Data Warehouse Service

Preguntas frecuentes

Edición	01
Fecha	2023-10-12

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Marcas registradas y permisos

El logotipo HUAWEI y otras marcas registradas de Huawei pertenecen a Huawei Technologies Co., Ltd.
Todas las demás marcas registradas y los otros nombres comerciales mencionados en este documento son propiedad de sus respectivos titulares.

Aviso

Es posible que la totalidad o parte de los productos, las funcionalidades y/o los servicios que figuran en el presente documento no se encuentren dentro del alcance de un contrato vigente entre Huawei Cloud y el cliente. Las funcionalidades, los productos y los servicios adquiridos se limitan a los estipulados en el respectivo contrato. A menos que un contrato especifique lo contrario, ninguna de las afirmaciones, informaciones ni recomendaciones contenidas en el presente documento constituye garantía alguna, ni expresa ni implícita.

Huawei está permanentemente preocupada por la calidad de los contenidos de este documento; sin embargo, ninguna declaración, información ni recomendación aquí contenida constituye garantía alguna, ni expresa ni implícita. La información contenida en este documento se encuentra sujeta a cambios sin previo aviso.

Índice

1 Preguntas frecuentes principales

Recientemente, hemos recopilado las preguntas frecuentes sobre GaussDB(DWS) con el **Chatbot inteligente**. Puede encontrar respuestas a sus preguntas en esta página.

Uso de sintaxis

Tabla 1-1 Uso de sintaxis

Clasificación	Preguntas frecuentes	URL de la página
1	● ¿Cómo puedo consultar el número de bits en una cadena de consulta? ● ¿Cómo se trunca una subcadena? ● ¿Cómo reemplazo algunas cadenas en el resultado devuelto? ● ¿Cómo devuelvo los primeros caracteres de una cadena? ● ¿Cómo puedo filtrar el encabezado y la cola de una cadena? ● ¿Cómo obtengo el número de bytes de una cadena especificada?	**Funciones y operadores de procesamiento de caracteres**
2	● ¿Cómo creo una tabla de particiones? ● ¿Cuáles son los tipos de particiones de tabla compatibles?	**CREATE TABLE PARTITION**
3	● ¿Cómo puedo ver todos los esquemas? ● ¿Cómo puedo ver todas las tablas de un esquema?	**Esquema**

Clasificación	Preguntas frecuentes	URL de la página
4	● ¿Cómo agrego una columna de tabla? ● ¿Cómo cambio un tipo de datos? ● ¿Cómo agrego una restricción NOT NULL a una columna? ● ¿Cómo configuro la clave principal? ● ¿Cómo se modifican los atributos de la tabla?	**ALTER TABLE**
5	● ¿Cómo uso las funciones de fecha? ● ¿Cómo uso pg_sleep()? ● ¿Cómo realizo la sustracción mensual? ● ¿Cómo uso las funciones para cambiar los tipos de fecha?	**Funciones y operadores de procesamiento de hora y fecha**
6	● ¿Cómo cambio las columnas de distribución? ● ¿Cómo cambio la columna de distribución a otra columna? ● ¿Por qué no se pueden actualizar los datos de la columna de distribución y se muestra el mensaje "Distributed key column can't be updated in current version"?	**¿Cómo cambio las columnas de distribución?**
7	● ¿Cómo gestiono las particiones? ● ¿Cómo agrego o elimino una partición? ● ¿Cómo cambio el nombre de una partición? ● ¿Cómo puedo consultar datos en una partición?	**Creación y gestión de tablas particionadas**
8	● ¿Cómo puedo consultar el número de filas en una partición? ● ¿Cómo fusiono dos particiones?	**ALTER TABLE PARTITION**
9	¿Cómo llamo al procedimiento almacenado?	**CALL**
10	● ¿Cómo creo una tabla? ● create table like	**CREATE TABLE**

Clasificación	Preguntas frecuentes	URL de la página
11	● ¿Cuál es el comando para la autorización? ● ¿Cómo uso la sintaxis de concesión? ● ¿Cómo otorgo los derechos de un usuario a otros usuarios? ● ¿Cómo puedo conceder permisos de tabla a un usuario? ● ¿Cómo puedo conceder permisos de base de datos a un usuario? ● ¿Cuáles son los permisos de la tabla externa?	**GRANT**
12	● ¿Cómo uso la función REPLACE? ● ¿Cómo uso to_timestamp? ● ¿Cómo puedo ocultar una marca de tiempo en un formato especificado? ● ¿Cómo utilizo current_timestamp? ● ¿Cómo uso to_number?	**Funciones de conversión de tipo**

Gestión de bases de datos

Tabla 1-2 Gestión de bases de datos

Clasificación	Preguntas frecuentes	URL de la página
1	● ¿Cómo puedo ver las definiciones de tabla? ● ¿Cómo puedo ver el DDL? ● ¿Cómo puedo ver la estructura de las vistas? ● ¿Cómo puedo consultar el tamaño de una base de datos o una tabla?	**Consultar el tamaño de la tabla y la base de datos**

Clasificación	Preguntas frecuentes	URL de la página
2	● ¿Cómo se borran los espacios de tablas? ● ¿Cómo uso la sintaxis VACUUM FULL? ● ¿Cómo puedo mejorar el rendimiento de la consulta después de un gran número de operaciones de adición, eliminación y modificación?	**VACUUM**
3	● ¿Cómo puedo comprobar si un usuario tiene permisos en la tabla actual? ● ¿Cómo puedo comprobar si un usuario tiene un permiso determinado en una tabla?	**¿Cómo puedo ver los permisos de tabla de un usuario?**
4	● ¿Cómo puedo consultar y terminar instrucciones bloqueadas? ● ¿Cómo puedo consultar las sentencias activas? ● ¿Cómo cancelo una sesión? ● ¿Qué debo hacer cuando bloquee los tiempos de espera fuera?	**Análisis de sentencias de SQL que se están ejecutando**
5	● ¿Hay una descripción detallada de los planes de ejecución de SQL? ● ¿Cómo puedo ver el plan de ejecución? ● ¿Cuáles son las diferencias entre Nested Loop, Hash Join y Merge Join?	**Inmersión profunda en el plan de ejecución de SQL**
6	● ¿Cómo cancelo el estado de solo lectura? ● ¿Qué debo hacer cuando la base de datos entra en el estado de solo lectura?	**Un clúster de GaussDB(DWS) se convierte en solo lectura y está bloqueado y los datos no se pueden escribir**
7	● ¿Cómo puedo recopilar estadísticas?	ANALYZE | ANALYSE

Clasificación	Preguntas frecuentes	URL de la página
8	● ¿Cómo configuro el optimizador? ● ¿Cómo puedo habilitar o deshabilitar nestloop? ● ¿Cómo puedo habilitar o deshabilitar mergejoin? ● ¿Cuáles son los parámetros que afectan al plan de ejecución?	**Configuración del método de optimizador**
9	● ¿Cómo cambio la zona horaria de la base de datos? ● ¿Cómo cambio la zona horaria?	**¿Cómo cambiar la zona horaria predeterminada de una base de datos cuando su tiempo es diferente del tiempo del sistema?**
10	● ¿Cómo puedo consultar definiciones de función? ● ¿Cómo uso las funciones de consulta de privilegios de acceso? ● ¿Cómo puedo consultar definiciones de vista?	**Funciones de información del sistema**
11	● ¿Cuáles son las especificaciones técnicas? ● ¿Cuáles son los tamaños de tabla de particiones compatibles? ● ¿Cuál es el volumen máximo de datos en una sola tabla? ● ¿Cuál es el número máximo de columnas soportadas por una tabla?	**Especificaciones técnicas**
12	● ¿Cuáles son las operaciones del desarrollador? ● ¿Cómo especifico si se habilita el uso del optimizador de un framework distribuido?	**Opciones de desarrollador**

Gestión de clúster

Tabla 1-3 Gestión de clúster

Clasificación	Preguntas frecuentes	URL de la página
1	● ¿Qué debo hacer cuando el uso del disco es demasiado alto? ● ¿Qué debo hacer si el disco está lleno? ● ¿Qué debo hacer si el clúster pasa a ser de solo lectura? ● ¿Qué debo hacer si se genera una alarma de disco? ● ¿Cómo configuro el umbral del disco? ● ¿Cómo puedo activar la suscripción de alarma de disco?	**Un clúster de GaussDB(DWS) se convierte en solo lectura y está bloqueado y los datos no se pueden escribir**
2	● ¿Cómo puedo ver la información básica del clúster? ● ¿Cómo puedo ver la EIP de un clúster? ● ¿Cómo puedo ver la dirección de conexión a la base de datos? ● ¿Cuáles son las variantes de clúster? ● ¿Cuál es la información básica del disco?	**Consulta de detalles del clúster**
3	¿Cómo puedo comprar un clúster?	**Creación de un clúster**
4	● ¿Qué debo hacer si el clúster está desequilibrado? ● ¿Cómo se realiza la conmutación del clúster primario/en espera?	**Realización de una conmutación de retorno principal/en espera**

2 Problemas generales

2.1 ¿Por qué son necesarios los almacenes de datos?

Situación actual y requisitos

Muchos datos (pedidos, existencias, materiales y pagos) se generan en los sistemas de operación de negocios y en la base de datos (transaccional) de las empresas.

Los responsables de la toma de decisiones categorizan y analizan los datos para la toma de decisiones empresariales.

Dificultades

La categorización y el análisis de datos implican el acceso simultáneo a los datos en múltiples tablas de base de datos. Es decir, múltiples tablas que se actualizan mediante diferentes transacciones pueden bloquearse al mismo tiempo, lo que puede causar complicaciones a los sistemas de base de datos durante las horas pico.

- El bloqueo de varias tablas aumenta la latencia de las consultas complejas.
- Las transacciones que están actualizando las tablas de la base de datos están bloqueadas, causando retrasos o interrupciones.

Solución

Los almacenes de datos sobresalen en la agregación y asociación de datos, por lo que los usuarios extraen más datos, obtienen más información y toman mejores decisiones. La minería requiere consultas complejas que involucran datos en varias tablas.

El proceso ETL copia los datos de las bases de datos de operaciones empresariales a los almacenes de datos para su análisis e informática. Los datos se pueden agregar desde múltiples sistemas de operaciones empresariales en un almacén de datos para una mejor asociación, análisis e información procesable.

Los almacenes de datos y las bases de datos estándar orientadas a las transacciones, como Oracle, SQL Server y MySQL, utilizan diferentes modos de diseño. Los almacenes de datos están optimizados en términos de agregación y asociación de datos, pero es posible que no se garanticen las funciones o el rendimiento de la transacción o la adición y eliminación de

datos. Por lo tanto, los almacenes de datos y las bases de datos se aplican a diferentes escenarios. Las bases de datos transaccionales se dedican al procesamiento de transacciones (operación comercial de las empresas), mientras que los almacenes de datos sobresalen en el análisis de datos complejos. En conclusión, las bases de datos se aplican a las actualizaciones de datos, mientras que los almacenes de datos se aplican al análisis de datos.

2.2 ¿Por qué debería usar la nube pública GaussDB(DWS)?

Los almacenes de datos convencionales no son prácticos para las empresas más pequeñas debido al alto costo, la selección y adquisición de dispositivos y sistemas que consumen mucho tiempo, y el complejo escalamiento horizontal.

GaussDB(DWS) en la nube pública es la mejor opción:

- Este servicio en la nube de almacenamiento de datos MPP distribuido es muy abierto, eficiente, compatible, escalable, y es fácil de O&M.
- Desarrollado en el núcleo FusionInsight de almacén de datos LibrA, permite a las empresas de la nube pública una experiencia mejor y consistente dentro y fuera de la nube.

  FusionInsight LibrA es un sistema de almacenamiento de datos distribuidos de última generación con derechos de propiedad intelectual independientes. Actualmente, es ampliamente utilizado en gobierno, finanzas y operadores. FusionInsight LibrA es compatible con bases de datos Postgres de código abierto, especialmente en sentencias SQL de Oracle y Teradata. Los ingenieros de GaussDB(DWS) han diseñado un núcleo de almacenes híbridos de fila-columna no solo para un análisis más rápido, sino también para el procesamiento de datos, como agregar, eliminar, modificar datos. FusionInsight LibrA cuenta con las tecnologías de optimización de costes y almacén, incluido la computación vectorial de código de máquina y entre/dentro paralelismo para operadores y nodos. Utiliza LLVM para optimizar el código local en los planes de consulta de compilación. Las consultas y análisis de datos más potentes abordan los puntos problemáticos del servicio y mejoran la experiencia del usuario.
- GaussDB(DWS) se puede utilizar fuera de la caja.

  La solicitud de GaussDB(DWS) en la nube pública toma solo unos minutos, lo que le libera del lento proceso de búsqueda y compra de almacenes de datos. Esto no solo simplifica la adquisición, sino que también reduce el costo y los requisitos para el uso de almacenes de datos. Las pequeñas y medianas empresas con acceso a GaussDB(DWS) pueden extraer sin problemas los valores de los datos para su desarrollo y formar información procesable.

2.3 ¿Debo elegir GaussDB(DWS) de la nube pública o RDS?

Ambos permiten ejecutar bases de datos relacionales convencionales en la nube y transferir cargas de gestión de bases de datos. Las bases de datos RDS son útiles para OLTP, informes y análisis, pero son menos capaces de manejar operaciones de lectura de una gran cantidad de datos (consultas complejas de solo lectura). GaussDB(DWS) es útil para OLAP al reducir las cargas de trabajo de análisis e informes de grandes conjuntos de datos en un orden de magnitud, gracias a su escala y recursos multinodo y algoritmos optimizados (**almacenamiento de columnas, ejecutadores vectorizados y marcos distribuidos**).

Puede escalar un clúster de GaussDB(DWS) para abordar datos y consultas complejos, o para manejar análisis abrumadores e informes de cargas de trabajo que afectan al rendimiento de OLTP.

La siguiente tabla muestra la comparación entre OLTP y OLAP.

Tabla 2-1 Comparación de características entre OLTP y OLAP

Características	RDS para OLTP	GaussDB(DWS) para OLAP
Usuario	Operaciones y gestión de bajo nivel	Tomadores de decisiones y alto gestión
Función	Procesamiento de operación diaria	Análisis y toma de decisiones
Diseño	Por aplicación	Por tema
Colaboración	Último, detallado, bidimensional, discreto	Histórico, integrado, multidimensional, unificado
Acceso	Decenas de registros de lectura y escritura	Millones de registros de lectura
Cobertura	Operaciones sencillas de lectura/ escritura	Consultas complejas
Tamaño de la base de datos	Cientos de GB	TB o PB

2.4 ¿Qué es la cuota de usuario?

Para los servicios de HUAWEI CLOUD, las cuotas limitan el número de recursos disponibles para los usuarios. Si necesita más, envíe un ticket de servicio para aumentar sus cuotas. Una vez aprobada, actualizaremos su cuota de recursos en consecuencia y le enviaremos una notificación. Para obtener más información sobre las operaciones de cuota, consulte **Cuotas**.

2.5 ¿Cuáles son las diferencias entre usuarios y roles?

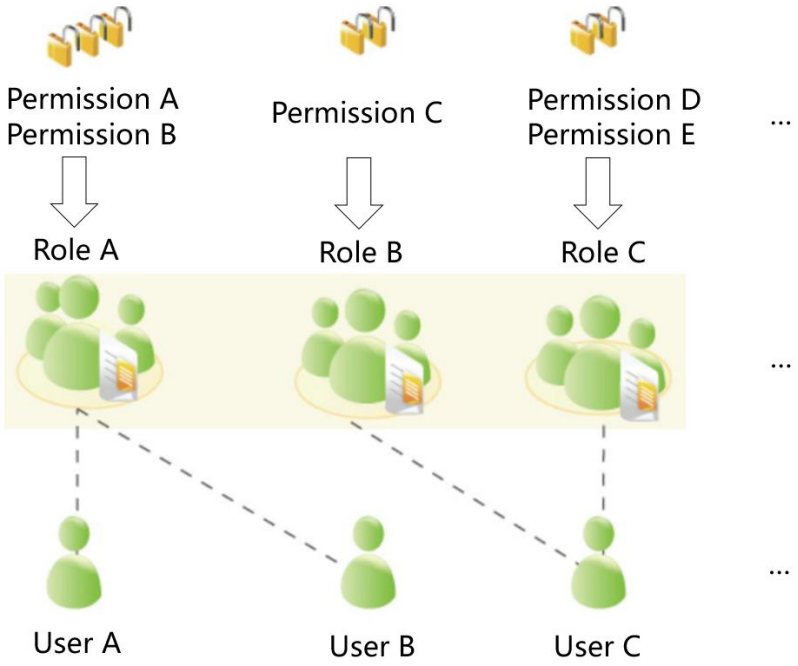
Los usuarios y los roles se comparten dentro de todo el clúster, pero sus datos no se comparten. Es decir, un usuario puede conectarse a cualquier base de datos, pero una vez que la conexión es exitosa, cualquier usuario puede acceder solo a la base de datos declarada en la solicitud de conexión.

- Un rol es un grupo de permisos. Por lo general, los roles se utilizan para ordenar los permisos. Los usuarios se utilizan para gestionar permisos y realizar operaciones.
- Un rol puede heredar permisos de otros roles. Todos los usuarios de un grupo de usuarios heredan automáticamente los permisos de operación del rol del grupo.
- En una base de datos, los permisos de los usuarios provienen de roles.
- Un grupo de usuarios es un grupo de usuarios que tienen el mismo permiso.

- Un usuario puede ser considerado como un rol con el permiso de inicio de sesión.
- Un rol puede considerarse como un usuario sin el permiso de inicio de sesión.

Los permisos proporcionados por Gauss (DWS) incluyen los permisos de operación y mantenimiento para los componentes en el plano de gestión. Puede asignar diferentes permisos a los usuarios según sea necesario. El plano de gestión utiliza roles para una mejor gestión de permisos. Puede seleccionar los permisos especificados y asignarlos a los roles de una manera unificada. De esta manera, los permisos se pueden ver y gestionar de manera centralizada.

En la siguiente figura se muestran las relaciones entre permisos, roles y usuarios en la gestión unificada de permisos.



GaussDB(DWS) proporciona varios permisos. Seleccione y asigne permisos a diferentes usuarios en función de los escenarios de servicio. Un rol se le pueden asignar uno o más permisos.

Después de que un rol es otorgado a un usuario a través de **GRANT**, el usuario tendrá todos los permisos del rol. Se recomienda que los roles se utilicen para conceder permisos de manera eficiente. Un usuario solo tiene permisos para sus propias tablas, pero no tiene permisos para las tablas de otros usuarios en sus esquemas.

- Al rol A se le asignan permisos de operación A y B. Después de asignar el rol A al usuario A y al usuario B, el usuario A y el usuario B pueden obtener permisos de operación A y B.
- Al rol B se le asigna el permiso de operación C. Después de asignar el rol B al usuario C, el usuario C puede obtener permisos de operación C.
- Al rol C se le asignan permisos de operación D y E. Después de asignar el rol C al usuario C, el usuario C puede obtener los permisos de operación D y F.

2.6 ¿Cuándo debo usar GaussDB(DWS) y MRS?

MRS funciona mejor con marcos de procesamiento de big data como Apache Spark, Hadoop y HBase, para procesar y analizar conjuntos de datos ultragrandes a través de código personalizado. Le permite controlar las configuraciones del clúster y el software instalado en el clúster.

GaussDB(DWS) funciona mejor con consultas complejas de una gran cantidad de datos estructurados. Su objetivo es reunir datos de diferentes fuentes, como inventario, finanzas y sistema minorista. Para garantizar la coherencia y la precisión de los informes empresariales, GaussDB(DWS) almacena los datos de una manera altamente estructurada. Esta estructura puede crear directamente la regla de coherencia de datos en la tabla de la base de datos. Además, GaussDB(DWS) es altamente compatible con instrucciones SQL estándar y la sintaxis de bases de datos convencionales compatibles con transacciones.

GaussDB(DWS) se prefiere cuando se desea realizar una consulta compleja de una gran cantidad de datos estructurados con alto rendimiento.

2.7 ¿Cómo verifico el tiempo de creación de un usuario de base de datos?

Método 1:

Cuando se crea un usuario de base de datos de GaussDB(DWS), si el momento en que el usuario entra en vigor (**VALID BEGIN**) es el mismo que el tiempo de creación del usuario, y el momento en que el usuario entra en vigor no se ha cambiado, puede comprobar la columna **valbegin** en la vista **PG_USER** para comprobar la hora de creación del usuario.

A continuación se presenta un ejemplo:

Cree **jerry** de usuario y establezca su hora de inicio de validez en su hora de creación actual.

```
CREATE USER jerry PASSWORD 'password' VALID BEGIN '2022-05-19 10:31:56';
```

Consulte los usuarios en la vista **PG_USER**. La columna **valbegin** indica el momento en que **jerry** entró en vigor, es decir, el momento en que se creó jerry.

```
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)
```

Método 2:

Compruebe la columna **passwordtime** en el catálogo del sistema **PG_AUTH_HISTORY**. Esta columna indica la hora en que se creó la contraseña inicial del usuario. Solo los usuarios con permisos de administrador del sistema pueden acceder al catálogo.

```
select roloid, min(passwordtime) as create_time from pg_auth_history group by
roloid order by roloid;
```

A continuación se presenta un ejemplo:

