Database and Application Migration UGO(UGO) 25.1.0

FAQs

Issue   01

Date   2025-01-30

Contents

1 Product Consulting

1.1 Instance Freezing, Release, and Deletion

Why Are My UGO Resources Released?

If your subscriptions have expired but not been renewed, or you are in arrears due to insufficient balance, your resources enter a grace period. If you still do not complete the payment or renewal after the grace period expires, you will enter a retention period. During the retention period, the resources are not available. If you still do not renew them or top up your account after the retention period ends, your resources will be released and your data stored will be deleted. For details, see **Service Suspension and Resource Release**.

Why Are My UGO Resources Frozen?

Your UGO resources may be frozen for a variety of reasons. The most common reason is that you are in arrears.

How Do I Unfreeze My Resources?

Frozen due to arrears: You can renew your resources or top up your account.

The frozen resources can be renewed, released, or deleted.

What Impacts Does Instance Freezing, Unfreezing or Release Have on My Services?

- After the UGO resources are frozen:
  - You can only view existing projects, and are not allowed to modify, delete, create projects or convert SQL statements.
  - You can manually delete UGO.
- After your resources are unfrozen, you can perform operations such as creation.
- After your resources are released, UGO will be deleted.

How Do I Renew My Resources?

After UGO expires, you can renew it on the **Renewals** page. For details, see **Renewal Management**.

Can My Resources Be Recovered After Being Released?

Deleted resources cannot be restored.

1.2 Which Schemas in Source Databases Are Ignored for Migration?

Oracle as the Source

Schemas that are not collected:

APEX_050000, AUDSYS, ADAMS, ANONYMOUS,

AURORAORBUNAUTHENTICATED, AWR_STAGE, APEX_030200, APEX_040200,

APEX_PUBLIC_USER, APPQOSSYS, BLAKE, CLARK, CSMIG,

CTXSYS, DBSNMP, DIP, DMSYS,

DSSYS, DEMO, DVSYS, DVF,

DBSFWUSER, EXFSYS, FLOWS_FILES,

GGSYS, GSMADMIN_INTERNAL, GSMCATUSER, GSMUSER

JONES, LBACSYS, MDDATA, MDSYS, MGMT_VIEW

OLAPSYS, ORACLE_OCM, ORDDATA, ORDPLUGINS

ORDSYS, OUTLN, OWBSYS, OWBSYS_AUDIT, OJVMSYS,

PERFSTAT, REMOTE_SCHEDULER_AGENT

SI_INFORMTN_SCHEMA, SPATIAL_CSW_ADMIN_USR,

SPATIAL_WFS_ADMIN_USR, SYS, SYSMAN, SPATIAL_CSW_ADMIN_USR, SYSBACKUP,

SYSKM, SYSDG, SYSRAC, SYS$UMF, SYSTEM,

TRACESVR, TSMSYS, WMSYS, XDB, XS$NULL, and GSMROOTUSER.

UGO does not collect the following tables:

- System tables, for example, SYS_EXPORT_SCHEMA_ %
- System objects starting with SYS_PLSQL are not collected.
- Deleted objects starting with BIN$ are not collected.
- Nested tables are not collected.
- Tables that are automatically generated by data pump jobs and recorded in the DBA_DATAPUMP_JOBS or USER_DATAPUMP_JOBS view are not collected.

MySQL as the Source

INFORMATION_SCHEMA, MYSQL, PERFORMANCE_SCHEMA, and SYS

PostgreSQL as the Source Database

information_schema and a system schema prefixed with **pg_**, for example, pg_catalog, pg_toast, or pg_temp_1

Microsoft SQL Server as the Source

GUEST, INFORMATION_SCHEMA, SYS, DB_OWNER, DB_ACCESSADMIN, DB_SECURITYADMIN, DB_DDLADMIN, DB_BACKUPOPERATOR, DB_DATAREADER, DB_DATAWRITER, DB_DENYDATAREADER, and DB_DENYDATAWRITER

1.3 What Are the Differences Between Syntax Conversion and Migration & Verification in a Migration Project?

Syntax conversion is the conversion of SQL scripts of source database objects into SQL scripts of the target database objects. The SQL scripts are not executed in the target database.

Migration & Verification is the sending of converted SQL statements to the target database for execution.

1.4 What Are the Database Schema Changes After an Oracle Database Is Migrated to the Target Database?

- The Oracle schemas are converted into the PostgreSQL schemas.
- The Oracle schemas are converted into the MySQL databases.

1.5 Why Cannot I Use Some Functions?

- You do not have the required permissions.

  For details about common UGO operations and corresponding actions, see **Permissions Management**.

  For details about how to apply for permissions, see **Creating a User and Granting Permissions**.

- Your account is frozen or restricted.

  You can only view existing projects, and are not allowed to modify, delete, create projects or convert SQL statements.

  In this case, you can click **Submit a service Ticket** as prompted. Alternatively, contact the customer service by choosing **Service Tickets > Create Service Ticket** in the upper right corner of the console.

1.6 What Is the Function of the dsc_ora_ext Schema Generated After Migration to the Target Database?

dsc_ora_ext is a custom schema compiled by UGO to enable the target database to implement certain functions in the Oracle source database.

If the source database is MySQL, the custom schema is **dsc_mys_ext**.

For example, UGO creates custom target database functions that provide the same functions as Oracle system functions, and it converts Oracle system function calls to custom function calls during migration.

1.7 What Should I Do If Data Collection Fails or Is Slow Due to the Small Values of Certain Oracle SGA Parameters?

If data collection fails or is slow because certain parameters in the Oracle system global area (SGA) are set to small values, run the following command:

show parameter SGA;

Command output:

```
NAME                          TYPE        VALUE
---------------------------- ----------- -------
sga_max_size                  big integer 796M
sga_target                    big integer 0
```

Contact the database administrator to increase the values of the SGA parameters.

ALTER SYSTEM SET sga_max_size=5G scope=spfile;

ALTER SYSTEM SET sga_target=5G scope=spfile;

📖 **NOTE**

The preceding parameter values are for reference only. If their value is too large, more resources are occupied. Contact the database administrator to set the parameters to appropriate values.

For details about the parameters, see the **Oracle official documentation**.

1.8 What Should I Do If Data Collection Fails and a Message SNAPSHOT TOO OLD Is Displayed?

If the collection fails or the error message "ORA-01555: snapshot too old" is displayed, check the UNDO parameters of the source Oracle database. Run the following command:

show parameter undo;

Command output:

```
NAME TYPE VALUE
undo_management string AUTO
undo_retention integer 28800
```

Contact the database administrator to increase the UNDO_RETENTION value.

ALTER SYSTEM SET UNDO_RETENTION =*N*;

Replace *N* with an appropriate value.

1.9 What Is the Mapping Between the GaussDB Versions Displayed on the UGO Console and Self-built GaussDB Versions?

Log in to your self-built GaussDB database and run the following command to query the version:

select version ();

Table 1-1 describes the mapping between self-built GaussDB database versions and the GaussDB versions displayed on the UGO console.

