《数据库》

openGauss数据库备份恢复指导手册



华为技术有限公司

目录

[前 言 3](#_Toc81318229)

[简介 3](#_Toc81318230)

[内容描述 3](#_Toc81318231)

[前置条件 3](#_Toc81318232)

[实验环境说明 3](#_Toc81318233)

[实验概览 4](#_Toc81318234)

[1 数据库备份恢复 5](#_Toc81318235)

[1.1 实验介绍 5](#_Toc81318236)

[1.1.1 关于本实验 5](#_Toc81318237)

[1.1.2 实验目的 5](#_Toc81318238)

[1.2 实验前提 5](#_Toc81318239)

[1.3 物理备份和恢复 8](#_Toc81318240)

[1.3.1 实验前提 8](#_Toc81318241)

[1.3.2 物理备份 8](#_Toc81318242)

[1.3.3 物理备份恢复 9](#_Toc81318243)

[1.4 逻辑备份和恢复 11](#_Toc81318244)

[1.4.1 实验前提 11](#_Toc81318245)

[1.4.2 gs\_dump 11](#_Toc81318246)

[1.4.3 gs\_dumpall 21](#_Toc81318247)

[1.4.4 gs\_restore 22](#_Toc81318248)

[附录一：Linux操作系统相关命令 32](#_Toc81318249)

# 前 言

## 简介

本指导书适用于openGauss数据库，通过该指导书可以了解openGauss数据库支持的两种备份恢复类型、多种备份恢复方案，并能够在备份和恢复过程中提供数据的可靠性保障机制。

## 内容描述

数据备份是保护数据安全的重要手段之一，为了更好的保护数据安全，openGauss数据库支持两种备份恢复类型、多种备份恢复方案，备份和恢复过程中提供数据的可靠性保障机制。

备份与恢复类型可分为逻辑备份与恢复、物理备份与恢复。

## 前置条件

* 由于本实验主要是在openEuler操作系统上进行数据库的备份和恢复，需要掌握Linux系统的基本操作和系统命令，详细请参见[**附录一**](#_附录一：Linux操作系统相关命令)。
* 实验前提和数据库恢复验证中需要掌握openGauss数据库的基本操作和SQL语法，openGauss数据库支持SQL2003标准语法，数据库基本操作参见[**附录二**](#_附录二：openGauss数据库基本操作)。

## 实验环境说明

* 组网说明

本实验环境为openGauss数据库管理系统，安装在华为云openEuler弹性服务器ECS上。

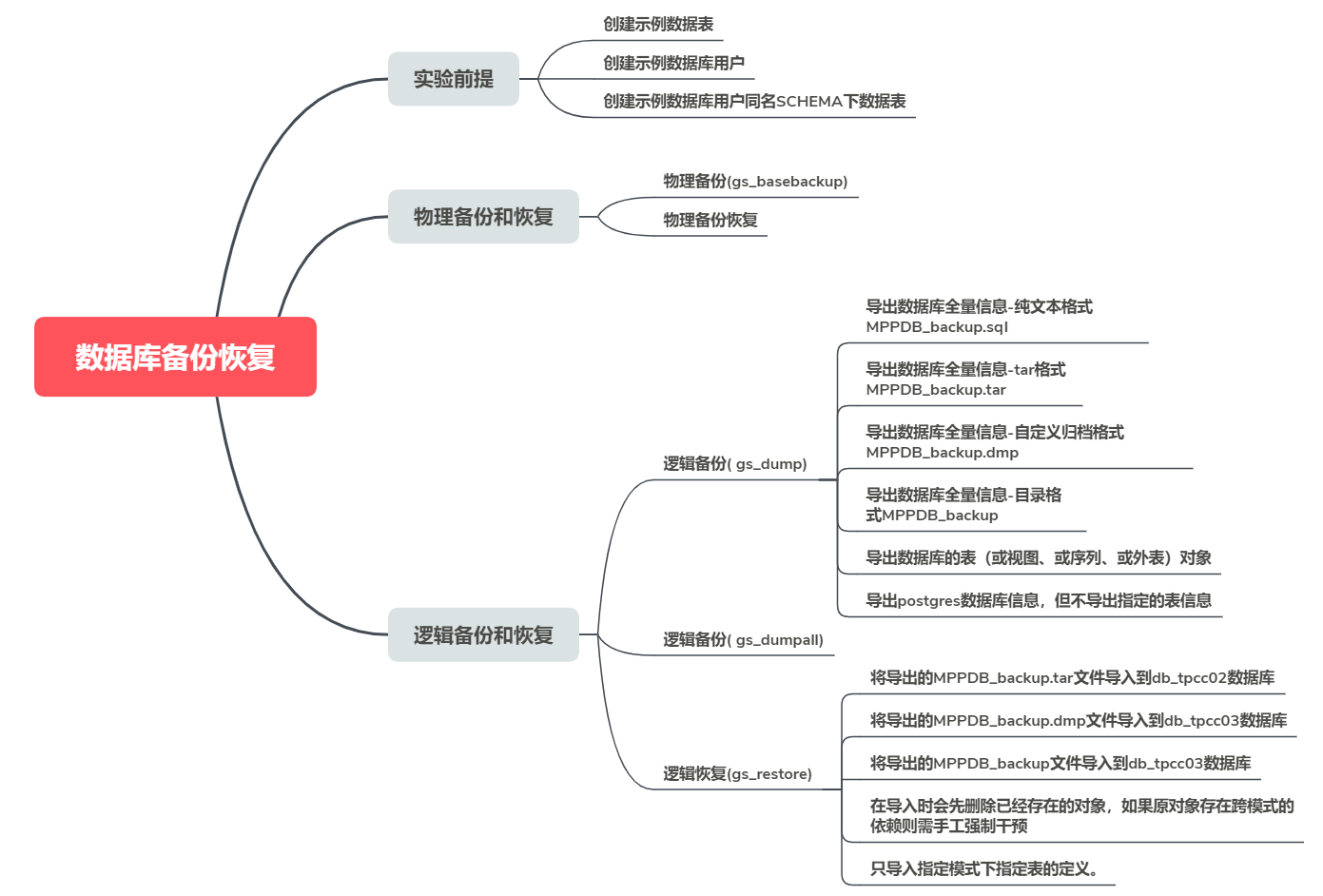
* 设备介绍

为了满足数据库原理与实践课程实验需要，建议每套实验环境采用以下配置：

配置明细表

|  |  |
| --- | --- |
| 软件名称 | 软件版本 |
| 数据库 | openGauss 2.0.0 |
| 操作系统 | openEuler 20.3LTS |

## 实验概览



本实验概览图

# 数据库备份恢复

## 实验介绍

### 关于本实验

本实验主要描述openGauss数据库支持的两种备份恢复类型、多种备份恢复方案，并能够在备份和恢复过程中提供数据的可靠性保障机制。

### 实验目的

掌握openGauss数据库逻辑备份与恢复方法；

掌握openGauss数据库物理备份与恢复的方法；

能够在备份和恢复过程中提供数据的可靠性保障机制。

## 实验前提

在对数据库进行备份前，对数据库进行如下操作：

切换到omm用户，以操作系统用户omm登录数据库主节点。

[root@ecs-bb09 ~]$ **su – omm**

启动数据库服务。

[omm@ecs-bb09 ~]$ **gs\_om -t start**

连接openGauss数据库。

[omm@ecs-bb09 ~]$ **gsql -d postgres -p 26000 -r**

创建customer\_t1表。

postgres=#**DROP TABLE IF EXISTS customer\_t1;**

postgres=#**CREATE TABLE customer\_t1   
(   
 c\_customer\_sk integer,   
 c\_customer\_id char(5),   
 c\_first\_name char(6),   
 c\_last\_name char(8)   
);**

当结果显示为如下信息，则表示创建成功。

CREATE TABLE

向表中插入数据。

postgres=# **INSERT INTO customer\_t1 (c\_customer\_sk, c\_customer\_id, c\_first\_name) VALUES**

**(3769, 'hello', DEFAULT) ,**

**(6885, 'maps', 'Joes'),**

**(4321, 'tpcds', 'Lily'),**

**(9527, 'world', 'James');**

显示结果为：

INSERT 0 4

查看表中的数据。

postgres=# **select \* from customer\_t1;**

c\_customer\_sk | c\_customer\_id | c\_first\_name | c\_last\_name

---------------+---------------+--------------+-------------

6885 | maps | Joes |

4321 | tpcds | Lily |

9527 | world | James |

3769 | hello | |

(4 rows)

