1 Connect to the DWS database **gaussdb**.

```
gsql -d gaussdb -p 8000 -h 192.168.x.xx -U dbadmin -W password -r
gaussdb=>
```

Step 2 Create user **u1**.

```
gaussdb=> CREATE USER u1 IDENTIFIED BY 'xxxxxxxx';
```

Step 3 Verify that user **u1** can access GaussDB.

```
gsql -d gaussdb -p 8000 -h 192.168.x.xx -U u1 -W password -r
gaussdb=>
```

Step 4 Connect to database **gaussdb** as administrator **dbadmin** and run the REVOKE command to revoke the **connect on database** permission of user **public**.

```
gsql -d gaussdb -h 192.168.x.xx -U dbadmin -p 8000 -r
gaussdb=> REVOKE CONNECT ON DATABASE gaussdb FROM public;
REVOKE
```

Step 5 Verify the result. Use **u1** to connect to the database. If the following information is displayed, the **connect on database** permission of user **u1** has been revoked successfully:

```
gsql -d gaussdb -p 8000 -h 192.168.x.xx -U u1 -W password -r
gsql: FATAL:  permission denied for database "gaussdb"
DETAIL:  User does not have CONNECT privilege.
```

----End

12.6.8 How Do I View the Table Permissions of a DWS 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 12-8 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

Column	Data Type	Description
privilege_type	character_data	Type of the granted permission. The value can be **SELECT**, **INSERT**, **UPDATE**, **DELETE**, **TRUNCATE**, **REFERENCES**, **ANALYZE**, **VACUUM**, **ALTER**, **DROP**, or **TRIGGER**.
is_grantable	yes_or_no	Indicates if the permission can be granted to other users. **YES** indicates that the permission can be granted to other users, and **NO** indicates that the permission cannot be granted to other users.
with_hierarchy	yes_or_no	Indicates if specific operations are allowed to be inherited at the table level. If the specific operation is **SELECT**, **YES** is displayed. Otherwise, **NO** is displayed.

In the preceding figure, user **u2** has all permissions of table **t2** in schema **u2** and the **SELECT** permission of table **t1** in schema **u1**.

information_schema.table_privileges can query only the permissions directly granted to the user, the **has_table_privilege()** function can query both directly granted permissions and indirect permissions (obtained by GRANT role to user). For example:

```
CREATE TABLE t1 (c1 int);
CREATE USER u1 password '********';
CREATE USER u2 password '********';
GRANT dbadmin to u2; //Indirectly grant permissions through roles.
GRANT SELECT on t1 to u1; // Directly grant the permission.

SET ROLE u1 password '********';
SELECT * FROM public.t1; // Directly grant the permission to access the table.
 c1
----
(0 rows)

SET ROLE u2 password '********';
SELECT * FROM public.t1; // Indirectly grant the permission to access the table.
 c1
----
(0 rows)

RESET role; //Switch back to dbadmin.
SELECT * FROM information_schema.table_privileges WHERE table_name = 't1'; // Can only view direct
grants.
 grantor |  grantee   | table_catalog | table_schema | table_name | privilege_type | is_grantable |
with_hierarchy
---------+------------+---------------+--------------+------------+----------------+--------------+----------------
 dbadmin | u1         | gaussdb       | public       | t1         | SELECT         | NO           | YES
(1 rows)

SELECT has_table_privilege('u2', 'public.t1', 'select'); // Can view both direct and indirect grants.
 has_table_privilege
---------------------
 t
(1 row)
```

Scenario 2: To **check whether a user has permissions on a table**, perform the following steps:

Step 1 Query the **pg_class** system catalog.

```
SELECT * FROM pg_class WHERE relname = 'tablename';
```

Check the **relacl** column. The command output is shown in the following figure. For details about the permission parameters, see **Table 12-9**.

- *rolename=xxxx/yyyy*: indicates that *rolename* has the *xxxx* permission on the table and the permission is obtained from *yyyy*.
- *=xxxx/yyyy*: indicates that **public** has the *xxxx* permission on the table and the permission is obtained from *yyyy*.