Table 1-1 Version mapping

Self-built GaussDB Version	GaussDB Version on the UGO Console
V500R002C10	GaussDB V2.0-2.7 Enterprise Edition
503.0.x	GaussDB V2.0-3.1 Enterprise Edition
503.1.x	GaussDB V2.0-3.2 Enterprise Edition
503.2.x	GaussDB V2.0-3.3 Enterprise Edition
505.0	GaussDB V2.0-8.0 Enterprise Edition
505.1	GaussDB V2.0-8.100 Enterprise Edition
505.2	GaussDB V2.0-8.200 Enterprise Edition
506.0	GaussDB V2.0-9.0 Enterprise Edition

1.10 What Are VPCEP Resource Quotas?

VPCEP resource quotas

When you create an object migration task, UGO uses the VPC Endpoint (VPCEP) service to connect to a database instance and perform the migration.

By default, you can create a maximum of 20 VPC endpoint services in a region and purchase a maximum of 50 VPC endpoint services in a region. If the existing resource quotas cannot meet your service requirements, you can request higher quotas.

Increasing Quotas

Step 1 Log in to the management console.

Step 2 In the upper right corner of the page, choose **Resources** > **My Quotas**. On the **Quotas** page, click **Increase Quota**.

Figure 1-1 Increasing quotas

Step 3 On the **Create Service Ticket** page, configure parameters as required. In the **Problem Description** area, enter the required quota and reason for the quota adjustment.

Figure 1-2 Creating a service ticket

Step 4 Agree to the agreement and click **Submit**.

----End

2 Database Connections

2.1 What Should I Do If I Cannot Connect to the Source Oracle Database During Database Evaluation Project Creation?

Check:

- Whether the network between the source database and UGO is connected. Currently, UGO can connect to the source database through a public network.
- Whether the network is stable (including the packet loss rate and delay). If the network quality of the source database is poor, the evaluation project may fail.
- Whether UGO is allowed by the source database firewall settings.
- Whether the database connection information is correct.

2.2 What Should I Do If I Failed To Connect to the Source Database as User sys?

User **sys** is the super administrator of Oracle databases. Generally, the Oracle database restricts the remote login of user **sys**. You are advised to connect to the source database as other database users.

2.3 How Do I Create GaussDB Databases Compatible with Source Databases?

Ensure that the compatibility mode used during database creation is the same as the GaussDB compatibility mode selected and confirmed during database evaluation.

Creating a GaussDB Database Compatible with Oracle

The automated conversion solution of UGO for migration from Oracle, PostgreSQL, and SQL Server to GaussDB is actually designed for migration to GaussDB compatible with Oracle. Perform the following steps to create a GaussDB database compatible with Oracle:

Step 1 Log in to your GaussDB instance as a user who has the permission to create databases.

Step 2 Create a GaussDB database compatible with Oracle.

Centralized:

create database *databasename* dbcompatibility = 'A';

Distributed:

create database *databasename* dbcompatibility = 'ORA';

Step 3 Check whether the database is created.

select * from pg_database where datname = '*databasename*';

- Centralized: If the value of **datcompatibility** is **A**, the database is created.
- Distributed: If the value of **datcompatibility** is **ORA**, the database is created.

----End

📖 **NOTE**

UGO does not support migration from PostgreSQL to GaussDB (PostgreSQL-compatible mode).

Creating a GaussDB Database Compatible with MySQL

If the source database type is MySQL or GoldenDB, perform the following steps to create a GaussDB database compatible with MySQL:

Step 1 Log in to your GaussDB instance as a user who has the permission to create databases.

Step 2 Create a GaussDB database compatible with MySQL.

GaussDB Centralized (B-compatible mode):

create database *databasename* dbcompatibility = 'B';

GaussDB Centralized (M-compatible mode):

create database *databasename* dbcompatibility = 'M';

Distributed:

create database *databasename* dbcompatibility = 'MySQL';

Step 3 Check whether the database is created.

select * from pg_database where datname = '*databasename*';

- GaussDB Centralized (B-compatible mode): If the value of **datcompatibility** is **B**, the database is created.

- GaussDB Centralized (M-compatible mode): If the value of **datcompatibility** is **M**, the database is created.
- GaussDB Distributed: If the value of **datcompatibility** is **MySQL**, the database is created.

----End

📖 **NOTE**

For migration from GoldenDB to GaussDB Centralized, only GaussDB Centralized (B-compatible mode) is supported.

2.4 What Should I Do If My Database Fails to be Connected?

Symptom

When UGO connected to a database, an error message was displayed, indicating that the connection failed.

Figure 2-1 Error message
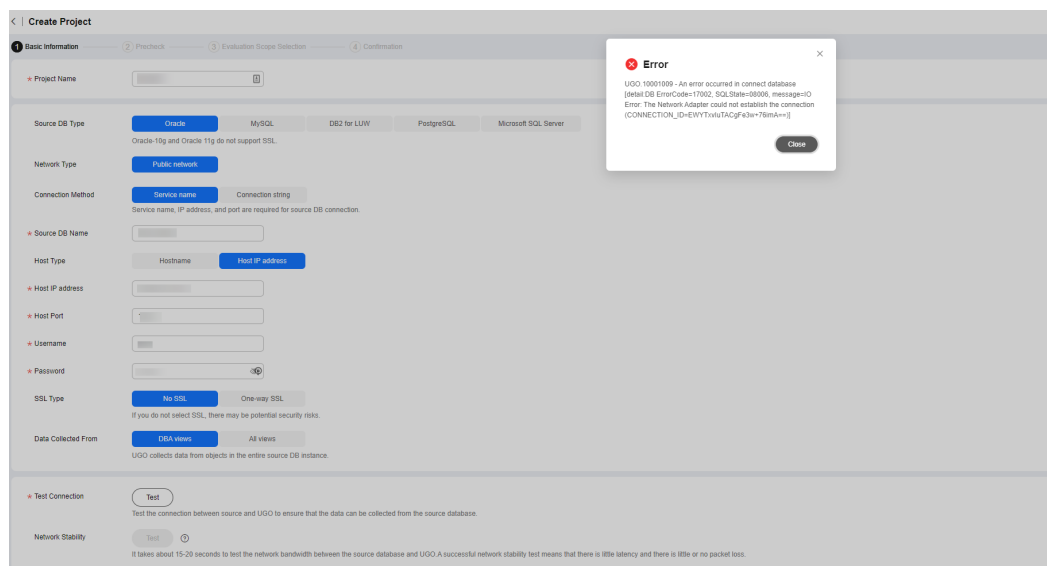
Possible Causes

1. The database connection information is incorrect.
2. The database user does not have permissions required to connect to the database.
3. The network between the database and UGO is disconnected.
4. The database firewall restricts UGO access to the database.
5. There are too many connections to the database.

Solution

1. Correct the database connection information.

2. Grant the connection permissions to the user.

3. Ensure that the network between the database and UGO is connected.

4. Allow UGO to access the database.

5. Adjust the maximum number of connections for the database.

2.5 SQL Server Connection Test Failed (Error Code=0, SQLState=08S01)

Symptom

When the source database type was Microsoft SQL Server and SSL was disabled, the connection test failed and message "ErrorCode=0, SQLState=08S01" was displayed. In addition, error message " Caused by: The server selected protocol version TLS10 is not accepted by client preferences [TLS13, TLS12]" was displayed in the log.

Causes

Even if SSL is disabled, a certificate can be generated over an underlying protocol of SQL Server to encrypt a login data packet. TLS 1.0 may be used in earlier versions but is disabled by default for security purposes in the current JRE.

Solution

Modifying JVM Configurations

Step 1 Log in to a UGO server as the **root** user.

Step 2 Confirm the JRE path (*JAVA_HOME*).