创建customer\_t2表。

postgres=#**DROP TABLE IF EXISTS customer\_t2;**

postgres=#**CREATE TABLE customer\_t2**

**(**

**c\_customer\_sk integer,**

**c\_customer\_id char(5),**

**c\_first\_name char(6),**

**c\_last\_name char(8)**

**);**

当结果显示为如下信息，则表示创建成功。

CREATE TABLE

向表中插入数据。

postgres=# **INSERT INTO customer\_t2 (c\_customer\_sk, c\_customer\_id, c\_first\_name) VALUES**

**(3769, 'hello', DEFAULT) ,**

**(6885, 'maps', 'Joes'),**

**(9527, 'world', 'James');**

显示结果为：

INSERT 0 3

查看表中数据。

postgres=# **select \* from customer\_t2;**

c\_customer\_sk | c\_customer\_id | c\_first\_name | c\_last\_name

---------------+---------------+--------------+-------------

3769 | hello | |

6885 | maps | Joes |

9527 | world | James |

(3 rows)

创建用户Lucy。

postgres=# **DROP user IF EXISTS lucy;**

postgres=# **CREATE USER lucy WITH PASSWORD "Bigdata@123";**

切换到Lucy用户。 （截图）

postgres=# **\c - lucy**

Password for user lucy:

Non-SSL connection (SSL connection is recommended when requiring high-security)

You are now connected to database "postgres" as user "lucy".

创建lucy shema的表。

postgres=# **DROP TABLE IF EXISTS lucy.mytable;**

postgres=# **CREATE TABLE mytable (firstcol int);**

显示结果为：

CREATE TABLE

向表中插入数据。

postgres=# **INSERT INTO mytable values (100);**

当结果显示为如下信息，则表示插入数据成功。

INSERT 0 1

查看表中数据。

postgres=# **SELECT \* from mytable;**

显示结果为：

firstcol   
----------   
 100   
(1 row)

退出数据库。

postgres=#**\q**

## 物理备份和恢复

openGauss部署成功后，在数据库运行的过程中，会遇到各种问题及异常状态。openGauss提供了gs\_basebackup工具做基础备份。gs\_basebackup工具由操作系统用户omm执行。

### 实验前提

在数据库备份之前创建存储备份文件的文件夹。

切换到omm用户，以操作系统用户omm登录数据库主节点。（连续实验则在上一步\q后直接切换到omm用户，不需要重新切换连接，可跳过此步骤）

[root@ecs-bb09 ~]$ **su - omm**

创建存储备份文件的文件夹。

[omm@ecs-bb09 ~]$ **mkdir -p /home/omm/physical/backup**

### 物理备份

如果数据库服务没有启动，就启动数据库服务（务必以操作系统用户omm启动数据库服务，如果没有请切换用户）。

[omm@ecs-bb09 ~]$ **gs\_om -t start**

结果显示为：

Starting cluster.

=========================================

=========================================

Successfully started.

将数据库进行物理备份。 （截图）

[omm@ecs-bb09 ~]$ **gs\_basebackup -D /home/omm/physical/backup -p 26000**

参数-D directory表示备份文件输出的目录，**必选项**。

结果显示为：

INFO: The starting position of the xlog copy of the full build is: 1/D5000028. The slot minimum LSN is: 0/0.

begin build tablespace list

finish build tablespace list

begin get xlog by xlogstream

check identify system success

send START\_REPLICATION 1/D5000000 success

keepalive message is received

keepalive message is received

keepalive message is received

keepalive message is received

keepalive message is received

keepalive message is received

keepalive message is received

keepalive message is received

切换到存储备份文件夹查看备份文件。

[omm@ecs-bb09 ~]$ **cd /home/omm/physical/backup**

[omm@ecs-bb09 ~]$ **ls**

显示如下：

backup\_label mot.conf pg\_errorinfo pg\_llog pg\_snapshots pg\_xlog server.key

base pg\_clog pg\_hba.conf pg\_multixact pg\_stat\_tmp postgresql.conf server.key.cipher

cacert.pem pg\_copydir pg\_hba.conf.bak pg\_notify pg\_tblspc postgresql.conf.bak server.key.rand

global pg\_csnlog pg\_hba.conf.lock pg\_replslot pg\_twophase postgresql.conf.lock

gswlm\_userinfo.cfg pg\_ctl.lock pg\_ident.conf pg\_serial PG\_VERSION server.crt

### 物理备份恢复

当数据库发生故障时需要从备份文件进行恢复。因为gs\_basebackup是对数据库按二进制进行备份，因此恢复时可以直接拷贝替换原有的文件，或者直接在备份的库上启动数据库。

说明：

若当前数据库实例正在运行，直接从备份文件启动数据库可能会存在端口冲突，这时，需要修改配置文件的port参数，或者在启动数据库时指定一下端口。

若当前备份文件为主备数据库，可能需要修改一下主备之间的复制连接。即配置文件中的postgre.conf中的replconninfo1，replconninfo2等。

当数据库发生问题需要从备份进行恢复时，步骤如下：

停止openGauss（务必以操作系统用户omm停止数据库服务，如果没有请切换用户）。

[omm@ecs-bb09 ~]$ **gs\_om -t stop**

显示的结果为：

Stopping cluster.

=========================================

Successfully stopped cluster.

=========================================

End stop cluster.

清理原库中的所有或部分文件。

[omm@ecs-bb09 ~]$ **cd /gaussdb/data/**

[omm@ecs-bb09 ~]$ **ls**

查看数据库节点文件夹名称（文件夹名称是数据库安装时定义的，不同数据可能不同）。

db1

查看文件列表如下：

[omm@ecs-bb09 ~]$ **cd db1**

[omm@ecs-bb09 ~]$ **ls**

base mot.conf pg\_errorinfo pg\_llog pg\_serial PG\_VERSION postmaster.opts

cacert.pem pg\_clog pg\_hba.conf pg\_location pg\_snapshots pg\_xlog server.crt

gaussdb.state pg\_copydir pg\_hba.conf.bak pg\_multixact pg\_stat\_tmp postgresql.conf server.key

global pg\_csnlog pg\_hba.conf.lock pg\_notify pg\_tblspc postgresql.conf.bak server.key.cipher

gswlm\_userinfo.cfg pg\_ctl.lock pg\_ident.conf pg\_replslot pg\_twophase postgresql.conf.lock server.key.rand

删除文件，对数据库文件进行破坏。

[omm@ecs-bb09 ~]$ **rm -rf \***

[omm@ecs-bb09 ~]$ **ls**

删除文件后，列表如下：

base pg\_clog pg\_csnlog pg\_llog pg\_multixact pg\_replslot pg\_snapshots pg\_tblspc pg\_xlog

global pg\_copydir pg\_errorinfo pg\_location pg\_notify pg\_serial pg\_stat\_tmp pg\_twophase

或者为空：

[omm@ecs-ecs-c9bf db1]$

使用数据库系统用户权限从备份中还原需要的数据库文件，＂/gaussdb/data/db1＂中db1是数据库节点文件夹名称，不同数据库可能不同请查看确认。

[omm@ecs-bb09 ~]$ **cp -r /home/omm/physical/backup/. /gaussdb/data/db1**

备份时间大概需要几分钟，恢复后文件列表如下：

[omm@ecs-bb09 ~]$ **cd /gaussdb/data/db1**

[omm@ecs-bb09 ~]$ **ls**

backup\_label.old mot.conf pg\_hba.conf pg\_multixact pg\_tblspc postgresql.conf.lock server.key.rand

base pg\_clog pg\_hba.conf.bak pg\_notify pg\_twophase postmaster.opts

cacert.pem pg\_copydir pg\_hba.conf.lock pg\_replslot PG\_VERSION postmaster.pid

gaussdb.state pg\_csnlog pg\_ident.conf pg\_serial pg\_xlog server.crt

global pg\_ctl.lock pg\_llog pg\_snapshots postgresql.conf server.key

gswlm\_userinfo.cfg pg\_errorinfo pg\_location pg\_stat\_tmp postgresql.conf.bak server.key.cipher

若数据库中存在链接文件，需要修改使其链接到正确的文件。

重启数据库服务器，并检查数据库内容，确保数据库已经恢复到所需的状态。

[omm@ecs-bb09 ~]$ **gs\_om -t start**

Starting cluster.