Take the following figure as an example:

joe=arwdDxtA: indicates that user **joe** has all permissions (**ALL PRIVILEGES**).

leo=arw/joe: indicates that user **leo** has the read, write, and modify permissions, which are granted by user **joe**.

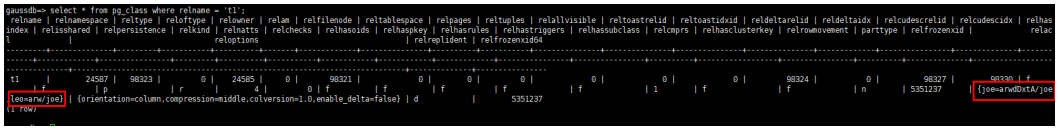


Table 12-9 Permissions parameters

Parameter	Description
r	SELECT (read)
w	UPDATE (write)
a	INSERT (insert)
d	DELETE
D	TRUNCATE
x	REFERENCES
t	TRIGGER
X	EXECUTE
U	USAGE
C	CREATE
c	CONNECT
T	TEMPORARY
A	ANALYZE|ANALYSE
arwdDxtA	ALL PRIVILEGES (for tables)
*	Actions for preceding permissions

Step 2 You can also use the **has_table_privilege** function to query user permissions on tables.


```
SELECT * FROM has_table_privilege('Username','Table_name', 'select');
```

For example, query whether user **joe** has the query permission on table **t1**.

```
SELECT * FROM has_table_privilege('joe','t1','select');
```

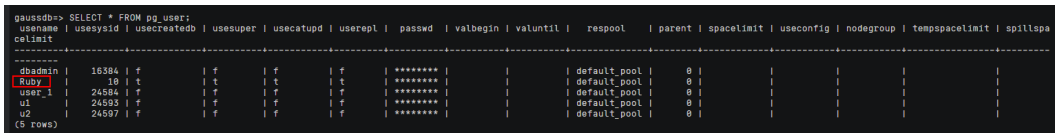
```
gaussdb=> select * from has_table_privilege('joe','t1','select');
 has_table_privilege
---------------------
 t
(1 row)
```

----End

12.6.9 Who Is the Ruby User in the DWS Database?

When you run the **SELECT * FROM pg_user** statement to view the users in the system, you will find the **Ruby** user with extensive permissions.

Ruby is an official O&M Account. After creating a DWS database, a **Ruby** account is automatically generated, involving no security risks; feel free to use it.



12.7 Database Performance

12.7.1 Why Is SQL Execution Slow After Using DWS for a Period of Time?

After a database is used for a period of time, the table data increases as services grow, or the table data is frequently added, deleted, or modified. As a result, bloating tables and inaccurate statistics are incurred, deteriorating database performance.

You are advised to periodically run **VACUUM FULL** and **ANALYZE** on tables that are frequently added, deleted, or modified. Perform the following operations:

Step 1 By default, 100 out of 30,000 records of statistics are collected. When a large amount of data is involved, the SQL execution is unstable, which may be caused by a changed execution plan. In this case, the sampling rate needs to be adjusted for statistics. You can run **set default_statistics_target** to increase the sampling rate, which helps the optimizer generate the optimal plan.

```
gaussdb=> set default_statistics_target=-2;
SET
```

Step 2 Run **ANALYZE** again. For details, see .


```
gaussdb=> ANALYZE customer_t1;
ANALYZE
```

----End

NOTE

To test whether disk fragments affect database performance, use the following function:

```
SELECT * FROM pgxc_get_stat_dirty_tables(30,100000);
```

12.7.2 Why Doesn't DWS Perform Better Than a Single-Node Database in Extreme Scenarios?

Due to the MPP architecture limitation in DWS, a few PostgreSQL system methods and functions cannot be pushed down to DNs for execution. As a result, performance bottlenecks occur only 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 DWS, but software code is optimized to reduce this kind of tasks. Therefore, the GTM does not have much of a bottleneck. In some scenarios, GTM-Free and GTM-Lite can be implemented.
- To ensure excellent performance, services need to be slightly modified for adaptation during migration from the application development mode of the traditional single-node database to that of the parallel database, especially for the traditional stored procedure nesting of Oracle.