Step 3 Modify the *<JAVA_HOME>*/**lib/security/java.security** file.

vi <JAVA_HOME>/lib/security/java.security

Step 4 Delete TLS v1 from the **jdk.tls.disabledAlgorithms** configuration item and save the modification.

Figure 2-2 Deleting TLS v1

Step 5 Restart UGO (*<UGO_HOME>* indicates the UGO installation directory).

python3 <UGO_HOME>/bin/ugoserver.py restart

----End

3 Evaluation Project

3.1 How Do I Select a Connection Method?

- **Service Name**: You need to specify the source database name, host IP address or host name, username and other parameters. UGO constructs a connection string based on these parameters.

- **Connection String**: JDBC URL, which must comply with the format specifications of the source database. You also need to specify database name and host name or IP address.

3.2 How Does UGO Collect Data from Source Databases?

Oracle as the Source Database Type

Unless otherwise specified, DDL information about Oracle objects is obtained using the **DBMS_METADATA.get_ddl system** package function.

The basic information about an object is obtained from system views, which consists of **DBA views** and **All views**. The following uses the **All views** as an example.

INDEX: During index collection, all constraint information is queried from the **ALL_INDEXES** view, filter out the indexes that are created based on primary keys or unique key constraints. (If an index is created before a constraint is created and the constraint name is the same as the index name, the index will not be collected.)

PostgreSQL as the Source Database Type

UGO uses the following method to collect each object of the source database.

- **SCHEMA**: Run **CREATE SCHEMA** *schema name*;

- **TABLE**: Query the **pg_catalog.pg_attribute**, **pg_catalog.pg_class**, **pg_catalog.pg_namespace**, and **pg_inherits** system views to obtain the table

name, field name, field type, and table partition information, and run the **CREATE TABLE** statement. Information about generated columns, table constraints, and foreign key constraints is not collected.

- **INDEX**: Query the **pg_indexes** view to obtain the indexdef field and run the **CREATE INDEX** statement.

- **VIEW**: Query the **pg_views** view to obtain the definition field and run the **CREATE VIEW** statement.

- **FUNCTION/PROCEDURE**: Use **pg_get_functiondef** to obtain the **CREATE FUNCTION** and **CREATE PROCEDURE** statements.

  Note that **pg_get_functiondef** cannot process aggregate functions. User-defined aggregate functions cannot be collected.

- **TRIGGER**: Use **pg_get_functiondef** to obtain the trigger function definition, use **pg_get_triggerdef** to obtain the **CREATE TRIGGER** statement, and run the **CREATE TRIGGER** statement.

There are no specific methods for object DDL collection in PostgreSQL. The DDLs of some objects are obtained by querying metadata. As a result, the collected DDL information may be inconsistent with the DDL information in the source database, or even the information may be missing.

If the collected DDLs cannot meet your requirements, you can use pg_dump or a third-party database client tool to obtain object DDLs, and then **convert the SQL statements**.

3.3 How Does UGO Check Database Permissions?

3.3.1 Oracle as the Source

Check Item	Description	Suggestion
DBMS_MET ADATA	Permission to retrieve metadata from the Oracle database dictionary. This permission is used to obtain the DDL of schema objects.	Statements: **GRANT SELECT_CATALOG_ROLE TO** *<user>*; **GRANT SELECT ANY DICTIONARY TO** *<user>*; or **GRANT DBA TO** *<user>*; If the role has been granted but the pre-check fails, run the following statement to check whether the role is DEFAULT_ROLE of the user: **SELECT GRANTED_ROLE, DEFAULT_ROLE FROM DBA_ROLE_PRIVS WHERE GRANTEE='***<user>***' AND GRANTED_ROLE IN ('SELECT_CATALOG_ROLE', 'DBA');** If the query result is **NO**, run the following statement to grant the role: **ALTER USER** *<user>* **DEFAULT ROLE SELECT_CATALOG_ROLE;** or **ALTER USER** *<user>* **DEFAULT ROLE DBA;**
Dynamic View	Permission to access various dynamic performance views. This permission is used to obtain basic database information.	DBA permission needs to be granted.
DDL Object Count Check	The system checks whether a user has the permission to access at least one schema where DDL statements need to be evaluated.	Ensure that at least one schema object where DDL statements need to be evaluated can be accessed, or the evaluation task cannot be created. Note: Schema objects in Oracle are not collected.

Check Item	Description	Suggestion
DBMS_MET ADATA SQL Formatting Parameters	Permission to check whether the check result of **DBMS_METADATA SQL Formatting Parameters** is **Alarm**. If so, the evaluation project can be created, but the collected SQL format may be incorrect. As a result, the evaluation and conversion fail.	Check whether the **DBMS_METADATA.SET_TRANSFORM_PA RAM(dbms_metadata.SESSION_TRANSF ORM, \\'CONSTRAINTS_AS_ALTER\\', false)** and **DBMS_METADATA.SET_TRANSFORM_PA RAM(dbms_metadata.SESSION_TRANSF ORM, \\'SQLTERMINATOR\\', true)** statements can be executed.

3.3.2 MySQL as the Source

Check Item	Description	Suggestion
MySQL database permission	By default, when UGO connects to a MySQL database, the user needs to obtain permissions to access the database.	Create a user. **CREATE USER *<user>* IDENTIFIEDBY passwd;** **GRANT SELECT ON mysql.* TO *<user>*;**
DDL Object Count Check	The system checks whether a user has the permission to access at least one schema where DDL statements need to be evaluated.	Ensure that at least one schema object where DDL statements need to be evaluated can be accessed, or the evaluation task cannot be created. Note: MySQL database objects are not checked.
Process	Permission to view all tables in information_schema.	Run the following statement: **GRANT PROCESS ON *.* TO *<user>*;**

Check Item	Description	Suggestion
Object collection	Permission to collect objects	Run the following statements: **GRANT SELECT ON** *<schema_name>*.*** **TO** *<user>*; **GRANT SHOW VIEW ON** *<schema_name>*.*** **TO** *<user>*; **GRANT TRIGGER ON** *<schema_name>*.*** **TO** *<user>*; NOTE To collect global objects, replace *<schema_name>.** with *.*.
Stored procedure and function	In MySQL 8.0.20 and later versions, if there are stored procedures and functions in the source database, grant the user **SHOW_ROUTINE** permission.	Run the following statement: **GRANT SHOW_ROUTINE ON *.* TO** *<user>*;

3.3.3 GoldenDB as the Source

Check Item	Description	Suggestion
GoldenDB system table mysql.user	Permission to collect information about USER and ROLE objects.	1. Create a user. **CREATE USER** *<user>* **IDENTIFIEDBY passwd;** **GRANT SELECT ON mysql.user TO** *<user>*;
Process	Permission to view all tables in information_schema.	Run the following statement: **GRANT PROCESS ON *.* TO db-user;**

Check Item	Description	Suggestion
Object collection	Permission to collect objects	Run the following statement: **GRANT SELECT ON** *<schema_name>*.*** **TO** *<user>*; **GRANT SHOW VIEW ON** *<schema_name>*.*** **TO** *<user>*; **GRANT TRIGGER ON** *<schema_name>*.*** **TO** *<user>*; NOTE ● To collect global objects, replace *<schema_name>.** with *.**. ● By default, the user created on any GoldenDB CN node can connect to all databases. To collect database objects, the user needs to be granted to the SELECT permission.
Stored procedure and function	In MySQL 8.0.20 and later versions, if there are stored procedures and functions in the source database, grant the user **SHOW_ROUTINE** permission.	Run the following statement: **GRANT SHOW_ROUTINE ON *.* TO** *<user>*;

3.3.4 Microsoft SQL Server as the Source

Table 3-1 Check items for source database Microsoft SQL Server

Check Item	Description	Suggestion
DDL query	Permission to obtain database objects	Grant the permission to obtain database objects to the user. Run the following statement: **GRANT SELECT ON OBJECT ;**
DDL Object Count Check	Permission to access at least one schema object for which DDL statements need to be evaluated	If a user does not have the permission to access tables, run the following statement to grant the View Definition permission: **GRANT VIEW DEFINITION ON DATABASE ::** *<database>* **TO** *<user>*; In addition, you must have at least one schema object. Otherwise, the evaluation project cannot be created. The number of objects in schemas in a SQL Server database is not counted.