Consulte la vista **PG_USER** para obtener el OID del **jerry** de usuario, que es el nombre **457386**. Consulte la columna **passwordtime** para obtener el tiempo de creación del **jerry** de usuario, que es el de **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)
```

2.8 Regiones y AZ

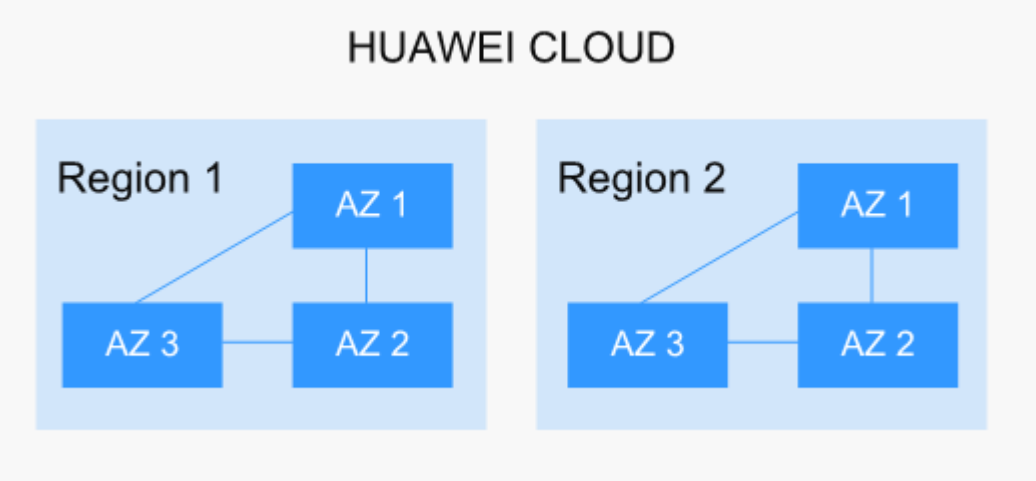
Conceptos

Una región y una zona de disponibilidad (AZ) identifican la ubicación de un centro de datos. Puede crear recursos en regiones y zonas de disponibilidad.

- **Regiones** se definen en términos de ubicación geográfica y latencia de red. Cada región tiene sus propios servicios públicos compartidos (ECS, EVS, OBS, VPC, EIP e IMS). Las regiones son comunes o dedicadas. Una región común proporciona servicios en la nube comunes disponibles para todos los tenants. Una región dedicada proporciona servicios de un tipo específico o solo para tenants específicos.
- Una **AZ** contiene uno o más centros de datos físicos. Cada AZ tiene instalaciones independientes de refrigeración, extinción de incendios, antihumedad y electricidad. La computación, la red, el almacenamiento y otros recursos en una AZ se dividen lógicamente en múltiples clústeres. Las AZ de una región están interconectadas a través de fibra óptica de alta velocidad, por lo que los sistemas implementados en las AZ pueden lograr una mayor disponibilidad.

La **Figura 2-1** muestra la relación entre las regiones y las AZ.

Figura 2-1 Regiones y AZ



Huawei Cloud ofrece servicios en todo el mundo. Puede seleccionar una región y AZ según sea necesario. Para obtener más información, consulte **Productos y servicios globales**.

¿Cómo selecciono una región?

Al seleccionar una región, tenga en cuenta lo siguiente:

- Localización

  Seleccione una región cercana a usted o a sus usuarios de destino para una menor latencia de red y un acceso más rápido. Las regiones en China continental proporcionan la misma infraestructura, calidad de red BGP y operaciones y configuraciones de recursos. Por lo tanto, si sus usuarios objetivo se encuentran en China continental, no es necesario tener en cuenta las diferencias de latencia de la red al seleccionar una región.

  Los países y regiones fuera de China continental, como Bangkok y Hong Kong, proporcionan servicios a los usuarios fuera de China continental. Si usted o sus usuarios objetivo están en China continental, estas regiones no se recomiendan debido a la alta latencia de acceso.

  – Si usted o sus usuarios objetivo se encuentran en la región de Asia Pacífico (excepto China continental), seleccione la región de **AP-Bangkok** o **AP-Singapore**.
  – Si usted o sus usuarios objetivo están en África, seleccione la región de **AF-Johannesburg**.
  – Si usted o sus usuarios objetivo están en Europa, seleccione la región **EU-Paris**.

- Precios de recursos

  Esto varía según la región. Para obtener más información, consulte los **detalles de precios**.

¿Cómo selecciono una AZ?

Tenga en cuenta sus requisitos para DR y latencia de red al seleccionar un AZ:

- Despliegue recursos en diferentes AZ en la misma región para fines de recuperación ante desastres.
- Despliegue recursos en la misma AZ para una latencia mínima.

Regiones y puntos de conexión

Cuando usa recursos con invocaciones a API, debe especificar el punto de conexión regional. Para obtener más información sobre las regiones y puntos de conexión de Huawei Cloud, consulte **Regiones y puntos de conexión**.

2.9 ¿Mis datos están seguros en GaussDB(DWS)?

Sí. En la era del big data, los datos se han convertido en un activo fundamental. La nube pública adherirá al compromiso asumido a lo largo de los años de no tocar sus aplicaciones o datos, ayudándole a proteger sus activos principales. Este es nuestro compromiso con los usuarios y la sociedad, sentando las bases para el éxito empresarial de la nube pública y sus socios.

GaussDB(DWS) es un sistema de almacenamiento de datos con seguridad de clase de telecomunicaciones para salvaguardar sus datos y privacidad. Además, GaussDB(DWS) en la nube pública ofrece calidad de operador, que puede satisfacer los requisitos de seguridad y privacidad de datos de gobiernos, organizaciones financieras y operadores. Por lo tanto, es ampliamente utilizado por diversas industrias. GaussDB(DWS) de la nube pública ganó la siguiente autenticación de seguridad:

- Laboratorio interno de seguridad cibernética (ICSL) en cumplimiento de los estándares de seguridad cibernética emitidos por las autoridades del UK.
- Certificación de Evaluación de Privacidad y Seguridad (PSA) para cumplir con los requisitos de la UE en materia de seguridad de datos y privacidad.

Seguridad de datos de servicio

GaussDB(DWS) se basa en la infraestructura de software de la nube pública, incluidos ECS y OBS. Tanto ECS como OBS ganaron la Certificación de nube confiable emitida por China Data Center Alliance en 2017.

Los datos de servicio de los usuarios de GaussDB(DWS) se almacenan en los ECS del clúster. Ni los usuarios ni los administradores de O&M de la nube pública pueden iniciar sesión en los ECS.

El sistema operativo de los ECS está reforzado para la seguridad, incluidos el refuerzo del núcleo, la instalación del último parche, el control de permisos, la gestión de puertos y el protocolo y el puerto anti-ataque.

GaussDB(DWS) proporciona medidas de seguridad completas, como políticas de contraseñas, autenticación, gestión de sesiones, gestión de permisos de usuario y auditoría de bases de datos.

Seguridad de datos de instantáneas

Las copias de seguridad de GaussDB(DWS) son instantáneas almacenadas en OBS. OBS ha pasado la autenticación de seguridad en la nube de confianza de China Data Center Alliance. Es compatible con funciones de control de permisos de acceso, acceso a claves y encriptación de datos. Los datos de instantáneas de GaussDB(DWS) solo se pueden usar para copias de seguridad y restauración de datos y ningún usuario puede acceder a ellos. Los administradores de GaussDB(DWS) pueden ver el espacio OBS ocupado por los datos de instantáneas en la consola de GaussDB(DWS) y las facturas de la nube pública .

Seguridad de acceso a la red

GaussDB(DWS) está completamente aislado entre las redes de capa 2 y capa 3 para cumplir con los requisitos de seguridad de los usuarios gubernamentales y financieros.

- GaussDB(DWS) se despliega en el entorno de ECS dedicado al tenant, que no se comparte con otros tenants. Por lo tanto, la fuga de datos debido al uso compartido de recursos informáticos es imposible físicamente.
- ECS en un clúster de GaussDB(DWS) se aíslan a través de VPC, evitando que los ECS sean descubiertos e intruidos por otros tenants.
- La red se divide en el plano de servicio y el plano de gestión. Los dos planos están aislados físicamente, lo que garantiza la seguridad de la red.
- Los tenants pueden personalizar de forma flexible el grupo de seguridad y las reglas de acceso.
- Software de aplicaciones externas de acceso GaussDB(DWS) a través de SSL.
- Los datos importados de OBS están cifrados.

2.10 ¿Cómo se asegura GaussDB(DWS)?

GaussDB(DWS) utiliza IAM y VPC para controlar el acceso del usuario y aislar la red del clúster. El acceso al clúster es a través de SSL y paquete de cifrado. Además, GaussDB(DWS) admite la autenticación de certificados digitales bidireccionales.

Los OS de nodo de cada clúster están reforzados para permitir un acceso válido solo a los archivos de sistema operativo.

2.11 ¿Puedo modificar el grupo de seguridad de un clúster de GaussDB(DWS)?

Sí. Después de crear un clúster de almacén de datos, no se puede cambiar su grupo de seguridad. Sin embargo, puede agregar, eliminar o modificar reglas del grupo de seguridad actual.

Para editar el grupo de seguridad del clúster:

1. Inicie sesión en la consola de gestión de GaussDB(DWS).
2. En el árbol de navegación de la izquierda, haga clic en **Clusters**.
3. En la lista de clústeres, busque el clúster objetivo y haga clic en el nombre del clúster. Se muestra **Basic Information**.
4. Busque el parámetro **Grupo de seguridad** y haga clic en el nombre del grupo de seguridad para cambiar a la página de **Grupos de seguridad** en la consola de VPC, en la que puede establecer el grupo de seguridad.

2.12 ¿Cómo se relacionan LibrA, GaussDB A y GaussDB(DWS)?

GaussDB(DWS) es una base de datos de procesamiento de datos en línea construida sobre la infraestructura y plataforma de nube pública. Evolucionó a partir de la GaussDB A de Huawei

(originalmente llamado FusionInsight LibrA). GaussDB A es un software de base de datos implementado en máquinas físicas. Para obtener más información, visite los siguientes sitios web:

- Versión 6.5.1 o anterior **https://support.huawei.com/enterprise/en/cloud-computing/gaussdb-200-pid-21407429**
- Versión 8.0.0:**https://support.huawei.com/enterprise/en/cloud-computing/gaussdb-a-pid-250949677**

2.13 What Is a Database/Data Warehouse/Data Lake/ Lakehouse?

The evolving Internet and IoT produce massive volumes of data. This data needs to be managed, using concepts like database, data warehouse, data lake, and lakehouse. What are these concepts? What are their relationships? What are the specific products and solutions? This document helps you understand them through comparison.

Database

A database is where data is organized, stored, and managed by data structure.

Databases have been used in computers since the 1960s, with the two prevailing data models (hierarchical and network), and data and applications were very interdependent. This limited database applications.

A database usually refers to a relational database. A relational database organizes data with a relational model, that is, data is stored in rows and columns. Therefore, database data is well-structured and independent with low redundancy. In 1970, relational databases were born to completely separate data from applications for software and have become an indispensable part of mainstream computer systems. Relational databases are the foundation of database products from all vendors, with relational API support even if a database is non-relational.

Relational databases process basic and routine transactions using OLTP, such as bank transactions.

Data Warehouse

Database growth has facilitated data growth. OLAP explores the relationship between data and mines more data value. However, it is difficult to share data between different databases, and data integration and analysis also face great challenges.

To overcome these challenges for enterprises, Bill Inmon, proposed the idea of data warehousing in 1990. The data warehouse runs on a unique storage architecture to perform OLAP on a large amount of the OLTP data accumulated over the years. In this way, enterprises can obtain valuable information from massive data quickly and effectively to make informed decisions. Thanks to data warehouses, the information industry has evolved from operational systems based on relational databases to decision support systems.

Unlike a database, a data warehouse has the following features:

- A data warehouse uses themes. It is built to support various services, with data coming from scattered operational data. Therefore, the required data needs to be extracted from multiple heterogeneous data sources, processed and integrated, and reorganized by theme.

- A data warehouse mainly supports enterprise decision analysis and the operations involved are focused on data query. Therefore, it improves the query speed and cuts the total cost of ownership (TCO) by optimizing table structures and storage modes.

Tabla 2-2 Comparison between data warehouses and databases

Dimension	Data Warehouse	Database
Application scenario	OLAP	OLTP
Data source	Multiple	Single
Data normalization	Denormalized schemas	Highly normalized static schemas
Data access	Optimized read operations	Optimized write operations

Data Lake

Data is an important asset for enterprises. Production and operations data are saved and distilled into effective management policies.

The data lake does that. It is a large data warehouse that centrally stores structured and unstructured data. It can store raw data of multiple data sources and types, meaning that data can be accessed, processed, analyzed, and transmitted without being structured first. The data lake helps enterprises quickly complete federated analysis of heterogeneous data sources and explore data value.

A data lake is in essence a solution that consists of a data storage architecture and data processing tools.

- The **storage architecture** must be scalable and reliable enough to store massive data of any type (structured, semi-structured, unstructured data).
- The two types of **processing tools** have separate functions:
  - The first type: migrates data into the lake, including defining sources, formulating synchronization policies, moving data, and compiling catalogs.
  - The second type then uses that data, including analyzing, mining, and using it. The data lake must be equipped with wide-ranging capabilities, such as comprehensive data and data lifecycle management, diversified data analytics, and secure data acquisition and release. These data governance tools help guarantee data quality, which can be compromised by a lack of metadata and turn the data lake into a data swamp.

Now with big data and AI, lake data is even more valuable and plays new roles. It represents more enterprise capabilities. For example, the data lake can centralize data management, helping enterprises build more optimized operation models. It also provides other enterprise capabilities such as prediction analysis and recommendation models. These models can stimulate further growth.

Just like any other warehouse and lake, one stores goods, or data, from one source while the other stores water, or data, from many sources.

Tabla 2-3 Comparison between data lakes and data warehouses

Dimension	Data Lake	Data Warehouse
Application scenario	Exploratory analytics (machine learning, data discovery, profiling, prediction)	Data analytics (based on historical structured data)
Cost	Low initial cost, high subsequent cost	High initial cost, low subsequent cost
Data quality	Massive raw data to be cleaned and normalized before use	High quality data that can be used as the basis of facts
Target user	Data scientists and data developers	Business analysts

Lakehouse

Although the application scenarios and architectures of a data warehouse and a data lake are different, they can cooperate to resolve problems. A data warehouse stores structured data and is ideal for quick BI and decision-making support, while a data lake stores data in any format and can generate larger value by mining data. Therefore, their convergence can bring more benefits to enterprises in some scenarios.

A lakehouse, the convergence of a data warehouse and a data lake, aims to enable data mobility and streamline construction. The key of the lakehouse architecture is to enable the free flow of data/metadata between the data warehouse and the data lake. The explicit-value data in the lake can flow to or even be directly used by the warehouse. The implicit-value data in the warehouse can also flow to the lake for long-term storage at a low cost and for future data mining.

Intelligent Data Solution

DataArts Studio is a data enablement platform that helps large government agencies and companies customize intelligent data resource management solutions. This solution can import all-domain data into the data lake, eliminating data silos, unleashing the value of data, and empowering data-driven digital transformation.

DataArts Studio features the FusionInsight intelligent data lake as its core. Around it are computing engines such as the database, data warehouse, data lake, and data platform. It provides comprehensive data enablement, covering data collection, aggregation, computing, asset management, and data openness.

Lake, warehouse, and database engines enable agile data lake construction, fast migration of GaussDB databases, and real-time analysis of the data warehouse. For more information, go to:

- Database
  - Relational databases include: **Relational Database Service (RDS)** , **GaussDB(for MySQL)**, **GaussDB** , **RDS for PostgreSQL** .
  - Non-relational database: **Document Database Service (DDS)**, GaussDB NoSQL (including **Influx**, **Redis**, **Mongo**, **Cassandra**)

- Data warehouse: **GaussDB(DWS)**
- Data lake and warehouse integration: **MapReduce Service (MRS)**, **Data Lake Insight (DLI)** .
- Data governance center: **DataArts Studio**.

2.14 How Are Dirty Pages Generated in GaussDB(DWS)?

Causes

By using the versioning concurrency control (MVCC) mechanism, GaussDB(DWS) can achieve consistency and concurrency for multiple transactions that access the database simultaneously. This mechanism has the benefit of avoiding read-write conflicts, but the drawback of causing disk bloat and dirty pages.

The scenarios are as follows:

- When the DELETE operation is performed on a table, data is logically deleted but not physically deleted from the disk.
- When the UPDATE operation is performed on a table, GaussDB(DWS) logically marks the data to be updated as delete and inserts new data.

For the DELETE and UPDATE operations in a table, the data marked as deleted is called discarded tuples. The proportion of discarded tuples in the entire table is the dirty page rate. Therefore, when the dirty page rate of a table is high, the proportion of data marked as deleted in the table is high.

Solution:

GaussDB(DWS) provides a system view for querying the dirty page rate. For details, see section **PGXC_STAT_TABLE_DIRTY**.

To solve the problem of disk space bloat caused by high dirty page rate, GaussDB(DWS) provides the VACUUM function to clear the data logically marked as deleted. For details, see **VACUUM**.

VACUUM does not release the allocated space. To completely reclaim the cleared space, run **VACUUM FULL**.

NOTA

- **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.

3 Uso de la base de datos

3.1 How Do I Change Distribution Columns?

In a data warehouse database, you need to carefully choose distribution columns for large tables, because they can affect your database and query performance. If an improper distribution key is used, data skew may occur after data is imported. As a result, the usage of some disks will be much higher than that of other disks, and the cluster may even become read-only. If the hash distribution policy is used and data skew occurs, the I/O performance of some 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.

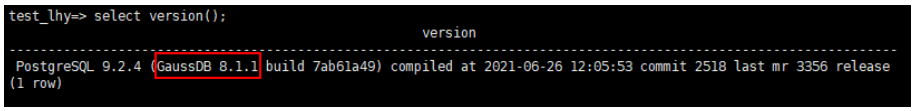
Choosing an Appropriate Distribution Column

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.



- **For 8.0.x and earlier versions, specify the distribution column when rebuilding a table.**

Paso 1 Use Data Studio or gsql in Linux to access the database.

Paso 2 Create a table.

NOTA

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);
```

Paso 3 Migrate data to the new table.

```
START TRANSACTION;
LOCK TABLE table1 IN ACCESS EXCLUSIVE MODE;
INSERT INTO table1_new SELECT * FROM table1;
COMMIT;
```

Paso 4 Verify that the table data has been migrated. Delete the original table.

```
SELECT COUNT(*) FROM table1_new;
DROP TABLE table1;
```

Paso 5 Replace the original table.

```
ALTER TABLE table1_new RENAME TO table1;
```

----Fin

3.2 How Do I View and Set the Database Character Encoding?

Viewing the Database Character Encoding

Use the **server_encoding** parameter to check the character set encoding of the current database. For example, the character encoding of database **music** is UTF8.

```
music=> SHOW server_encoding;
 server_encoding
-----------------
 UTF8
(1 row)
```

Setting the Database Character Encoding

NOTA

GaussDB(DWS) does not support the modification of the character encoding format of a created database.

If you need to specify the character encoding format of a database, use **template0** and the **CREATE DATABASE** syntax to create a database. To make your database compatible with most characters, you are advised to use the UTF8 encoding when creating a database.

CREATE DATABASE syntax

```
CREATE DATABASE database_name
    [ [ WITH ] { [ OWNER [=] user_name ] |
                 [ TEMPLATE [=] template ] |
                 [ ENCODING [=] encoding ] |
                 [ LC_COLLATE [=] lc_collate ] |
                 [ LC_CTYPE [=] lc_ctype ] |
                 [ DBCOMPATIBILITY [=] compatibility_type ] |
                 [ CONNECTION LIMIT [=] connlimit ]}[...] ];
```

- **TEMPLATE [=] template**

  Indicates the template name, that is, the name of the template to be used to create the database. GaussDB(DWS) creates a database by copying a database template. GaussDB(DWS) has two initial template databases **template0** and **template1** and a default user database **gaussdb**.

  Value range: an existing database name. If this is not specified, the system copies **template1** by default. Its value cannot be **gaussdb**.

  AVISO

  Currently, database templates cannot contain sequences. If sequences exist in the template library, database creation will fail.

- **ENCODING [=] encoding**

  Character encoding used by the database. The value can be a character string (for example, **SQL_ASCII'**) or an integer number.

  If this parameter is not specified, the encoding of the template database is used by default. The encoding of template databases **template0** and **template1** depends on the OS by default. The character encoding of **template1** cannot be changed. To change the encoding, use **template0** to create a database.

  Value range: **GBK**, **UTF8**, and **Latin1**

  AVISO

  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;
```

3.3 What Do I Do If Date Type Is Automatically Converted to the Timestamp Type During Table Creation?

When creating a database, you can set the **DBCOMPATIBILITY** parameter to the compatible database type. The value of **DBCOMPATIBILITY** can be **ORA**, **TD**, and **MySQL**, indicating Oracle, Teradata, and MySQL databases, respectively. If this parameter is

not specified during database creation, the default value **ORA** is used. In ORA compatibility mode, the date type is automatically converted to timestamp(0). The date type is only supported in the MySQL compatibility mode.

To solve the problem, you need to change the compatibility mode to MySQL. The compatibility mode of an existing database cannot be changed. It can only be specified during creation of the database. GaussDB(DWS) supports the MySQL compatibility mode in cluster version 8.1.1 and later. To configure this mode, run the following commands:

```
gaussdb=> CREATE DATABASE mydatabase 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 trings 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)
```

3.4 Do I Need to Run VACUUM FULL and ANALYZE on Common Tables Periodically?

Yes.

For tables that are frequently added, deleted, or modified, you need to periodically perform **VACUUM FULL** and **ANALYZE** to reclaim the disk space occupied by updated or deleted data, preventing performance deterioration caused by data bloat and inaccurate statistics.

- Generally, you are advised to perform **ANALYZE** after a large number of **adding or modification** operations are performed on a table.
- After a table is deleted, you are advised to run **VACUUM** rather than **VACUUM FULL**. However, you can run **VACUUM FULL** in some particular cases, such as when you want to physically narrow a table to decrease the occupied disk space after deleting most rows of the table. For details about the differences between **VACUUM** and **VACUUM FULL**, see **VACUUM and VACUUM FULL**.

Syntax

Perform **ANALYZE** on a table.

```
ANALYZE table_name;
```

Perform **ANALYZE** on all tables (non-foreign tables) in the database.

```
ANALYZE;
```

Perform **VACUUM** on a table.

```
VACUUM table_name;
```

Perform **VACUUM FULL** on a table.

```
VACUUM FULL table_name;
```

For details, see **VACUUM** and **ANALYZE | ANALYSE**.

NOTA

- If the physical space usage does not decrease after you run the **VACUUM FULL** command, check whether there were other active transactions (started before you delete data transactions and not ended before you run **VACUUM FULL**). If yes, run this command again when the transactions have finished.
- In version 8.1.3 or later, **VACUUM/VACUUM FULL** can be invoked on the management plane. For details, see **Intelligent O&M**.

VACUUM and VACUUM FULL

In GaussDB(DWS), the **VACUUM** operation is like a vacuum cleaner used to absorb dust. Here, "dust" means old data. If the data is not cleared in a timely manner, the database space will bloat, causing performance deterioration or even system breakdown.

Purposes of VACUUM:

- Solve space bloat: Clear obsolete tuples and corresponding indexes, which include the tuple (and index) of a committed **DELETE** transaction, the old version (and index) of an **UPDATE** transaction, the inserted tuple (and index) of a rolled back **INSERT** transaction, the new version (and index) of an **UPDATE** transaction, and the tuple (and index) of a **COPY** transaction.
- VACUUM FREEZE: Prevents system breakdown caused by transaction ID wraparound. It converts transaction IDs smaller than OldestXmin to freeze xids, update relfrozenxids in a table, and update relfrozenxids and truncate clogs in a database.
- Update statistics: **VACUUM ANALYZE** updates statistics, enabling the optimizer to select a better way to execute SQL statements.

The VACUUM statement includes **VACUUM** and **VACUUM FULL**. Currently, **VACUUM** can only work on row-store tables. **VACUUM FULL** can be used to release space of column-store tables. For details, see the following table.

Tabla 3-1 VACUUM and VACUUM FULL

Item	VACUUM	VACUUM FULL
Clearing space	If the deleted record is at the end of a table, the space occupied by the deleted record is physically released and returned to the operating system. If the data is not at the end of a table, the space occupied by dead tuples in the table or index is set to be available for reuse.	Despite the position of the deleted data, the space occupied by the data is physically released and returned to the operating system. When data is inserted, a new disk page is allocated.
Lock type	Shared lock. The **VACUUM** operation can be performed in parallel with other operations.	Exclusive lock. All operations based on the table are suspended during execution.
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.

3.5 ¿Necesito ajustar una clave de distribución después de establecer una clave principal?

No, solo necesita ajustar la clave principal. De forma predeterminada, la primera columna de la clave principal se selecciona como clave de distribución. Si ambos están definidos, la clave principal debe contener la clave de distribución.

3.6 ¿Es GaussDB(DWS) compatible con los procedimientos almacenados de PostgreSQL?

Sí.

GaussDB(DWS) es compatible con los procedimientos almacenados de PostgreSQL. Para obtener más información, consulte **Procedimientos almacenados**.

3.7 ¿Qué son las tablas particionadas, las particiones y las claves de partición?

Tabla particionada: La partición se refiere a dividir lo que lógicamente es una tabla grande en piezas físicas más pequeñas basadas en esquemas específicos. La tabla basada en la lógica se llama una tabla particionada, y una pieza física se llama una partición. Los datos se almacenan en estas piezas físicas más pequeñas, a saber, particiones, en lugar de la tabla particionada lógica más grande.

Partición: En el sistema distribuido GaussDB(DWS), la partición de datos es para dividir horizontalmente los datos de la tabla dentro de un nodo basado en una política especificada. La tabla se divide en particiones que no se superponen dentro de un rango específico.

Clave de partición: Una clave de partición es un conjunto ordenado de una o más columnas de tabla. Los valores de las claves de partición de tabla se utilizan para determinar la partición de datos a la que pertenece una fila.

3.8 ¿Cómo puedo exportar la estructura de la tabla?

Se recomienda utilizar el cliente gráfico de Data Studio para exportar datos de tabla. Puede exportar datos desde:

- Una tabla específica
- Todas las tablas de un esquema
- Todas las tablas de una base de datos

Para obtener más información, consulte **Exportación de datos de tabla**.

Alternativamente, utilice **gs_dump** y **gs_dumpall** para exportar datos. Puedes hacer lo siguiente:

- Exportar una única base de datos:
  - Exportación a nivel de base de datos
  - Exportación a nivel de modo
  - Exportación a nivel de tabla
- Exportar todas las bases de datos:
  - Exportación a nivel de base de datos
  - Exportación de objetos globales de cada base de datos

Para obtener más información, consulte **gs_dump** y **gs_dumpall**.

3.9 How Can I Delete Table Data Efficiently?

Yes. **TRUNCATE** is more efficient than **DELETE** for deleting massive data.

For details, see **TRUNCATE**.

Function

TRUNCATE quickly removes all rows from a database table.

It has the same effect as the unconditional **DELETE**, but **TRUNCATE** is faster, especially for large tables, because it does not scan tables.

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 **p_15** partition.

```
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;
```

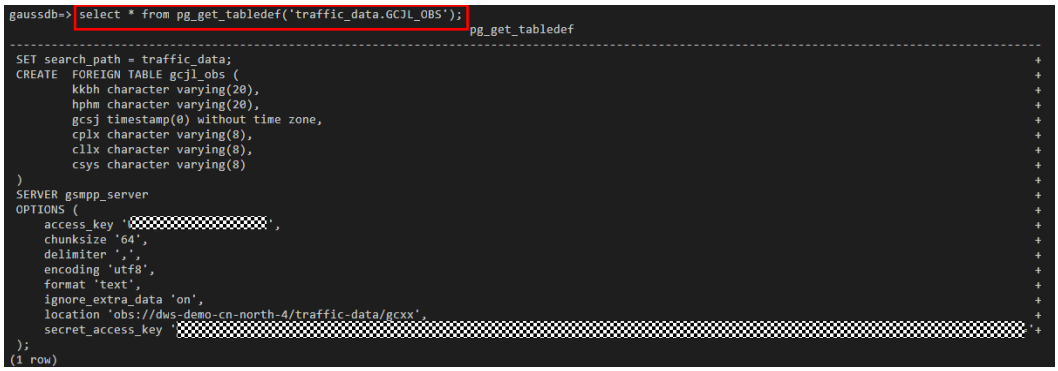
3.10 How Do I View Foreign Table Information?

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');
```



3.11 If No Distribution Column Is Specified, How Will Data Be Stored?

NOTA

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)
```

3.12 How Do I Replace the Null Result with 0?

When OUTER JOIN (LEFT JOIN, RIGHT JOIN, and FULL JOIN) is executed, the match failure in the outer join generates a large number of NULL values. You can replace these null values with 0.

You can use the **COALESCE** function to do that. This function returns the first non-null parameter value in the parameter list. For example:

```
SELECT coalesce(NULL,'hello');
 coalesce
----------
 hello
(1 row)
```

Use left join to join the tables **course1** and **course2**.

```
SELECT * FROM course1;
  stu_id  |  stu_name  |     cour_name
----------+------------+--------------------
 20110103 | ALLEN      | Math
 20110102 | JACK       | Programming Design
 20110101 | MAX        | Science
```



```
(3 rows)

SELECT * FROM course2;
 cour_id |     cour_name      | teacher_name
---------+--------------------+--------------
    1002 | Programming Design | Mark
    1001 | Science            | Anne
(2 rows)

SELECT course1.stu_name,course2.cour_id,course2.cour_name,course2.teacher_name
FROM course1 LEFT JOIN course2 ON course1.cour_name = course2.cour_name ORDER BY
1;
  stu_name   | cour_id |     cour_name      | teacher_name
-------------+---------+--------------------+--------------
 ALLEN       |         |                    |
 JACK        |    1002 | Programming Design | Mark
 MAX         |    1001 | Science            | Anne
(3 rows)
```

Use the **COALESCE** function to replace null values in the query result with 0 or other non-zero values:

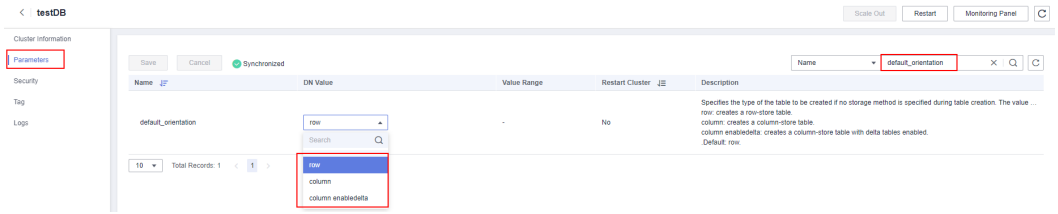
```
SELECT course1.stu_name,
 coalesce(course2.cour_id,0) AS cour_id,
 coalesce(course2.cour_name,'NA') AS cour_name,
 coalesce(course2.teacher_name,'NA') AS teacher_name
 FROM course1
 LEFT JOIN course2 ON course1.cour_name = course2.cour_name
 ORDER BY 1;
  stu_name   | cour_id |     cour_name      | teacher_name
-------------+---------+--------------------+--------------
 ALLEN       |       0 | NA                 | NA
 JACK        |    1002 | Programming Design | Mark
 MAX         |    1001 | Science            | Anne
(3 rows)
```

3.13 How Do I Check Whether a Table Is Row-Stored or Column-Stored?

The storage mode of a table is controlled by the ORIENTATION parameter in the table creation statement. **row** indicates row storage, and **column** indicates column storage.

In 8.1.2 and earlier versions, if **ORIENTATION** is not specified, row storage is used by default.

In versions later than 8.1.3, the **default_orientation** parameter can be used to control the storage mode. If the parameter **ORIENTATION** is not specified during table creation, the table storage mode is based on the value of **default_orientation**. **row** indicates a row-store table, **column** indicates a column-store table, and **column enabledelta** indicates that a column-store table with delta enabled is created. The GUC parameter can be configured on the GaussDB(DWS) console, as shown in the following figure.



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)
```

3.14 How Do I Query the Information About GaussDB(DWS) Column-Store Tables?

The following SQL statements are used to query common information about column-store tables:

Create a column-store partitioned table **my_table**.

```
CREATE TABLE my_table
(
    product_id INT,
    product_name VARCHAR2(40),
    product_quantity INT
 )
WITH (ORIENTATION = COLUMN)
PARTITION BY range(product_quantity)
(
partition my_table_p1 values less than(600),
partition my_table_p2 values less than(800),
partition my_table_p3 values less than(950),
partition my_table_p4 values less than(1000));
INSERT INTO my_table VALUES(1011, 'tents', 720);
INSERT INTO my_table VALUES(1012, 'hammock', 890);
INSERT INTO my_table VALUES(1013, 'compass', 210);
INSERT INTO my_table VALUES(1014, 'telescope', 490);
INSERT INTO my_table VALUES(1015, 'flashlight', 990);
INSERT INTO my_table VALUES(1016, 'ropes', 890);
```

Run the following command to view the created column-store partitioned table:

```
SELECT * FROM my_table;
 product_id | product_name | product_quantity
------------+--------------+------------------
       1013 | compass      |              210
```



```
      1014 | telescope    |              490
      1011 | tents        |              720
      1015 | flashlight   |              990
      1012 | hammock      |              890
      1016 | ropes        |              890
(6 rows)
```

Querying the Boundary of a Partition

```
SELECT relname, partstrategy, boundaries FROM pg_partition where parentid=(select
parentid from pg_partition where relname='my_table');
   relname   | partstrategy | boundaries
-------------+--------------+------------
 my_table    | r            |
 my_table_p1 | r            | {600}
 my_table_p2 | r            | {800}
 my_table_p3 | r            | {950}
 my_table_p4 | r            | {1000}
(5 rows)
```

Querying the Number of Columns in a Column-Store Table

```
SELECT count(*) FROM ALL_TAB_COLUMNS where table_name='my_table';
 count
-------
     3
(1 row)
```

Querying Data Distribution on DNs

```
SELECT table_skewness('my_table');
           table_skewness
-------------------------------------
 ("dn_6007_6008        ",3,50.000%)
 ("dn_6009_6010        ",2,33.333%)
 ("dn_6003_6004        ",1,16.667%)
 ("dn_6001_6002        ",0,0.000%)
 ("dn_6005_6006        ",0,0.000%)
 ("dn_6011_6012        ",0,0.000%)
(6 rows)
```

Querying the Names of the Cudesc and Delta Tables in Partition P1 on a DN

```
EXECUTE DIRECT ON (dn_6003_6004) 'select a.relname from pg_class a, pg_partition
b where (a.oid=b.reldeltarelid or a.oid=b.relcudescrelid) and
b.relname=''my_table_p1''';
       relname
----------------------
 pg_delta_part_60317
 pg_cudesc_part_60317
(2 rows)
```

3.15 Why Sometimes the GaussDB(DWS) Query Indexes Become Invalid?

Creating indexes for tables can improve database query performance. However, sometimes indexes cannot be used in a query plan. This section describes several common reasons and optimization methods.