Solutions:

- Alternatively, contact technical support to modify and optimize services.

12.7.3 How Do I View SQL Execution Records in a Certain Period When DWS Reads and Writes 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 query, see .
- For details about historical SQL query, see .

12.7.4 DWS CPU Resource Isolation

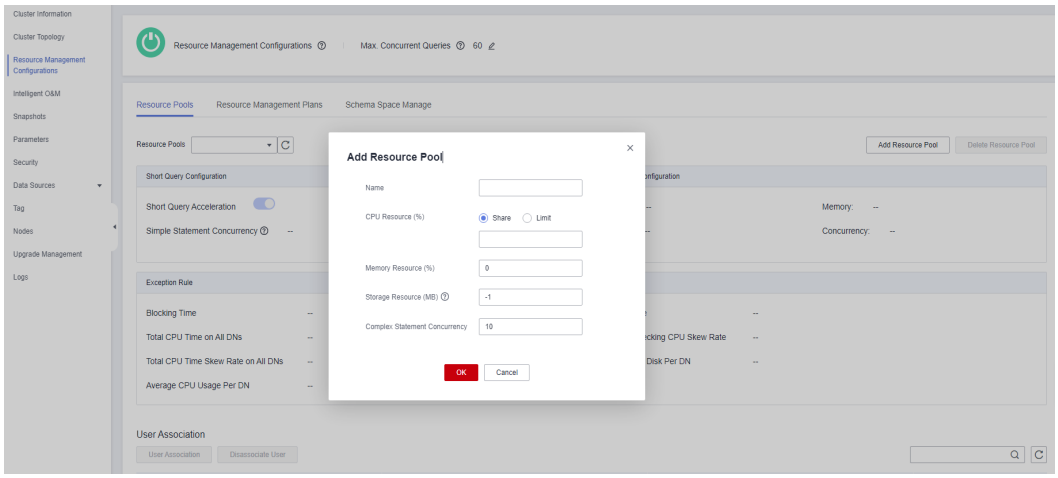
Overview of CPU Resource Management

In different service scenarios, system resources (CPU, memory, I/O, and storage resources) of the database are properly allocated to queries to ensure query performance, and service stability.

DWS allows you to divide resources into different resource pools based on service requirements, with resources in separate pools being isolated from one another. Then, you can associate database users with these resource pools. When a user starts a SQL query, the query will be transferred to the resource pool associated with the user. You can specify the number of queries that can be concurrently executed in a resource pool, the upper limit of memory used for a single query, and the memory and CPU resources that can be used by a resource pool. In this way, you can limit and isolate the resources occupied by different workloads.

DWS primarily utilizes cgroups to manage and control CPU resources, which involves the CPU, cpuacct, and cpuset subsystems. The CPU share is implemented based on the **cpu.shares** of the CPU subsystem. The CPU control is triggered only when the OS CPU is fully occupied. CPU limit is implemented based on the **cpuset**. The **cpuacct** subsystem is used to monitor the CPU resource usage.

To add a resource pool on the DWS console, you can choose **Share** or **Limit** to manage CPU resources as needed.



CPU Share

CPU Share: Percentage of CPU time that can be used by users associated with the current resource pool to execute jobs.

The share has two meanings:

- **Share**: The CPU is shared by all Cgroups, and other Cgroups can use idle CPU resources.
- **Limit**: When the CPU is fully loaded during peak hours, Cgroups preempt CPU resources based on their limits.

CPU share is implemented based by cpu.shares and takes effect only when the CPU is fully loaded. When the CPU is idle, there is no guarantee that a Cgroup will

preempt CPU resources appropriate to its quota. There can still be resource contention when the CPU is idle. Tasks in a Cgroup can use CPU resources without restriction. Although the average CPU usage may not be high, CPU resource contention may still occur at a specific time.

For example, 10 jobs are running on 10 CPUs, and one job is running on each CPU. In this case, any job request for CPU resources will be responded instantly, and there is no contention. If 20 jobs are running on 10 CPUs, the CPU usage may still not be high because the jobs do not always occupy the CPU and may wait for I/O and network resources. The CPU resources seem idle. However, if 2 or more jobs request one CPU at the same time, CPU resource contention occurs, affecting job performance.