Check Item	Description	Suggestion
View Definition	Whether a user has the View Definition permission	A user without the the View Definition permission cannot query metadata. Run the following statement to grant the permission: **GRANT VIEW DEFINITION ON DATABASE :: *<database>* TO *<user>*;**

3.3.5 PostgreSQL as the Source Database Type

Check Item	Description	Suggestion
DDL Object Count Check	The system checks whether a user has the permission to access at least one schema where DDL statements need to be evaluated.	Ensure that at least one schema object can be accessed, or the evaluation task cannot be created. Note: System objects in PostgreSQL are not checked.

☐ NOTE

In the pre-check phase during evaluation project creation, the system only checks whether there is at least one object in the source database needs to be evaluated.

3.4 What Are the Possible Causes for an Object Collection Failure?

Object collection can fail for various reasons. Some of the possible reasons are as follows:

- The network to the source database is unavailable.

  In this case, restore the network connection between UGO and the source database and re-create an evaluation project.

- The target database could not be verified.

  If a namespace is specified in the table creation statement, check whether the namespace exists in the target database. If no, create it.

  If a view or function fails to be created, check whether it depends on the target database table. If yes, ensure that the table is successfully created.

3.5 In GaussDB, How Do I Configure a Search Path If A Statement Without Schema Name Fails To Be Executed?

You can execute the following command in GaussDB to see if the **aa** table exists.

create schema sch1;

create table sch1.aa(col int);

select * from aa;

If it does not exist, the actual error message is as follows: -ERROR: The table **aa** does not exist.

LINE 1: select * from aa;

^

SQL state: 42P01

Character: 15

This is because there is no **sch1** in the search path.

Run following statement to add **sch1** to the search path:

set search_path = "$user",public,sch1;

Run the SELECT statement.

select * from aa;

In this case, no error occurs.

> 📖 **NOTE**
>
> To obtain the current search_path, run the following statement:
> **show search_path;**

3.6 What Is Native Compatibility, Conversion Compatibility, Partial Compatibility, or Incompatibility?

What is native compatibility, conversion compatibility, partial compatibility, or incompatibility?

- **Native compatibility**: The object syntax of the source database is fully compatible with that of the target database.
- **Conversion compatibility**: UGO can convert the object syntax of source database to be compatible with the target database.
- **Partial compatibility**: UGO can convert the object syntax of the source database to be compatible with the target database, but there may be some performance or functional differences after conversion. Before performing

syntax conversion, you can select a new conversion configuration that meets the customer's service requirements.

- **Incompatibility**: UGO cannot convert the object syntax of the source database to be compatible with the target database.

Native compatible objects, conversion compatible objects, and partial compatible objects collectively referred to as UGO-supported objects.

3.7 What Is the Relationship Between Migration Risk (Top 10 risk SQL) and Risky SQL Summary?

There is no relationship between migration risk and risky SQL summary.

Migration Risks (Top 10 risk SQL): describes the 10 slow SQL statements using up the most CPU and memory resources on the source database over the past 7 days.

Risky SQL Summary: describes migration risks from the perspective of the target database. You need to pay attention to some functions that are not directly supported by the target database.

Risks are classified into functional and performance risks.

- Functional risks: risks that affect the database functions. For example, data types (such as value ranges) that are not directly supported by the target database can be summarized in terms of data_type_mismatch, table_def_mismatch and sequence_limitation.
- Performance risks: risks that affect the database performance. For example, GaussDB does not support partition intervals, which can be summarized in terms of distribution and partitioning.

3.8 What Are Reconstruction Statistics and How Are Reconstruction Points Measured?

Each clause or keyword in the source syntax can be considered a reconstruction point in the migration.

Successful reconstruction points are collected based on native-supported objects, UGO-supported objects, and supported objects with risks.

Failed reconstruction points are collected based on UGO-unsupported objects.

3.9 What Should I Do If An Object Collection Error (Closed Connection) Is Displayed During Evaluation Project Creation?

Symptom

When an evaluation project is being created, **Project Status** becomes **Stopped. Object Collection Error**. After you click **Object Collection Error**, **Closed Connection** is displayed in the **Error** column.

Figure 3-1 Error display

Schema Name	Object Type	Object Name	Error
CDATA10	TABLE	BASE_SERVICE_EVENT_T	Closed Connection
CDATA10	TABLE	BASE_SITEMAP_T	Closed Connection
CDATA10	TABLE	BASE_TABLE_BACKUP_IGNORE_T	Closed Connection

Causes

The collection time for UGO to collect DDL information from the source database was set to 60 seconds. If the size of the database objects to be collected are too large or no data is returned within 60 seconds due to poor database performance or network connection, there may be an object collection error.

Solution

Method 1: Manually submit the SQL statements of the objects that fail to be collected.

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

Step 2 In the navigation pane on the left, choose **Schema Migration** > **DB Evaluation**.

Step 3 Click the project name to go to the **Source DB Analysis** page.

Step 4 Click **View Object Details** in the **Object Statistics** area and select a failed schema.

Step 5 Click **Edit SQL** in the **Operation** column and manually import the SQL statement, and save it.

Step 6 Locate the desired project on the project list page and click **Resume** in the **Operation** column.

----End

Method 2: Change the collection time.

Step 1 Locate the desired project and click **Resume** in the **Operation** column.

Figure 3-2 Resuming a project
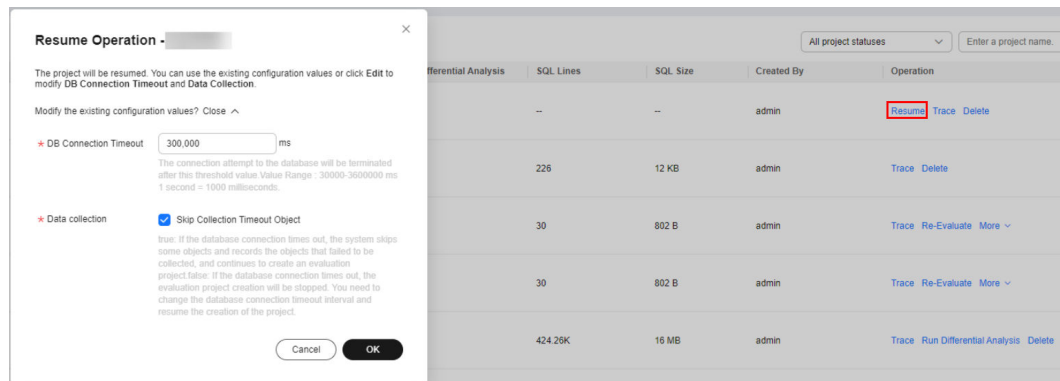
Step 2 Change the value of **DB Connection Timeout**.