=========================================

=========================================

Successfully started.

## 逻辑备份和恢复

通过逻辑导出对数据进行备份，逻辑备份只能基于备份时刻进行数据转储，所以恢复时也只能恢复到备份时保存的数据。逻辑备份无法备份故障点和备份点之间的数据，逻辑备份适合备份很少变化的数据，当这些数据因误操作被损坏时，可以通过逻辑备份进行快速恢复。如果通过逻辑备份进行全库恢复，通常需要重建数据库，导入备份数据来完成，对于可用性要求很高的数据库，这种恢复时间太长，通常不被采用。由于逻辑备份具有平台无关性，所以更为常见的是逻辑备份被作为一个数据迁移及移动的主要手段。

### 实验前提

在数据库备份之前创建存储备份文件的文件夹。

切换到omm用户，以操作系统用户omm登录数据库主节点。（连续实验可跳过此步骤）

[root@ecs-bb09 ~]$ **su - omm**

创建存储备份文件的文件夹。

[omm@ecs-bb09 ~]$ **mkdir -p /home/omm/logical/backup**

### gs\_dump

gs\_dump是openGauss用于导出数据库相关信息的工具，用户可以自定义导出一个数据库或其中的对象（模式、表、视图等）。支持导出的数据库可以是默认数据库postgres，也可以是自定义数据库。

主要功能：

gs\_dump可以创建四种不同的导出文件格式，通过[-F或者--format=]选项指定，具体如表所示。

导出文件格式说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 格式名称 | -F的参数数值 | 说明 | 建议 | 对应导入工具 |
| 纯文本格式 | p | 纯文本脚本文件包含SQL语句和命令。命令可以由gsql命令行终端程序执行，用于重新创建数据库对象并加载表数据。 | 小型数据库，一般推荐纯文本格式。 | 使用gsql工具恢复数据库对象前，可根据需要使用文本编辑器编辑纯文本导出文件。 |
| 自定义归档格式 | c | 一种二进制文件。支持从导出文件中恢复所有或所选数据库对象。 | 中型或大型数据库，推荐自定义归档格式。 | 使用gs\_restore可以选择要从自定义归档导出文件中导入相应的数据库对象。 |
| 目录归档格式 | d | 该格式会创建一个目录，该目录包含两类文件，一类是目录文件，另一类是每个表和blob对象对应的数据文件。 | 无 |
| tar归档格式 | t | tar归档文件支持从导出文件中恢复所有或所选数据库对象。tar归档格式不支持压缩且对于单独表大小应小于8GB。 | 无 |

参数说明：

gs\_dump参数说明

|  |  |
| --- | --- |
| 参数 | 参数说明 |
| -f, --file=FILENAME | 将输出发送至指定文件或目录。如果省略该参数，则使用标准输出。如果输出格式为（-F c/-F d/-F t）时，必须指定-f参数。如果-f的参数值含有目录，要求目录对当前用户具有读写权限，并且不能指定已有目录。 |
| -F, --format=c|d|t|p | 选择输出格式。格式如下：  p|plain：输出一个文本SQL脚本文件（默认）。  c|custom：输出一个自定义格式的归档，并且以目录形式输出，作为gs\_restore输入信息。  d|directory：该格式会创建一个目录，该目录包含两类文件，一类是目录文件，另一类是每个表和blob对象对应的数据文件。  t|tar：输出一个tar格式的归档形式，作为gs\_restore输入信息。 |
| -a, --data-only | 只输出数据，不输出模式\（数据定义\）。转储表数据、大对象和序列值。 |
| -c, --clean | 在将创建数据库对象的指令输出到备份文件之前，先将清理（删除）数据库对象的指令输出到备份文件中。 |
| -n, --schema=SCHEMA | 只转储与模式名称匹配的模式，此选项包括模式本身和所有它包含的对象。如果该选项没有指定，所有在目标数据库中的非系统模式将会被转储。写入多个-n选项来选择多个模式。 |
| --include-table-file=FILENAME | 指定需要dump的表文件。 |
| -h, --host=HOSTNAME | 指定主机名称。如果数值以斜杠开头，则被用作到Unix域套接字的路径。缺省从PGHOST环境变量中获取（如果已设置），否则，尝试一个Unix域套接字连接。该参数只针对openGauss外，对openGauss内本机只能用127.0.0.1。 |
| -U, --username=NAME | 指定所连接主机的用户名。不指定连接主机的用户名时，用户默认系统管理员。 |
| -W, --password=PASSWORD | 指定用户连接的密码。如果主机的认证策略是trust，则不会对系统管理员进行密码验证，即无需输入-W选项；如果没有-W选项，并且不是系统管理员，＂Dump Restore工具＂会提示用户输入密码。 |

#### 注意事项

如果openGauss有任何本地数据要添加到template1数据库，请将gs\_dump的输出恢复到一个真正的空数据库中，否则可能会因为被添加对象的定义被复制，出现错误。要创建一个无本地添加的空数据库，需从template0而非template1复制，例如：

**CREATE DATABASE foo WITH TEMPLATE template0;**

tar归档形式的文件大小不得超过8GB（tar文件格式的固有限制）。tar文档整体大小和任何其他输出格式没有限制，操作系统可能对此有要求。

由gs\_dump生成的转储文件不包含优化程序用来做执行计划决定的统计数据。因此，最好从某转储文件恢复之后运行ANALYZE以确保最佳效果。转储文件不包含任何ALTER DATABASE…SET命令，这些设置由gs\_dumpall转储，还有数据库用户和其他完成安装设置。

使用gs\_dump转储数据库为SQL文本文件或其它格式的操作：示例中＂Bigdata@123＂表示数据库用户密码＂/home/omm/logical/backup/MPPDB\_backup.sql＂表示导出的文件；＂26000＂表示数据库服务器端口；＂postgres＂表示要访问的数据库名。

说明：导出操作时，请确保该目录存在并且当前的操作系统用户对其具有读写权限。

#### gs\_dump备份示例1

执行gs\_dump，导出postgres数据库全量信息，导出的**MPPDB\_backup.sql文件格式**为纯文本格式。

以操作系统用户omm登录数据库主节点。（连续实验可跳过此步骤）

[root@ecs-bb09 ~]$ **su - omm**

执行gs\_dump，导出的MPPDB\_backup.sql文件格式为纯文本格式。（截图）

[omm@ecs-bb09 ~]$ **gs\_dump -U omm -W Bigdata@123 -f /home/omm/logical/backup/MPPDB\_backup.sql -p 26000 postgres -F p**

执行后显示为:

gs\_dump[port='26000'][postgres][2020-07-29 14:00:49]: The total objects number is 348.

gs\_dump[port='26000'][postgres][2020-07-29 14:00:49]: [100.00%] 348 objects have been dumped.

gs\_dump[port='26000'][postgres][2020-07-29 14:00:49]: dump database postgres successfully

gs\_dump[port='26000'][postgres][2020-07-29 14:00:49]: total time: 176 ms

切换到backup文件夹，查看MPPDB\_backup.sql文件。

[omm@ecs-bb09 ~]$ **ll /home/omm/logical/backup/**

total 4.0K

-rw------- 1 omm dbgrp 2.7K Jul 29 14:00 MPPDB\_backup.sql

[omm@ecs-bb09 ~]$ **cat /home/omm/logical/backup/MPPDB\_backup.sql**

--

-- PostgreSQL database dump

--

SET statement\_timeout = 0;

SET xmloption = content;

SET client\_encoding = 'SQL\_ASCII';

SET standard\_conforming\_strings = on;

SET check\_function\_bodies = false;

SET client\_min\_messages = warning;

--

-- Name: postgres; Type: COMMENT; Schema: -; Owner: omm

--

COMMENT ON DATABASE postgres IS 'default administrative connection database';

--

-- Name: lucy; Type: SCHEMA; Schema: -; Owner: lucy

--

CREATE SCHEMA lucy;

ALTER SCHEMA lucy OWNER TO lucy;

SET search\_path = lucy;

SET default\_tablespace = '';