Reason 1: The Returned Result Sets Are Large.

The following uses Seq Scan and Index Scan on a row-store table as an example:

- Seq Scan: searches table records in sequence. All records are retrieved during each scan. This is the simplest and most basic table scanning method, and its cost is high.
- Index Scan: searches the index first, find the target location (pointer) in the index, and then retrieve data on the target page.

Index scan is faster than sequence scan in most cases. However, if the obtained result sets account for a large proportion (more than 70%) of all data, Index Scan needs to scan indexes before reading table data. This makes it slower table scan.

Reason 2: ANALYZE Is Not Performed In a Timely Manner.

ANALYZE is used to update table statistics. If **ANALYZE** is not executed on a table or a large amount of data is added to or deleted from a table after **ANALYZE** is executed, the statistics may be inaccurate, which may cause a query to skip the index.

Optimization method: Run the **ANALYZE** statement on the table to update statistics.

Reason 3: Filtering Conditions Contains Functions or Implicit Data Type Conversion

If calculation, function, or implicit data type conversion is contained in filter criteria, indexes may fail to be selected.

For example, when a table is created, indexes are created in columns **a**, **b**, and **c**.

```
create table test(a int, b text, c date);
```

- Perform calculation on the indexed columns.

  The following command output indicates that both **where a = 101** and **where a = 102 - 1** use the index in column a, but **where a + 1 = 102** does not use the index.

```
explain verbose select * from test where a  = 101;
                                                QUERY PLAN
------------------------------------------------------------------------------------
------------------------------
  id |                   operation                    | E-rows | E-distinct |
E-memory | E-width | E-costs
 ----+------------------------------------------------+--------+------------
+----------+---------+---------
   1 | ->  Streaming (type: GATHER)                   |      1 |
|          |      44 | 16.27
   2 |    ->  Index Scan using index_a on public.test |      1 |            |
1MB      |      44 | 8.27

 Predicate Information (identified by plan id)
 ---------------------------------------------
   2 --Index Scan using index_a on public.test
         Index Cond: (test.a = 101)

 Targetlist Information (identified by plan id)
 ----------------------------------------------
   1 --Streaming (type: GATHER)
         Output: a, b, c
         Node/s: dn_6005_6006
   2 --Index Scan using index_a on public.test
         Output: a, b, c
         Distribute Key: a

   ====== Query Summary =====
 ------------------------------
 System available mem: 3358720KB
 Query Max mem: 3358720KB
```



```
 Query estimated mem: 1024KB
(24 rows)
 explain verbose select * from test where a  = 102 - 1;
                                                 QUERY PLAN
-------------------------------------------------------------------------------
------------------------------
  id |                   operation                   | E-rows | E-distinct |
E-memory | E-width | E-costs
 ----+-----------------------------------------------+--------+------------
+----------+---------+---------
   1 | ->  Streaming (type: GATHER)                  |      1 |
|          |      44 | 16.27
   2 |    ->  Index Scan using index_a on public.test |      1 |            |
1MB      |      44 | 8.27

 Predicate Information (identified by plan id)
 ---------------------------------------------
   2 --Index Scan using index_a on public.test
         Index Cond: (test.a = 101)

 Targetlist Information (identified by plan id)
 ---------------------------------------------
   1 --Streaming (type: GATHER)
         Output: a, b, c
         Node/s: dn_6005_6006
   2 --Index Scan using index_a on public.test
         Output: a, b, c
         Distribute Key: a

   ====== Query Summary =====
 -------------------------------
 System available mem: 3358720KB
 Query Max mem: 3358720KB
 Query estimated mem: 1024KB
(24 rows)
explain verbose select * from test where a + 1 = 102;
                                           QUERY PLAN
-------------------------------------------------------------------------------
--------------
  id |           operation            | E-rows | E-distinct | E-memory | E-
width | E-costs
 ----+--------------------------------+--------+------------+----------
+---------+---------
   1 | ->  Streaming (type: GATHER)   |      1 |            |          |
44 | 22.21
   2 |    ->  Seq Scan on public.test |      1 |            | 1MB      |
44 | 14.21

 Predicate Information (identified by plan id)
 ---------------------------------------------
   2 --Seq Scan on public.test
         Filter: ((test.a + 1) = 102)

 Targetlist Information (identified by plan id)
 ---------------------------------------------
   1 --Streaming (type: GATHER)
         Output: a, b, c
         Node/s: All datanodes
   2 --Seq Scan on public.test
         Output: a, b, c
         Distribute Key: a

   ====== Query Summary =====
 -------------------------------
 System available mem: 3358720KB
 Query Max mem: 3358720KB
 Query estimated mem: 1024KB
(24 rows)
```


Optimization method: Use constants instead of expressions, or put constant calculation on the right of the equal sign (=).

- Use functions on indexed columns.

  According to the following execution result, if a function is used on an indexed column, the index fails to be selected.

```
explain verbose select * from test where to_char(c, 'yyyyMMdd') =
to_char(CURRENT_DATE,'yyyyMMdd');
                                                                        QUERY PLAN
------------------------------------------------------------------------------------
-------------------------------------------------------------
  id |           operation            | E-rows | E-distinct | E-memory | E-
width | E-costs
 ----+--------------------------------+--------+------------+----------
+---------+---------
   1 | ->  Streaming (type: GATHER)   |      1 |            |          |
44 | 22.28
   2 |    ->  Seq Scan on public.test |      1 |            | 1MB      |
44 | 14.28

                                              Predicate Information
(identified by plan id)

------------------------------------------------------------------------------------
-------------------------------------------------------------
   2 --Seq Scan on public.test
         Filter: (to_char(test.c, 'yyyyMMdd'::text) =
to_char(('2022-11-30'::pg_catalog.date)::timestamp with time zone,
'yyyyMMdd'::text))

 Targetlist Information (identified by plan id)
 -----------------------------------------------
   1 --Streaming (type: GATHER)
         Output: a, b, c
         Node/s: All datanodes
   2 --Seq Scan on public.test
         Output: a, b, c
         Distribute Key: a

   ====== Query Summary =====
 -------------------------------
 System available mem: 3358720KB
 Query Max mem: 3358720KB
 Query estimated mem: 1024KB
(24 rows)
explain verbose select * from test where c = current_date;
                                                  QUERY PLAN
------------------------------------------------------------------------------------
-------------------------------
  id |                   operation                    | E-rows | E-distinct |
E-memory | E-width | E-costs
 ----+------------------------------------------------+--------+------------
+----------+---------+---------
   1 | ->  Streaming (type: GATHER)                   |      1 |
|          |      44 | 16.27
   2 |    ->  Index Scan using index_c on public.test |      1 |            |
1MB      |      44 | 8.27

         Predicate Information (identified by plan id)
 ------------------------------------------------------------
   2 --Index Scan using index_c on public.test
         Index Cond: (test.c = '2022-11-30'::pg_catalog.date)

 Targetlist Information (identified by plan id)
 -----------------------------------------------
   1 --Streaming (type: GATHER)
         Output: a, b, c
         Node/s: All datanodes
```



```
   2 --Index Scan using index_c on public.test
         Output: a, b, c
         Distribute Key: a

   ====== Query Summary =====
 -------------------------------
 System available mem: 3358720KB
 Query Max mem: 3358720KB
 Query estimated mem: 1024KB
(24 rows)
```

Optimization method: Do not use unnecessary functions on indexed columns.

- Implicit conversion of data types.

  This scenario is common. For example, the type of column **b** is Text, and the filtering condition is **where b = 2**. During plan generation, the Text type is implicitly converted to the Bigint type, and the actual filtering condition changes to **where b::bigint = 2**. As a result, the index in column **b** becomes invalid.

```
explain verbose select * from test where b = 2;
                                        QUERY PLAN
------------------------------------------------------------------------------
--------------
  id |           operation            | E-rows | E-distinct | E-memory | E-
width | E-costs
 ----+--------------------------------+--------+------------+----------
+---------+---------
   1 | ->  Streaming (type: GATHER)   |      1 |            |          |
44 | 22.21
   2 |    ->  Seq Scan on public.test |      1 |            | 1MB      |
44 | 14.21

 Predicate Information (identified by plan id)
 -----------------------------------------
   2 --Seq Scan on public.test
         Filter: ((test.b)::bigint = 2)

 Targetlist Information (identified by plan id)
 -----------------------------------------
   1 --Streaming (type: GATHER)
         Output: a, b, c
         Node/s: All datanodes
   2 --Seq Scan on public.test
         Output: a, b, c
         Distribute Key: a

   ====== Query Summary =====
 -------------------------------
 System available mem: 3358720KB
 Query Max mem: 3358720KB
 Query estimated mem: 1024KB
(24 rows)
explain verbose select * from test where b = '2';
                                                QUERY PLAN
------------------------------------------------------------------------------
------------------------------
  id |                   operation                   | E-rows | E-distinct |
E-memory | E-width | E-costs
 ----+-----------------------------------------------+--------+------------
+----------+---------+---------
   1 | ->  Streaming (type: GATHER)                  |      1 |
|          |      44 | 16.27
   2 |    ->  Index Scan using index_b on public.test |      1 |            |
1MB      |      44 | 8.27

 Predicate Information (identified by plan id)
 -----------------------------------------
   2 --Index Scan using index_b on public.test
         Index Cond: (test.b = '2'::text)
```



```
Targetlist Information (identified by plan id)
--------------------------------------------
  1 --Streaming (type: GATHER)
        Output: a, b, c
        Node/s: All datanodes
  2 --Index Scan using index_b on public.test
        Output: a, b, c
        Distribute Key: a

  ====== Query Summary =====
------------------------------
System available mem: 3358720KB
Query Max mem: 3358720KB
Query estimated mem: 1024KB
(24 rows)
```

Optimization method: Use constants of the same type as the indexed column to avoid implicit type conversion.

Scenario 4: Hashjoin Is Replaced with Nestloop + Indexscan.

When two tables are joined, the number of rows in the result set filtered by the WHERE condition in one table is small, thus the number of rows in the final result set is also small. In this case, the effect of nestloop+indexscan is better than that of hashjoin. The better execution plan is as follows:

You can see that the Index Cond: (t1.b = t2.b) at layer 5 has pushed the join condition down to the base table scanning.

```
explain verbose select t1.a,t1.b from t1,t2 where t1.b=t2.b and t2.a=4;
 id |                    operation                       | E-rows | E-distinct | E-
memory | E-width | E-costs
----+----------------------------------------------------+--------+------------
+----------+---------+---------
  1 | ->  Streaming (type: GATHER)                       |     26 |
|          |       8 | 17.97
  2 |    ->  Nested Loop (3,5)                           |     26 |            |
1MB      |       8 | 11.97
  3 |       ->  Streaming(type: BROADCAST)               |      2 |            |
2MB      |       4 | 2.78
  4 |          ->  Seq Scan on public.t2                 |      1 |            |
1MB      |       4 | 2.62
  5 |       ->  Index Scan using t1_b_idx on public.t1 |     26 |            |
1MB      |       8 | 9.05
(5 rows)

 Predicate Information (identified by plan id)
-----------------------------------------------
   4 --Seq Scan on public.t2
         Filter: (t2.a = 4)
   5 --Index Scan using t1_b_idx on public.t1
         Index Cond: (t1.b = t2.b)
(4 rows)

 Targetlist Information (identified by plan id)
------------------------------------------------
   1 --Streaming (type: GATHER)
         Output: t1.a, t1.b
         Node/s: All datanodes
   2 --Nested Loop (3,5)
         Output: t1.a, t1.b
   3 --Streaming(type: BROADCAST)
         Output: t2.b
         Spawn on: datanode2
         Consumer Nodes: All datanodes
   4 --Seq Scan on public.t2
```



```
         Output: t2.b
         Distribute Key: t2.a
   5 --Index Scan using t1_b_idx on public.t1
         Output: t1.a, t1.b
         Distribute Key: t1.a
(15 rows)

   ====== Query Summary =====
---------------------------------
 System available mem: 9262694KB
 Query Max mem: 9471590KB
 Query estimated mem: 5144KB
(3 rows)
```

If the optimizer does not select such an execution plan, you can optimize it as follows:

```
set enable_index_nestloop = on;
set enable_hashjoin = off;
set enable_seqscan = off;
```

Reason 5: The Scan Method Is Incorrectly Specified by Hints.

GaussDB(DWS) plan hints can specify three scan method: tablescan, indexscan, and indexonlyscan.

- Table Scan: full table scan, such as Seq Scan of row-store tables and CStore Scan of column-store tables.
- Index Scan: scans indexes and then obtains table records based on the indexes.
- Index-Only Scan: scans indexes, which cover all required results. Compared with the index scan, the index-only scan covers all queried columns. In this way, only indexes are retrieved, and data records do not need to be retrieved.

In Index-Only Scan scenarios, Index Scan specified by a hint will be invalid.

```
explain verbose select/*+ indexscan(test)*/ b from test where b = '1';
WARNING:  unused hint: IndexScan(test)
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------
-------------------------------
  id |                       operation                       | E-rows | E-distinct
| E-memory | E-width | E-costs
 ----+-------------------------------------------------------+--------+------------
+----------+---------+---------
   1 | ->  Streaming (type: GATHER)                         |      1 |
|          |      32 | 16.27
   2 |    ->  Index Only Scan using index_b on public.test |      1 |
| 1MB      |      32 | 8.27

   Predicate Information (identified by plan id)
 -------------------------------------------------
   2 --Index Only Scan using index_b on public.test
         Index Cond: (test.b = '1'::text)

   Targetlist Information (identified by plan id)
 -------------------------------------------------
   1 --Streaming (type: GATHER)
         Output: b
         Node/s: All datanodes
   2 --Index Only Scan using index_b on public.test
         Output: b
         Distribute Key: a

   ====== Query Summary =====
 -------------------------------
 System available mem: 3358720KB
 Query Max mem: 3358720KB
```



```
 Query estimated mem: 1024KB
(24 rows)
explain verbose select/*+ indexonlyscan(test)*/ b from test where b = '1';
                                                 QUERY PLAN
------------------------------------------------------------------------------------------
------------------------------
  id |                       operation                       | E-rows | E-distinct
| E-memory | E-width | E-costs
 ----+-------------------------------------------------------+--------+------------
+----------+---------+---------
   1 | ->  Streaming (type: GATHER)                          |      1 |
|          |      32 | 16.27
   2 |    ->  Index Only Scan using index_b on public.test |      1 |
| 1MB      |      32 | 8.27

   Predicate Information (identified by plan id)
 -----------------------------------------------
   2 --Index Only Scan using index_b on public.test
         Index Cond: (test.b = '1'::text)

   Targetlist Information (identified by plan id)
 -----------------------------------------------
   1 --Streaming (type: GATHER)
         Output: b
         Node/s: All datanodes
   2 --Index Only Scan using index_b on public.test
         Output: b
         Distribute Key: a

   ====== Query Summary =====
 -------------------------------
 System available mem: 3358720KB
 Query Max mem: 3358720KB
 Query estimated mem: 1024KB
(24 rows)
```

Optimization method: Correctly specify Index scan and Index-Only Scan.

Reason 6: Incorrect Use of GIN Index in Full-Text Retrieval

To accelerate text search, you can create a GIN index for full-text search.

```
CREATE INDEX idxb ON test using gin(to_tsvector('english',b));
```

When creating the GIN index, you must use the 2-argument version of to_tsvector. Only when the query also uses the 2-argument version and the arguments are the same as that in the Gin index, the GIN index can be called.

> **NOTA**
>
> 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.

3.16 ¿Cómo uso una función definida por el usuario para reescribir la función CRC32()?

Actualmente, GaussDB(DWS) no tiene una función integrada de **CRC32()**. En su lugar, puede usar la función definida por el usuario de GaussDB(DWS) para reescribir la función **CRC32()**.

- CRC32(expr)
- Descripción: Calcula la redundancia cíclica. El parámetro de entrada **expr** es una cadena. Si el parámetro es NULL, se devuelve NULL. De lo contrario, se devuelve un valor sin signo de 32 bits después del cálculo de redundancia.

Ejemplo de reescritura de la función **CRC32()** usando la sentencia de función definida por el usuario de GaussDB(DWS):

```
CREATE OR REPLACE FUNCTION crc32(text_string text) RETURNS bigint AS $$
DECLARE
    val bigint;
    i int;
    j int;
    byte_length int;
    binary_string bytea;
BEGIN
    IF text_string is null THEN
        RETURN null;
    ELSIF text_string = '' THEN
        RETURN 0;
    END IF;

    i = 0;
    val = 4294967295;
    byte_length = bit_length(text_string) / 8;
    binary_string = decode(replace(text_string, E'\\', E'\\\\'), 'escape');
    LOOP
        val = (val # get_byte(binary_string, i))::bigint;
        i = i + 1;
        j = 0;
        LOOP
            val = ((val >> 1) # (3988292384 * (val & 1)))::bigint;
            j = j + 1;
            IF j >= 8 THEN
                EXIT;
            END IF;
        END LOOP;
        IF i >= byte_length THEN
            EXIT;
        END IF;
    END LOOP;
    RETURN (val # 4294967295);
END
$$ IMMUTABLE LANGUAGE plpgsql;
```

Verifique el resultado de reescritura.

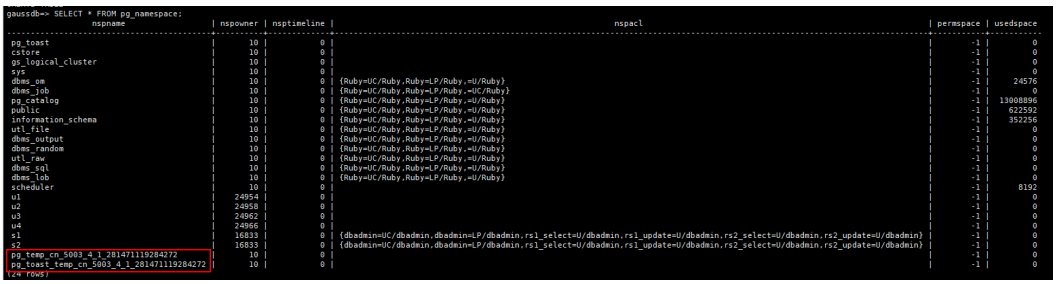

```
select crc32(null),crc32(''),crc32('1');
 crc32 | crc32 |   crc32
-------+-------+------------
       |     0 | 2212294583
(1 row)
```

Para obtener detalles sobre cómo usar funciones definidas por el usuario, consulte la sección **CREATE FUNCTION**.

3.17 ¿Cuáles son los esquemas que comienzan con pg_toast_temp* o pg_temp*?

Cuando consulta la lista de esquemas, el resultado de la consulta puede contener esquemas que comiencen por **pg_temp*** o **pg_toast_temp*** como se muestra en la siguiente figura.

```
SELECT * FROM pg_namespace;
```



Estos esquemas se crean cuando se crean tablas temporales. Cada sesión tiene un esquema independiente que comienza por **pg_temp** para asegurarse de que las tablas temporales solo son visibles para la sesión actual. Por lo tanto, no se recomienda eliminar manualmente los esquemas que comienzan con **pg_temp** o **pg_toast_temp** durante las operaciones rutinarias.

Las tablas temporales solo son visibles en la sesión actual y se eliminan automáticamente después de que finalice la sesión. También se eliminan los esquemas correspondientes.

3.18 Solutions to Inconsistent GaussDB(DWS) Query Results

In GaussDB(DWS), sometimes a SQL query may get different results. This problem is most likely caused by improper syntax or usage. To avoid this problem, use the syntax correctly. The following are some examples of query results inconsistency along with the solutions.

Window Function Results Are Incompletely Sorted

Scenario:

In the window function **row_number()**, column **c** of table **t3** is queried after sorting. The two query results are different.

```
select * from t3 order by 1,2,3;
 a | b | c
---+---+---
 1 | 2 | 1
 1 | 2 | 2
 1 | 2 | 3
(3 rows)
```



```
select c,rn from (select c,row_number() over(order by a,b) as rn from t3) where
rn = 1;
 c | rn
---+----
1  |  1
(1 row)
select c,rn from (select c,row_number() over(order by a,b) as rn from t3) where
rn = 1;
 c | rn
---+----
 3 |  1
(1 row)
```

Analysis:

As shown above, run **select c,rn from (select c,row_number() over(order by a,b) as rn from t3) where rn = 1;** twice, the results are different. That is because duplicate values **1** and **2** exist in the sorting columns **a** and **b** of the window function while their values in column **c** are different. As a result, when the first record is obtained based on the sorting result in columns **a** and **b**, the obtained data in column **c** is random, as a result, the result sets are inconsistent.

Solution:

The values in column **c** need to be added to the sorting.

```
select c,rn from (select c,row_number() over(order by a,b,c) as rn from t3) where
rn = 1;
 c | rn
---+----
 1 |  1
(1 row)
```

Using Sorting in Subviews/Subqueries

Scenario

After table **test** and view **v** are created, the query results are inconsistent when sorting is used to query table **test** in a subquery.

```
CREATE TABLE test(a serial ,b int);
INSERT INTO test(b) VALUES(1);
INSERT INTO test(b) SELECT b FROM test;
...
INSERT INTO test(b) SELECT b FROM test;
CREATE VIEW v as SELECT * FROM test ORDER BY a;
```

Problem SQL:

```
select * from v limit 1;
 a | b
---+---
 3 | 1
(1 row)

select * from (select * from test order by a) limit 10;
 a  | b
----+---
 14 | 1
(1 row)

select * from test order by a limit 10;
 a | b
---+---
 1 | 1
(1 row)
```

Analysis:

ORDER BY is invalid for subviews and subqueries.

Solution:

You are not advised to use **ORDER BY** in subviews and subqueries. To ensure that the results are in order, use **ORDER BY** in the outermost query.

LIMIT in Subqueries

Scenario: When **LIMIT** is used in a subquery, the two query results are inconsistent.

```
select * from (select a from test limit 1 ) order by 1;
 a
---
 5
(1 row)

select * from (select a from test limit 1 ) order by 1;
 a
---
 1
(1 row)
```

Analysis:

The LIMIT in the subquery causes random results to be obtained.

Solution:

To ensure the stability of the final query result, do not use **LIMIT** in subqueries.

Using String_agg

Scenario: When **string_agg** is used to query the table **employee**, the query results are inconsistent.

```
select * from employee;
 empno | ename  |   job   | mgr  |      hiredate       |  sal  | comm | deptno
-------+--------+---------+------+---------------------+-------+------+--------
  7654 | MARTIN | SALEMAN | 7698 | 2022-11-08 00:00:00 | 12000 | 1400 |     30
  7566 | JONES  | MANAGER | 7839 | 2022-11-08 00:00:00 | 32000 |    0 |     20
  7499 | ALLEN  | SALEMAN | 7698 | 2022-11-08 00:00:00 | 16000 |  300 |     30
(3 rows)

select count(*) from (select deptno, string_agg(ename, ',') from employee group
by deptno) t1, (select deptno, string_agg(ename, ',') from employee group by
deptno) t2 where t1.string_agg = t2.string_agg;
 count
-------
     2
(1 row)

select count(*) from (select deptno, string_agg(ename, ',') from employee group
by deptno) t1, (select deptno, string_agg(ename, ',') from employee group by
deptno) t2 where t1.string_agg = t2.string_agg;
 count
-------
     1
(1 row)
```

Analysis:

The **string_agg** function is used to concatenate data in a group into one row. However, if you use **string_agg(ename, ',')**, the order of concatenated results needs to be specified. For

example, in the preceding statement, **select deptno, string_agg(ename, ',') from employee group by deptno;**

can output either of the following:

```
30 | ALLEN,MARTIN
```

Or:

```
30 |MARTIN,ALLEN
```

In the preceding scenario, the result of subquery **t1** may be different from that of subquery **t2** when deptno is **30**.

Solution:

Add **ORDER BY** to **String_agg** to ensure that data is concatenated in sequence.

```
select count(*) from (select deptno, string_agg(ename, ',' order by ename desc)
from employee group by deptno) t1 ,(select deptno, string_agg(ename, ',' order by
ename desc) from employee group by deptno) t2 where t1.string_agg = t2.string_agg;
```

Database Compatibility Mode

Scenario: The query results of empty strings in the database are inconsistent.

database1 (TD-compatible):

```
td=# select '' is null;
 isnull
--------
 f
(1 row)
```

database2 (ORA compatible):

```
ora=# select '' is null;
 isnull
--------
 t
(1 row)
```

Analysis:

The empty string query results are different because the syntax of the empty string is different from that of the null string in different database compatibility.

Currently, GaussDB(DWS) supports three types of database compatibility: Oracle, TD, and MySQL. The syntax and behavior vary depending on the compatibility. For details about the compatibility differences, see **Syntax Compatibility Differences Among Oracle, Teradata, and MySQL**

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 vary according to the database compatibility configuration item **behavior_compat_options**. For details about the parameter options, see **behavior_compat_options**.

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.

3.19 Which System Catalogs That the VACUUM FULL Operation Cannot Be Performed on?

VACUUM FULL can be performed on all GaussDB(DWS) system catalogs. However, during the process, level 8 locks will be imposed on the system catalogs, and services involving these system catalogs will be blocked.

The suggestions are based on database versions:

8.1.3 and Later Versions

- For clusters of version 8.1.3 or later, **AUTO VACUUM** is enabled by default (controlled by the **autovacuum** parameter). After you set the parameter, the system automatically performs **VACUUM FULL** on all system catalogs and row-store tables.
  - If the value of **autovacuum_max_workers** is **0**, neither on the system catalogs nor on ordinary tables will **VACUUM FULL** be automatically performed.
  - If **autovacuum** is set to **off**, **VACUUM FULL** will be automatically performed on ordinary tables, but not system catalogs.
- This applies only to row-store tables. To automatically trigger **VACUUM** for column-store tables, you need to configure intelligent scheduling tasks on the management console. For details, see **O&M plan**.

8.1.1 and Earlier Versions

1. Reforming **VACUUM FULL** on the following system catalogs affects all services. Perform this operation in an idle time window or when services are stopped.
   - **pg_statistic** (Statistics information. You are advised not to clear it because it affects service query performance.)
   - pg_attribute
   - pgxc_class
   - pg_type
   - pg_depend
   - pg_class
   - pg_index
   - pg_proc
   - pg_partition
   - pg_object
   - pg_shdepend
2. The following system catalogs affect resource monitoring and table size query interfaces, but do not affect other services.
   - gs_wlm_user_resource_history
   - gs_wlm_session_info
   - gs_wlm_instance_history
   - gs_respool_resource_history
   - pg_relfilenode_size
3. Other system catalogs do not occupy space and do not need to be cleared.
4. During routine O&M, you are advised to monitor the sizes of these system catalogs, and collect statistics every week. If the space must be reclaimed, clear the space based on the sizes of the system tables.