Step 3 Click **OK**.

----**End**

3.10 What Should I Do If "ErrorCode=4036" Is Displayed During Evaluation Project Creation?

Symptom

When an evaluation project is being created, **Project Status** becomes **Stopped**. After you move the mouse to **Stopped**, the error message "ErrorCode=4036" is displayed.

Figure 3-3 Error message
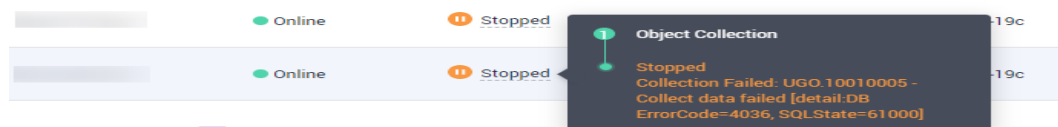
Causes

The value of **pga_aggregate_limit** is inappropriate and there is an error **pga-memory-used-by-the-instance-exceeds-pga-aggregate-limit** in the database.

Solution

Step 1 Log in to the source Oracle database as the **oracle** user.

Step 2 Run the following command to query the value of **pga_aggregate_limit**.

show parameter pga_aggregate_limit;

Step 3 Run the following command to change the value of **pga_aggregate_limit**.

alter system set pga_aggregate_limit = 0;

Step 4 **Log in to UGO** and restore the project.

----End

3.11 What Should I Do If "ErrorCode=17002" Is Displayed During Evaluation Project Creation?

Symptom

When an evaluation project is being created, **Project Status** becomes **Stopped**. After you move the mouse to **Stopped**, the error message "ErrorCode=17002" is displayed.

Causes

During peak hours or when multiple database evaluation tasks are being created at the same time, the number of database connections will reach the upper limit. As a result, UGO cannot obtain valid connections from the source database and several projects are in the **Stopped** status.

Solution

Step 1 Log in to the source Oracle database as the **oracle** user.

Step 2 Query the maximum number of database connections.

show parameter processes;

Step 3 Create **spfile**.

create spfile from pfile;

Step 4 Restart the source database.

shutdown immediate;

startup

Step 5 Run the following command to change the maximum number of connections for the source database:

alter system set processes = 2000 scope = spfile;

Step 6 Restart the source database and check whether the maximum number of connections is changed successfully.

shutdown immediate;

startup

show parameter processes;

Step 7 **Log in to UGO** and restore the project.

----End

3.12 What Do I Do If the Assessment Task Is Connected but the Pre-check Fails?

Symptom

The DBA permission was granted to the connection user who created the evaluation project. The connection test of the evaluation task was successful, but the pre-check failed.

Possible Cause

The user has the DBA permission, but is not the default DBA role.

Solution

Step 1 Log in to the source database and run the following SQL statement to check whether the current user is the default DBA role:

select * from dba_role_privs where grantee='user_name';

📖 **NOTE**

- If the value of **default_role** of **dba** is **no**, the role is not the default DBA role.
- **user_name** indicates the name of the user used to connect to the source database.

Step 2 Run the following SQL statement to change the role to the default one:

ALTER USER user_name DEFAULT ROLE dba;

Step 3 After the modification, perform the pre-check again.

----End

3.13 What Is the Impact of the Collection on the Source Database?

- When a UGO evaluation task connects to the source database to collect objects, the execution time of the **SELECT** or **ALTER** statement in the source Oracle database increases by about 0.1s.
- The following table describes the impact on the source database performance.

Table 3-2 Impact on the source database performance

Source Database Type	Configuration	CPU Usage	Memory Usage	I/O Usage	Application Query Impact
Oracle database without application load	vCPUs: 48 Memory: 188 GB Storage: 511 GB	0.04% -> 17.03%	1.5% -> 2.56%	0.1% -> 2.8%	N/A
Oracle database with application load simulated in TPC-H	vCPUs: 48 Memory: 188 GB Storage: 511 GB	0.04% -> 19.63%	1.52% -> 2.54%	0.1% -> 10.2%	Increased by 5%

◻ **NOTE**

- Source: Lab test data. The specific impact depends on the actual situation.
- The TPC-H is a well-known benchmark used by many database vendors. It consists of a suite of business oriented ad-hoc queries and concurrent data modifications. The queries and the data populating the database have been chosen to have broad industry-wide relevance.

4 Migration Project

4.1 Why Is There No Available Evaluation Project During Migration Project Creation?

The evaluation project creation is not yet complete.

Before migration project creation, ensure that there is at least one evaluation where the target database has been confirmed.

4.2 When Should I Use the Conversion Config Function?

Before the migration, if multiple migration solutions are available in the source database application system, you can click **Conversion Config** to select the optimal configuration solution (based on the source database system and your requirements).

4.3 What Should I Do If SQL Modifications Failed to Be Saved During Object Correction?

After you modify a SQL statement on the object correction page and save the modifications, an error message shown in the following figure may be displayed:

Figure 4-1 Error message
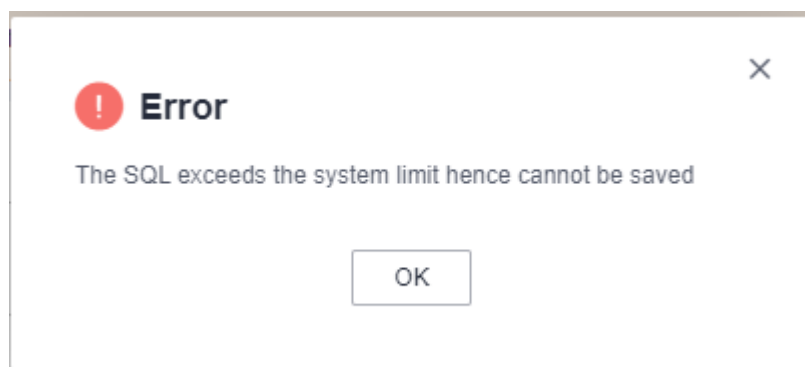
Reason: The statement length exceeds the upper limit of 1 MB.

Suggestion: Adjust the statement length.

4.4 How Do I Solve ERROR: syntax error at or near "@"?

Symptom

In GaussDB Centralized V2.0-3.2, V2.0-3.3, and V2.0-8.0 Enterprise Edition, after **set @a = 1;** was entered, the following error was displayed.

Figure 4-2 Error message

Causes

The **enable_set_variables** parameter is not configured.

Solution

Method 1: Configuring the value of enable_set_variables and applying the new value in a specified database

Step 1 Connect to the database as the **root** user.

Step 2 Run the following command to configure the **enable_set_variables** parameter:

alter database *databasename* set b_format_behavior_compat_options = 'enable_set_variables';

----End

Method 2: Configuring the value of enable_set_variables and applying the new value in a specified instance

Step 1 Connect to the instance as the **root** user.

Step 2 Run the following command to switch to the instance user:

su - omm

 NOTE

In the preceding command, **omm**"indicates the instance user. Change it based on the site requirements.

Step 3 Run the following command to set the instance parameters:

gs_guc reload -Z datanode -N all -I all -c "b_format_behavior_compat_options='enable_set_variables'";

----End

4.5 What Are the Possible Causes for an Object Verification Failure?

Some of the possible reasons are as follows:

- The target database failed to be verified.