CPU Limit

CPU Limit: specifies the percentage of the maximum number of CPU cores that can be used by a database user in the resource pool.

The limit has two meanings:

- Dedicated: The CPU is dedicated to a Cgroup. Other Cgroups with quotas cannot use idle CPU resources.
- Quota: Only the CPU resources in the allocated quota can be used. Idle CPU resources of other Cgroups cannot be preempted.

CPU limit is implemented based on cpuset.cpu. You can set a proper quota to implement absolute isolation of CPU resources between Cgroups. In this way, tasks of different Cgroups will not affect each other. However, the absolute CPU isolation will cause idle CPU resources in a Cgroup to be wasted. Therefore, the limit cannot be too large. A larger limit may not bring a better performance.

For example, in one case, 10 jobs are running on 10 CPUs and the average CPU usage is about 5%. In another case, 10 jobs are running on 5 CPUs and the average CPU usage is about 10%. According to the preceding analysis, although the CPU usage is low when 10 jobs run on five CPUs. However, CPU resource contention still exists. Therefore, the performance of running 10 jobs on 10 CPUs is better than that of running 10 jobs on 5 CPUs. However, it is not the more CPUs, the better. If ten jobs run on 20 CPUs, at any time point, at least 10 CPUs are idle. Therefore, theoretically, running 10 jobs on 20 CPUs does not have better performance than running 10 CPUs. For a Cgroup with a concurrency of N, if the number of allocated CPUs is less than N, the job performance is better with more CPUs. However, if the number of allocated CPUs is greater than N, the job performance will not be improved with more CPUs.

Application Scenarios of CPU Resource Management

The CPU limit and CPU share both have their own advantages and disadvantages. CPU share can fully utilize CPU resources. However, resources of different Cgroups are not completely isolated, which may affect the query performance. CPU limit can implement absolute isolation of CPU resources. However, idle CPU resources will be wasted. Compared with CPU limit, CPU share has higher CPU usage and overall job throughput. Compared with CPU share, CPU limit has complete CPU isolation, which can better meet the requirements of performance-sensitive users.

If CPU contention occurs when multiple types of jobs are running in the database system, you can select different CPU resource control modes based on different scenarios.

- Scenario 1: Fully utilize CPU resources. Focus on the overall CPU throughput instead of the performance of a single type of jobs.

  Suggestion: You are advised not to isolate CPUs for individual users as any CPU management can impact overall CPU usage.

- Scenario 2: A certain degree of CPU resource contention and performance loss are allowed. When the CPU is idle, the CPU resources are fully utilized. When the CPU is fully loaded, each service type needs to use the CPU proportionally.

  Suggestion: You can use CPU share to improve the overall CPU usage while implementing CPU isolation and control when the CPUs are fully loaded.

- Scenario 3: Some jobs are sensitive to performance and CPU resource waste is allowed.

  Suggestion: You can use CPU limit to implement absolute CPU isolation between different types of jobs.

12.7.5 Why Do Regular DWS Users Run Statements Slower Than User dbadmin?

There are three main scenarios where regular DWS users experience slower execution compared to user **dbadmin**.

Scenario 1: Impact of Resource Management on Common Users

Common users often find themselves waiting in various queues, such as the global queue or the CCN queue.

1. Reasons for common users waiting in queues or global queues

   The primary reason for this queueing is the high number of active statements exceeding the maximum value set for **max_active_statements**. Administrators are exempt from queueing as they are not subject to any control measures. Modify the **max_active_statements** parameter on the DWS console.

   a. Log in to the DWS console.

   b. In the navigation pane on the left, choose **Cluster** > **Cluster List**.

   c. In the cluster list, find the target cluster and click the cluster name. The **Cluster Information** page is displayed.