   The statement is as follows:

   ```
   SELECT c.oid,c.relname, c.relkind, pg_relation_size(c.oid) AS size  FROM
   pg_class c WHERE c.relkind IN ('r') AND c.oid <16385 ORDER BY size DESC;
   ```


3.20 In Which Scenarios Would a Statement Be "idle in transaction"?

When user SQL information is queried in the **PGXC_STAT_ACTIVITY** view, the **state** column in the query result sometimes displays **idle in transaction**. **idle in transaction** indicates that the backend is in a transaction, but no statement is being executed. This status indicates that a statement has been executed. Therefore, the value of query_id is 0, but the transaction has not been committed or rolled back. Statements in this state have been executed and do not occupy CPU and I/O resources, but they occupy connection resources such as connections and concurrent connections.

If a statement is in the **idle in transaction** state, rectify the fault by referring to the following common scenarios and solutions:

Scenario 1: A Transaction Is Started But Not Committed, and the Statement Is in the "idle in transaction" State

BEGIN/START TRANSACTION is manually executed to start a transaction. After statements are executed, **COMMIT/ROLLBACK** is not executed. View the **PGXC_STAT_ACTIVITY**:

```
SELECT state, query, query_id FROM pgxc_stat_activity;
```

The result shows that the statement is in the **idle in transaction** state.

```
       state        |                          query                          |      query_id
--------------------+---------------------------------------------------------+--------------------
 active             |                                                         |                  0
 idle               |                                                         |                  0
 idle               |                                                         |                  0
 active             | WLM fetch collect info from data nodes                  |  73464968921613282
 active             | WLM calculate space info process                        |                  0
 active             | WLM monitor update and verify local info                |  73464968921613276
 active             | WLM arbiter sync info by CCN and CNs                    |                  0
 idle in transaction | select count(1) from t group by a order by 1 desc limit 1; |                  0
 idle               |                                                         |                  0
 active             | select state,query,query_id from pgxc_stat_activity;    |  73464968921613283
 active             |                                                         |                  0
 idle               |                                                         |                  0
 idle               |                                                         |                  0
 active             | WLM fetch collect info from data nodes                  | 145522562959541153
 active             | WLM calculate space info process                        |                  0
 active             | WLM monitor update and verify local info                | 145522562959541123
 active             | WLM arbiter sync info by CCN and CNs                    |                  0
 active             | SELECT * FROM pg_stat_activity                          |  73464968921613283
 idle               |                                                         |                  0
(19 rows)
```

Solution: Manually execute **COMMIT/ROLLBACK** on the started transaction.

Scenario 2: After a DDL Statement in a Stored Procedure Is Executed, Other Nodes of the Stored Procedure Is In the "idle in transaction" State

Create a stored procedure:

```
CREATE OR REPLACE FUNCTION public.test_sleep()
RETURNS void
LANGUAGE plpgsql
AS $$

BEGIN
```



```
    truncate t1;
    truncate t2;
    EXECUTE IMMEDIATE 'select pg_sleep(6)';
    RETURN;
END$$;
```

View the **PGXC_STAT_ACTIVITY** view:

```
SELECT coorname,pid,query_id,state,query,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:

SAVEPOINT and **RELEASE** statements are automatically generated by the system when a stored procedure with **EXCEPTION** is executed. In versions later than 8.1.0, **SAVEPOINT** is not delivered to CNs. GaussDB(DWS) stored procedures with **EXCEPTION** are implemented based on subtransactions, the mapping is as follows:

```
begin
    (Savepoint s1)
    DDL/DML
exception
    (Rollback to s1)
    (Release s1)
...
end
```

If there is **EXCEPTION** in a stored procedure when it is started, a subtransaction will be started. If there is 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 **RELEAS** messages, the stored procedure triggers multiple exceptions. In this case, you have to analyze whether the logic of the stored procedure is proper.

3.21 How Does GaussDB(DWS) Implement Row-to-Column and Column-to-Row Conversion?

This section describes how to use SQL statements to convert rows to columns and convert columns to rows in GaussDB(DWS).

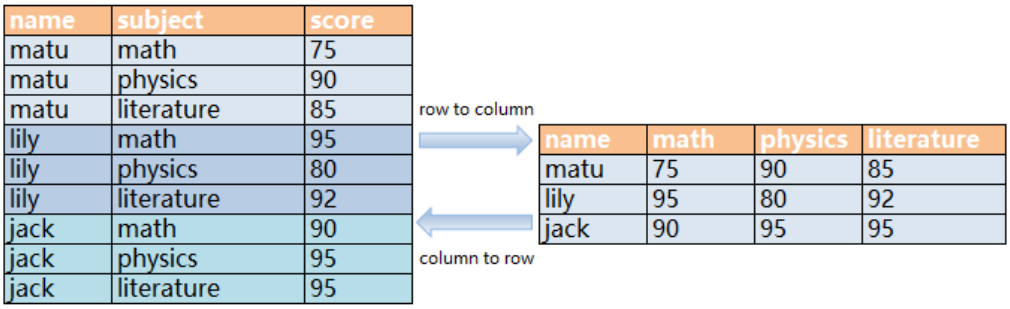
Scenario

Use a student score table as an example:

Teachers record the score of each subject of each student in a table, but students care only 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 sores 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.

Figura 3-1 Diagram



- Rows-to-column conversion

  Convert multiple rows of data into one row, or convert one column of data into multiple columns.

- Column-to-row conversion

  Convert a row of data into multiple rows, or convert multiple columns of data into one column.

Example

- Create a row-store table **students_info**, and insert data into the table.

```
CREATE TABLE students_info(name varchar(20),subject varchar(100),score
bigint) distribute by hash(name);
INSERT INTO students_info VALUES('lily','math',95);
INSERT INTO students_info VALUES('lily','physics',80);
INSERT INTO students_info VALUES('lily','literature',92);
INSERT INTO students_info VALUES('matu','math',75);
INSERT INTO students_info VALUES('matu','physics',90);
INSERT INTO students_info VALUES('matu','literature',85);
```



```
INSERT INTO students_info VALUES('jack','math',90);
INSERT INTO students_info VALUES('jack','physics',95);
INSERT INTO students_info VALUES('jack','literature',95);
```

View information about the **students_info** table.

```
SELECT * FROM students_info;
 name |  subject   | score
------+------------+-------
 matu | math       |    75
 matu | physics    |    90
 matu | literature |    85
 lily | math       |    95
 lily | physics    |    80
 lily | literature |    92
 jack | math       |    90
 jack | physics    |    95
 jack | literature |    95
```

- Create a column-store table **students_info1**, and insert data into the table.

```
CREATE TABLE students_info1(name varchar(20), math bigint, physics bigint,
literature bigint) with (orientation = column) distribute by hash(name);
INSERT INTO students_info1 VALUES('lily',95,80,92);
INSERT INTO students_info1 VALUES('matu',75,90,85);
INSERT INTO students_info1 VALUES('jack',90,95,95);
```

View information about table **students_info1**.

```
SELECT * FROM students_info1;
 name | math | physics | literature
------+------+---------+------------
 matu |   75 |      90 |         85
 lily |   95 |      80 |         92
 jack |   90 |      95 |         95
(3 rows)
```

Static row-to-column conversion

Static row-to-column conversion requires you to manually specify the column names using the given values. If no value is given to a column, the default value **0** is assigned to the column.

```
SELECT name,
sum(case when subject='math' then score else 0 end) as math,
sum(case when subject='physics' then score else 0 end) as physics,
sum(case when subject='literature' then score else 0 end) as literature FROM
students_info GROUP BY name;
 name | math | physics | literature
------+------+---------+------------
 matu |   75 |      90 |         85
 lily |   95 |      80 |         92
 jack |   90 |      95 |         95
(3 rows)
```

Dynamic row-to-column conversion

For clusters of 8.1.2 or later, you can use **GROUP_CONCAT** to generate column-store statements.

```
SELECT group_concat(concat('sum(IF(subject = ''', subject, ''', score, 0)) AS "',
name, '"'))FROM students_info;
      group_concat

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-------------------------------------------
```



```
-----------------------
 sum(IF(subject = 'literature', score, 0)) AS "jack",sum(IF(subject =
'literature', score, 0)) AS "lily",sum(IF(subject = 'literature', score, 0)) AS
"matu",sum(IF(subject = 'math', score, 0)) AS "jack",sum(IF
(subject = 'math', score, 0)) AS "lily",sum(IF(subject = 'math', score, 0)) AS
"matu",sum(IF(subject = 'physics', score, 0)) AS "jack",sum(IF(subject =
'physics', score, 0)) AS "lily",sum(IF(subject = 'physics
', score, 0)) AS "matu"
(1 row)
```

In 8.1.1 and earlier versions, you can use **LISTAGG** to generate column-store statements.

```
SELECT listagg(concat('sum(case when subject = ''', subject, ''' then score else
0 end) AS "', subject, '"'),',') within GROUP(ORDER BY 1)FROM (select distinct
subject from students_info);

                    listagg

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-------------------------------------------
--
 sum(case when subject = 'literature' then score else 0 end) AS
"literature",sum(case when subject = 'physics' then score else 0 end) AS
"physics",sum(case when subject = 'math' then score else 0 end) AS "math
"
(1 row)
```

Dynamically rebuild the view:

```
CREATE OR REPLACE FUNCTION build_view()
RETURNS VOID
LANGUAGE plpgsql
AS $$ DECLARE
sql text;
rec record;
BEGIN
sql := 'select LISTAGG(
    CONCAT( ''sum(case when subject = ''''''', subject, ''''''' then score else 0
end) AS "''', subject, '''"'' )
    ,'','' ) within group(order by 1) from (select distinct subject from
students_info);';
    EXECUTE sql INTO rec;
    sql := 'drop view if exists get_score';
    EXECUTE sql;
    sql := 'create view get_score as select name, ' || rec.LISTAGG || ' from
students_info group by name';
    EXECUTE sql;
END$$;
```

Rebuild the database:

```
CALL build_view();
```

Query view:

```
SELECT * FROM get_score;
 name | literature | physics | math
------+------------+---------+------
 matu |         85 |      90 |   75
 lily |         92 |      80 |   95
 jack |         95 |      95 |   90
(3 rows)
```

Column-to-Row Conversion

Use **UNION ALL** to merge subjects (math, physics, and literature) into one column. The following is an example:


```
SELECT * FROM
(
SELECT name, 'math' AS subject, math AS score FROM students_info1
union all
SELECT name, 'physics' AS subject, physics AS score FROM students_info1
union all
SELECT name, 'literature' AS subject, literature AS score FROM students_info1
)
order by name;
 name |  subject   | score
------+------------+-------
 jack | math       |    90
 jack | physics    |    95
 jack | literature |    95
 lily | math       |    95
 lily | physics    |    80
 lily | literature |    92
 matu | math       |    75
 matu | physics    |    90
 matu | literature |    85
(9 rows)
```

3.22 What Are the Differences Between Unique Constraints and Unique Indexes?

- The concepts of a unique constraint and a unique index are different.

  A unique constraint specifies that the values in a column or a group of columns are all unique. If **DISTRIBUTE BY REPLICATION** is not specified, the column table that contains only unique values must contain distribution columns.

  A unique index is used to ensure the uniqueness of a field value or the value combination of multiple fields. **CREATE UNIQUE INDEX** creates a unique index.

- The functions of a unique constraint and a unique index are different.

  Constraints are used to ensure data integrity, and indexes are used to facilitate query.

- The usages of a unique constraint and a unique index are different.

  a. Both unique constraints and unique indexes can be used to ensure the uniqueness of column values which can be NULL.

  b. When a unique constraint is created, a unique index with the same name is automatically created. The index cannot be deleted separately. When the constraint is deleted, the index is automatically deleted. A unique constraint uses a unique index to ensure data uniqueness. GaussDB(DWS) row-store tables support unique constraints, but column-store tables do not.

  c. A created unique index is independent and can be deleted separately. Currently in GaussDB(DWS), unique indexes can only be created using B-Tree.

  d. If you want to have both a unique constraint and a unique index on a column, and they can be deleted separately, you can create a unique index and then a unique constraint with the same name.

  e. If a field in a table is to be used as a foreign key of another table, the field must have a unique constraint (or it is a primary key). If the field has only a unique index, an error is reported.

Example: Create a composite index for two columns, which is not required to be a unique index.

```
CREATE TABLE t (n1 number,n2 number);
CREATE INDEX t_idx ON t(n1,n2);
```


You can use the index **t_idx** created in the preceding example 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 (n1) USING INDEX t_idx;
```

3.23 What Are the Differences Between Functions and Stored Procedures?

Functions and stored procedures are two common objects in database management systems. They have similarities and differences in implementing specific functions. Understanding their characteristics and application scenarios is important for properly designing the database structure and improving database performance.

Tabla 3-2 Differences between functions and stored procedures

Function	Stored procedures
Both can be used to implement specific functions. Both functions and stored procedures can encapsulate a series of SQL statements to complete certain specific operations.	
Both can receive input parameters and perform corresponding operations based on the parameters.	
The identifier of a function is **FUNCTION**.	The identifier of the stored procedure is **PROCEDURE**.
A function must return a specific value of the specified numeric type.	A stored procedure can have no return value, one return value, or multiple return values. You can use output parameters to return results or directly use the SELECT statement in a stored procedure to return result sets.
Functions are used to return single values, for example, a number calculation result, a string processing result, or a table.	Stored procedures are used for DML operations, for example, inserting, updating, and deleting data in batches.

- **Creating and Invoking a Function**

  Create the **emp** table and insert data into the table. The table data is as follows:

```
SELECT * FROM emp;
 empno | ename |   job    | mgr  |      hiredate       |   sal   |  comm  |
deptno
-------+-------+----------+------+---------------------+---------+--------
+--------
  7369 | SMITH | CLERK    | 7902 | 1980-12-17 00:00:00 |  800.00 |
|     20
  7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 00:00:00 | 1600.00 | 300.00
|     30
  7566 | JONES | MANAGER  | 7839 | 1981-04-02 00:00:00 | 2975.00 |
|     20
  7521 | WARD  | SALESMAN | 7698 | 1981-02-22 00:00:00 | 1250.00 | 500.00
|     30
(4 rows)
```


Create the **emp_comp** function to accept two numbers as input and return the calculated value.

```
CREATE OR REPLACE FUNCTION emp_comp (
    p_sal           NUMBER,
    p_comm          NUMBER
) RETURN NUMBER
IS
BEGIN
    RETURN (p_sal + NVL(p_comm, 0)) * 24;
END;
/
```

Run the **SELECT** command to invoke the function:

```
SELECT ename "Name", sal "Salary", comm "Commission", emp_comp(sal, comm)
"Total Compensation" FROM emp;
 Name  | Salary  | Commission | Total Compensation
-------+---------+------------+--------------------
 SMITH |  800.00 |            |           19200.00
 ALLEN | 1600.00 |     300.00 |           45600.00
 JONES | 2975.00 |            |           71400.00
 WARD  | 1250.00 |     500.00 |           42000.00
(4 rows)
```

- **Creating and Invoking a Stored Procedure**

  Create the **MATCHES** table and insert data into the table. The table data is as follows:

  ```
  SELECT * FROM MATCHES;
   matchno | teamno | playerno | won | lost
  ---------+--------+----------+-----+------
         1 |      1 |        6 |   3 |    1
         7 |      1 |       57 |   3 |    0
         8 |      1 |        8 |   0 |    3
         9 |      2 |       27 |   3 |    2
        11 |      2 |      112 |   2 |    3
  (5 rows)
  ```

  Create the stored procedure **delete_matches** to delete all matches that a specified player participates in.

  ```
  CREATE PROCEDURE delete_matches(IN p_playerno INTEGER)
  AS
  BEGIN
     DELETE FROM MATCHES WHERE playerno = p_playerno;
  END;
  /
  ```

  Invoke the stored procedure **delete_matches**.

  ```
  CALL delete_matches(57);
  ```

  Query the **MATCHES** table again. The returned result indicates that the data of the player whose **playerno** is **57** has been deleted.

  ```
  SELECT * FROM MATCHES;
   matchno | teamno | playerno | won | lost
  ---------+--------+----------+-----+------
        11 |      2 |      112 |   2 |    3
         8 |      1 |        8 |   0 |    3
         1 |      1 |        6 |   3 |    1
         9 |      2 |       27 |   3 |    2
  (4 rows)
  ```


4 Gestión de clúster

4.1 ¿Qué hago si la creación de un clúster de GaussDB(DWS) ha fallado?

Solución de problemas

Compruebe que tiene suficiente cuota para crear el clúster.

Asistencia técnica

Si no se puede identificar el fallo, envíe un ticket de servicio para informar del problema: Inicie sesión en la consola de gestión y elija **Service Tickets** > **Create Service Ticket**.

4.2 ¿Cómo puedo borrar y recuperar el espacio de almacenamiento?

Después de eliminar los datos almacenados en los almacenes de datos de GaussDB(DWS), es posible que se generen datos sucios debido a que el espacio en disco no se libera. Esto da como resultado un desperdicio de espacio en disco y deteriora el rendimiento de la creación y restauración de instantáneas. A continuación se describe el impacto en el sistema y la operación posterior para borrar el espacio en disco:

Puntos que vale la pena mencionar durante la limpieza y recuperación de espacio de almacenamiento:

- Los datos innecesarios deben eliminarse para liberar el espacio de almacenamiento.
- Las operaciones de lectura y escritura frecuentes pueden afectar al uso adecuado de la base de datos. Por lo tanto, es una buena práctica limpiar y recuperar el espacio de almacenamiento cuando no está en horas pico.
- El tiempo de borrado de datos depende de los datos almacenados en la base de datos.

Realice los siguientes pasos para borrar y recuperar el espacio de almacenamiento:

1. Conéctese a la base de datos. Para obtener más información, consulte **Métodos de conexión a un clúster**.

2. Ejecute el siguiente comando para borrar y recuperar el espacio de almacenamiento:

   VACUUM FULL;

   De forma predeterminada, se eliminan las tablas en las que el usuario actual tiene el permiso. Otras tablas son omitidas.

   Se muestra la siguiente información una vez que se borra el espacio:

   ```
   VACUUM
   ```

   NOTA

   - **VACUUM FULL** recupera todo el espacio de fila caducado, sin embargo, requiere un bloqueo exclusivo en cada tabla que se procesa y puede tardar mucho tiempo en completarse en tablas de base de datos grandes y distribuidas. Se recomienda hacer **VACUUM FULL** a las tablas especificadas. Si desea hacer **VACUUM FULL** a toda la base de datos, se recomienda hacerlo durante el mantenimiento de la base de datos.
   - La información estadística se perderá si utiliza el parámetro **FULL**. Para recopilar las estadísticas, agregue la palabra clave **ANALYZE**, por ejemplo, **VACUUM FULL ANALYZE;**.

     Para obtener más información acerca de **VACUUM**, vea **VACUUM** en la *Referencia de sintaxis de SQL*.

4.3 ¿Puedo cambiar mis nodos de clúster a otra región después de la compra?

No. Los nodos de clúster no se pueden utilizar en todas las regiones.

En su lugar, cancele el orden en la región original, coloque un nuevo orden en la región deseada y cree un clúster.

4.4 ¿Por qué se redujo el almacenamiento usado después de la expansión horizontal?

Causas posibles

Si no ejecuta **VACUUM** para borrar y recuperar el espacio de almacenamiento antes de la expansión horizontal, es posible que los datos eliminados de GaussDB(DWS) no liberen el espacio ocupado en disco.

Durante la expansión horizontal, el sistema redistribuye los datos porque el volumen de datos de servicio en los nodos originales es significativamente mayor que el de los nodos recién añadidos. Cuando se inicia la redistribución, el sistema realiza automáticamente **VACUUM** para liberar el espacio de almacenamiento. Esto provoca una gran caída en la capacidad.

Procedimiento de manejo

Se recomienda borrar y recuperar periódicamente el espacio de almacenamiento ejecutando **VACUUM FULL** para evitar la expansión de los datos.

Si el espacio de almacenamiento utilizado sigue siendo grande después de ejecutar **VACUUM FULL**, analice si la variante de clúster existente cumple con los requisitos de servicio. Si no, ampliar el clúster.

4.5 ¿Cómo puedo ver las métricas de nodo (CPU, memoria y uso de disco)?

Puede ver la capacidad utilizada de la CPU, la memoria y los discos de un clúster en la consola de gestión de Cloud Eye. Realice los siguientes pasos para ver la información:

Paso 1 Inicie sesión en la consola de GaussDB(DWS) y haga clic en **Viewing Metric** junto a un clúster.

Paso 2 Haga clic en  para volver a la página **Cloud Service Monitoring** y cambie a la página **Data Warehouse Node** y haga clic en **View Metric** a la derecha del nodo correspondiente para ver su uso del disco.

----Fin

4.6 ¿GaussDB(DWS) soporta el nodo único para un entorno de aprendizaje?

Sí. En GaussDB(DWS), puede crear un clúster de almacén de datos híbrido en modo independiente. Si el nombre de la variante de nodo seleccionado contiene **h1** (por ejemplo, **dwsx2.h1.xlarge.2.c6**), el almacén de datos híbrido solo admite el despliegue independiente, que no proporciona capacidades de HA. El coste de almacenamiento se puede reducir a la mitad. Un almacén de datos independiente se puede restaurar mediante la reconstrucción automática de ECS, y su fiabilidad de datos está garantizada por el mecanismo multicopia de EVS. Es menos costoso que otras variantes y es una buena opción para servicios ligeros.

4.7 ¿GaussDB(DWS) es compatible con BMS?

Sí. Puede enviar un ticket de servicio para solicitar las variantes de BMS. GaussDB(DWS) utiliza ECS solo en Huawei Cloud por defecto.

4.8 How Is the Disk Space or Capacity of GaussDB(DWS) Calculated?

1. Total disk capacity of GaussDB(DWS): For a three-node cluster, if each node is 320 GB, the total capacity is 960 GB. When 1 GB data is stored, GaussDB(DWS) stores 1 GB data on two nodes due to duplication, a security mechanism, thereby occupying a total of 2 GB space. As a result, more than 2 GB space is occupied if metadata and indexes are calculated. Therefore, a three-node cluster with a total capacity of 960 GB can store 480 GB data. This mechanism ensures data security.

   When you purchase nodes on the console, you are billed by the available capacity of a node. For example, the actual space of **dws.m3.xlarge** is 320 GB and the available space displayed is 160 GB, the space you will be billed for.

2. Check the disk usage of a single node.

   Similarly, if the total capacity is 960 GB and there are three data nodes, the disk capacity of each node is 320 GB.

Log in to the Gauss(DWS) console and choose **Monitoring** > **Node Monitoring** > **Overview** to view the usage of disks and other resources on each node.

NOTA

- The disk space displayed on the **Node Management** page is the total capacity of all disks, that is, system disks and data disks, in the data warehouse cluster. The disk space displayed on the **Overview** page is only the available space for storing table data in the cluster. In addition, tables in the data warehouse cluster have backup copies, these copies also occupy the disk storage.
- If the cluster is read-only and an alarm for insufficient disk space is generated, expand the cluster capacity by following the instructions provided in **Scaling Out a Cluster**.

4.9 ¿Cuáles son las bases de datos gaussdb y postgres de GaussDB(DWS)?

Las bases de datos **gaussdb** y **postgres** son bases de datos integradas de GaussDB(DWS). Puede crear esquemas y tablas en ellos. Sin embargo, se recomienda volver a crear una base de datos y crear esquemas y tablas en la nueva base de datos.

4.10 ¿Cómo configuro el número máximo de sesiones al agregar una regla de alarma en Cloud Eye?

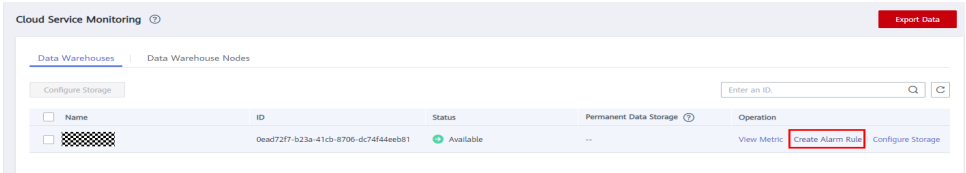
Después de conectarse a una base de datos, ejecute la siguiente instrucción SQL para comprobar el número máximo de sesiones simultáneas globalmente:

```
show max_active_statements;
```

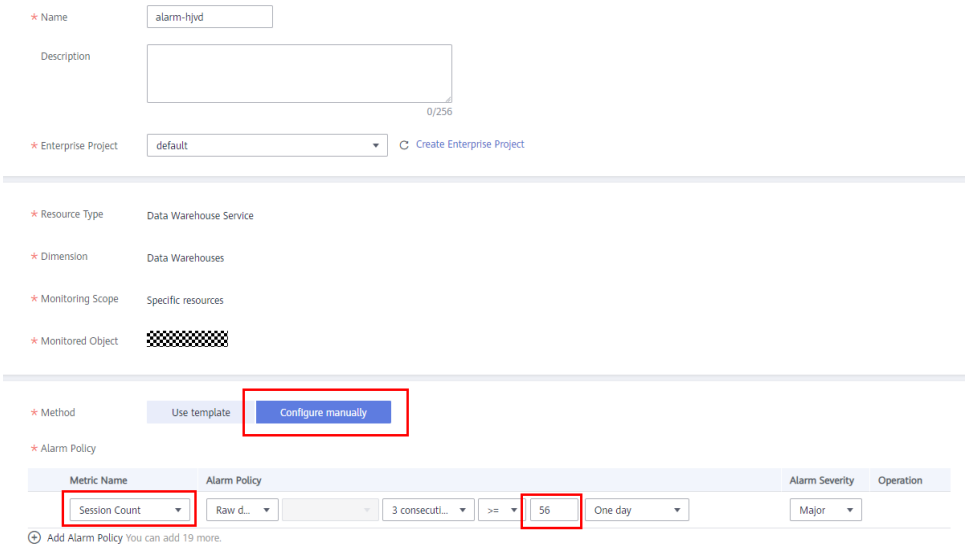
Vaya a la consola de Cloud Eye y establezca el umbral entre el 70% y el 80% del valor obtenido. Por ejemplo, si el valor de **max_active_statements** es **80**, establezca el umbral en **56** (80 x 70%).

Procedimiento:

1. Vaya a la página **Clusters** en la consola de gestión de GaussDB(DWS).
2. Haga clic en **View Metric** en la columna **Operation** del clúster de destino para ir a la consola de Cloud Eye.
3. Haga clic en < en la esquina superior izquierda de la página mostrada y haga clic en **Create Alarm Rule** del clúster de destino.



4. Establezca **Method** en **Configure manually**, **Metric Name** en **Session Count**, **Alarm Policy** en **56**, y **Alarm Severity** en **Major**. A continuación, haga clic en **Create**.



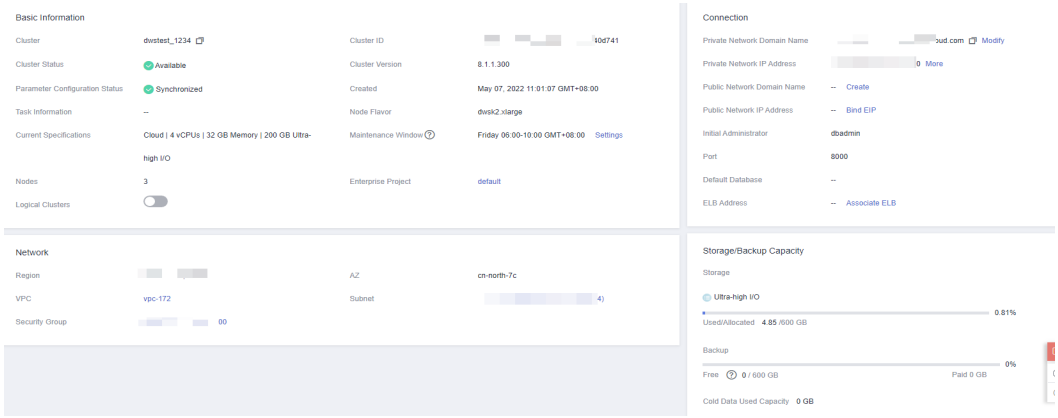
4.11 ¿Cómo puedo determinar si un clúster utiliza la arquitectura x86 o Arm?