  If a tablespace is specified in the table creation statement, check whether the tablespace exists in the target database. If not, create it.

  If a view or function fails to be created, check whether it depends on the target database table. If so, ensure that the table is successfully created.

  Check whether the correct conversion configurations are selected. Some conversion configurations will retain the syntax of the source database. If the syntax is incompatible with that of the target database, the migration will fail. You need to select proper conversion configurations as required.

4.6 What Are the Impacts of GUC Parameter Settings of GaussDB on Migration?

4.6.1 GUC Parameters of GaussDB (Oracle-Compatible Mode)

For GaussDB databases compatible with Oracle syntax, you can set the GUC parameters to enhance the compatibility. For details, see **Table 4-1**.

⚠ CAUTION

GaussDB provides many operation parameters. Configurations of these parameters affect the behavior of the database system. Before changing these parameters, learn the impact of these parameters on the database. Otherwise, unexpected results may occur. For details, see **GUC Parameter Usage**.

Table 4-1 GUC parameters of GaussDB (Oracle-compatible mode)

Oracle-Compatible Configuration Item	Compatibility Configuration Item	Compatibility Configuration Item Name	Supported Database	Enable	Description
behavior_compat_options	display_leading_zero	Specifies how floating point numbers are displayed. 0 before the decimal point is displayed	Centralized / Distributed	Yes	Centralized: **GUC Parameters**. Distributed: **Parameter Details**
	end_month_calculate	Specifies the calculation logic of the add_months function.	Centralized / Distributed	Yes	
	bind_procedure_searchpath	Specifies the search path of the database object in a stored procedure for which no schema name is specified.	Centralized / Distributed	Yes	
	correct_to_number	Specifies the compatibility of the to_number() result.	Centralized / Distributed	Yes	
	unbind_divide_bound	Specifies the range check on the result of integer division.	Centralized / Distributed	Yes	
	convert_string_digit_to_numeric	Determines whether to convert numeric constants of the character string type to those of the numeric type before these two types are compared.	Centralized / Distributed	Yes	
	return_null_string	Specifies how to display the empty result (empty string '') of the lpad() and rpad() functions.	Centralized / Distributed	Yes	

Oracle-Compatible Configuration Item	Compatibility Configuration Item	Compatibility Configuration Item Name	Supported Database	Enable	Description
	merge_update_multi	Controls the UPDATE behavior if a piece of target data in the target table conflicts with multiple pieces of source data When **MERGE INTO … WHEN MATCHED THEN UPDATE** and **INSERT … ON DUPLICATE KEY UPDATE** are used.	Centralized / Distributed	Yes	
	plstmt_implicit_savepoint	Determines whether the execution of an UPDATE statement in a stored procedure has an independent subtransaction.	Centralized / Distributed	Yes	
	hide_tailing_zero	Controls configuration item for numeric display.	Centralized / Distributed	Yes	
	rownum_type_compat	Specifies the ROWNUM type.	Centralized	Yes	
	aformat_null_test	Specifies the logic for checking whether the row type is not null.	Centralized / Distributed	Yes	
	aformat_regexp_match	Determines the matching behavior of regular expression functions.	Centralized / Distributed	Yes	
	compat_cursor	Determines the compatibility behavior of implicit cursor states.	Centralized	Yes	

Oracle-Compatible Configuration Item	Compatibility Configuration Item	Compatibility Configuration Item Name	Supported Database	Enable	Description
	proc_outparam_override	Determines the overloading of output parameters of a stored procedure.	Centralized / Distributed	Yes	
	proc_implicit_for_loop_variable	Controls the behavior of the FOR_LOOP query statement in a stored procedure.	Centralized	Yes	
	allow_procedure_compile_check	Controls the compilation check of the SELECT and OPEN CURSOR statements in a stored procedure.	Centralized	Yes	
	plsql_security_definer	Determines whether the definer permission is used by default when a stored procedure is created.	Centralized / Distributed	Yes	
	plpgsql_dependency	Determines whether a function, stored procedure, or package containing undefined objects can be created.	Centralized	Yes	
plsql_compile_check_options	plsql_expression_check	If the **plsql_expression_check** parameter is enabled, the **plpgsql_dependency** parameter must also be enabled.	Centralized 503.1	Yes	
a_format_version	10c	Specifies database platform compatibility configuration items.	Centralized / Distributed	Yes	

Oracle-Compatible Configuration Item	Compatibility Configuration Item	Compatibility Configuration Item Name	Supported Database	Enable	Description
a_format_dev_version	s1	Specifies the compatible minor version of the database platform.	Centralized / Distributed 503.0	Yes	
	s2	Specifies the compatible minor version of the database platform.	Centralized / Distributed 503.1	Yes	
	s3	Specifies the compatible minor version of the database platform.	Centralized / Distributed 503.2	Yes	
	s4	Specifies the compatible minor version of the database platform.	Centralized / Distributed 505.0	Yes	
	s5	Specifies the compatible minor version of the database platform.	Centralized / Distributed 505.1	Yes	
	s6	Specifies the compatible minor version of the database platform.	Centralized / Distributed 505.2	Yes	
sql_beta_feature	a_style_coerce	Affects the decode function.	Centralized / Distributed	Yes	

📖 NOTE

> Set the **a_format_dev_version** parameter as follows:
>
> - For GaussDB V2.0-3.1 Enterprise Edition, set the value to **s1**.
> - For GaussDB V2.0-3.2 Enterprise Edition, set the value to **s2**.
> - For GaussDB V2.0-3.3 Enterprise Edition, set the value to **s3**.
> - For GaussDB V2.0-8.0 Enterprise Edition, set the value to **s4**.
> - For GaussDB V2.0-8.100 Enterprise Edition, set the value to **s5**.
> - For GaussDB V2.0-8.200 Enterprise Edition and GaussDB V2.0-9.0 Enterprise Edition, set the value to **s6**.
>
> For details about the versions, see **1.9 What Is the Mapping Between the GaussDB Versions Displayed on the UGO Console and Self-built GaussDB Versions?**

4.6.2 GUC Parameters of GaussDB (MySQL-Compatible Mode)

For GaussDB databases compatible with MySQL syntax, you can set the GUC parameters to enhance the compatibility. For details, see **Table 4-2**.

⚠️ CAUTION

GaussDB provides many operation parameters. Configurations of these parameters affect the behavior of the database system. Before changing these parameters, learn the impact of these parameters on the database. Otherwise, unexpected results may occur.

Table 4-2 GUC parameters of GaussDB (MySQL-compatible mode)

MySQL-Compatible Configuration Item	Compatibility Configuration Item	Compatibility Configuration Item Name	Supported Database	Enable
b_format_behavior_compat_options	enable_set_variables	Takes effect only for session variables of centralized GaussDB instances.	Centralized	Yes
b_format_version	5.7	Specifies database platform compatibility configuration items.	Centralized/Distributed 505.0	Yes
b_format_dev_version	s1	Specifies the compatible minor version of the database platform.	Centralized/Distributed 505.0	Yes

MySQL-Compatible Configuration Item	Compatibility Configuration Item	Compatibility Configuration Item Name	Supported Database	Enable
	s2	Specifies the compatible minor version of the database platform.	Centralized/Distributed 505.1	Yes

☐ **NOTE**

Set the **b_format_dev_version** parameter as follows:

● For GaussDB V2.0-8.0 Enterprise Edition, set the value to **s1**.