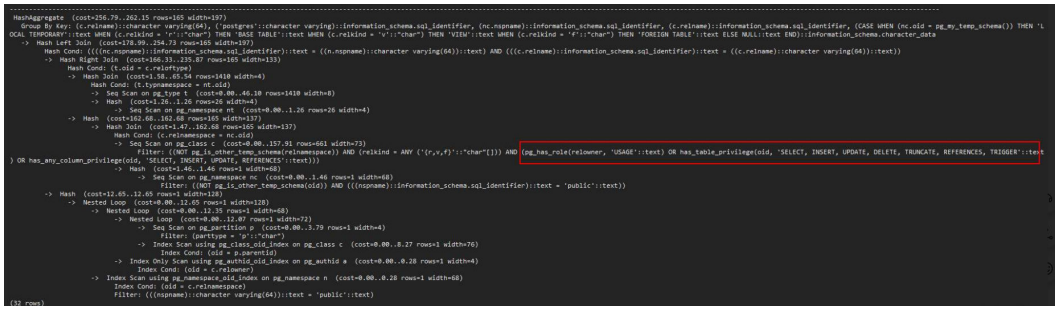
   d. Click **Parameter Modifications**, search for **max_active_statements**, modify its value, and click **Save**. After confirming that the value is correct, click **Save**.

Scenario 2: Executing OR Conditions on Common User Statements Is Time-Consuming

The **OR** conditions in the execution plans contain permission-related checks. This scenario usually occurs when the system view is used. For example, in the following SQL statement:


```
SELECT distinct(dtp.table_name),
 ta.table_catalog,
 ta.table_schema,
 ta.table_name,
 ta.table_type
from information_schema.tables ta left outer join DBA_TAB_PARTITIONS dtp
on (dtp.schema = ta.table_schema and dtp.table_name = ta.table_name)
where ta.table_schema = 'public';
```

Part of the execution plan is as follows:



In the system view, the **OR** condition is used for permission check.

```
pg_has_role(c.relowner, 'USAGE'::text) OR has_table_privilege(c.oid, 'SELECT, INSERT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER'::text) OR has_any_column_privilege(c.oid, 'SELECT, INSERT, UPDATE, REFERENCES'::text)
```

true is always returned for **pg_has_role** of the **dbadmin** use. Therefore, the conditions after **OR** do not need to be checked.

However, the **OR** condition of common users needs to be checked one by one. If there are a large number of tables in the database, the execution time of common users is longer than that of **dbadmin**.

In this scenario, if the number of output result sets is small, you can set **set enable_hashjoin** and **set enable_seqscan** to **off**, to use the index+nestloop plan.

Scenario 3: Differences in Resource Pool Allocation for Common Users and Administrators

Check whether the resource pools corresponding to the user are the same. If they are different, check whether the tenant resources allocated to the two resource pools are different.

```
SELECt * FROM pg_user;
```

12.7.6 Which Factors Affect Single-Table Query Performance in DWS?

DWS employs the shared-nothing architecture, where data is stored in a distributed manner; consequently, the design of the distribution key, the volume of data stored in a single table, and the number of partitions impact the overall query performance of that table.

1. Distribution Key Design

   By default, DWS takes the first column of the primary key as the distribution key. When you define both a primary key and a distribution key for a table, the distribution key must be a subset of the primary key. Distribution keys

determine data distribution among partitions. If distribution keys are well distributed among partitions, query performance can be improved.

If the distribution key is incorrectly selected, data skew may occur after data is imported. The usage of some disks may be much higher than that of other disks, and the cluster may become read-only in some extreme cases. Proper selection of distribution keys is critical to table query performance. In addition, proper distribution keys enable data indexes to be created and maintained more quickly.

2. Data Volume Stored in a Single Table

   The larger the amount of data stored in a single table, the poorer the query performance. If a table contains a large amount of data, you need to store the data in partitions. To convert an ordinary table to a partitioned table, you need to create a partitioned table and import data to it from the ordinary table. When you design tables, plan whether to use partitioned tables based on service requirements.

   To partition a table, comply with the following principles:

   - Use fields with obvious ranges for partitioning, for example, date or region.
   - The partition name must reflect the data characteristics of the partition. For example, its format can be Keyword+Range characteristics.
   - Set the upper limit of a partition to **MAXVALUE** to prevent data overflow.