Procedimiento

Paso 1 Inicie sesión en la consola de gestión de GaussDB(DWS).

Paso 2 Elija **Clusters**. Todos los clústeres se mostrarán de forma predeterminada.

Paso 3 En la lista de clústeres, haga clic en el nombre de un clúster para ir a la página **Cluster Information**. En el área **Basic Information**, compruebe las especificaciones de nodo del clúster.



Paso 4 Determine la arquitectura del clúster en función de las especificaciones del nodo. En la siguiente tabla se describen las especificaciones.

Tabla 4-1 Descripción de variante

Nodo	Núcleos de vCPU	Memoria (GB)	Arquitectura	Tipo
dws2.olap.4xlarge.m6	16	128	x86	ECS/storage-compute decoupling EVS
dws2.olap.8xlarge.m6	32	256	x86	ECS/storage-compute decoupling EVS
dws2.olap.16xlarge.m6	64	512	x86	ECS/storage-compute decoupling EVS
dws2.olap.4xlarge.kc1	16	64	Arm	ECS/storage-compute decoupling EVS
dws2.olap.4xlarge.km1	16	128	Arm	ECS/storage-compute decoupling EVS
dws2.olap.6xlarge.km1	24	192	Arm	ECS/storage-compute decoupling EVS
dws2.olap.8xlarge.km1	32	256	Arm	ECS/storage-compute decoupling EVS
dws2.olap.12xlarge.km1	48	384	Arm	ECS/storage-compute decoupling EVS
dws2.m6.4xlarge.8	16	128	x86	ECS/EVS
dws2.m6.8xlarge.8	32	256	x86	ECS/EVS
dws2.m6.16xlarge.8	64	512	x86	ECS/EVS
dws2.km1.4xlarge.8	16	128	Arm	ECS/EVS
dws2.km1.6xlarge.8	24	192	Arm	ECS/EVS
dws2.km1.8xlarge.8	32	256	Arm	ECS/EVS
dws2.km1.12xlarge.8	48	384	Arm	ECS/EVS
dws2.km1.xlarge	4	32	Arm	ECS/EVS
dws2.kc1.4xlarge	16	64	Arm	ECS/EVS
dws2.olap.4xlarge.i3	16	128	x86	ECS/Local passthrough
dws2.olap.8xlarge.i3	32	256	x86	ECS/Local passthrough
dws2.olap.16xlarge.i3	64	512	x86	ECS/Local passthrough
dws2.olap.4xlarge.ki1	16	64	Arm	ECS/Local passthrough

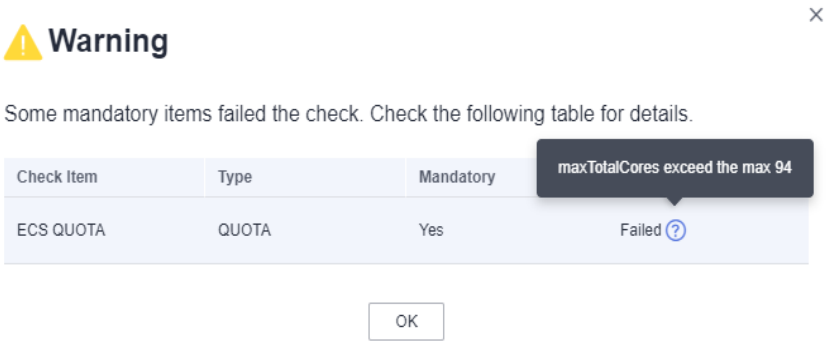
Nodo	Núcleos de vCPU	Memoria (GB)	Arquitectura	Tipo
dws2.olap.8xlarge.ki1	32	128	Arm	ECS/Local passthrough
dws2.olap.16xlarge.ki1	64	228	Arm	ECS/Local passthrough
dws2.physical.ki1ne.4xlarge	128	512	Arm	BMS
dws2.physical.ki1ne.4xlarge.a	128	512	Arm	BMS
dws2.physical.c6sd.6xlarge	104	768	x86	BMS
dws2.physical.c6sd.6xlarge.7	104	768	x86	BMS
dws2.physical.c6sd.6xlarge.7.cbg	104	768	x86	BMS
dws2.physical.c6sd.6xlarge.a.7	104	768	x86	BMS
dws2.physical.io6.3xlarge.4a	104	384	x86	BMS
dws2.physical.c6d.6xlarge.3a.cbg	88	768	x86	BMS
dws2.physical.c6sd.6xlarge.a.7.cbg	104	768	x86	BMS
dws2.physical.c6sd.6xlarge.1.cbg	104	768	x86	BMS

----Fin

4.12 ¿Qué debo hacer si falla la comprobación de expansión horizontal?

Síntoma

Después de hacer clic en **OK** durante la expansión o agregar nodos inactivos, se muestra un mensaje de advertencia y no puede ir al siguiente paso.



Análisis de las causas

Antes de enviar una tarea de expansión, el sistema comprueba elementos como las cuotas de recursos y los permisos de IAM. Los elementos anormales impedirán que se envíen tareas de expansión para evitar fallas de expansión.

Solución

- Si la comprobación de la cuota falla, compruebe si la cuota de recursos es suficiente. Si los nodos disponibles son insuficientes, haga clic en **Increase Quota** para enviar una orden de trabajo para aumentar la cuota de nodos.
- Permisos de IAM: Este problema se produce cuando un usuario solo tiene una cuenta de IAM. En este caso, los permisos como los permisos de VPC y EVC/BMS deben asignarse a la cuenta de IAM. O bien, el usuario puede usar la cuenta principal para realizar la expansión hacia fuera.
- Si no hay elementos anormales pero el cuadro de diálogo sigue existiendo, póngase en contacto con el soporte técnico.

4.13 ¿Cuándo debo agregar CN o ampliar un clúster?

Introducción a la concurrencia de CN

CN es la abreviatura de Coordinator Node. Un CN es un componente importante de GaussDB(DWS) y está estrechamente relacionado con los usuarios. Proporciona interfaces a aplicaciones externas, optimiza los planes de ejecución globales, distribuye los planes de ejecución a DataNodes y resume y procesa los resultados de la ejecución. Un CN es una interfaz para las aplicaciones externas. La capacidad de concurrencia de la CN determina la concurrencia de servicio.

La concurrencia de CN se determina mediante los siguientes parámetros:

- **max_connections**: especifica el número máximo de conexiones simultáneas a la base de datos. Este parámetro afecta a la capacidad de procesamiento simultáneo del clúster. El valor predeterminado depende de las especificaciones del clúster. Para obtener más información, consulte **Gestión de conexiones a bases de datos**.
- **max_active_statements**: especifica el número máximo de trabajos simultáneos. Este parámetro se aplica a todos los trabajos de un CN. El valor predeterminado es **60**, lo que indica que se pueden ejecutar un máximo de 60 trabajos al mismo tiempo. Otros trabajos serán puestos en cola.

¿Agregar los CN o ampliar un clúster?

- Conexiones insuficientes: cuando se crea un clúster por primera vez, el número predeterminado de los CN en el clúster es 3, que puede cumplir con los requisitos básicos de conexión del cliente. Si el clúster tiene un gran número de solicitudes simultáneas y el número de conexiones a cada CN es grande, o el uso de CPU de un CN excede su capacidad, se recomienda agregar CN. Para más detalles, véase **CNs**.
- Capacidad y rendimiento de almacenamiento insuficientes: si su empresa crece y tiene requisitos más altos en cuanto a capacidad y rendimiento de almacenamiento, o si la CPU del clúster es insuficiente, le aconsejamos que amplíe el clúster. Para obtener más información, consulte **Expansión de clúster**.

Con la expansión de los nodos de clúster, se necesitan más CN para cumplir con los requisitos de distribución de GaussDB(DWS). En resumen, la adición de CN no requiere necesariamente expansión de clústeres. Sin embargo, después de la expansión de grupo, es posible que sea necesario agregar los CN.

4.14 ¿Cuáles son los escenarios de cambiar el tamaño de un clúster, cambiar la variante del nodo, ampliar y reducir?

Cambiar el tamaño de un clúster tiene un gran impacto en sus cargas de trabajo. Es similar a migrar un clúster antiguo a uno nuevo, con cambios en los nodos y las especificaciones. Se recomienda realizar operaciones ligeras, como expansión, reducción y cambio de variante. En la siguiente tabla se enumeran los escenarios de aplicación de las opciones de modificación del clúster.

Tabla 4-2 Opciones de modificación de clúster

Opción	Escenario de aplicación	Notas
Expansión horizontal	Si su empresa crece y tiene mayores requisitos de capacidad de almacenamiento y rendimiento, o si la CPU del clúster es insuficiente, se le aconseja que amplíe el clúster.	No se pueden agregar nodos a un almacén de datos híbrido (independiente).
Reducción horizontal	Durante las horas no pico cuando una gran cantidad de capacidad del clúster está inactiva, puede reducir el número de nodos para reducir los costos.	Un almacén de datos híbrido (modo de clúster) no se puede reducir un clúster independiente.

Opción	Escenario de aplicación	Notas
Cambio de la variante del nodo	Esta opción cambia las variantes de clúster (incluida la CPU, la memoria y otros) para cumplir con los requisitos de servicio. No cambia el número de nodos.	Actualmente, solo puede cambiar las variantes de los clústeres de almacén de datos en la nube y los clústeres de almacén de datos en streaming que solo utilizan recursos de ECS y EVS para cómputo y almacenamiento.
Cambio de todas las especificaciones	Puede cambiar el tamaño del clúster cuando: ● Los clústeres son clústeres de BMS o no admiten los cambios de variante. ● Desea cambiar la topología del clúster en lugar de expansión o reducción que simplemente agrega nodos o elimina nodos anillo por anillo. ● El clúster está envejecido y desea cambiarlo a un clúster nuevo sin migrar datos.	Actualmente, solo se admiten clústeres de almacén de datos estándar.

4.15 ¿Cómo debo seleccionar entre un clúster de muchos nodos con variante pequeña y un clúster de tres nodos de variante grande con los mismos núcleos de CPU y memoria?

- De muchos nodos con la variante pequeña:

  Si el volumen de datos es pequeño y necesita escalar nodos, pero tiene un presupuesto limitado, puede seleccionar un clúster de varios nodos con la variante pequeña.

  Por ejemplo, un clúster con la variante pequeña (dwsx2.h.2xlarge.4.c6) con 8 núcleos y 32 GB de memoria puede proporcionar capacidades informáticas sólidas. El clúster tiene un gran número de nodos y puede procesar altas solicitudes simultáneas. En este caso, solo necesita asegurarse de que la velocidad de red entre nodos es normal para evitar la limitación del rendimiento del clúster.
- De tres nodos con la variante grande:

  Si tiene que procesar una gran cantidad de datos, tiene un alto requisito de cómputo y tiene un alto presupuesto, puede seleccionar un clúster de tres nodos con la variante grande.

Por ejemplo, un clúster con la variante grande (dws2.m6.8xlarge.8) con 32 núcleos y 256 GB de memoria tiene una capacidad de procesamiento de CPU más rápida y una memoria más grande, y puede procesar datos más rápidamente. Sin embargo, el clúster tiene nodos limitados, lo que puede causar un bajo rendimiento en escenarios de alta simultaneidad.

4.16 ¿Cuáles son las diferencias entre las SSD en la nube y las SSD locales?

Las SSD en la nube son compatibles con expansión. Por lo tanto, se recomienda usarlas. Las diferencias entre las SSD en la nube y las SSD locales son las siguientes:

- **SSD en la nube**
  - Las SSD en la nube utilizan EVS como medio de almacenamiento y se pueden escalar a medida que crecen los datos. Esto es más flexible.
  - Las SSD en la nube no están vinculadas a la variante de ECS. Por lo tanto, puede ajustar las variantes de las SSD en la nube cuando sea necesario.
- **SSD locales**
  - Las SSD locales utilizan los discos locales de ECS como medio de almacenamiento. Tienen una capacidad fija y un rendimiento más alto, pero no se pueden escalar.
  - Si la capacidad es insuficiente, hay que agregar más nodos para aumentar la capacidad. No se pueden ajustar las variantes de las SSD locales.

4.17 ¿Cuáles son las diferencias entre el almacenamiento de datos en caliente y el almacenamiento de datos en frío?

La mayor diferencia entre el almacenamiento de datos calientes y el almacenamiento de datos fríos radica en los medios de almacenamiento.

- Los datos activos se consultan o actualizan con frecuencia y tienen altos requisitos de tiempo de respuesta de acceso. Se almacena en **DN data disks**.
- Los datos fríos no se actualizan y se consultan ocasionalmente, y no tienen altos requisitos de tiempo de respuesta de acceso. Se almacena en **OBS**.

Los diferentes medios de almacenamiento determinan el costo, el rendimiento y los escenarios de aplicación de los dos modos de almacenamiento, como se muestra en **Tabla 4-3**.

Tabla 4-3 Diferencias entre almacenamiento de datos en caliente y en frío

Almacenamiento	Lectura y escritura	Costo	Capacidad	Escenario de aplicación
Almacenamiento en caliente	Rápida	Alto	Fija y restringida	Este modo es aplicable a escenarios en los que el volumen de datos es limitado y necesita ser leído y actualizado con frecuencia.

Almacenamiento	Lectura y escritura	Costo	Capacidad	Escenario de aplicación
Almacenamiento en frío	Lento	Bajo	Grande e ilimitada	Este modo es aplicable a escenarios como el archivado de datos. Cuenta con costo bajo y capacidad grande.

4.18 What Do I do if the Scale-in Button Is Dimmed?

Symptom

When a user performs a scale-in operation, the **Scale In** button is unavailable and the user cannot proceed to the next scale-in operation.

Possible Causes

The system verifies the cluster's eligibility for scaling in before each operation. The **Scale In** button is dimmed if the cluster does not qualify.

Solution

Check the cluster configuration information and check whether the scale-in meets the following conditions:

- The cluster consists of rings of four or five hosts each, with primary, standby, and secondary 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.
- A yearly/monthly cluster to be scaled in cannot be in the grace period.

5 Conexión de bases de datos

5.1 ¿Cómo se comunican las aplicaciones con GaussDB(DWS)?

Para que las aplicaciones se comuniquen con GaussDB(DWS), asegúrese de que las redes entre ellas estén conectadas. En la siguiente tabla se enumeran los escenarios de conexión más comunes.

Tabla 5-1 Comunicación entre aplicaciones y GaussDB(DWS)

Escenario		Descripción	Tipo de conexión compatible
Nube	**La aplicación de servicio y GaussDB(DWS) están en la misma VPC en la misma región**	Dos direcciones IP privadas en la misma VPC pueden comunicarse directamente entre sí.	● gsql ● Data Studio ● JDBC/ODBC Para obtener más modos de conexión, consulte **Métodos de conexión a un clúster**.
	Aplicaciones de servicio y GaussDB(DWS) están en diferentes VPCs en la misma región	Después de crear una conexión de **VPC peering** entre dos VPC, las dos direcciones IP privadas pueden comunicarse directamente entre sí.	
	Las aplicaciones de servicio y GaussDB(DWS) están en diferentes regiones	Después de establecer una **cloud connection** (CC) entre dos regiones, las dos regiones se comunican entre sí con direcciones IP privadas.	

Escenario		Descripción	Tipo de conexión compatible
In situ y en la nube	**Las aplicaciones de servicio se despliegan en centros de datos locales y necesitan comunicarse con GaussDB(DWS).**	● Utilice el **public IP address or domain name** de GaussDB(DWS) para la comunicación. ● Use **Direct Connect** (DC) para la comunicación.	

La aplicación de servicio y GaussDB(DWS) están en la misma VPC en la misma región

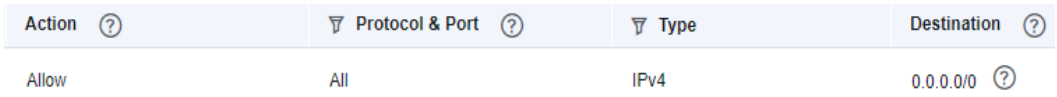
Para garantizar una baja latencia de servicio, se recomienda que desplegar aplicaciones de servicio y GaussDB(DWS) en la misma región. Por ejemplo, si se despliega una aplicación de servicio en un ECS, se recomienda que desplegar el clúster de almacén de datos en la misma VPC que el ECS. De esta manera, la aplicación puede comunicarse directamente con GaussDB(DWS) con una dirección IP de intranet. En este caso, despliegue el clúster de almacén de datos en la misma región y VPC donde reside el ECS.

Por ejemplo, si el ECS se despliega en la **CN-Hong Kong** seleccione **CN-Hong Kong** para el clúster de GaussDB(DWS) y asegúrese de que el clúster de GaussDB(DWS) y el ECS estén ambos en la **VPC1**. La dirección IP privada del ECS es **192.168.120.1** y la dirección IP privada de GaussDB(DWS) es **192.168.120.2**. Por lo tanto, pueden comunicarse entre sí con direcciones IP privadas.

Los puntos clave en la comprobación de comunicación son la regla de salida de ECS y la regla de entrada GaussDB(DWS). El procedimiento de comprobación es el siguiente:

Paso 1 Comprobar las reglas salientes de ECS:

Asegúrese de que la regla de salida del grupo de seguridad ECS permita el acceso. Si no se permite el acceso, consulte la **Configuración de reglas del grupo de seguridad**.



Paso 2 Comprobar las reglas entrantes de GaussDB(DWS):

Si no se configura ningún grupo de seguridad cuando se crea GaussDB(DWS), la regla de entrada predeterminada permite el acceso TCP desde todas las direcciones IPv4 y el puerto 8000. Para garantizar la seguridad, también puede permitir solo una dirección IP. Para obtener más información, consulte **¿Cómo configuro una lista blanca para proteger clústeres disponibles por una dirección IP pública?**

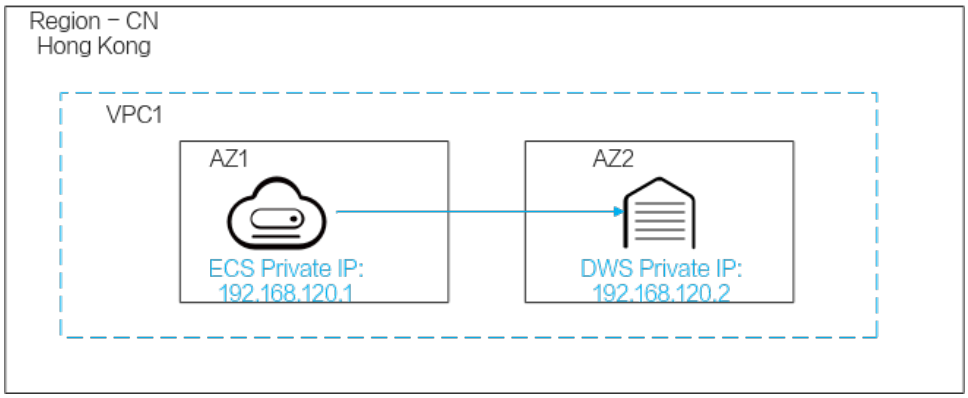
Paso 3 Inicie sesión en el ECS. Si se puede hacer ping la dirección IP interna de GaussDB(DWS), la conexión de red es normal. Si la dirección IP no se puede hacer ping, compruebe la configuración anterior. Si el ECS tiene un firewall, compruebe la configuración del firewall.

----Fin

Ejemplo de uso de gsql para la conexión:

gsql -d gaussdb -h *192.168.120.2* **-p 8000 -U dbadmin -W** *password* **-r**

Figura 5-1 Acceso por las direcciones IP privadas



Aplicaciones de servicio y GaussDB(DWS) están en diferentes VPCs en la misma región

Para garantizar una baja latencia de servicio, se recomienda que desplegar aplicaciones de servicio y GaussDB(DWS) en la misma región. Por ejemplo, si se despliegan las aplicaciones de servicio en un ECS, se recomienda que desplegar el clúster de almacén de datos en la misma VPC que el ECS. Si se selecciona una VPC diferente para el clúster del almacén de datos, el ECS no puede conectarse directamente a GaussDB(DWS).

Por ejemplo, tanto ECS como GaussDB(DWS) se despliegan en **CN-Hong Kong**, pero ECS está en VPC1 y GaussDB(DWS) está en VPC2. En este caso, necesita crear una **Interconexión de VPC** entre VPC1 y VPC2 para que ECS pueda acceder a GaussDB(DWS) utilizando la dirección IP privada de GaussDB(DWS).

Los puntos clave para comprobar la comunicación son las reglas salientes de ECS, las reglas entrantes de GaussDB(DWS) y las interconexiones de VPC. El procedimiento de comprobación es el siguiente:

Paso 1 **Comprobar las reglas salientes de ECS:**

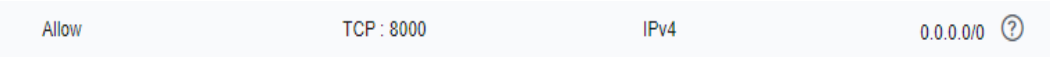
Asegúrese de que la regla de salida del grupo de seguridad ECS permita el acceso. Si no se permite el acceso, consulte la **Configuración de reglas del grupo de seguridad**.

Action	Protocol & Port	Type	Destination
Allow	All	IPv4	0.0.0.0/0

Paso 2 **Comprobar las reglas entrantes de GaussDB(DWS):**

Si no se configura ningún grupo de seguridad cuando se crea GaussDB(DWS), la regla de entrada predeterminada permite el acceso TCP desde todas las direcciones IPv4 y el puerto

8000. Para garantizar la seguridad, también puede permitir solo una dirección IP. Para obtener más información, consulte **¿Cómo configuro una lista blanca para proteger clústeres disponibles por una dirección IP pública?**



Paso 3 Crear una **Interconexión de VPC** entre la VPC1 donde está ECS y la VPC2 donde está GaussDB(DWS).

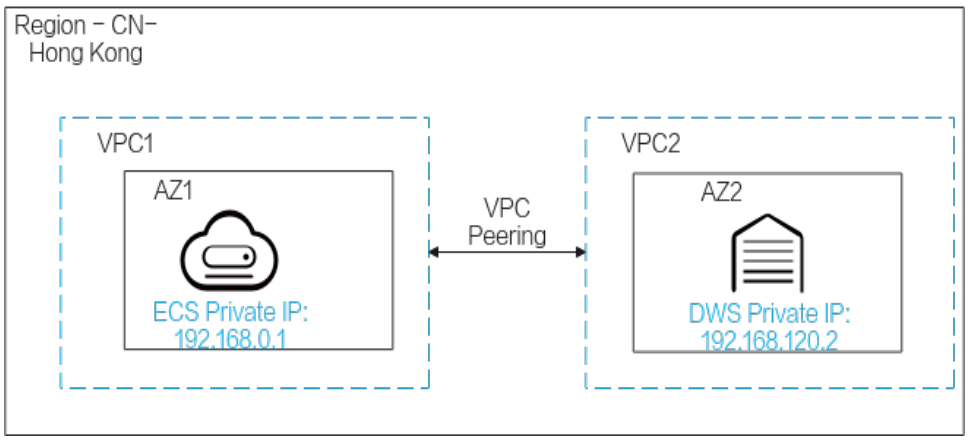
Paso 4 Inicie sesión en el ECS. Si se puede hacer ping la dirección IP interna de GaussDB(DWS), la conexión de red es normal. Si la dirección IP no se puede hacer ping, compruebe la configuración anterior. Si el ECS tiene un firewall, compruebe la configuración del firewall.

----Fin

Ejemplo de uso de gsql para la conexión:

gsql -d gaussdb -h *192.168.120.2* **-p 8000 -U dbadmin -W** *password* **-r**

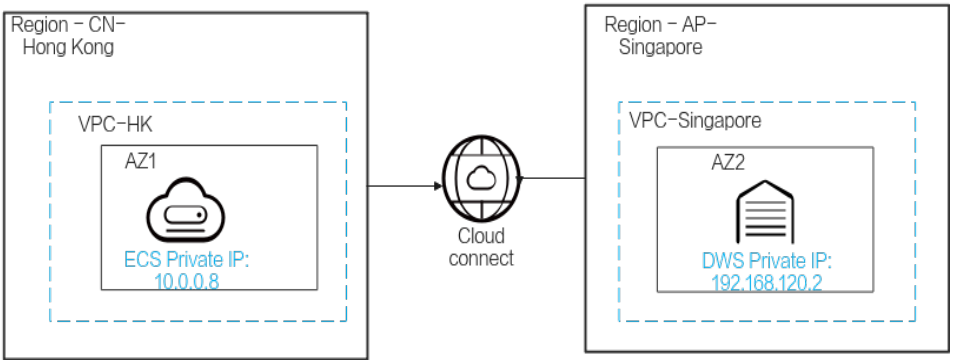
Figura 5-2 Acceso por la interconexión de VPC



Las aplicaciones de servicio y GaussDB(DWS) están en diferentes regiones

Si la aplicación de servicio y GaussDB(DWS) están en diferentes regiones, por ejemplo, ECS está en **CN-Hong Kong** y GaussDB(DWS) está en **AP-Singapore**, necesita establecer un **Cloud Connect** entre las dos regiones para la comunicación.

Figura 5-3 Acceso por la conexión en la nube



Las aplicaciones de servicio se despliegan en centros de datos locales y necesitan comunicarse con GaussDB(DWS).

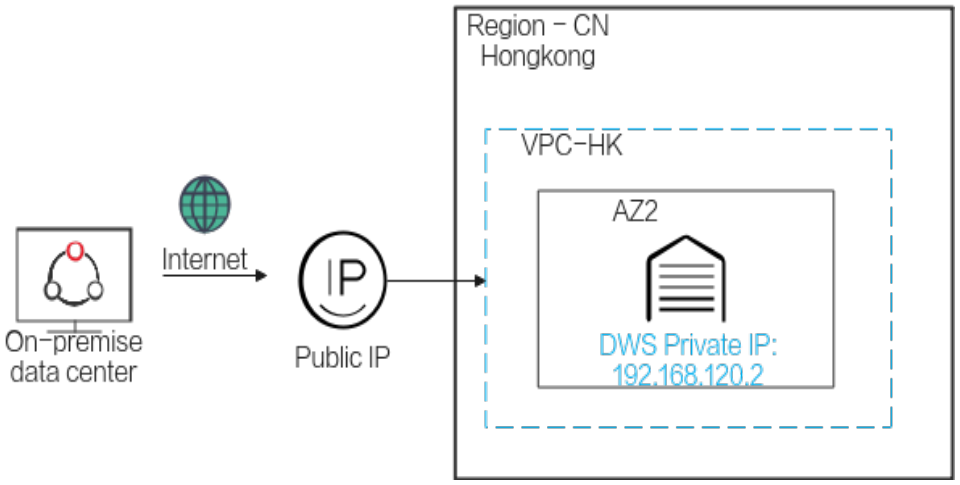
Si las aplicaciones de servicio no están en la nube sino en el centro de datos local, deben comunicarse con GaussDB(DWS) en la nube.

- **Escenario 1**: Las aplicaciones de servicio local se comunican con GaussDB(DWS) con direcciones IP públicas de GaussDB(DWS).