SET default\_with\_oids = false;

--

-- Name: mytable; Type: TABLE; Schema: lucy; Owner: lucy; Tablespace:

--

CREATE TABLE mytable (

firstcol integer

)

WITH (orientation=row, compression=no);

ALTER TABLE lucy.mytable OWNER TO lucy;

SET search\_path = public;

--

-- Name: customer\_t1; Type: TABLE; Schema: public; Owner: omm; Tablespace:

--

CREATE TABLE customer\_t1 (

a integer,

b integer,

c integer,

d integer

)

WITH (orientation=row, compression=no);

ALTER TABLE public.customer\_t1 OWNER TO omm;

……

REVOKE ALL ON SCHEMA public FROM PUBLIC;

REVOKE ALL ON SCHEMA public FROM omm;

GRANT ALL ON SCHEMA public TO omm;

GRANT USAGE ON SCHEMA public TO PUBLIC;

--

-- PostgreSQL database dump complete

--

#### gs\_dump备份示例2

执行gs\_dump，导出postgres数据库全量信息，导出的**MPPDB\_backup.tar文件格式**为tar格式。

以操作系统用户omm登录数据库主节点。（连续实验可跳过此步骤）

[root@ecs-bb09 ~]$ **su - omm**

执行gs\_dump，导出的MPPDB\_backup.tar文件格式为tar格式。

[omm@ecs-bb09 ~]$ **gs\_dump -U omm -W Bigdata@123 -f /home/omm/logical/backup/MPPDB\_backup.tar -p 26000 postgres -F t**

运行后的结果为：

gs\_dump[port='26000'][postgres][2020-07-28 16:19:14]: The total objects number is 341.

gs\_dump[port='26000'][postgres][2020-07-28 16:19:14]: [100.00%] 341 objects have been dumped.

gs\_dump[port='26000'][postgres][2020-07-28 16:19:14]: dump database postgres successfully

gs\_dump[port='26000'][postgres][2020-07-28 16:19:14]: total time: 169 ms

查看生成的文件信息。

[omm@ecs-bb09 ~]$ **ll /home/omm/logical/backup/**

total 12K

-rw------- 1 omm dbgrp 1.5K Jul 28 13:47 MPPDB\_backup.sql

-rw------- 1 omm dbgrp 8.0K Jul 28 19:31 MPPDB\_backup.tar

#### gs\_dump备份示例3

执行gs\_dump，导出postgres数据库全量信息，导出的**MPPDB\_backup.dmp文件格式**为自定义归档格式。

以操作系统用户omm登录数据库主节点。（连续实验可跳过此步骤）

[root@ecs-bb09 ~]$ **su - omm**

执行gs\_dump，导出的MPPDB\_backup.dmp文件格式为自定义归档格式。

[omm@ecs-bb09 ~]$ **gs\_dump -U omm -W Bigdata@123 -f /home/omm/logical/backup/MPPDB\_backup.dmp -p 26000 postgres -F c**

运行后的结果为：

gs\_dump[port='26000'][postgres][2020-07-28 19:32:20]: The total objects number is 341.

gs\_dump[port='26000'][postgres][2020-07-28 19:32:20]: [100.00%] 341 objects have been dumped.

gs\_dump[port='26000'][postgres][2020-07-28 19:32:20]: dump database postgres successfully

gs\_dump[port='26000'][postgres][2020-07-28 19:32:20]: total time: 167 ms

查看生成的文件信息。

[omm@ecs-bb09 ~]$ **ll /home/omm/logical/backup/**

total 16K

-rw------- 1 omm dbgrp 2.5K Jul 28 19:32 MPPDB\_backup.dmp

-rw------- 1 omm dbgrp 1.5K Jul 28 13:47 MPPDB\_backup.sql

-rw------- 1 omm dbgrp 8.0K Jul 28 19:31 MPPDB\_backup.tar

#### gs\_dump备份示例4

执行gs\_dump，导出postgres数据库全量信息，导出的**MPPDB\_backup文件格式为目录格式**。

以操作系统用户omm登录数据库主节点。（连续实验可跳过此步骤）

[root@ecs-bb09 ~]$ **su - omm**

执行gs\_dump，导出的MPPDB\_backup文件格式为目录格式。

[omm@ecs-bb09 ~]$ **gs\_dump -U omm -W Bigdata@123 -f /home/omm/logical/backup/MPPDB\_backup -p 26000 postgres -F d**

运行后的结果为：

gs\_dump[port='26000'][postgres][2020-07-28 19:35:59]: The total objects number is 341.

gs\_dump[port='26000'][postgres][2020-07-28 19:35:59]: [100.00%] 341 objects have been dumped.

gs\_dump[port='26000'][postgres][2020-07-28 19:35:59]: dump database postgres successfully

gs\_dump[port='26000'][postgres][2020-07-28 19:35:59]: total time: 173 ms

查看生成的文件信息。

[omm@ecs-bb09 ~]$ **ll /home/omm/logical/backup/**

total 20K

drwx------ 2 omm dbgrp 4.0K Jul 28 19:35 MPPDB\_backup

-rw------- 1 omm dbgrp 2.5K Jul 28 19:32 MPPDB\_backup.dmp

-rw------- 1 omm dbgrp 1.5K Jul 28 13:47 MPPDB\_backup.sql

-rw------- 1 omm dbgrp 8.0K Jul 28 19:31 MPPDB\_backup.tar

#### gs\_dump备份示例5

执行gs\_dump，导出postgres数据库的**表（或视图、或序列、或外表）对象**，例如表customer\_t1。

以操作系统用户omm登录数据库主节点。（连续实验可跳过此步骤）

[root@ecs-bb09 ~]$ **su - omm**

执行gs\_dump，导出的表customer\_t1。

[omm@ecs-bb09 ~]$ **gs\_dump -U omm -W Bigdata@123 -f /home/omm/logical/backup/bkp\_shl2.sql -t public.customer\_t1 -p 26000 postgres**

运行后的结果为：

gs\_dump[port='26000'][postgres][2020-08-15 19:54:12]: The total objects number is 336.

gs\_dump[port='26000'][postgres][2020-08-15 19:54:12]: [100.00%] 336 objects have been dumped.

gs\_dump[port='26000'][postgres][2020-08-15 19:54:12]: dump database postgres successfully

gs\_dump[port='26000'][postgres][2020-08-15 19:54:12]: total time: 157 ms

查看生成的文件信息。

[omm@ecs-bb09 ~]$ **ll /home/omm/logical/backup/**

total 480K

-rw------- 1 omm dbgrp 939 Aug 15 19:54 bkp\_shl2.sql

drwx------ 2 omm dbgrp 4.0K Aug 15 19:39 MPPDB\_backup

-rw------- 1 omm dbgrp 112K Aug 15 19:38 MPPDB\_backup.dmp

-rw------- 1 omm dbgrp 110K Aug 15 19:33 MPPDB\_backup.sql

-rw------- 1 omm dbgrp 248K Aug 15 19:36 MPPDB\_backup.tar

查看生成的sql文件。

[omm@ecs-bb09 ~]$ **cat /home/omm/logical/backup/bkp\_shl2.sql**

--

-- PostgreSQL database dump

--

SET statement\_timeout = 0;

SET xmloption = content;

SET client\_encoding = 'SQL\_ASCII';

SET standard\_conforming\_strings = on;

SET check\_function\_bodies = false;

SET client\_min\_messages = warning;

SET search\_path = public;

SET default\_tablespace = '';

SET default\_with\_oids = false;

--

-- Name: customer\_t1; Type: TABLE; Schema: public; Owner: omm; Tablespace:

--

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=row, compression=no);

ALTER TABLE public.customer\_t1 OWNER TO omm;

--

-- Data for Name: customer\_t1; Type: TABLE DATA; Schema: public; Owner: omm

--

COPY customer\_t1 (c\_customer\_sk, c\_customer\_id, c\_first\_name, c\_last\_name) FROM stdin;

3769 hello \N \N

6885 maps Joes \N

4321 tpcds Lily \N

9527 world James \N

\.