● For GaussDB V2.0-8.100 Enterprise Edition, set the value to **s2**.

For details about the versions, see **1.9 What Is the Mapping Between the GaussDB Versions Displayed on the UGO Console and Self-built GaussDB Versions?**

● The S1 compatibility configuration item affects the following content:

– NEW() function, last_day() function, date_sub(date, INTERVAL expr unit), datediff(expr1, expr2), day()/dayofmonth(), and dayname()

– dayofweek(), dayofyear(), extract(unit FROM date), from_days(days), from_unixtime(unix_timestamp[,format]), get_format({DATE | TIME | DATETIME | TIMESTAMP}, {'EUR' | 'USA' | 'JIS' | 'ISO' | 'INTERNAL'}), hour(), makedate(year,dayofyear), microsecond(), minute(), month(), monthname(), period_add(period, month_number)

– TIMESTAMPDIFF, yearweek(date[, mode]), year(), weekofyear(date), weekday(), week(date[, mode]), utc_timestamp(), utc_time(), period_diff(p1,p2), second().

– quarter(), str_to_date(str, format), subdate(expr, days), time_format(time, format), ifnull(expr1, expr2), database(), current_date, current_time, current_timestamp

– round(arg1, arg2), localtime([precision]), localtimestamp, dbtimezone, timenow(), numtodsinterval(num, interval_unit), numtoyminterval(num, interval_unit)

– new_time(date, timezone1,timezone2), sysdate([precision]), add_months(d,n), months_between(d1, d2), convert_tz(dt, from_tz, to_tz)

– adddate(date, INTERVAL expr unit), date_format(date, format)

● The S2 compatibility configuration item affects:

– Compatibility behaviors controlled by s1.

– The following functions:

■ When data is imported to the **AUTO_INCREMENT** column or the **BATCH INSERT** execution plan is inserted, if **0**, **NULL**, and determined values are used together, the auto-increment count is updated immediately when the determined value is inserted, the subsequent **0** or **NULL** increases automatically based on the determined value.

- The like operator does not report an error when an escape character is at the end of the matching string.

- The priorities of the sorting rules for character sets and character orders are changed.

  - The following syntax:

    - The CREATE TABLE table_name LIKE source_table syntax is supported.

    - The syntaxes CREATE TABLE table_name LIKE source_table and CREATE TABLE table_name (LIKE source_table) cannot specify **INCLUDING** and **EXCLUDING** options. By default, **INCLUDING ALL** is specified.

    - The LOAD DATA syntax is supported. Some syntax functions in gs_loader that are consistent with the LOAD DATA syntax will change.

    - The collate clause can be specified by set names.

    - The syntax for changing table names, such as ALTER TABLE and RENAME TABLE, is involved. For example, if the character string corresponding to the new table name starts with **#MySQL50#** and is followed by other characters, **#MySQL50#** will be ignored.

4.7 How Do I Set GaussDB GUC Parameters?

Setting GUC Parameters for GaussDB Centralized

GaussDB Centralized provides two methods for setting GUC parameters:

1. Use gsql, a management platform, or other client tools to connect to GaussDB and run the following SQL statements to set the database-level GUC parameters:
   ALTER DATABASE *<database_name>* **SET behavior_compat_options = 'bind_procedure_searchpath,truncate_numeric_tail_zero,plsql_security_definer,proc_out param_override,aformat_null_test,rownum_type_compat,allow_procedure_compile_ch eck,proc_implicit_for_loop_variable,plstmt_implicit_savepoint,end_month_calculate,dis able_rewrite_nesttable,plpgsql_dependency,display_leading_zero,correct_to_number,u nbind_divide_bound,convert_string_digit_to_numeric,hide_tailing_zero,return_null_stri ng,aformat_regexp_match,compat_cursor,enable_funcname_with_argsname,tableof_el em_constraints,merge_update_multi';**
   ALTER DATABASE *<database_name>* **SET plsql_compile_check_options = 'plsql_expression_check';**
   ALTER DATABASE *<database_name>* **SET sql_beta_feature = 'a_style_coerce';**
   ALTER DATABASE *<database_name>* **SET a_format_version = '10c';**
   ALTER DATABASE *<database_name>* **SET a_format_dev_version = 's6';**

2. Log in to a GaussDB server and use the gs_guc tool to run the following commands to set the instance-level GUC parameters:
   su - omm
   gs_guc reload -Z coordinator -Z datanode -N all -I all -c "behavior_compat_options='bind_procedure_searchpath,truncate_numeric_tail_zero,pls ql_security_definer,proc_outparam_override,aformat_null_test,rownum_type_compat,a llow_procedure_compile_check,proc_implicit_for_loop_variable,plstmt_implicit_savepoi

```
nt,end_month_calculate,disable_rewrite_nesttable,plpgsql_dependency,display_leadin
g_zero,correct_to_number,unbind_divide_bound,convert_string_digit_to_numeric,hide_t
ailing_zero,return_null_string,aformat_regexp_match,compat_cursor,enable_funcname
_with_argsname,tableof_elem_constraints,merge_update_multi'"
gs_guc reload -Z coordinator  -Z datanode -N all -I all -c
"plsql_compile_check_options='plsql_expression_check'"
gs_guc reload -Z datanode -N all -I all -c  "sql_beta_feature='a_style_coerce'"
gs_guc reload -Z coordinator -Z datanode -N all -I all -c "a_format_version='10c'"
gs_guc reload -Z coordinator -Z datanode -N all -I all -c "a_format_dev_version='s6'"
```

☐☐ NOTE

> The preceding SQL statements and commands are applied to migration from Oracle to
> GaussDB Centralized V2.0-8.0 Enterprise Edition. For other migration flows, replace the
> parameter names and values before and after equal signs (=) in the preceding
> commands with those displayed on the pre-check page.

Setting GUC Parameters for GaussDB Distributed

You cannot modify GUC parameters of distributed GaussDB instances on the
cloud. Contact GaussDB O&M personnel to modify the parameters.

For on-premises distributed GaussDB instances, you can only log in to a server and
use the gs_guc tool to run the following commands to set the instance-level GUC
parameters:

```
gs_guc reload -Z coordinator -Z datanode -N all -I all -c
"behavior_compat_options='bind_procedure_searchpath,truncate_numeric_tail_zero,plpgsql_s
ecurity_definer,proc_outparam_override,aformat_null_test,plstmt_implicit_savepoint,end_m
onth_calculate,disable_rewrite_nesttable,display_leading_zero,correct_to_number,unbind_di
vide_bound,convert_string_digit_to_numeric,hide_tailing_zero,return_null_string,aformat_re
gexp_match,enable_funcname_with_argsname,tableof_elem_constraints,merge_update_mu
lti'";
gs_guc reload -Z coordinator  -Z datanode -N all -I all -c "a_format_version='10c'";
gs_guc reload -Z coordinator -Z datanode -N all -I all -c "a_format_dev_version='s6'";
gs_guc reload -Z datanode -N all -I all -c  "sql_beta_feature='a_style_coerce'";
```

☐☐ NOTE

> The preceding SQL statements and commands are applied to migration from Oracle to
> GaussDB Centralized V2.0-8.0 Enterprise Edition. For other migration flows, replace the
> parameter names and values before and after equal signs (=) in the preceding commands
> with those displayed on the pre-check page.