  Ejemplo de uso de gsql para la conexión:

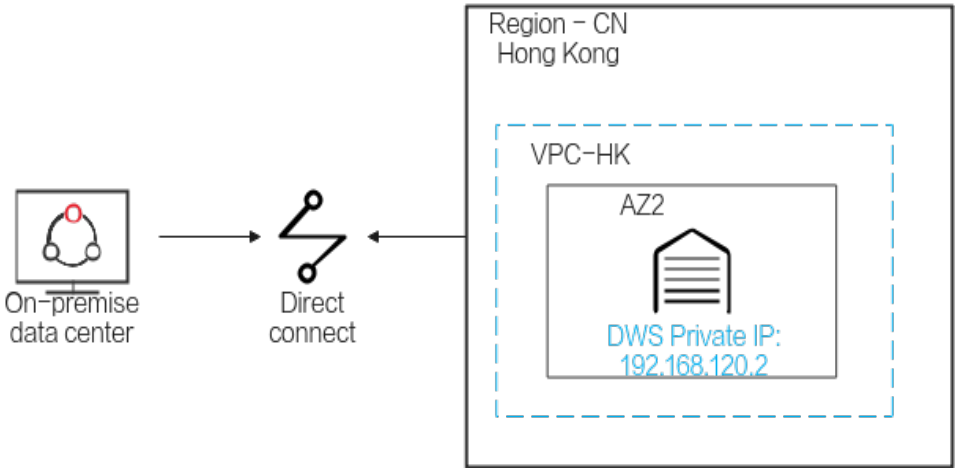
  gsql -d gaussdb -h *public_IP_address* **-p 8000 -U dbadmin -W** *password* **-r**

  Figura 5-4 Acceso por las direcciones IP públicas

- **Escenario 2**: Los servicios locales no pueden acceder a la red externa. En este caso, se requiere **Direct Connect** para la comunicación.

  Figura 5-5 Acceso por la conexión directa

5.2 ¿GaussDB(DWS) es compatible con clientes de terceros y controladores JDBC y ODBC?

Sí, pero se recomiendan los clientes y controladores de GaussDB(DWS). A diferencia de los clientes y controladores de PostgreSQL de código abierto, los clientes y controladores de GaussDB(DWS) tienen dos ventajas clave:

- **Endurecimiento de la seguridad**: Los controladores de PostgreSQL solo admiten la autenticación de MD5, pero los controladores de GaussDB(DWS) admiten SHA256 y MD5.
- **Mejora del tipo de datos**: Los controladores de GaussDB(DWS) admiten nuevos tipos de datos smalldatetime y tinyint.

GaussDB(DWS) soporta clientes de PostgreSQL de código abierto y controladores JDBC y ODBC.

Las versiones de cliente y controlador compatibles son:

- PostgreSQL psql 9.2.4 o posterior
- PostgreSQL JDBC Driver 9.3-1103 o posterior
- PSQL ODBC 09.01.0200 o posterior

Para obtener más información sobre cómo usar JDBC/ODBC para conectarse a GaussDB(DWS), véase la **Guía: Desarrollo basado en JDBC o ODBC**.

5.3 ¿Puedo conectarme a nodos de clúster de GaussDB(DWS) usando SSH?

No se admite el acceso directo. Las VM en la capa inferior de GaussDB(DWS) sirven como nodos de cálculo para el análisis de datos. Acceda a las bases de datos de clúster mediante la dirección de acceso a la red pública o privada en su lugar.

5.4 ¿Qué debo hacer si no puedo conectarme a un clúster de almacenamiento de datos?

Solución de problemas

Verificación:

- Si el estado del clúster es normal.
- Si el comando de conexión, nombre de usuario, contraseña, dirección IP y puerto son correctos.
- Si el tipo de sistema operativo y la versión del cliente son correctos.
- Si el cliente está instalado correctamente.

Si la conexión del clúster falló en la nube pública, verifique los siguientes elementos:

- Si los ECS están en la misma AZ, VPC, subred y grupo de seguridad que el clúster.
- Si las reglas entrantes y salientes del grupo de seguridad son correctas.

Si la conexión del clúster falla a través de Internet, confirme los elementos siguientes:

- Si su red está conectada a Internet.
- Si el firewall bloqueó el acceso.
- Si necesita acceder a Internet a través de un proxy.

Asistencia técnica

Si no se puede identificar el fallo, envíe un ticket de servicio para informar del problema: Inicie sesión en la consola de gestión y elija **Service Tickets** > **Create Service Ticket**.

5.5 ¿Por qué no se me notificó un fallo al desvincular la EIP cuando GaussDB(DWS) está conectado por Internet?

Después de que la EIP esté libre, la red puede desconectarse. Sin embargo, la capa TCP no detecta una conexión física defectuosa a tiempo debido a la configuración Keepalive. Como resultado, los clientes gsql, ODBC y JDBC tampoco pueden identificar el error de red a tiempo.

La duración cuando la base de datos envía el mensaje de desconexión al cliente depende de la configuración de Keepalive. El algoritmo específico para calcular la duración es:

keepalive_time + **keepalive_probes** x **keepalive_intvl**

Los valores de keepalive afectan a la estabilidad de la comunicación de la red. Ajustarlos a la presión de servicio y a las condiciones de la red.

En Linux, ejecute el comando **sysctl** para modificar los siguientes parámetros:

- net.ipv4.tcp_keepalive_time
- net.ipv4.tcp_keeaplive_probes
- net.ipv4.tcp_keepalive_intvl

Por ejemplo, si desea cambiar el valor de **net.ipv4.tcp_keepalive_time**, ejecute el siguiente comando para cambiarlo a **120**.

sysctl net.ipv4.tcp_keepalive_time=120

En Windows, modifique la siguiente información de configuración en el Registro **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters**:

- KeepAliveTime
- KeepAliveInterval
- TcpMaxDataRetransmissions (equivalent to **tcp_keepalive_probes**)

NOTA

Si no puede encontrar los parámetros anteriores en el registro **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters**, agregue estos parámetros. Abra **Registry Editor**, haga clic con el botón derecho en el área en blanco de la derecha y seleccione **Create** > **DWORD (32-bit) Value** para agregar estos parámetros.

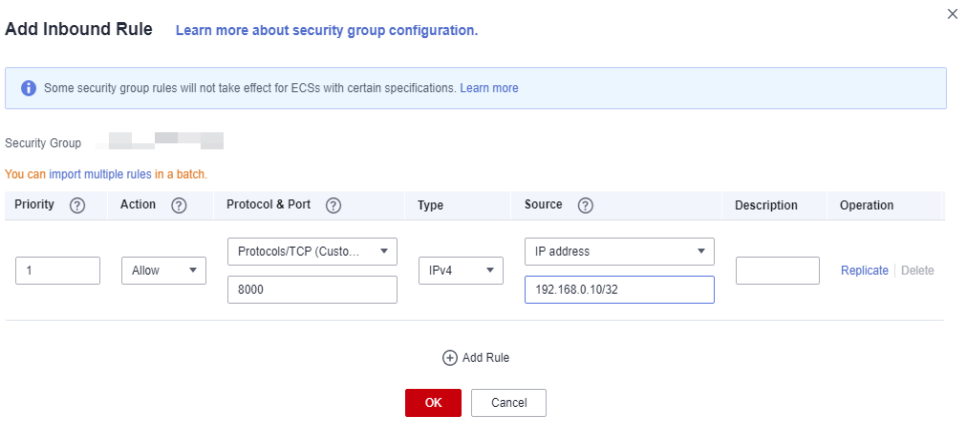
5.6 ¿Cómo configuro una lista blanca para proteger los clústeres disponibles por una dirección IP pública?

También puede iniciar sesión en la **consola de gestión de VPC** para crear manualmente un grupo de seguridad. A continuación, vuelva a la página para crear clústeres de almacén de datos, haga clic en el botón  junto a la lista desplegable **Security Group** para actualizar la página y seleccione el nuevo grupo de seguridad.

Para permitir que el cliente de GaussDB(DWS) se conecte al clúster, debe agregar una regla de entrada al nuevo grupo de seguridad para conceder el permiso de acceso al puerto de la base de datos del clúster GaussDB(DWS).

- **Protocol**: **TCP**
- **Port**: **8000** Utilice el conjunto de puertos de base de datos al crear el clúster GaussDB(DWS). Este puerto se utiliza para recibir conexiones de cliente a GaussDB(DWS).
- **Source**: Seleccione la **IP address** y utilice la dirección de IP del host del cliente, por ejemplo, **192.168.0.10/32**.

Figura 5-6 Adición de una regla de entrada



Se agregará la lista blanca.

6 Importación y exportación de datos

6.1 What Are the Differences Between Data Formats Supported by OBS and GDS Foreign Tables?

The file formats supported by OBS and GDS foreign tables are as follows:

OBS supports ORC, TEXT, JSON, CSV, CARBONDATA and PARQUET file formats for data import and ORC, CSV, and TEXT file formats for data export. The default format is TEXT.

GDS supports the following file formats: TEXT, CSV, and FIXED. The default format is TEXT.

6.2 ¿Cómo puedo importar datos incrementales usando una tabla externa de OBS?

Cuando se utiliza una tabla externa de OBS para importar datos, **INSERT** importa los datos a una tabla física local. Cuando se actualizan los datos de OBS, no es necesario volver a ejecutar la sentencia **INSERT**. Puede utilizar la sentencia **MERGE INTO**.

6.3 How Can I Import Data to GaussDB(DWS)?

GaussDB(DWS) supports efficient data import from multiple data sources. The following lists typical data import modes. For details, see **Importing Data**.

- Importing data from the OBS

  Upload data to OBS and then export it to GaussDB(DWS) clusters. Data formats such as CSV and TEXT are supported.

- Inserting data with **INSERT** statements

  Use the gsql client tool provided by GaussDB(DWS) or the JDBC/ODBC driver to write data to GaussDB(DWS) from upper-layer applications. GaussDB(DWS) supports complete database transaction-level CRUD operations. This is the simplest method and is applicable to scenarios with small data volume and low concurrency.

- Importing data from MRS with MRS as the ETL.
- Importing data with the **COPY FROM STDIN** command

  Run the **COPY FROM STDIN** command to write data to a table.
- Importing data from a remote server to GaussDB(DWS) using GDS

  Use the GDS data import function provided by GaussDB(DWS) to import data files from a common file system (for example, an ECS).
- Migrating data to GaussDB(DWS) using CDM

6.4 ¿Cuántos datos de servicio puede almacenar un almacén de datos?

Cada nodo de un clúster de almacén de datos tiene una capacidad de almacenamiento predeterminada de 1.49 TB, 2.98 TB, 4.47 TB, 160 GB, 1.68 TB o 13.41 TB. Un clúster puede alojar de 3 a 256 nodos y la capacidad total de almacenamiento del clúster se expande proporcionalmente a medida que crece la escala del clúster.

Para mejorar la fiabilidad, cada nodo tiene una copia, que ocupa la mitad del espacio de almacenamiento.

El sistema de GaussDB(DWS) realiza copias de seguridad de los datos y genera índices, archivos temporales de caché y registros de ejecución, que ocupan espacio de almacenamiento. Por lo tanto, el espacio de almacenamiento real de cada nodo es aproximadamente la mitad de la capacidad de almacenamiento total.

6.5 ¿Cómo uso \Copy para importar y exportar datos?

GaussDB(DWS) es un servicio totalmente gestionado en la nube. Los usuarios no pueden iniciar sesión en el fondo para importar o exportar datos mediante **COPY**, por lo que la sintaxis **COPY** está deshabilitada. Se recomienda almacenar archivos de datos en OBS y utilizar las tablas externas de OBS para importar datos. Si desea utilizar **COPY** para importar y exportar datos, realice las siguientes operaciones:

1. Coloque el archivo de datos en el cliente.
2. Utilice gsql para conectarse al clúster de destino.
3. Ejecute el siguiente comando para importar datos. Introduzca el nombre de directorio y el nombre de archivo del archivo de datos en el cliente y especifique la opción de importación en **with**. El comando es casi el mismo que el comando **COPY** común. Solo necesita agregar una barra invertida (\) antes del comando. Cuando los datos se importan correctamente, no se mostrará ninguna notificación.
   ```
   \copy tb_name from '/directory_name/file_name' with(...);
   ```
4. Ejecute el siguiente comando para exportar datos a un archivo local. Conserve la configuración predeterminada de los parámetros.
   ```
   \copy table_name to '/directory_name/file_name';
   ```
5. Especifique el parámetro **copy_option** para exportar datos a un archivo CSV.
   ```
   \copy table_name to '/directory_name/file_name' CSV;
   ```
6. Utilice **with** para especificar parámetros, exportando datos como archivos CSV que utilizan barras verticales (|) como delimitadores.
   ```
   \copy table_name to '/directory_name/file_name' with(format 'csv',delimiter
   '|') ;
   ```


6.6 ¿Cómo puedo implementar la importación de tolerancia a fallos entre diferentes bibliotecas de codificación

Para importar datos de la base de datos A (UTF8) a la base de datos B (GBK), puede haber un error de falta de coincidencia de conjunto de caracteres que hace que la importación de datos falle.

Para importar una pequeña cantidad de datos, ejecute el comando **\COPY**. El procedimiento es el siguiente:

Paso 1 Crear las bases de datos A y B. El formato de codificación de la base de datos A es UTF8, y el de la base de datos B es GBK.

```
postgres=> CREATE DATABASE A ENCODING 'UTF8' template = template0;
postgres=> CREATE DATABASE B ENCODING 'GBK' template = template0;
```

Paso 2 Ver la lista de la base de datos. Puede ver las bases de datos creadas A y 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)
```

Paso 3 Cambie a la base de datos A e introduzca la contraseña de usuario. Cree una tabla denominada **test01** e inserte datos en la tabla.

```
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
```

Paso 4 Ejecute el comando **\COPY** para exportar datos de la biblioteca UTF8 en formato Unicode al archivo **test01.dat**.

```
\copy test01 to '/opt/test01.dat' with (ENCODING 'Unicode');
```


Paso 5 Cambie a la base de datos B y cree una tabla con el mismo nombre **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);
```

Paso 6 Ejecute el comando **\COPY** para importar el archivo **test01.dat** a la base de datos B.

```
\copy test01 from '/opt/test01.dat' with (ENCODING
'Unicode' ,COMPATIBLE_ILLEGAL_CHARS 'true');
```

NOTA

- El parámetro de tolerancia a errores **COMPATIBLE_ILLEGAL_CHARS** especifica que se toleran caracteres no válidos durante la importación de datos. Los caracteres no válidos se convierten y, a continuación, se importan a la base de datos. No se muestra ningún mensaje de error. La importación no se interrumpe.
- El formato BINARY no es compatible. Cuando se importan datos de dicho formato, se producirá el error "cannot specify bulkload compatibility options in BINARY mode" (no se pueden especificar opciones de compatibilidad de carga masiva en modo BINARY).
- El parámetro solo es válido para la importación de datos mediante la opción **COPY FROM**.

Paso 7 Ver datos en la tabla **test01** de la base de datos B.

```
b=> select * from test01;
 c_customer_sk | c_customer_id | c_first_name | c_last_name
---------------+---------------+--------------+-------------
          3769 | hello         | Grace        |
           456 | good          |              |
(2 rows)
```

Paso 8 Después de realizar las operaciones anteriores, los datos se importan de la base de datos A (UTF8) a la base de datos B (GBK).

----Fin

6.7 Can I Import and Export Data to and from OBS Across Regions?

No. No, GaussDB(DWS) does not support OBS data import or export across regions. The GaussDB(DWS) cluster and OBS must be in the same region.

For example, direct data export from a GaussDB(DWS) cluster in Beijing 4 to OBS in Beijing 1 is not supported. However, you can first export the data to Beijing 4 OBS and then use CDM to export the data to Beijing 1 OBS.

6.8 ¿Cómo importo datos de GaussDB(DWS)/Oracle/MySQL/SQL Server a GaussDB(DWS) (Migración de toda la base de datos)?

Los datos heterogéneos se pueden importar a GaussDB(DWS) con CDM. Puede migrar una base de datos completa de Oracle, MySQL, SQL Server o GaussDB(DWS) a una base de datos de GaussDB(DWS). Para obtener más información, consulte **Creación de un trabajo de migración de base de datos completo**.

También puede almacenar datos en OBS y luego volcar los datos en GaussDB(DWS). Para obtener más información, consulte **Acerca de la importación de datos paralelos desde OBS**.

6.9 ¿Puedo importar datos por la red pública/externa con GDS?

No. El servidor de GDS y GaussDB(DWS) solo pueden comunicarse entre sí en la intranet. Cada DN en el clúster de GaussDB(DWS) se utiliza para conectarse al servidor de GDS en paralelo para importar una gran cantidad de datos. El servidor de GDS y el clúster deben estar en la misma red. Si GDS se despliega en un servidor sin conexión, el firewall debe estar habilitado y el clúster necesita una EIP. Sin embargo, un clúster solo puede vincularse a una EIP, y la importación de datos con múltiples DN no puede implementarse.

6.10 ¿Cuáles son los factores que afectan al rendimiento de importación de GaussDB(DWS)?

El rendimiento de importación de GaussDB(DWS) se ve afectado por los siguientes factores:

1. Especificaciones del clúster: E/S de disco, rendimiento de red, memoria y especificaciones de CPU
2. Planificación de servicios: tipo de campos de tabla, comprimir y almacén de filas o almacén de columnas
3. Almacenamiento de datos: clúster local, OBS
4. Modo de importación de datos

7 Cuenta, Contraseña y Permiso

7.1 ¿Cómo implementa GaussDB(DWS) el aislamiento de la carga de trabajo?

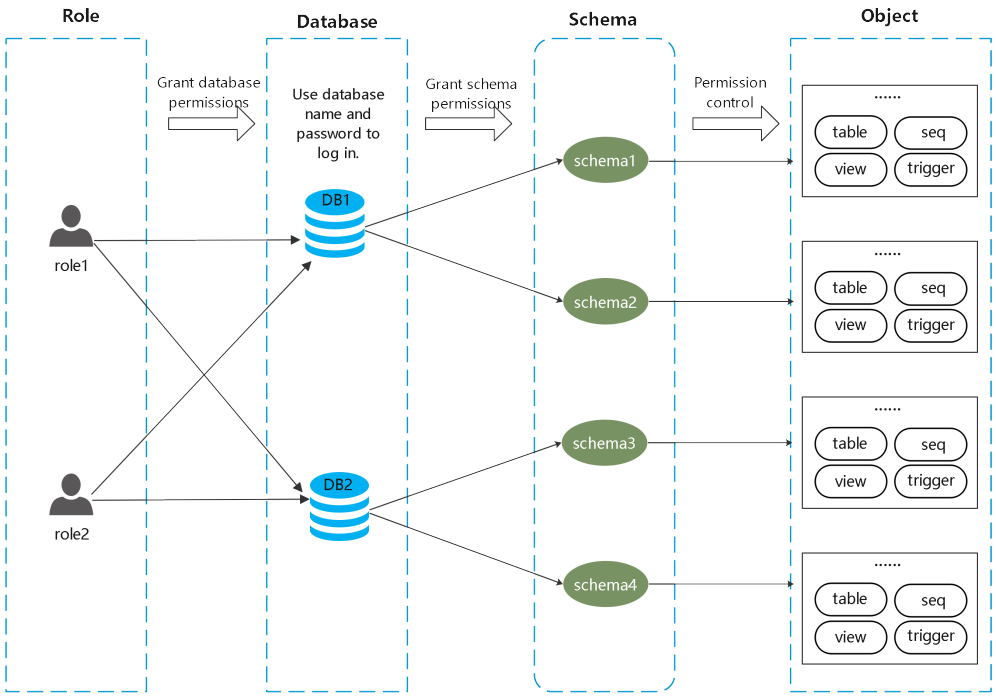
Aislamiento de la carga de trabajo

En GaussDB(DWS), puede aislar cargas de trabajo con configuraciones de bases de datos y esquemas. Sus diferencias son las siguientes:

- Las bases de datos no pueden comunicarse entre sí y compartir muy pocos recursos. Sus conexiones y permisos pueden ser aislados.
- Los esquemas comparten más recursos que las bases de datos. Los permisos de usuario en esquemas y objetos subordinados se pueden configurar de forma flexible mediante la sintaxis **GRANT** y **REVOKE**.

Se recomienda utilizar esquemas para aislar servicios para mayor comodidad y uso compartido de recursos. Se recomienda que los administradores del sistema creen esquemas y bases de datos y, a continuación, asignen los permisos necesarios a los usuarios.

Figura 7-1 Se utiliza para el control de permisos.



Base de datos

Una base de datos es una colección física de objetos de base de datos. Los recursos de las bases de datos diferentes están completamente aislados (excepto algunos objetos compartidos). Las bases de datos se utilizan para aislar cargas de trabajo. Los objetos de diferentes bases de datos no pueden tener acceso entre sí. Por ejemplo, no se puede tener acceso a los objetos de la base de datos B en la base de datos A. Por lo tanto, al iniciar sesión en un clúster, debe conectarse a la base de datos especificada.

Esquema

En una base de datos, los objetos de base de datos se dividen y se aíslan lógicamente en función de los esquemas.

Con la gestión de permisos, puede acceder y operar objetos en diferentes esquemas en la misma sesión. Los esquemas contienen objetos a los que las aplicaciones pueden acceder, como tablas, índices, datos de diversos tipos, funciones y operadores.

Los objetos de base de datos con el mismo nombre no pueden existir en el mismo esquema, pero los nombres de objetos en diferentes esquemas pueden ser los mismos.

```
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
```


Los esquemas dividen lógicamente las cargas de trabajo. Estas cargas de trabajo son interdependientes con los esquemas. Por lo tanto, si un esquema contiene objetos, eliminarlo causará errores con la información de dependencia mostrada.

```
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.
```

Cuando se elimina un esquema, se utiliza la opción **CASCADE** para eliminar los objetos que dependen del esquema.

```
gaussdb=> DROP SCHEMA myschema_1 CASCADE;
NOTICE:  drop cascades to table myschema_1.t1
gaussdb=> DROP SCHEMA
```

Usuario/Rol

Los usuarios y los roles se utilizan para implementar el control de permisos en el servidor de base de datos (clúster). Son los propietarios y ejecutores de las cargas de trabajo de clúster y gestionan todos los permisos de objetos en clústeres. Un rol no está limitado a una base de datos específica. Sin embargo, cuando inicia sesión en el clúster, debe especificar explícitamente un nombre de usuario para garantizar la transparencia de la operación. Los permisos de un usuario para una base de datos se pueden especificar mediante la gestión de permisos.

Un usuario es el sujeto de permisos. La gestión de permisos es en realidad el proceso de decidir si un usuario puede realizar operaciones en objetos de base de datos.

Gestión de permisos

La gestión de permisos en GaussDB(DWS) se divide en tres categorías:

- Permiso del sistema

  Los permisos del sistema también se denominan atributos de usuario, incluidos **SYSADMIN**, **CREATEDB**, **CREATEROLE**, **AUDITADMIN** y **LOGIN**.

  Solo se pueden especificar mediante la sintaxis **CREATE ROLE** o **ALTER ROLE**. El permiso **SYSADMIN** se puede conceder y revocar usando **GRANT ALL PRIVILEGE** y **REVOKE ALL PRIVILEGE** respectivamente. Los permisos del sistema no pueden ser heredados por un usuario de un rol y no se pueden conceder mediante **PUBLIC**.

- Permisos

  Otorgar los permisos de un rol o usuario a uno o más roles o usuarios. En este caso, cada rol o usuario puede considerarse como un conjunto de uno o más permisos de base de datos.

  Si se especifica **WITH ADMIN OPTION**, el miembro puede a su vez conceder permisos en el rol a otros y revocar permisos en el rol también. Si se cambia o revoca un rol o usuario a quien se conceden ciertos permisos, los permisos heredados del rol o usuario también cambian.

  Un administrador de base de datos puede concederles permisos y revocarlos de cualquier rol o usuario. Los roles que tienen permiso **CREATEROLE** pueden conceder o revocar la membresía en cualquier rol que no sea un administrador.

- Permiso de objeto

  Los permisos de un objeto de base de datos (tabla, vista, columna, base de datos, función, esquema o espacio de tabla) se pueden conceder a un rol o usuario. El comando

GRANT se puede utilizar para conceder permisos a un usuario o rol. Estos permisos concedidos se agregan a los existentes.

Ejemplo de aislamiento de esquema

Ejemplo 1:

De forma predeterminada, el propietario de un esquema tiene todos los permisos sobre los objetos del esquema, incluido el permiso de eliminación. El propietario de una base de datos tiene todos los permisos sobre los objetos de la base de datos, incluido el permiso de eliminación. Por lo tanto, se recomienda controlar estrictamente la creación de bases de datos y esquemas. Cree bases de datos y esquemas como administrador y asigne permisos relacionados a los usuarios.

Paso 1 Asigne el permiso para crear esquemas en la base de datos **testdb** al usuario **user_1** como usuario **dbadmin**.

```
testdb=> GRANT CREATE ON DATABASE testdb to user_1;
GRANT
```

Paso 2 Cambiar al usuario **user_1**.

```
testdb=> SET SESSION AUTHORIZATION user_1 PASSWORD '********';
SET
```

Cree un esquema denominado **myschema_2** en la base de datos **testdb** como **user_1**.

```
testdb=> CREATE SCHEMA myschema_2;
CREATE SCHEMA
```

Paso 3 Cambiar al administrador **dbadmin**.

```
testdb=> RESET SESSION AUTHORIZATION;
RESET
```

Cree **table t1** en el esquema **myschema_2** como administrador **dbadmin**.

```
testdb=> CREATE TABLE myschema_2.t1(a int, b int) DISTRIBUTE BY HASH(b);
CREATE TABLE
```

Paso 4 Cambiar al usuario **user_1**.

```
testdb=> SET SESSION AUTHORIZATION user_1 PASSWORD '********';
SET
```

Eliminar la tabla **t1** creada por el administrador **dbadmin** en el esquema **myschema_2** como usuario **user_1**.

```
testdb=> drop table myschema_2.t1;
DROP TABLE
```

----Fin

Ejemplo 2:

Debido al aislamiento lógico de los esquemas, los objetos de base de datos deben verificarse tanto en el nivel de esquema como en el nivel de objeto.

Paso 1 Conceda el permiso en la tabla **myschema.t1** a **user_1**.

```
gaussdb=> GRANT SELECT ON TABLE myschema.t1 TO user_1;
GRANT
```

Paso 2 Cambiar al usuario **user_1**.

```
SET SESSION AUTHORIZATION user_1 PASSWORD '********';
SET
```


Consultar la tabla **myschema.t1**.

```
gaussdb=> SELECT * FROM myschema.t1;
ERROR:  permission denied for schema myschema
LINE 1: SELECT * FROM myschema.t1;
```

Paso 3 Cambiar al administrador **dbadmin**.

```
gaussdb=> RESET SESSION AUTHORIZATION;
RESET
```

Otorgar el permiso de la tabla **myschema.t1** al usuario **user_1**.

```
gaussdb=> GRANT USAGE ON SCHEMA myschema TO user_1;
GRANT
```

Paso 4 Cambiar al usuario **user_1**.

```
gaussdb=> SET SESSION AUTHORIZATION user_1 PASSWORD '********';
SET
```

Consultar la tabla **myschema.t1**.

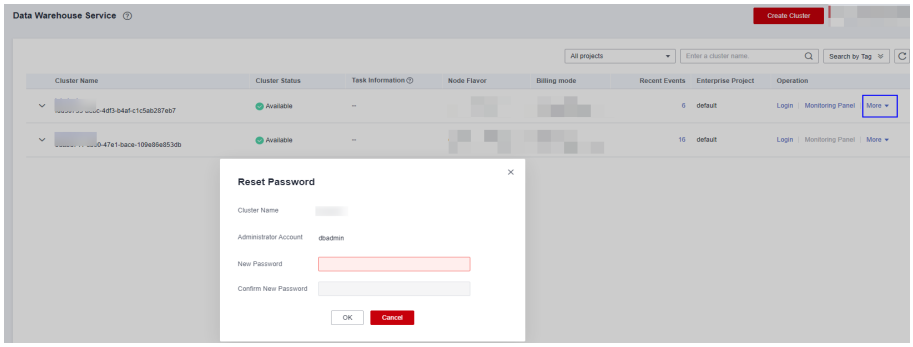
```
gaussdb=> SELECT * FROM myschema.t1;
 a | b
---+---
(0 rows)
```

----Fin

7.2 How Do I Change the Password of a Database Account When the Password Expires?

- To change the password of the database administrator **dbadmin**, log in to the console and choose **More** > **Reset Password** in cluster row.