;

--

-- PostgreSQL database dump complete

--

#### gs\_dump备份示例6

执行gs\_dump，导出postgres数据库信息，但**不导出/home/MPPDB\_temp.sql中指定的表信息**。导出的MPPDB\_backup.sql文件格式为纯文本格式。

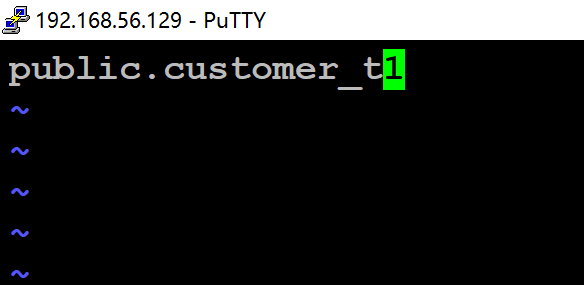
以操作系统用户omm登录数据库主节点。（连续实验可跳过此步骤）

[root@ecs-bb09 ~]$ **su - omm**

创建MPPDB\_temp.sql，填写不导出的表的表名。例如customer\_t1表。

[omm@ecs-bb09 ~]$ **vi /home/omm/logical/MPPDB\_temp.sql**

输入＂i＂，切换到INSERT模式，输入＂**public.customer\_t1**＂,按下＂ESC＂，并输入＂:wq＂后回车保存后退出。



执行gs\_dump，导出postgres数据库信息，但不导出MPPDB\_temp.sql中指定的表信息。

[omm@ecs-bb09 ~]$ **gs\_dump -U omm -W Bigdata@123 -p 26000 postgres --exclude-table-file=/home/omm/logical/MPPDB\_temp.sql -f /home/omm/logical/backup/MPPDB\_backup2.sql**

运行后的结果为：

gs\_dump[port='26000'][postgres][2020-08-15 20:15:24]: The total objects number is 403.

gs\_dump[port='26000'][postgres][2020-08-15 20:15:24]: [100.00%] 403 objects have been dumped.

gs\_dump[port='26000'][postgres][2020-08-15 20:15:24]: dump database postgres successfully

gs\_dump[port='26000'][postgres][2020-08-15 20:15:24]: total time: 238 ms

查看生成的文件信息。

[omm@ecs-bb09 ~]$ **ll /home/omm/logical/backup/**

total 592K

-rw------- 1 omm dbgrp 939 Aug 15 19:54 bkp\_shl2.sql

drwx------ 2 omm dbgrp 4.0K Aug 15 19:39 MPPDB\_backup

-rw------- 1 omm dbgrp 110K Aug 15 20:15 MPPDB\_backup2.sql

-rw------- 1 omm dbgrp 112K Aug 15 19:38 MPPDB\_backup.dmp

-rw------- 1 omm dbgrp 110K Aug 15 19:33 MPPDB\_backup.sql

-rw------- 1 omm dbgrp 248K Aug 15 19:36 MPPDB\_backup.tar

查看生成的sql文件，确认没有备份customer\_t1表。

[omm@ecs-bb09 ~]$ **cat /home/omm/logical/backup/MPPDB\_backup2.sql**

--

-- PostgreSQL database dump

--

SET statement\_timeout = 0;

SET xmloption = content;

SET client\_encoding = 'SQL\_ASCII';

SET standard\_conforming\_strings = on;

SET check\_function\_bodies = false;

SET client\_min\_messages = warning;

--

-- Name: postgres; Type: COMMENT; Schema: -; Owner: omm

--

COMMENT ON DATABASE postgres IS 'default administrative connection database';

--

-- Name: lucy; Type: SCHEMA; Schema: -; Owner: lucy

--

CREATE SCHEMA lucy;

ALTER SCHEMA lucy OWNER TO lucy;

SET search\_path = lucy;

SET default\_tablespace = '';

SET default\_with\_oids = false;

--

-- Name: mytable; Type: TABLE; Schema: lucy; Owner: lucy; Tablespace:

--

CREATE TABLE mytable (

firstcol integer

)

WITH (orientation=row, compression=no);

ALTER TABLE lucy.mytable OWNER TO lucy;

SET search\_path = public;

REVOKE ALL ON SCHEMA public FROM PUBLIC;

REVOKE ALL ON SCHEMA public FROM omm;

GRANT ALL ON SCHEMA public TO omm;

GRANT USAGE ON SCHEMA public TO PUBLIC;

--

-- PostgreSQL database dump complete

--

### gs\_dumpall

gs\_dumpall是openGauss用于导出所有数据库相关信息工具，它可以导出openGauss数据库的所有数据，包括默认数据库postgres的数据、自定义数据库的数据、以及openGauss所有数据库公共的全局对象。

gs\_dumpall在导出openGauss所有数据库时分为两部分：

gs\_dumpall自身对所有数据库公共的全局对象进行导出，包括有关数据库用户和组，表空间以及属性（例如，适用于数据库整体的访问权限）信息。

gs\_dumpall通过调用gs\_dump来完成openGauss中各数据库的SQL脚本文件导出，该脚本文件包含将数据库恢复为其保存时的状态所需要的全部SQL语句。

以上两部分导出的结果为纯文本格式的SQL脚本文件，使用gsql运行该脚本文件可以恢复openGauss数据库。

**参数说明：**

gs\_dumpall参数说明

|  |  |
| --- | --- |
| 参数 | 参数说明 |
| -f, –filename=FILEN | 将输出发送至指定文件。如果这里省略，则使用标准输出。 |
| -a, –data-only | 只转储数据，不转储模式（数据定义）。 |
| -c, –clean | 在重新创建数据库之前，执行SQL语句清理（删除）这些数据库。针对角色和表空间的转储命令已添加。 |
| -g, –globals-only | 只转储全局对象（角色和表空间），无数据库。 |
| -s, –schema-only | 只转储对象定义（模式），而非数据。 |
| -t, –tablespaces-only | 只转储表空间，不转储数据库或角色。 |
| -h, –host | 指定主机的名称。该参数只针对openGauss外，对openGauss内本机只能用127.0.0.1。 |
| -l, –database | 指定所连接的转储全局对象的数据库名称，并去寻找还有其他哪些数据库需要被转储。如果没有指定，会使用postgres数据库，如果postgres数据库不存在，会使用template1。 |
| -p, –port | 指定服务器所监听的TCP端口或本地Unix域套接字后缀，以确保连接。默认值设置为PGPORT环境变量。在开启线程池情况下，建议使用 pooler port，即监听端口+1。 |
| -U, –username | 所连接的用户名。 |
| -W, –password | 指定用户连接的密码。如果主机的认证策略是trust，则不会对系统管理员进行密码验证，即无需输入-W选项；如果没有-W选项，并且不是系统管理员，＂Dump Restore工具＂会提示用户输入密码。 |

#### 操作步骤：

切换到omm用户，以操作系统用户omm登录数据库主节点。（连续实验可跳过此步骤）

[root@ecs-bb09 ~]$ **su - omm**

使用gs\_dumpall一次导出openGauss的所有数据库。

[omm@ecs-bb09 ~]$ **gs\_dumpall -f /home/omm/logical/backup/bkp2.sql -p 26000**

结果显示为:

gs\_dump[port='26000'][dbname='postgres'][2020-07-28 20:39:29]: The total objects number is 345.

gs\_dump[port='26000'][dbname='postgres'][2020-07-28 20:39:29]: [100.00%] 345 objects have been dumped.