3. Number of Partitions

   Tables and indexes can be divided into smaller and easier-to-manage units. This significantly reduces search space and improves access performance.

   The number of partitions affects the query performance. If the number of partitions is too small, the query performance may deteriorate.

   DWS supports range partitioning and list partitioning. In range partitioning, records are divided and inserted into multiple partitions of a table. Each partition stores data of a specific range (ranges in different partitions do not overlap). List partitioning is only supported by clusters version 8.1.3 or later.

When designing a data warehouse, you need to consider these factors and perform experiments to determine the optimal design scheme.

12.7.7 How Do I Optimize a SQL Query with Many CASE WHEN Conditions?

In service queries, the **CASE WHEN** statement checks conditions. Too many unnecessary **CASE WHEN** statements in an SQL query can affect performance.

```
SELECT
  SUM(CASE WHEN a > 1 THEN 1 ELSE 0 END) AS a1,
  SUM(CASE WHEN a > 2 THEN 1 ELSE 0 END) AS a2,
  ...
FROM test
WHERE dt = '20241225';
```

In this example, the **CASE WHEN** statement must run for each branch, which increases the query time and affects performance.

DWS offers these optimization policies to address this issue:

Using a Temporary Result Set or Subquery

Extract the complex **CASE WHEN** calculations into a temporary result set or subquery. This avoids repeating the same logic in the main query.

Create a subquery to calculate the intermediate result.

```
SELECT
  sub.a1,
  sub.a2
FROM (
  SELECT
    sum(case when a > 1 then 1 else 0 end) AS a1,
    sum(case when a > 2 then 1 else 0 end) AS a2
  FROM test
  WHERE dt = '20241225'
) sub;


SELECT
    SUM(case_when_a1) as a1,
    SUM(case_when_a2) as a2,
    ...
FROM (
    SELECT
        CASE WHEN a > 1 THEN 1 ELSE 0 END AS case_when_a1,
        CASE WHEN a > 2 THEN 1 ELSE 0 END AS case_when_a2,
        ...
    FROM test
    WHERE dt = '20241225'
) AS subquery;
```

Using a User-Defined Function

Encapsulate the **CASE WHEN** logic in a function. Then, call the function in your query instead of rewriting the **CASE WHEN** logic multiple times.

Create a simple function count_a_gt_value.

```
CREATE OR REPLACE FUNCTION count_a_gt_value(val INT)
RETURNS INT AS $$
DECLARE
    result INT;
BEGIN
    SELECT sum(CASE WHEN a > val THEN 1 ELSE 0 END)
      INTO result
      FROM test
      WHERE dt = '20241225';
    RETURN result;
END;
$$ LANGUAGE plpgsql;
```

Use the user-defined function count_a_gt_value for query.

```
SELECT
  count_a_gt_value(1) AS a1,
  count_a_gt_value(2) AS a2
FROM test;
```

12.8 Backup and Restoration

12.8.1 Why Is It Slow to Create a DWS Automated Snapshot?

This happens when the data to be backed up is large. Automated snapshots are incremental backups, and the lower the frequency you set (for example, one week), the longer it takes. Increase backup frequency to speed up the process.

The following table lists the snapshot backup and restoration rates. (The rates are obtained from the lab test environment with local SSDs as the backup media. The rates are for reference only. The actual rate depends on your disk, network, and bandwidth resources.)

- Backup rate: 200 MB/s/DN
- Restoration rate: 125 MB/s/DN

12.8.2 Does a DWS Snapshot Have the Same Function as an EVS Snapshot?L

No.

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 DWS snapshot is a full backup and an incremental backup of a cluster at a point in time. It records the data in the current database and cluster information, including the number of nodes, node specifications, and database administrator username. Snapshots can be created manually or automatically.

When restoring data from a snapshot to a cluster, DWS creates a cluster based on the cluster information recorded in the snapshot and restores database information from the snapshot data.

For details, see "Backing Up and Restoring a DWS Cluster" in the *Data Warehouse Service (DWS) User 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*.