  Figura 7-2 Resetting the password of user dbadmin

  For security, the following two parameters manage account passwords. Log in to the console, click the cluster name and switch to the parameter modification page to modify the parameters.

  – **failed_login_attempts**: maximum number of consecutive incorrect password attempts before the account is locked. Run the following statement as user **dbadmin** to unlock the account:
    ```
    ALTER USER user_name ACCOUNT UNLOCK;
    ```
  – **password_effect_time**: validity period of the account password, in days. The default value is **90**.

- You can also connect to the database and run the **ALTER USER** command to change the password validity period of a database account (common user and administrator dbadmin).

```
ALTER USER username PASSWORD EXPIRATION 90;
```

7.3 ¿Cómo puedo conceder permisos de tabla a un usuario?

Esta sección describe cómo conceder a los usuarios los permisos SELECT, INSERT, UPDATE o completos de tablas a los usuarios.

Sintaxis

```
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 ];
```

Escenario

Supongamos que hay usuarios **u1**, **u2**, **u3**, **u4**, y **u5** y cinco esquemas con el nombre de estos usuarios. Sus requisitos de permiso son los siguientes:

- El usuario **u2** es un usuario de sólo lectura y requiere el permiso SELECT para la tabla **u1.t1**.
- El usuario **u3** requiere el permiso SELECT para la tabla **u1.t1**.
- El usuario **u3** requiere el permiso UPDATE para la tabla **u1.t1**.
- El usuario **u5** requiere todos los permisos de la tabla **u1.t1**.

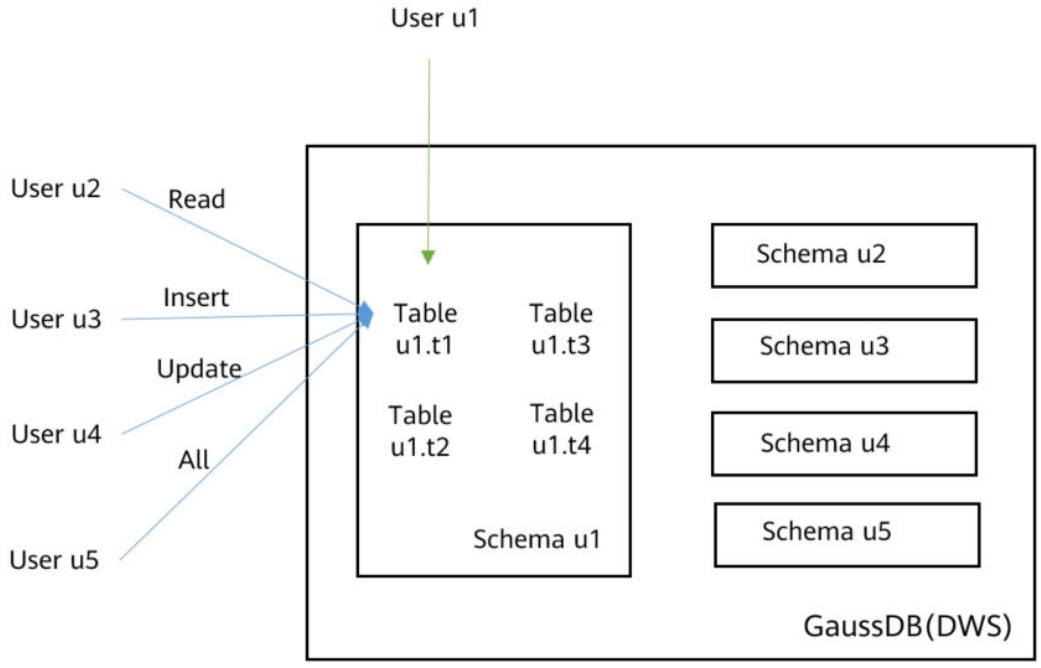


Tabla 7-1 Permisos de la tabla u1.t1

Usuario	Tipo	Sentencia de GRANT	Consultar	Ingresar	Actualizar	Eliminar
u1	Propietario	-	√	√	√	√
u2	Usuario de solo lectura	`GRANT SELECT ON u1.t1 TO u2;`	√	x	x	x
u3	Usuario INSERT	`GRANT INSERT ON u1.t1 TO u3;`	x	√	x	x
u4	Usuario UPDATE	`GRANT SELECT,UPDATE ON u1.t1 TO u4;` **AVISO** El permiso UPDATE se debe conceder junto con el permiso SELECT, o puede producirse una fuga de información.	√	x	√	x
u5	Usuarios con todos los permisos	`GRANT ALL PRIVILEGES ON u1.t1 TO u5;`	√	√	√	√

Procedimiento

Realice los siguientes pasos para conceder y verificar permisos:

Paso 1 Conéctese a su base de datos como **dbadmin**. Ejecute las siguientes instrucciones para crear usuarios **u1** a **u5**. De forma predeterminada, se crearán cinco esquemas con el nombre de los usuarios.

```
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}';
```


Paso 2 Crear tabla **u1.t1** en el esquema **u1**.

```
CREATE TABLE u1.t1 (c1 int, c2 int);
```

Paso 3 Insertar dos registros en la tabla.

```
INSERT INTO u1.t1 VALUES (1,2);
INSERT INTO u1.t1 VALUES (1,2);
```

Paso 4 Conceder permisos de esquema a los usuarios.

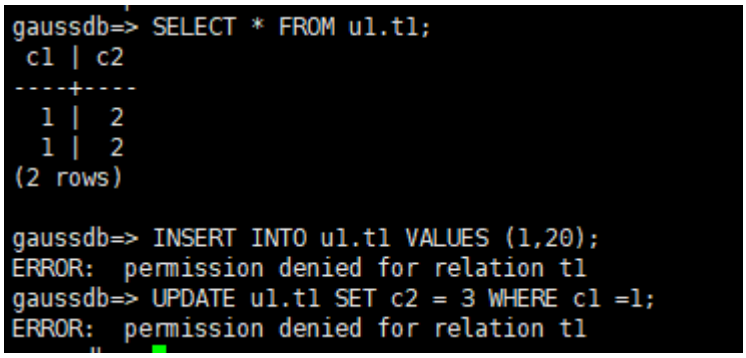
```
GRANT USAGE ON SCHEMA u1 TO u2,u3,u4,u5;
```

Paso 5 Otorgar al usuario **u2** el permiso para consultar la tabla **u1.t1**.

```
GRANT SELECT ON u1.t1 TO u2;
```

Paso 6 Iniciar una nueva sesión y conéctese a la base de datos como usuario **u2**. Comprobar que el usuario **u2** puede consultar la tabla **u1.t1** pero no puede escribir en la tabla ni modificarla.

```
SELECT * FROM u1.t1;
INSERT INTO u1.t1 VALUES (1,20);
UPDATE u1.t1 SET c2 = 3 WHERE c1 =1;
```

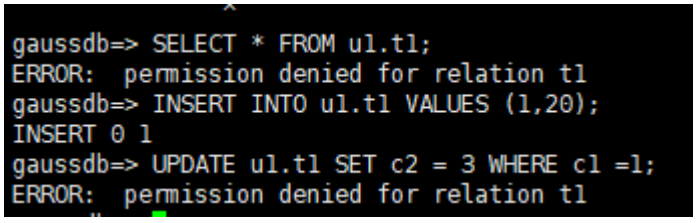


Paso 7 En la sesión iniciada por el usuario **dbadmin**, conceda permisos a los usuarios **u3**, **u4** y **u5**.

```
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.
```

Paso 8 Inicie una nueva sesión y conéctese a la base de datos como usuario **u3**. Compruebe que el usuario **u3** puede consultar la tabla **u1.t1** pero no puede consultar o modificar la tabla.

```
SELECT * FROM u1.t1;
INSERT INTO u1.t1 VALUES (1,20);
UPDATE u1.t1 SET c2 = 3 WHERE c1 =1;
```



Paso 9 Inicie una nueva sesión y conéctese a la base de datos como usuario **u4**. Compruebe que el usuario **u4** puede modificar y consultar la tabla **u1.t1**, pero no puede insertar datos en la tabla.

```
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
```

Paso 10 Inicie una nueva sesión y conéctese a la base de datos como usuario **u5**. Verifique que el usuario **u5** pueda consultar, insertar, modificar y eliminar datos en la tabla **u1.t1**.

```
SELECT * FROM u1.t1;
INSERT INTO u1.t1 VALUES (1,20);
UPDATE u1.t1 SET c2 = 3 WHERE c1 =1;
DELETE FROM u1.t1;
```

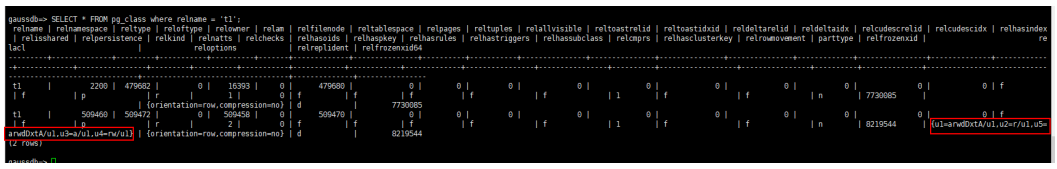
```
gaussdb=> SELECT * FROM u1.t1;
 c1 | c2
----+----
  1 |  3
  1 |  3
  1 |  3
(3 rows)

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
```

Paso 11 En la sesión iniciada por el usuario **dbadmin**, ejecute la función has_table_privilege para consultar los permisos del usuario.

```
SELECT * FROM pg_class WHERE relname = 't1';
```

Compruebe la columna **relacl** en la salida del comando. *rolename=xxxx/yyyy* indica que *rolename* tiene el permiso *xxxx* en la tabla y el permiso se obtiene de *yyyy*.

En la siguiente figura se muestra la salida del comando.



- **u1=arwdDxtA/u1** indica que **u1** es el propietario y tiene permisos completos.
- **u2=r/u1** indica que **u2** tiene el permiso de lectura.
- **u3=a/u1** indica que **u3** tiene el permiso de inserción.
- **u4=rw/u1** indica que **u4** tiene los permisos de lectura y actualización.
- **u5=arwdDxtA/u1** indica que **u5** tiene permisos completos.

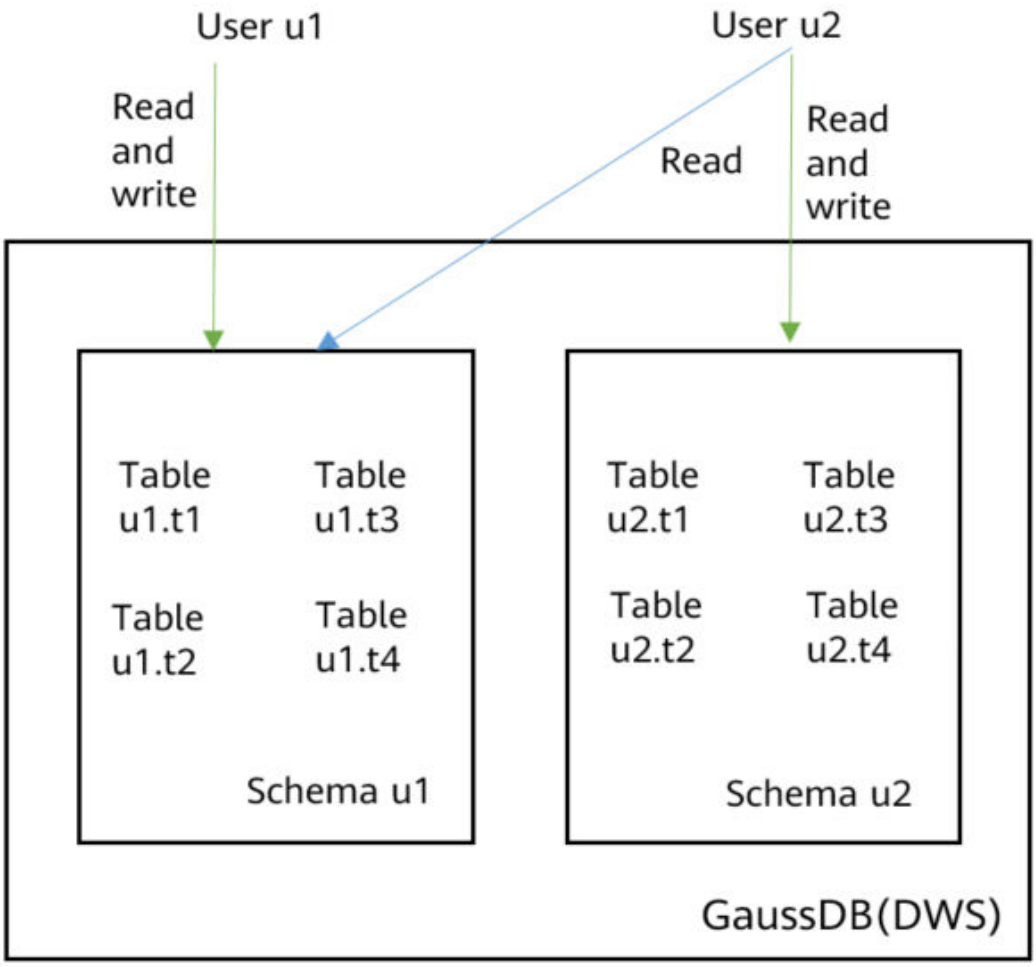
----Fin

7.4 How Do I Grant Schema Permissions to a User?

This section describes how to grant the query permission for a schema as an example. For more information, see **How Do I Grant Table Permissions to a User?** :

- Permission for a table in a schema
- Permission for all the tables in a schema
- Permission for tables to be created in the schema

Assume that there are users **u1** and **u2**, and two schemas named after them. User **u2** needs to access tables in schema **u1**.



Paso 1 Connect to your database as **dbadmin**. Run the following statements to create users **u1** and **u2**. Two schemas will be created and named after the users by default.

```
CREATE USER u1 PASSWORD '{password}';
CREATE USER u2 PASSWORD '{password}';
```

Paso 2 Create tables **u1.t1** and **u1.t2** in schema **u1**.

```
CREATE TABLE u1.t1 (c1 int, c2 int);
CREATE TABLE u1.t2 (c1 int, c2 int);
```


Paso 3 Grant the access permission of schema **u1** to user **u2**.

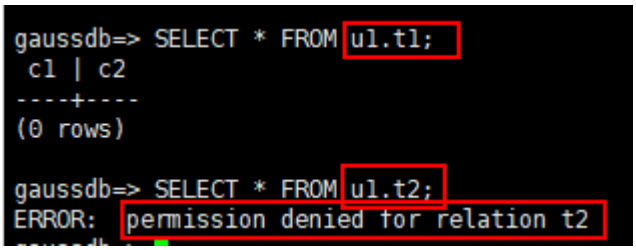
```
GRANT USAGE ON SCHEMA u1 TO u2;
```

Paso 4 Grant user **u2** the permission to query table **u1.t1** in schema **u1**.

```
GRANT SELECT ON u1.t1 TO u2;
```

Paso 5 Start a new session and connect to the database as user **u2** Verify that user **u2** can query the **u1.t1** table but not the **u1.t2** table.

```
SELECT * FROM u1.t1;
SELECT * FROM u1.t2;
```

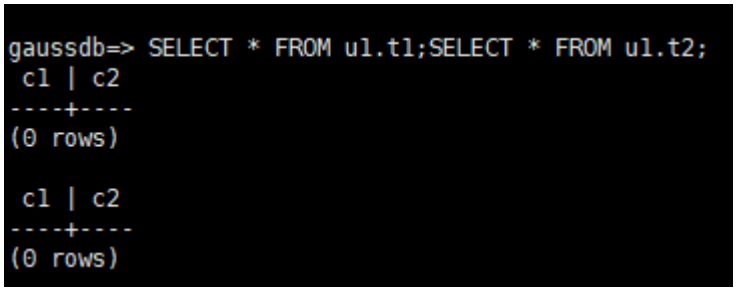


Paso 6 In the session started by user **dbadmin**, grant user **u2** the permission to query all the tables in schema **u1**.

```
GRANT SELECT ON ALL TABLES IN SCHEMA u1 TO u2;
```

Paso 7 In the session started by user **u2**, verify that **u2** can query all tables.

```
SELECT * FROM u1.t1;
SELECT * FROM u1.t2;
```

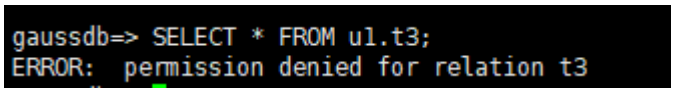


Paso 8 In the session started by user **dbadmin**, create table **u1.t3**.

```
CREATE TABLE u1.t3 (c1 int, c2 int);
```

Paso 9 In the session started by user **u2**, verify that user **u2** does not have the query permission for **u1.t3**. It indicates that user **u2** has the permission to access all the existing tables in schema **u1**, but not the tables to be created in the future.

```
SELECT * FROM u1.t3;
```



Paso 10 In the session started by user **dbadmin**, grant user **u2** the permission to query the tables to be created in schema **u1**. Create table **u1.t4**.

NOTA

ALTER DEFAULT PRIVILEGES is used to grant permissions on objects to be created.

```
ALTER DEFAULT PRIVILEGES FOR ROLE u1 IN SCHEMA u1 GRANT SELECT ON TABLES TO u2;
CREATE TABLE u1.t4 (c1 int, c2 int);
```


Paso 11 In the session started by user **u2**, verify that user **u2** can access table **u1.t4**, but does not have the permission to access **u1.t3**. To let the user access table **u1.t3**, you can grant permissions by performing **Paso 4**.

```
SELECT * FROM u1.t4;
```

```
gaussdb=> SELECT * FROM u1.t4;
 c1 | c2
----+----
(0 rows)
```

----Fin

7.5 How Do I Create a Database Read-only User?

Scenario

In service development, database administrators use schemas to classify data. For example, in the financial industry, liability data belong to schema **s1**, and asset data belong to schema **s2**.

Now you have to create a read-only user **user1** in the database. The user can access all tables (including new tables to be created in the future) in schema **s1** for daily reading, but cannot insert, modify, or delete data.

Principles

DWS provides role-based user management. You need to create a read-only role **role1** and grant the role to **user1**. For details, see **Role-based Access Control**.

Procedure

Paso 1 Connect to the DWS database as user **dbadmin**.

Paso 2 Run the following SQL statement to create role **role1**:

```
CREATE ROLE role1 PASSWORD disable;
```

Paso 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.
```

Paso 4 Run the following SQL statement to grant the role **role1** to the actual user **user1**:

```
GRANT role1 TO user1;
```

Paso 5 Read all table data in schema **s1** as read-only user **user1**.

----Fin

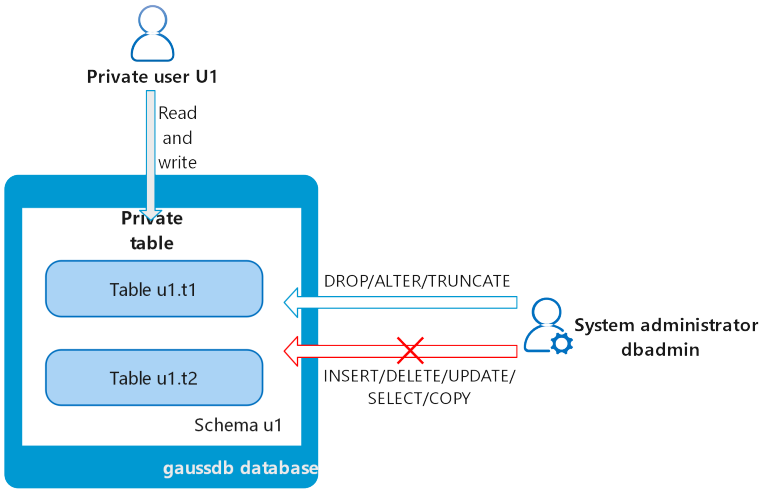
7.6 How Do I Create Private Database Users and Tables?

Scenario

The system administrator **dbadmin** has the permission to access tables created by common users by default. When **Separation of Permissions** 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.

Figura 7-3 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

Paso 1 Connect to the DWS database as user **dbadmin**.

Paso 2 Run the following SQL statement to create private user **u1**:

```
CREATE USER u1 WITH INDEPENDENT IDENTIFIED BY 'password';
```

Paso 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');
```

Paso 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"
```

Paso 5 Run the **DROP** statement as user **dbadmin** to delete the table **test**.

```
DROP TABLE u1.test;
```

```
gaussdb=> drop table u1.test;
DROP TABLE
```

----Fin

7.7 ¿Cómo revoco el permiso de CONNECT ON DATABASE de un usuario?

Escenario

En un servicio, el permiso del usuario **u1** para conectarse a una base de datos debe ser revocado. Una vez que el comando **REVOKE CONNECT ON DATABASE *gaussdb* FROM u1;** se ejecuta correctamente, el usuario **u1** todavía puede conectarse a la base de datos. Esto significa que la revocación no tiene efecto.

Análisis de las causas

Si ejecuta el comando **REVOKE CONNECT ON DATABASE gaussdb from u1** para revocar los permisos de **u1** de usuario, la revocación no surte efecto porque el permiso **CONNECT** de la base de datos se concede a **PUBLIC**. Por lo tanto, debe especificar **PUBLIC**.

- GaussDB(DWS) proporciona un **PUBLIC** grupo definido implícitamente que contiene todos los roles. De forma predeterminada, todos los usuarios y roles nuevos tienen los permisos de **PUBLIC**. Para revocar los permisos de **PUBLIC** de un usuario o rol, o

volver a concederles estos permisos, agregue la palabra clave **PUBLIC** en la instrucción **REVOKE** o **GRANT**.

- GaussDB(DWS) otorga los permisos para objetos de ciertos tipos al **PUBLIC**. De forma predeterminada, **PUBLIC** no concede permisos en tablas, columnas, secuencias, orígenes de datos externos, servidores externos, esquemas y tablas, pero **PUBLIC** concede los siguientes permisos;
  - **CONNECT** el permiso de una base de datos
  - **CREATE TEMP TABLE** el permiso de una base de datos
  - **EXECUTE** el permiso de una función
  - **USAGE** el permiso para idiomas y tipos de datos (incluidos dominios)
- Un propietario de objeto puede revocar los permisos predeterminados concedidos a **PUBLIC** y conceder permisos a otros usuarios según sea necesario.

Ejemplo de operaciones

Ejecute el siguiente comando para revocar el permiso del usuario **u1** para acceder al **gaussdb** de la base de datos:

Paso 1 Conéctese a la base de datos de GaussDB(DWS) **gaussdb**.

```
gsql -d gaussdb -p 8000 -h 192.168.x.xx -U dbadmin -W password -r
gaussdb=>
```

Paso 2 Cree el usuario **u1**.

```
gaussdb=> CREATE USER u1 IDENTIFIED BY 'xxxxxxxx';
```

Paso 3 Verifique que el usuario **u1** pueda acceder a GaussDB.

```
gsql -d gaussdb -p 8000 -h 192.168.x.xx -U u1 -W password -r
gaussdb=>
```

Paso 4 Conéctese a la base de datos **gaussdb** como administrador **dbadmin** y ejecute el comando REVOKE para revocar el permiso de conexión en la base de datos del usuario **public**.

```
gsql -d gaussdb -h 192.168.x.xx -U dbadmin -p 8000 -r
gaussdb=> REVOKE CONNECT ON DATABASE gaussdb FROM public;
REVOKE
```

Paso 5 Verifique el resultado. Utilice **u1** para conectarse a la base de datos. Si se muestra la siguiente información, el permiso **connect on database** del usuario **u1** se ha revocado correctamente:

```
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.
```

----Fin

7.8 ¿Cómo puedo ver los permisos de tabla de un usuario?

Escenario 1: Ejecute el comando **information_schema.table_privileges** a **view the table permissions of a user**. Por ejemplo:

```
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)
```

Tabla 7-2 Columnas de table_privileges

Columna	Tipo de datos	Descripción
grantor	sql_identifier	Concedidor de permisos
grantee	sql_identifier	Concesionario del permiso
table_catalog	sql_identifier	Base de datos donde se encuentra la tabla
table_schema	sql_identifier	Esquema donde se encuentra la tabla
table_name	sql_identifier	Nombre de la tabla
privilege_type	character_data	Tipo de permiso concedido. El valor puede ser **SELECT**, **INSERT**, **UPDATE**, **DELETE**, **TRUNCATE**, **REFERENCES**, **ANALYZE**, **VACUUM**, **ALTER**, **DROP** o **TRIGGER**.
is_grantable	yes_or_no	Indica si el permiso se puede conceder a otros usuarios. **YES** indica que el permiso se puede conceder a otros usuarios, y **NO** indica que el permiso no se puede conceder a otros usuarios.
with_hierarchy	yes_or_no	Indica si se permiten operaciones específicas heredadas en el nivel de tabla. Si la operación específica es **SELECT**, se muestra **YES**. De lo contrario, se muestra **NO**.

En la figura anterior, el usuario **u2** tiene todos los permisos de la tabla **t2** en el esquema **u2** y el permiso **SELECT** de la tabla **t1** en el esquema **u1**.

information_schema.table_privileges solo puede consultar los permisos concedidos directamente al usuario, la función **has_table_privilege()** puede consultar tanto los permisos concedidos directamente como los permisos indirectos (obtenidos por el rol de GRANT al usuario). Por ejemplo:

```
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
----
```



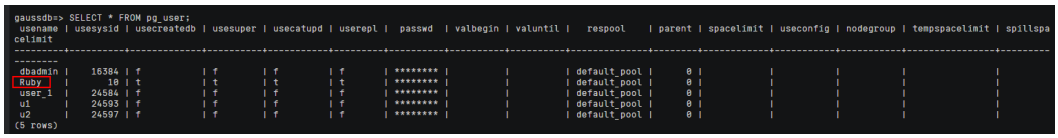
```
(0 rows)

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)
```

Escenario 2: Para **check whether a user has permissions on a table**, realice los siguientes pasos:

7.9 Who Is User Ruby?

When you run the **SELECT * FROM pg_user** statement to view users in the current system, you may see user **Ruby** many times, who has many permissions.

User **Ruby** is an official O&M account. After a GaussDB(DWS) database is created, user **Ruby** is generated by default. There is no security risk in regard to user **Ruby**.



8 Rendimiento de bases de datos

8.1 Why Is SQL Execution Slow After Long GaussDB(DWS) Usage?

After a database is used for a period of time, the table data increases as services grow, or the table data is frequently added, deleted, or modified. As a result, bloating tables and inaccurate statistics are incurred, deteriorating database performance.

You are advised to periodically run **VACUUM FULL** and **ANALYZE** on tables that are frequently added, deleted, or modified. Perform the following operations:

Paso 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
```

Paso 2 Run **ANALYZE** again. For details, see **ANALYZE | ANALYSE**.

```
gaussdb=> ANALYZE customer_t1;
ANALYZE
```

----Fin

NOTA

To test whether disk fragments affect database performance, use the following function:

```
SELECT * FROM pgxc_get_stat_dirty_tables(30,100000);
```


8.2 ¿Por qué GaussDB(DWS) funciona peor que una base de datos de un solo servidor en los escenarios extremos?