gs\_dump[port='26000'][dbname='postgres'][2020-07-28 20:39:29]: dump database dbname='postgres' successfully

gs\_dump[port='26000'][dbname='postgres'][2020-07-28 20:39:29]: total time: 172 ms

gs\_dumpall[port='26000'][2020-07-28 20:39:29]: dumpall operation successful

gs\_dumpall[port='26000'][2020-07-28 20:39:29]: total time: 213 ms

查看生成的文件信息。

[omm@ecs-bb09 ~]$ **ll /home/omm/logical/backup/**

total 48K

-rw------- 1 omm dbgrp 6.2K Feb 5 15:36 bkp2.sql

-rw------- 1 omm dbgrp 934 Feb 5 15:33 bkp\_shl2.sql

drwx------ 2 omm dbgrp 4.0K Feb 5 15:32 MPPDB\_backup

-rw------- 1 omm dbgrp 3.0K Feb 5 15:35 MPPDB\_backup2.sql

-rw------- 1 omm dbgrp 4.1K Feb 5 15:32 MPPDB\_backup.dmp

-rw------- 1 omm dbgrp 3.0K Feb 5 15:30 MPPDB\_backup.sql

-rw------- 1 omm dbgrp 14K Feb 5 15:31 MPPDB\_backup.tar

### gs\_restore

gs\_restore是openGauss提供的针对gs\_dump导出数据的导入工具。通过此工具可由gs\_dump生成的导出文件进行导入。

gs\_restore工具由操作系统用户omm执行。

主要功能包含：

导入到数据库：

如果连接参数中指定了数据库，则数据将被导入到指定的数据库中。其中，并行导入必须指定连接的密码。

导入到脚本文件：

如果未指定导入数据库，则创建包含重建数据库所必须的SQL语句脚本并写入到文件或者标准输出。等效于直接使用gs\_dump导出为纯文本格式。

**参数说明：**

gs\_restore参数说明

|  |  |
| --- | --- |
| 参数 | 参数说明 |
| -d, –dbname=NAME | 连接数据库dbname并直接导入到该数据库中。 |
| -f, –file=FILENAME | 指定生成脚本的输出文件，或使用-l时列表的输出文件。  默认是标准输出。 |
| -F, –format=c|d|t | 指定归档格式。由于gs\_restore会自动决定格式，因此不需要指定格式。  取值范围：  c/custom：该归档形式为gs\_dump的自定义格式。  d/directory：该归档形式是一个目录归档形式。  t/tar：该归档形式是一个tar归档形式。 |
| -a, -data-only | 只导入数据，不导入模式（数据定义）。gs\_restore的导入是以追加方式进行的。 |
| -c, –clean | 在重新创建数据库对象前，清理（删除）已存在于将要还原的数据库中的数据库对象。 |
| -e, –exit-on-error | 当发送SQL语句到数据库时如果出现错误，请退出。默认状态下会继续，且在导入后会显示一系列错误信息。 |
| -n, –schema=NAME | 只导入已列举的模式中的对象。该选项可与-t选项一起用以导入某个指定的表。  多次输入-n \_schemaname\_可以导入多个模式。 |
| -s, –schema-only | 只导入模式（数据定义），不导入数据（表内容）。当前的序列值也不会导入。 |
| -t, –table=NAME | 只导入已列举的表定义、数据或定义和数据。该选项与-n选项同时使用时，用来指定某个模式下的表对象。-n参数不输入时，默认为PUBLIC模式。多次输入-n -t可以导入指定模式下的多个表。 |
| -h, --host=HOSTNAME | 指定的主机名称。如果取值是以斜线开头，他将用作Unix域套接字的目录。默认值取自PGHOST环境变量；如果没有设置，将启动某个Unix域套接字建立连接。  该参数只针对openGauss外，对openGauss内本机只能用127.0.0.1。 |
| -p, --port=PORT | 指定服务器所监听的TCP端口或本地Unix域套接字后缀，以确保连接。默认值设置为PGPORT环境变量。  在开启线程池情况下，建议使用 pooler port，即监听端口+1。 |
| -U, --username=NAME | 所连接的用户名。 |
| -W, --password=PASSWORD | 指定用户连接的密码。如果主机的认证策略是trust，则不会对系统管理员进行密码验证，即无需输入-W参数；如果没有-W参数，并且不是系统管理员，＂gs\\_restore＂会提示用户输入密码。 |

#### gs\_restore导入示例1

执行gs\_restore，将导出的**MPPDB\_backup.tar文件（tar格式）**导入到db\_tpcc01数据库。

切换到omm用户，以操作系统用户omm登录数据库主节点。（连续实验可跳过此步骤）

[root@ecs-bb09 ~]$ **su - omm**

连接openGauss数据库。

[omm@ecs-bb09 ~]$ **gsql -d postgres -p 26000 -r**

创建数据库。

postgres=# **DROP DATABASE IF EXISTS db\_tpcc01;**

postgres=# **CREATE DATABASE db\_tpcc01;**

当结果显示为如下信息，则表示创建成功。

CREATE DATABASE

退出数据库。

postgres=# **\q**

执行gs\_restore，将导出的MPPDB\_backup.tar文件（tar格式）导入到db\_tpcc01数据库。（截图）

[omm@ecs-bb09 ~]$ **gs\_restore /home/omm/logical/backup/MPPDB\_backup.tar -p 26000 -d db\_tpcc01**

如果成功，显示如下：

start restore operation ...

table customer\_t1 complete data imported !

Finish reading 11 SQL statements!

end restore operation ...

restore operation successful

total time: 19 ms

连接db\_tpcc01数据库。

[omm@ecs-bb09 ~]$ **gsql -d db\_tpcc01 -p 26000 -r**

查看数据库中恢复的customer\_t1表。

db\_tpcc01=# **select \* from customer\_t1;**

c\_customer\_sk | c\_customer\_id | c\_first\_name | c\_last\_name

---------------+---------------+--------------+-------------

6885 | maps | Joes |

4321 | tpcds | Lily |

9527 | world | James |

3769 | hello | |

(4 rows)

退出db\_tpcc01数据库。

db\_tpcc01=# **\q**

#### gs\_restore导入示例2

执行gs\_restore，将导出的**MPPDB\_backup.dmp文件（自定义归档格式）**导入到db\_tpcc02数据库。

切换到omm用户，以操作系统用户omm登录数据库主节点。（连续实验可跳过此步骤）

[root@ecs-bb09 ~]$ **su - omm**

连接openGauss数据库。

[omm@ecs-bb09 ~]$ **gsql -d postgres -p 26000 -r**

创建数据库。

postgres=# **DROP DATABASE IF EXISTS db\_tpcc02;**

postgres=# **CREATE DATABASE db\_tpcc02;**

当结果显示为如下信息，则表示创建成功。

CREATE DATABASE

退出数据库。

postgres=# **\q**

执行gs\_restore，将导出的MPPDB\_backup.dmp文件（自定义归档格式）导入到db\_tpcc02数据库。

[omm@ecs-bb09 ~]$ **gs\_restore /home/omm/logical/backup/MPPDB\_backup.dmp -p 26000 -d db\_tpcc02**

如果成功，显示如下：

start restore operation ...

table customer\_t1 complete data imported !

Finish reading 11 SQL statements!

end restore operation ...

restore operation successful

total time: 19 ms

连接db\_tpcc02数据库。

[omm@ecs-bb09 ~]$ **gsql -d db\_tpcc02 -p 26000 -r**

查看数据库中恢复的customer\_t1表。

db\_tpcc02=# **select \* from customer\_t1;**

c\_customer\_sk | c\_customer\_id | c\_first\_name | c\_last\_name

---------------+---------------+--------------+-------------

6885 | maps | Joes |

4321 | tpcds | Lily |

9527 | world | James |

3769 | hello | |

(4 rows)

退出db\_tpcc02数据库。

db\_tpcc02=# **\q**

#### gs\_restore导入示例3

执行gs\_restore，将导出的**MPPDB\_backup文件（目录格式）**导入到postgres数据库。

切换到omm用户，以操作系统用户omm登录数据库主节点。

[root@ecs-bb09 ~]$ **su - omm**

连接openGauss数据库。

[omm@ecs-bb09 ~]$ **gsql -d postgres -p 26000 -r**

创建数据库。

postgres=# **DROP DATABASE IF EXISTS db\_tpcc03;**

postgres=#  **CREATE DATABASE db\_tpcc03;**

当结果显示为如下信息，则表示创建成功。

CREATE DATABASE

退出数据库。

postgres=# **\q**

执行gs\_restore，将导出的MPPDB\_backup文件（目录格式）导入到db\_tpcc03数据库。

[omm@ecs-bb09 ~]$ **gs\_restore /home/omm/logical/backup/MPPDB\_backup -p 26000 -d db\_tpcc03**

如果成功，显示如下：

start restore operation ...

table customer\_t1 complete data imported !