4.8 Why Is the Number of Indexes Different After Oracle Data is Migrated to GaussDB?

Symptom

After data is migrated from Oracle to GaussDB, the total number of indexes is
different in the source and target databases.

Causes

During UGO collection, the primary key and unique constraints, and indexes
generated by the system are filtered out.

Solution

Connect to the Oracle database and run the following SQL statements to query the number of indexes:

1. Query the number of indexes displayed on the UGO evaluation page.

```
SELECT count(*)
FROM (select owner, index_name, status  from ALL_INDEXES  WHERE OWNER in  ('schema_name') ) i l
LEFT OUTER JOIN
( WITH
cons_ind AS -- Primary key and unique key constraints
     (SELECT constraint_name
      FROM ALL_CONSTRAINTS
      WHERE owner  in  ('schema_name')   AND constraint_type IN ('P', 'U')  AND constraint_name  =
index_name),

cons_col AS --Column information of the constraints
     (SELECT ci.constraint_name, cc.position, cc.column_name
      FROM cons_ind ci, ALL_CONS_COLUMNS cc
      WHERE cc.owner  in  ('schema_name')   AND cc.constraint_name = ci.constraint_name),

ind_col AS -- Column information of the indexes
     (SELECT ic.index_name, ic.column_position, ic.column_name
      FROM cons_ind ci, ALL_IND_COLUMNS ic
      WHERE ic.index_owner  in   ('schema_name')   AND ic.index_name  = ci.constraint_name),

not_match_ind AS -- The column information of the constraints does not match that of the indexes.
     (SELECT cc.constraint_name, ic.index_name
      FROM cons_col cc FULL OUTER JOIN ind_col ic
      ON cc.constraint_name  = ic.index_name
      AND cc.position = ic.column_position
      AND cc.column_name = ic.column_name
      WHERE cc.constraint_name IS NULL OR ic.index_name IS NULL )

  SELECT ci.constraint_name
  FROM cons_ind ci
  LEFT OUTER JOIN ( SELECT constraint_name FROM not_match_ind
           WHERE constraint_name IS NOT NULL
           UNION
           SELECT index_name FROM not_match_ind
           WHERE index_name IS NOT NULL ) nm
 ON ci.constraint_name = nm.constraint_name
 WHERE nm.constraint_name IS NULL ) fil_cons_not
ON il.index_name = fil_cons_not.constraint_name
WHERE fil_cons_not.CONSTRAINT_NAME is null;
```

2. Query the total number of primary key and unique constraints.

```
SELECTcount(*)
     FROM ALL_CONSTRAINTS
     WHERE owner  in  ('schema_name')   AND constraint_type IN ('P', 'U')  AND constraint_name  =
index_name;
```

3. Query the number of objects generated by the system.

```
select owner,OBJECT_NAME, object_type as objectType, count(*) as count
   from dba_objects
   where object_type  in ('INDEX')
   and owner in  ('schema_name')
   and OBJECT_NAME not in( SELECT index_name
FROM (select owner, index_name, status  from ALL_INDEXES  WHERE OWNER in  ('schema_name') ) il
LEFT OUTER JOIN
( WITH
cons_ind AS -- Primary key and unique key constraints
     (SELECT constraint_name
      FROM ALL_CONSTRAINTS
      WHERE owner  in  ('schema_name')   AND constraint_type IN ('P', 'U')  AND constraint_name  =
index_name),
cons_col AS --Column information of constraints
     (SELECT ci.constraint_name, cc.position, cc.column_name
```

```
        FROM cons_ind ci, ALL_CONS_COLUMNS cc
        WHERE cc.owner  in  ('schema_name')   AND cc.constraint_name = ci.constraint_name),
ind_col AS -- Column information of the indexes
      (SELECT ic.index_name, ic.column_position, ic.column_name
       FROM cons_ind ci, ALL_IND_COLUMNS ic
       WHERE ic.index_owner  in   ('schema_name')   AND ic.index_name  = ci.constraint_name),
not_match_ind AS -- The column information of the constraints does not match that of the indexes.
      (SELECT cc.constraint_name, ic.index_name
       FROM cons_col cc FULL OUTER JOIN ind_col ic
       ON cc.constraint_name  = ic.index_name
       AND cc.position = ic.column_position
       AND cc.column_name = ic.column_name
       WHERE cc.constraint_name IS NULL OR ic.index_name IS NULL )
  SELECT ci.constraint_name
  FROM cons_ind ci
  LEFT OUTER JOIN ( SELECT constraint_name FROM not_match_ind
          WHERE constraint_name IS NOT NULL
          UNION
          SELECT index_name FROM not_match_ind
          WHERE index_name IS NOT NULL ) nm
  ON ci.constraint_name = nm.constraint_name
  WHERE nm.constraint_name IS NULL ) fil_cons_not
ON il.index_name = fil_cons_not.constraint_name
WHERE fil_cons_not.CONSTRAINT_NAME is null)
and  OBJECT_NAME not in  (SELECT constraint_name
        FROM ALL_CONSTRAINTS
        WHERE owner  in  ('schema_name')   AND constraint_type IN ('P', 'U')  AND constraint_name  =
index_name)
   group by owner, object_type,OBJECT_NAME;
```

📖 **NOTE**

- The total number of indexes queried in the source Oracle database is equal to the sum of the preceding three parts.

- *schema_name* indicates the name of the schemas to be migrated. Replace it with the actual schema name.

4.9 What Should I Do If There Are ctid, xc_node_id, and tableoid Columns in GaussDB?

Symptom

To migrate a heterogeneous database to GaussDB, you need to manually rename the **ctid**, **xc_node_id**, **tableoid**, and **ctrd** columns.

```
ugo=>
ugo=> create table test_row(tableoid int);
ERROR:  column name "tableoid" conflicts with a system column name
ugo=>
ugo=> create table test_row1(xc_node_id int);
ERROR:  column name "xc_node_id" conflicts with a system column name
ugo=>
ugo=>
ugo=> create table test_row2(ctid int);
ERROR:  column name "ctid" conflicts with a system column name
ugo=>
```

Causes

There are the system columns **ctid**, **xc_node_id**, **tableoid**, and **ctrd** both in GaussDB and the source database. You need to manually rename these columns when migrating data to GaussDB.

Solution

Step 1 Run the following command to connect to the GaussDB database:

gsql -U username -d database -p 4000 -W *pwd*

📖 **NOTE**

> In the preceding command, *username* indicates the database username, *database* indicates the database name, and *pwd* indicates the password of the username.

Step 2 Run the following SQL statement to rename the columns **ctid**, **xc_node_id**, **tableoid**, and **ctrd**:

create table test("CTRD" int);

create table test("XC_NODE" int);

create table test("TABLEO" int);

create table test("CTRD" int);

----End

4.10 Why Is Data Inconsistent When SELECT Statements Without ORDER BY Are Used for Query in Distributed GaussDB?

Symptom

Figure 4-3 There is no ORDER BY in SELECT statements.

Causes

In distributed GaussDB, data is evenly stored on each DN based on distribution keys. If ORDER BY is not specified in a SELECT query, data is randomly read from DNs. Therefore, data consistency cannot be ensured.

Solution

Use ORDER BY in SQL statements for query. The column followed by ORDER BY is the distribution key.

Figure 4-4 SELECT statements with ORDER BY