Debido a la limitación de la arquitectura MPP de GaussDB(DWS), algunos métodos y funciones de PostgreSQL no se pueden enviar a los DN para su ejecución. Como resultado, se producen cuellos de botella de rendimiento en los CN.

Explicación:

- Una operación puede ejecutarse concurrentemente sólo cuando es lógicamente una operación concurrente. Por ejemplo, SUM realizada en todos los DN simultáneamente debe centralizar el resumen final en un CN. En este caso, la mayor parte del trabajo de resumen se ha completado en DN, por lo que el trabajo en el CN es relativamente ligero.
- En algunos escenarios, la operación debe ejecutarse centralmente en un nodo. Por ejemplo, la asignación de un nombre global único a un ID de transacción se implementa usando el sistema GTM. Por lo tanto, el GTM es también un componente único a nivel mundial (activo/en espera). Todas las tareas únicas a nivel mundial se implementan a través del GTM en GaussDB(DWS), pero el código de software está optimizado para reducir este tipo de tareas. Por lo tanto, el GTM no tiene muchos cuellos de botella. En algunos escenarios, GTM-Free y GTM-Lite pueden ser implementados.
- Para garantizar un excelente rendimiento, los servicios deben modificarse ligeramente para su adaptación durante la migración desde el modo de desarrollo de aplicaciones de la base de datos tradicional de un solo nodo al de la base de datos paralela, especialmente para el anidamiento de procedimientos almacenados tradicionales de Oracle.

Soluciones:

- Si se produce un problema de este tipo, consulte "Optimización del rendimiento de consultas" en la ***Guía para desarrolladores de Data Warehouse Service (DWS)***.
- Alternativamente, póngase en contacto con el soporte técnico para modificar y optimizar los servicios.

8.3 ¿Cómo puedo ver registros de ejecución de SQL en un cierto período cuando las solicitudes de lectura y escritura están bloqueadas?

Puede utilizar la característica de SQL superior para ver las sentencias de SQL ejecutadas en un período especificado. Se pueden ver las instrucciones de SQL del CN actual o de todos los CN.

TopSQL le permite ver las sentencias de SQL históricas y en tiempo real.

- Para obtener detalles sobre la consulta de sentencia SQL en tiempo real, consulte la sección **TopSQL en tiempo real**.
- Para obtener detalles sobre cómo consultar sentencias de SQL históricas, consulte la sección **TopSQL histórico**.

8.4 What Do I Do If My Cluster Is Unavailable Because of Insufficient Space?

You can use a snapshot to restore your cluster to a new one that has larger storage space, and then delete the old cluster to avoid resource waste. You can learn cluster storage by checking **Available Storage**. To restore your cluster to a new one, see **Restoring a Snapshot to a New Cluster** To delete a cluster, see **Deleting a Cluster**.

NOTA

This method is only supported by the standard data warehouse.

8.5 What is Operator Spilling in GaussDB(DWS)?

During query execution, if the cluster memory is insufficient, the database will choose to store the temporary results to the disk. When the disk storage used by the temporary results exceeds a certain value, users will receive an alarm: "data spilling exceeds the threshold". What does spilling mean?

Operator Spilling to Disks

Any computing consumes memory space. If too much memory is consumed, the memory for running other jobs will be insufficient, causing unstable job running. Therefore, you need to limit the memory usage of query statements to ensure job running stability.

If a job running requires 500 MB memory but only 300 MB memory is allocated to it, data that is not used temporarily needs to be written to disks, and only data that is being used is retained in the memory. This is called data spilling. It is also called **operator spilling**. Large size of data spilled to disk may cause 100% disk usage. If it happens, the database will turn to read-only and query performance will be greatly affected. Therefore, GaussDB(DWS) imposes a limit on operator spilling. If the operator spilling exceeds the limit, an error is reported and the query exits.

Which operators can spill?

Operators that can spill to disk include **Hash(VecHashJoin)**, **Agg(VecAgg)**, **Sort(VecSort)**, **Material(VecMaterial)**, **SetOp(VecSetOp)**, and **WindowAgg(VecWindowAgg)**. They can be vectorized or non-vectorized.

Which parameters can be used to control the operator spilling?

- **work_mem**: sets spilling threshold. Disk usage exceeding this parameter will cause operator spilling. This parameter takes effect only when the memory is not self-adaptive. (**enable_dynamic_workload=off**). It ensures concurrent throughput and the performance of a single query job. Therefore, you need to optimize the parameter based on the output of **Explain Performance**.
- **temp_file_limit**: limits the size of files spilled to disks. You are advised to set this parameter based on the site requirements to prevent spilled files from using up the disk space. Spilled files exceeding the value of this parameter will cause an error.

How Do I Know Whether a Statement Is Spilled to Disks?

- Check the spill files. The spill files are stored in the **base/pgsql_tmp** directory of the instance directory. The spill files are named *pgsql_tmp$queryid_$pid*. You can determine which SQL statement is spilled to the disk based on the **queryid**.
- Check the waiting view (**pgxc_thread_wait_status**). If **write file** is displayed in the waiting view, there are temporary results spilled to disks.
- Check the execution plan (**EXPLAIN PERFORMANCE**). Keywords such as **spill**, **written disk**, and **temp file num** indicates that there is an operator spilling.
- Check whether the **spill_info** column in **Real-time TopSQL** or **Historical TopSQL** contains spill information. If this column is not empty, data has been spilled to disks on DNs. (Prerequisite: The topsql function has been enabled.)

How Do I Avoid Spilling?

When operators are spilled to disks, operator calculation data is written to disks. Compared with memory access, disk operations are slow, causing performance deterioration and query response time deterioration. Therefore, try to avoid operator spilling during query execution. You are advised to use the following methods:

- Reduce the intermediate result set: If the intermediate result set is too large, you can add filter criteria to reduce the size of the intermediate result set.
- Avoid data skew: If there is a severe data skew, spilling will occur on a DN that has a large amount of data.
- Perform **ANALYZE** in a timely manner: When the statistics are inaccurate, the number of rows may be estimated to be smaller. As a result, the plan is not optimal and data is spilled to disks.
- Perform single-point optimization: Perform optimization on single SQL statements.
- If the memory is not self-adaptive, and the intermediate result set cannot be reduced, increase the value of **work_mem** as proper.
- If the memory is self-adaptive, increase the available memory of the database as much as possible to reduce the probability of data spilling.

8.6 GaussDB(DWS) CPU Resource Management

Overview of CPU Resource Management

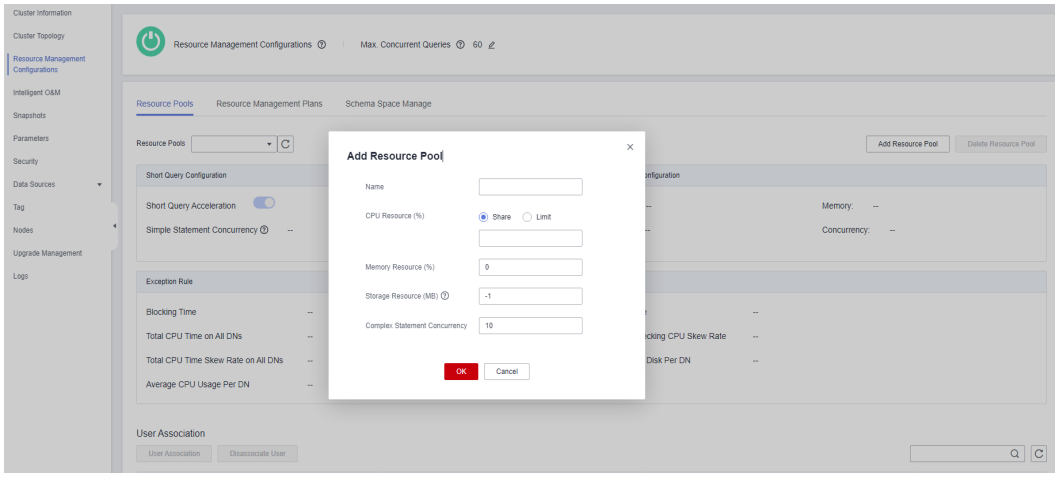
In different service scenarios, system resources (CPU, memory, I/O, and storage resources) of the database are properly allocated to queries to ensure query performance, and service stability.

GaussDB(DWS) provides the resource management function. You can put resources into different resource pools, which are isolated from each other. Then, you can associate database users with these resource pools. When a user starts a SQL query, the query will be transferred to the resource pool associated with the user. You can specify the number of queries that can be concurrently executed in a resource pool, the upper limit of memory used for a single query, and the memory and CPU resources that can be used by a resource pool. In this way, you can limit and isolate the resources occupied by different workloads.

GaussDB(DWS) uses cgroups to manage and control CPU resources, involving the CPU, cpuacct, and cpuset subsystems. CPU share is implemented based on the CPU subsystem

cpu.shares. The advantages of CPU share are as follows: CPU control is not triggered when the OS CPU is not fully occupied. CPU limit is implemented based on cpuset, which is a CPU subsystem used to monitor CPU resource usage.

When adding a resource pool on the GaussDB(DWS) management console, you should choose between **Share** and **Limit**.



CPU Share

CPU Share: Percentage of CPU time that can be used by users associated with the current resource pool to execute jobs.

The share has two meanings:

- **Share**: The CPU is shared by all Cgroups, and other Cgroups can use idle CPU resources.
- **Limit**: When the CPU is fully loaded during peak hours, Cgroups preempt CPU resources based on their limits.

CPU share is implemented based by cpu.shares and takes effect only when the CPU is fully loaded. When the CPU is idle, there is no guarantee that a Cgroup will preempt CPU resources appropriate to its quota. There can still be resource contention when the CPU is idle. Tasks in a Cgroup can use CPU resources without restriction. Although the average CPU usage may not be high, CPU resource contention may still occur at a specific time.

For example, 10 jobs are running on 10 CPUs, and one job is running on each CPU. In this case, any job request for CPU resources will be responded instantly, and there is no contention. If 20 jobs are running on 10 CPUs, the CPU usage may still not be high because the jobs do not always occupy the CPU and may wait for I/O and network resources. The CPU resources seem idle. However, if 2 or more jobs request one CPU at the same time, CPU resource contention occurs, affecting job performance.

CPU Limit

CPU Limit: specifies the percentage of the maximum number of CPU cores that can be used by a database user in the resource pool.

The limit has two meanings:

- Dedicated: The CPU is dedicated to a Cgroup. Other Cgroups cannot use idle CPU resources.

- Quota: Only the CPU resources in the allocated quota can be used. Idle CPU resources of other Cgroups cannot be preempted.

CPU limit is implemented based on cpuset.cpu. You can set a proper quota to implement absolute isolation of CPU resources between Cgroups. In this way, tasks of different Cgroups will not affect each other. However, the absolute CPU isolation will cause idle CPU resources in a Cgroup to be wasted. Therefore, the limit cannot be too large. A larger limit may not bring a better performance.

For example, in one case, 10 jobs are running on 10 CPUs and the average CPU usage is about 5%. In another case, 10 jobs are running on 5 CPUs and the average CPU usage is about 10%. According to the preceding analysis, although the CPU usage is low when 10 jobs run on five CPUs. However, CPU resource contention still exists. Therefore, the performance of running 10 jobs on 10 CPUs is better than that of running 10 jobs on 5 CPUs. However, it is not the more CPUs, the better. If ten jobs run on 20 CPUs, at any time point, at least 10 CPUs are idle. Therefore, theoretically, running 10 jobs on 20 CPUs does not have better performance than running 10 CPUs. For a Cgroup with a concurrency of N, if the number of allocated CPUs is less than N, the job performance is better with more CPUs. However, if the number of allocated CPUs is greater than N, the job performance will not be improved with more CPUs.

Application Scenarios of CPU Resource Management

The CPU limit and CPU share both have their own advantages and disadvantages. CPU share can fully utilize CPU resources. However, resources of different Cgroups are not completely isolated, which may affect the query performance. CPU limit can implement absolute isolation of CPU resources. However, idle CPU resources will be wasted. Compared with CPU limit, CPU share has higher CPU usage and overall job throughput. Compared with CPU share, CPU limit has complete CPU isolation, which can better meet the requirements of performance-sensitive users.

If CPU contention occurs when multiple types of jobs are running in the database system, you can select different CPU resource control modes based on different scenarios.

- Scenario 1: Fully utilize CPU resources. Focus on the overall CPU throughput instead of the performance of a single type of jobs.

  Suggestion: You are not advised to isolate CPUs between users. No matter which type of CPU control is implemented, the overall CPU usage is affected.
- Scenario 2: A certain degree of CPU resource contention and performance loss are allowed. When the CPU is idle, the CPU resources are fully utilized. When the CPU is fully loaded, each service type needs to use the CPU proportionally.

  Suggestion: You can use CPU share to improve the overall CPU usage while implementing CPU isolation and control when the CPUs are fully loaded.
- Scenario 3: Some jobs are sensitive to performance and CPU resource waste is allowed.

  Suggestion: You can use CPU limit to implement absolute CPU isolation between different types of jobs.

8.7 Why the Tasks Executed by an Ordinary User Are Slower Than That Executed by the dbadmin User?

The execution speed of an ordinary user is slower than that of the dbadmin user in the following scenarios:

Scenario 1: Ordinary users are subject to resource management.

Ordinary users queuing: **waiting in queue/waiting in global queue/waiting in ccn queue**

1. Ordinary users will be **waiting in queue/waiting in global queue** when the number of active statements exceeds the value of **max_active_statements**. While administrators do not need to queue.

   You can increase the value of this parameter or clear some statements to avoid queuing.

   Change the value of **max_active_statements** on the management console.

   a. Log in to the GaussDB(DWS) management console.

   b. In the navigation pane on the left, click **Clusters**.

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

   d. Go to the **Parameter Modifications** page of the cluster, search for the **max_active_statements** parameter, change its value, and click **Save**.

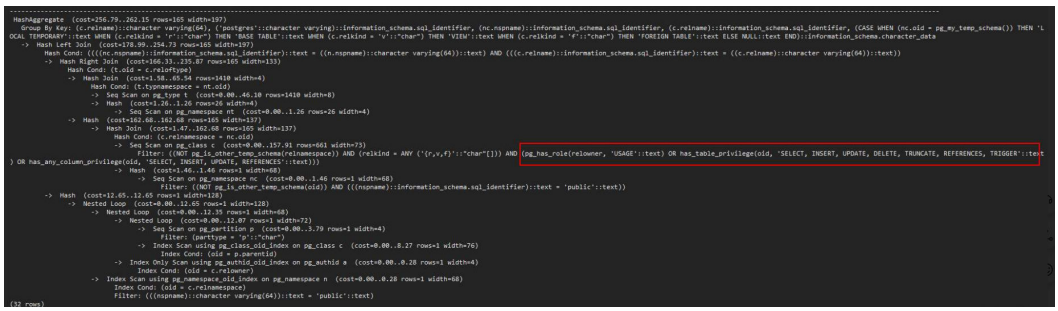
2. It takes a long time for ordinary users to wait in the ccn queue. When dynamic resource management is enabled (**enable_dynamic_workload** is set to **on**), if the concurrency is high and the available memory is small, ordinary users may get into this state when executing statements. Administrators are not controlled. You can stop some statements or increase the memory parameter value. If the memory usage of each DN is not high, you can disable the dynamic resource management parameter **enable_dynamic_workload** by setting it to **off**.

Scenario 2: The OR condition in the execution plan checks the statements executed by common users one by one. This consumes a lot of time.

The **OR** conditions in the execution plans contain permission-related checks. This scenario usually occurs when the system view is used. For example, in the following SQL statement:

```
SELECT distinct(dtp.table_name),
 ta.table_catalog,
 ta.table_schema,
 ta.table_name,
 ta.table_type
from information_schema.tables ta left outer join DBA_TAB_PARTITIONS dtp
on (dtp.schema = ta.table_schema and dtp.table_name = ta.table_name)
where ta.table_schema = 'public';
```

Part of the execution plan is as follows:



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.

While the **OR** conditions of an ordinary user need to be checked one by one. If there are a large number of tables in the database, the execution time of the ordinary user is longer than that of the **dbadmin** user.

In this scenario, if the number of output result sets is small, you can set **set enable_hashjoin** and **enable_seqscan** to **off**, to use the index+nestloop plan.

Scenario 3: The resource pools allocated to ordinary users and administrators are different.

Run the following command to check whether the resource pools corresponding to an ordinary user are the same as that of the administrator user. If they are different, check whether the tenant resources allocated to the two users are different.

```
SELECt * FROM pg_user;
```

8.8 What Are the Factors Related to the Single-Table Query Performance in GaussDB(DWS)?

GaussDB(DWS) uses the shared-nothing architecture, and data is stored in a distributed manner. Therefore, the distribution key, data volume, and number of partitions affect the overall query performance of a single table.

1. Distribution Key Design

   By default, GaussDB(DWS) takes the first column of the primary key as the distribution key. When you define both a primary key and a distribution key for a table, the distribution key must be a subset of the primary key. Distribution keys determine data distribution among partitions. If distribution keys are well distributed among partitions, query performance can be improved.

   If the distribution key is incorrectly selected, data skew may occur after data is imported. The usage of some disks may be much higher than that of other disks, and the cluster may become read-only in some extreme cases. Proper selection of distribution keys is critical to table query performance. In addition, proper distribution keys enable data indexes to be created and maintained more quickly.

2. Data Volume Stored in a Single Table

   The larger the amount of data stored in a single table, the poorer the query performance. If a table contains a large amount of data, you need to store the data in partitions. To convert an ordinary table to a partitioned table, you need to create a partitioned table and import data to it from the ordinary table. When you design tables, plan whether to use partitioned tables based on service requirements.

   To partition a table, comply with the following principles:

   – Use fields with obvious ranges for partitioning, for example, date or region.
   – The partition name must reflect the data characteristics of the partition. For example, its format can be Keyword+Range characteristics.
   – Set the upper limit of a partition to **MAXVALUE** to prevent data overflow.

3. Number of Partitions

Tables and indexes can be divided into smaller and easier-to-manage units. This significantly reduces search space and improves access performance.

The number of partitions affects the query performance. If the number of partitions is too small, the query performance may deteriorate.

GaussDB(DWS) supports range partitioning and list partitioning. In range partitioning, records are divided and inserted into multiple partitions of a table. Each partition stores data of a specific range (ranges in different partitions do not overlap). List partitioning is supported only by clusters of 8.1.3 and later versions.

When designing a data warehouse, you need to consider these factors and perform experiments to determine the optimal design scheme.

9 Copia de respaldo y restauración de instantáneas

9.1 ¿Por qué la creación de una instantánea automatizada es tan lenta?

Esto ocurre cuando los datos que se van a hacer una copia de seguridad son grandes. Las instantáneas automatizadas son copias de seguridad incrementales, y cuanto menor sea la frecuencia establecida (por ejemplo, una semana), más tiempo tardará. Aumente la frecuencia de backup para acelerar el proceso.

En la siguiente tabla se enumeran las tasas de copia de respaldo y restauración de instantáneas. (Las tasas se obtienen del entorno de pruebas de laboratorio con SSD locales como medio de copia de respaldo. Las tarifas son solo de referencia. La velocidad real depende de los recursos de disco, red y ancho de banda.)

- Velocidad de copia de respaldo: 200 MB/s/DN
- Tasa de restauración: 125 MB/s/DN

9.2 Does a DWS Snapshot Have the Same Function as an EVS Snapshot?

No.

GaussDB(DWS) snapshots are used to restore all the configurations and service data of a cluster. EVS snapshots are used to restore the service data of a data disk or system disk within a specific time period.

DWS Snapshot

A GaussDB(DWS) snapshot is a full or incremental backup of a GaussDB(DWS) cluster at a specific point in time. It records the current database data and cluster information, including the number of nodes, node specifications, and administrator name. Snapshots can be created manually or automatically.

When a snapshot is used for restoration, GaussDB(DWS) creates a new cluster based on the cluster information recorded in the snapshot and restores data from the snapshot.

For more information about snapshots, see **Managing Snapshots**.

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 more information about snapshots, see **EVS Snapshots**.

10 Facturación

10.1 How Do I Renew the Service?

About Renewal

Currently, GaussDB(DWS) supports the pay-per-use billing mode and yearly/monthly billing mode.

- In yearly/monthly mode, you pay only once when purchasing the service and no extra fees will be charged during your use of the service. After a yearly/monthly subscription expires, the resources will enter a retention period. If you want to continue using the resources, renew the subscription. For details, see **Resource Expiration Description**.
- Pay-per-use mode: The system settles the fees by hour. You can use the service as long as your account balance is sufficient. If your account balance is insufficient, it will be in arrears. Top up your account in a timely manner.

Renewal Modes

Pay-per-use

To top up your account, perform the following steps:

Paso 1 Log in to the management console.

Paso 2 Choose **Billing Center > Renewal** in the upper right of the page.

Paso 3 In the navigation pane on the left, click **Overview**. Then click **Top Up** so your balance is sufficient to continue services.

----Fin

Yearly/Monthly

Perform the following operations to renew your account:

Paso 1 Log in to the management console.

Paso 2 Click **Service List** on the left and go to the **Data Warehouse Service** page.

Paso 3 On the **Clusters** page, locate the target cluster and click **Renew**.

Paso 4 Complete the renewal as prompted.

----Fin

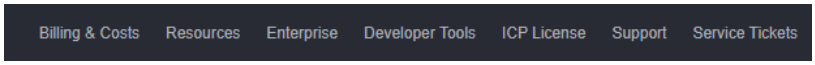
10.2 Is Refund Supported?

Yes. GaussDB(DWS) support 5-day full refund. Within the first 5 days of purchase, you can request a full refund. The full payment will be refunded, excluding the used cash coupons and discount coupons. Each account can request a 5-day full refund for up to 10 times within a calendar year (starting from January 1).

Unsubscribing from a Discount Package

Paso 1 Log in to the management console.

Paso 2 In the upper right corner, click **Billing & Costs**.



Paso 3 In the navigation tree on the left, choose **Orders > Unsubscriptions**.

Paso 4 Find the resource in the list, click **Unsubscribe from Resources** in the row where the resource is located, and unsubscribe from the resource as prompted.

For details, go to **Unsubscription Rules**.

----Fin

10.3 How Am I Billed for Scheduled Synchronization of GaussDB(DWS) Data to a PostgreSQL Database?

You can choose either of the following methods to periodically synchronize GaussDB(DWS) data to a PostgreSQL database.

- Use GaussDB(DWS) to export data to OBS, and then import the data from OBS to the PostgreSQL database to implement data synchronization. In this process, fees are generated only when data is stored in OBS. For details about the billing of OBS data storage, see **Object Storage Service Pricing Pricing Details**.

10.4 How Can I Try Out GaussDB(DWS) for Free?

Only new users can participate in the free trial campaign. If your account has never created a data warehouse cluster but passed the real-name authentication, you are eligible for a one-month free trial of GaussDB(DWS).

To apply for the free trial, log in to the GaussDB(DWS) management console and click **Apply for free trial**. Free trial packages cannot be used across regions. Select the region based on your requirements.

After subscribing to a free trial package, you can log in to the GaussDB(DWS) management console to create a cluster with the corresponding region, node flavor, and node quantity

within the free trial period. The system will not bill you for the created cluster. If you choose to use other node types, you will be charged at the standard pay-per-use rate. For details, see **GaussDB (DWS) Pricing Details**.

When the one-month free trial ends, you can delete the cluster to avoid fees. You can also retain the cluster, but you will be charged at the regular pay-per-use rate.

10.5 ¿Por qué se me dedujeron las tarifas después de que venciera mi prueba gratuita de GaussDB(DWS)?

Como se indica en la página de actividad de GaussDB(DWS), el clúster que creó no se liberará automáticamente después de que expire la prueba gratuita. Si no elimina el clúster manualmente, el sistema deducirá la tarifa por el clúster.

Para evitar cargos adicionales, inicie sesión en la consola de gestión de GaussDB(DWS) y vaya a la página de **Clusters** para eliminar su clúster si no desea usarlo después de que finalice el período de prueba. Si el clúster tiene una EIP, seleccione **Release the EIP bound to the cluster**. en el cuadro de diálogo **Delete Cluster**. Si no libera la EIP, se facturará según la regla de precios de la EIP de VPC.

10.6 ¿Por qué no puedo ver un clúster después de suscribirme a una prueba gratuita de GaussDB(DWS)?

El sistema no crea automáticamente un clúster después de suscribirse a una prueba gratuita. Inicie sesión en la consola de gestión de GaussDB(DWS) para crear una manualmente.

10.7 How Can I Stop GaussDB(DWS) Billing?

The following describes how to stop billing in pay-per-use and yearly/monthly.

- **Pay-per-use**

  If you no longer use a GaussDB(DWS) cluster that is in pay-per-use mode, delete it and its resources to stop the billing. To do so, log in to the GaussDB(DWS) management console and go to the **Clusters** page. The following table lists the billable items of GaussDB(DWS).

  Tabla 10-1 GaussDB(DWS) billable items

Billing Item	Description
Data warehouse node	The billing will be stopped after the cluster is deleted. For details about how to delete a cluster, see **Deleting a Cluster**.

Billing Item	Description
Snapshot storage space	When a cluster is deleted, its automated snapshots are deleted, but its manual snapshots are retained. After you delete the manual snapshots, the billing is stopped. To delete the manual snapshots, log in to the GaussDB(DWS) management console and go to the **Clusters** page. If the manual snapshots are still retained after the cluster is deleted and the total snapshot size exceeds the free space, the excess part will be billed based on the OBS billing rate.
(Optional) EIP and bandwidth	If a cluster has an EIP, select **Release the EIP bound to the cluster.** when deleting the cluster. The billing will be stopped after the EIP is released. If you do not release the EIP, it will be billed as per VPC EIP pricing rules.
(Optional) DEK	If you have enabled the **Encrypt DataStore** function for a cluster and purchased a key on DEW, the key is not deleted after you delete the cluster. Manually unsubscribe from and delete the key to stop billing. To do this, log in to the DEW management console and choose **Data Encryption Workshop > Key Pair Service**.

- Yearly/Monthly package

  GaussDB(DWS) clusters billed in yearly/monthly mode support unconditional unsubscription within five days from your purchase. For details, see **Yearly/Monthly Unsubscription**. If you do not unsubscribe from the package, the system does not refund the fees.

10.8 Does Pay-per-Use Billing Stop When My Cluster Stops?

No. The system settles the fees by hour for the pay-per-use mode. You can use the service as long as your account balance is sufficient. To reduce costs:

- Delete the purchased clusters if you will not be needing them, and create new ones when needed.
- Switch to the yearly/monthly mode that allows you to use the service within the specific period of time without additional fees.

10.9 ¿Por qué el botón de compra no está disponible cuando creo un clúster?

Las posibles causas del botón de compra no disponible durante la creación del clúster son las siguientes:

- La variante a comprar podría estar agotada, o la región no tiene tal variante.

  Solución: Antes de comprar, compruebe si la variante está disponible en la región o compre un paquete después de crear el clúster. Después del paquete, el paquete se asocia automáticamente con el clúster.

- La cuenta podría estar en mora o estar restringida, por lo que no se pueden crear nuevos recursos.

  Solución: Si su saldo es insuficiente, recargue la cuenta para cancelar la cantidad vencida.

10.10 How Do I Unfreeze a Cluster?

Cause Analysis

If your account balance is insufficient and fee deduction fails, the retention period starts. During the retention period, the service resources will be frozen and cannot be used, but resources and data are reserved.

Handling Procedure

To unfreeze a cluster, you need to top up your account to ensure that the account balance is not **0**. For details, see **How Do I Renew the Service?** After a cluster is unfrozen, the cluster status changes to **Available**.

10.11 ¿Puedo congelar o cerrar un clúster de GaussDB(DWS) para detener la facturación?

No. Si no necesita un clúster, elimínelo y sus recursos.