Finish reading 11 SQL statements!

end restore operation ...

restore operation successful

total time: 19 ms

连接db\_tpcc03数据库。

[omm@ecs-bb09 ~]$ **gsql -d db\_tpcc03 -p 26000 -r**

查看数据库中恢复的customer\_t1表。

db\_tpcc02=# **select \* from customer\_t1;**

c\_customer\_sk | c\_customer\_id | c\_first\_name | c\_last\_name

---------------+---------------+--------------+-------------

6885 | maps | Joes |

4321 | tpcds | Lily |

9527 | world | James |

3769 | hello | |

(4 rows)

退出db\_tpcc03数据库。

db\_tpcc03=# **\q**

#### gs\_restore导入示例4

执行gs\_restore，使用自定义归档格式的MPPDB\_backup.dmp文件来进行如下导入操作。导入时在重新创建数据库对象前，清理（删除）已存在于将要还原的数据库中的数据库对象。

展示报错信息。

由于在备份示例1已经将所有对象恢复到db\_tpcc01数据库中，所以在再次导入时如果不先删除已经存在的对象，会出现报错信息。

[omm@ecs-bb09 ~]$ **gs\_restore -W Bigdata@123 /home/omm/logical/backup/MPPDB\_backup.dmp -p 26000 -d db\_tpcc01**

显示如下：

start restore operation ...

……

Error from TOC entry 601; 1259 41883 TABLE customer\_t1 omm

could not execute query: ERROR: relation "customer\_t1" already exists

Command was: CREATE TABLE customer\_t1 (

c\_customer\_sk integer,

c\_customer\_id character(5),

c\_first\_name character(6),

c\_l...

table customer\_t1 complete data imported !

Finish reading 11 SQL statements!

end restore operation ...

WARNING: errors ignored on restore: 3

restore operation successful

total time: 19 ms

连接db\_tpcc01数据库。

[omm@ecs-bb09 ~]$ **gsql -d db\_tpcc01 -p 26000 -r**

查看数据库中恢复的customer\_t1表。

db\_tpcc01=# **select \* from customer\_t1;**

c\_customer\_sk | c\_customer\_id | c\_first\_name | c\_last\_name

---------------+---------------+--------------+-------------

6885 | maps | Joes |

4321 | tpcds | Lily |

9527 | world | James |

3769 | hello | |

6885 | maps | Joes |

4321 | tpcds | Lily |

9527 | world | James |

3769 | hello | |

(8 rows)

原有数据表并没有被删除，导入的数据表以数据追加的方式导入数据。

退出数据库。

db\_tpcc01=# **\q**

输入以下命令，使用-c参数手工删除依赖，导入完成后再重新创建。

[omm@ecs-bb09 ~]$ **gs\_restore -W Bigdata@123 /home/omm/logical/backup/MPPDB\_backup.dmp -p 26000 -d db\_tpcc01 -e -c**

显示如下：

start restore operation ...

Finish reading 11 SQL statements!

end restore operation ...

restore operation successful

total time: 14 ms

说明：如果原对象存在跨模式的依赖则需手工强制干预。

连接db\_tpcc01数据库。

[omm@ecs-bb09 ~]$ **gsql -d db\_tpcc01 -p 26000 -r**

查看数据库中恢复的customer\_t1表。

db\_tpcc01=# **select \* from customer\_t1;**

c\_customer\_sk | c\_customer\_id | c\_first\_name | c\_last\_name

---------------+---------------+--------------+-------------

6885 | maps | Joes |

4321 | tpcds | Lily |

9527 | world | James |

3769 | hello | |

(8 rows)

退出数据库。

db\_tpcc01=# **\q**

原有数据表已经被清除，恢复的是导入的数据表。

#### gs\_restore导入示例5

执行gs\_restore，使用自定义归档格式的MPPDB\_backup.dmp文件来进行如下导入操作。**只导入PUBLIC模式下表customer\_t1的定义。**

切换到omm用户，以操作系统用户omm登录数据库主节点。

[root@ecs-bb09 ~]$ **su - omm**

连接openGauss数据库。

[omm@ecs-bb09 ~]$ **gsql -d postgres -p 26000 -r**

创建数据库。

postgres=# **DROP DATABASE IF EXISTS db\_tpcc04;**

postgres=#  **CREATE DATABASE db\_tpcc04;**

当结果显示为如下信息，则表示创建成功。

CREATE DATABASE

退出数据库。

postgres=# **\q**

执行gs\_restore，只导入PUBLIC模式下表customer\_t1的定义。

[omm@ecs-bb09 ~]$ **gs\_restore /home/omm/logical/backup/MPPDB\_backup.dmp -p 26000 -d db\_tpcc04 -n public -t customer\_t1**

如果成功，显示如下：

start restore operation ...

table customer\_t1 complete data imported !

Finish reading 17 SQL statements!

end restore operation ...

restore operation successful

total time: 18 ms

连接db\_tpcc04数据库。

[omm@ecs-bb09 ~]$ **gsql -d db\_tpcc04 -p 26000 -r**

查看数据库中恢复的customer\_t1表。

db\_tpcc04# **select \* from customer\_t1;**

c\_customer\_sk | c\_customer\_id | c\_first\_name | c\_last\_name

---------------+---------------+--------------+-------------

6885 | maps | Joes |

4321 | tpcds | Lily |

9527 | world | James |

3769 | hello | |

(4 rows)

查看postgres数据库中LUCY模式下的表。

db\_tpcc04=# **select \* from lucy.mytable;**

显示结果如下：

ERROR: schema "lucy" does not exist

LINE 1: select \* from lucy.mytable;

查看postgres数据库中PUBLIC模式下的customer\_t2表。

db\_tpcc04=# **select \* from customer\_t2;**

显示结果如下：

ERROR: relation "customer\_t2" does not exist on dn\_6001

LINE 1: select \* from customer\_t2;

说明只恢复了PUBLIC shema模式下的的customer\_t1表。

退出数据库。

postgres=# **\q**

本实验结束。

# 附录一：Linux操作系统相关命令

Linux中的命令格式为：command [options] [arguments] 中括号表示可选的，即有些命令不需要选项也不需要参数，但有的命令在运行时需要多个选项或参数。

* options(选项)：选项是调整命令执行行为的开关，选项的不同决定了命令的显示结果不同。
* agruments(参数)：参数是指命令的作用对象。

## vi/vim

文本编辑器，若文件存在则是编辑，若不存在则是创建并编辑文本。

命令语法：

vim [参数]

参数说明：可编辑的文件名。

命令示例：

* 编辑名为clusterconfig的xml文本：

vim clusterconfig.xml

注：

vim编辑器有以下三种模式：

* 正常模式：其它模式下按Esc或Ctrl+[进入，左下角显示文件名或为空。
* 插入模式：正常模式下按i键进入，左下角显示--INSERT--。
* 可视模式：正常模式下按v键进入，左下角显示--VISUAL--。

退出命令（正常模式下）：

* :wq 保存并退出。
* :q! 强制退出并忽略所有更改。
* :e! 放弃所有修改，并打开原有文件。

## cd

显示当前目录的名称，或切换当前的目录（打开指定目录）。

命令语法：

cd [参数]

参数说明：

* 无参数：切换用户当前目录。
* . ：表示当前目录；
* .. ：表示上一级目录；
* ~ ：表示home目录；
* / ：表示根目录。

命令示例：

* 切换到usr目录下的bin目录中：

cd /usr/bin

* 切换到用户home目录：

cd

* 切换到当前目录(cd后面接一个.)：

cd .

* 切换到当前目录上一级目录(cd后面接两个.)：

cd ..

* 切换到用户home目录：

cd ~

* 切换到根目录下：

cd /

注：切换目录需要理解绝对路径和相对路径这两个概念。

* 绝对路径：在Linux中，绝对路径是从/（即根目录）开始的，例如 /opt/software、/etc/profile, 如果目录以 / 就是绝对目录。
* 相对路径：是以 . 或 .. 开始的目录。 . 表示用户当前操作所在的位置，而 .. 表示上级目录。例如 ./gs\_om 表示当前目录下的文件或者目录。

## mv

文件或目录改名(move (rename) files)或将文件或目录移入其它位置,经常用来备份文件或者目录。

命令语法：

mv [选项] 参数1 参数2

常用选项：

* -b：若需覆盖文件，则覆盖前先行备份。

参数说明：

* 参数1：源文件或目录。
* 参数2：目标文件或目录。

命令示例：

* 将文件python重命名为python.bak：

mv python python.bak

* 将/physical/backup目录下的所有文件和目录移到/data/dbn1目录下：

mv /physical/backup/\* /data/dbn1

## curl

在Linux中curl是一个利用URL规则在命令行下工作的文件传输工具。支持文件的上传和下载，是综合传输工具。

命令语法：

curl [选项] [URL]

常用选项：

* -A/--user-agent <string>：设置用户代理发送给服务器；
* -C/--continue-at <offset>：断点续转；
* -D/--dump-header <file>：把header信息写入到该文件中；
* -e/--referer：来源网址；
* -o/--output：把输出写到该文件中；
* -O/--remote-name：把输出写到该文件中，保留远程文件的文件名；
* -s/--silent：静默模式。不输出任何东西；
* -T/--upload-file <file>：上传文件；
* -u/--user <user[:password]>：设置服务器的用户和密码；
* -x/--proxy <host[:port]>：在给定的端口上使用HTTP代理；
* -#/--progress-bar：进度条显示当前的传送状态。

参数说明：

* URL：指定的文件传输URL地址。

命令示例：

* 将url(<https://mirrors.huaweicloud.com/repository/conf/CentOS-7-anon.repo)的内容保存到/etc/yum.repos.d/CentOS-Base.repo>文件中。

curl -o /etc/yum.repos.d/CentOS-Base.repo <https://mirrors.huaweicloud.com/repository/conf/CentOS-7-anon.repo>

* 如果在传输过程中掉线，可以使用-C的方式进行续传。

curl -C -O <https://mirrors.huaweicloud.com/repository/conf/CentOS-7-anon.repo>

## yum

Shell 前端软件包管理器。基于 RPM 包管理，能够从指定的服务器自动下载 RPM 包并且安装，可以自动处理依赖性关系，并且一次安装所有依赖的软体包，无须繁琐地一次次下载和安装。

命令语法：

yum [options] [command] [package ...]

常用选项：

* -h：查看帮助；
* -y：当安装过程提示选择全部为 ＂yes＂；
* -q：不显示安装的过程。

参数说明：

* command：要进行的操作。
* package：安装的包名。

命令示例：

* 列出所有可更新的软件清单命令：

yum check-update

* 更新所有软件命令：

yum update

* 列出所有可安裝的软件清单命令：

yum list

* 安装指定的软件：

yum install -y libaio-devel flex bison ncurses-devel glibc.devel patch lsb\_release wget python3

## wget

wget是Linux下下载文件的最常用命令。wget支持HTTP,HTTPS和FTP协议,支持自动下载,即可以在用户退出系统后在后台执行,直到下载结束。

命令语法:

wget [选项] [URL]

常用选项:

* -c：接着下载没下载完的文件；
* -b：启动后转入后台执行；
* -P：指定下载目录；
* -O：变更下载文件名；
* --ftp-user --ftp-password：使用FTP用户认证下载。

参数说明:

* 指定的文件下载URL地址。

命令示例：

* 下载openGauss数据库安装文件到当前文件夹：

wget <https://opengauss.obs.cn-south-1.myhuaweicloud.com/1.0.0/x86/openGauss-1.0.0-CentOS-64bit.tar.gz>

* 使用wget断点续传：

wget –c <https://opengauss.obs.cn-south-1.myhuaweicloud.com/1.0.0/x86/openGauss-1.0.0-CentOS-64bit.tar.gz>

## ln

为某一个文件在另外一个位置建立一个同步的链接（软硬链接，不带选项为硬链接）。

当需要在不同的目录，用到相同的文件时，就不需要在每一个需要要的目录下都放一个必须相同的文件，我们只要在某个固定的目录，放上该文件，然后在 其它的目录下用ln命令链接（link）它就可以，不必重复的占用磁盘空间。

命令语法：

ln [选项] 参数1 参数2

常用选项：

* -b --删除，覆盖以前建立的链接；
* -d --允许超级用户制作目录的硬链接；
* -s --软链接(符号链接)。

参数说明：

* 参数1：源文件或目录。
* 参数2：被链接的文件或目录。

命令示例：

* 为python3文件创建软链接/usr/bin/python，如果python3丢失，/usr/bin/python将失效：

ln -s python3 /usr/bin/python

* 为python3创建硬链接/usr/bin/python，python3与/usr/bin/python的各项属性相同：

ln python3 /usr/bin/python

## mkdir

创建指定的名称的目录，要求创建目录的用户在当前目录中具有写权限，并且指定的目录名不能是当前目录中已有的目录。

命令语法：

mkdir [选项] [参数]

常用选项：

* -p --可以是一个路径名称。此时若路径中的某些目录尚不存在,加上此选项后,系统将自动建立好那些尚不存在的目录,即一次可以建立多个目录（递归）；
* -v --每次创建新目录都显示信息；
* -m --设定权限<模式> (类似chmod)，而不是rwxrwxrwx减umask。

参数说明：

* 需要创建的目录。

命令示例：

* 创建一个空目录：

mkdir test

* 递归创建多个目录：

mkdir -p /opt/software/openGauss

* 创建权限为777的目录(目录的权限为rwxrwxrwx)：

mkdir –m 777 test

## chmod命令

更改文件权限。

命令语法：

chmod [选项] <mode> <file...>

常用选项：

* -R, --以递归的方式对目前目录下的所有文件与子目录进行相同的权限变更。

参数说明：

* mode：权限设定字串，详细格式如下 ：

[ugoa...][[+-=][rwxX]...][,...]，

其中，[ugoa...]：u 表示该档案的拥有者，g 表示与该档案的拥有者属于同一个群体(group)者，o 表示其他以外的人，a 表示所有（包含上面三者）；[+-=]：+ 表示增加权限，- 表示取消权限，= 表示唯一设定权限；[rwxX]：r 表示可读取，w 表示可写入，x 表示可执行，X表示只有当该档案是个子目录或者该档案已经被设定过为可执行。

* file：文件列表（单个或者多个文件、文件夹）。

命令示例：

* 设置所有用户可读取文件 cluterconfig.xml：

chmod ugo+r cluterconfig.xml

或

chmod a+r cluterconfig.xml

* 设置当前目录下的所有档案与子目录皆设为任何人可读写：

chmod -R a+rw \*

数字权限使用格式：

* 这种使用方式中，规定数字4、2和1表示读、写、执行权限，即r=4,w=2,x=1。
* 例：rwx = 7（4+2+1）；rw = 6（4+2）；r-x = 5 （4+0+1）；r-- = 4（4+0+0）；--x = 1（0+0+1）；

每个文件都可以针对三个粒度，设置不同的rwx(读写执行)权限。即我们可以用用三个8进制数字分别表示 拥有者 、群组 、其它组( u、 g 、o)的权限详情，并用chmod直接加三个8进制数字的方式直接改变文件权限。语法格式为 ：

chmod <abc> file...

其中，a,b,c各为一个数字，分别代表User、Group、及Other的权限，相当于简化版的chmod u=权限,g=权限,o=权限 file...，而此处的权限将用8进制的数字来表示User、Group、及Other的读、写、执行权限。

命令示例：

* 赋予cluterconfig.xml文件可读可写可执行权限（所有权限）：

chmod 777 cluterconfig.xml

* 赋予/opt/software/openGauss目录下所有文件及其子目录 用户所有权限组可读可执行权限，其他用户可读可执行权限：

chmod R 755 /opt/software/openGauss

## chown

利用 chown 将指定文件的拥有者改为指定的用户或组，用户可以是用户名或者用户ID；组可以是组名或者组ID；文件是以空格分开的要改变权限的文件列表，支持通配符。只有系统管理者(root)才有这样的权限。使用权限 : **root**。

命令语法：

chown [选项] user[:group] file...

常用选项:

* -c : 显示更改的部分的信息；
* -f : 忽略错误信息；
* -R : 处理指定目录以及其子目录下的所有文件。

参数说明

* user : 新的文件拥有者的使用者 ID。
* group : 新的文件拥有者的使用者组(group)。
* flie：文件。

命令示例：

* 将文件 file1.txt 的拥有者设为omm，群体的使用者dbgrp:

chown omm:dbgrp /opt/software/openGauss/clusterconfig.xml

* 将目前目录下的所有文件与子目录的拥有者皆设为omm，群体的使用者dbgrp:

chown -R omm:dbgrp \*

## ls

列出文件和目录的内容。

命令语法：

ls [选项] [参数]

常用选项：

* -l --以长格式显示，列出文件的详细信息，如创建者，创建时间，文件的读写权限列表等等；
* -a --列出文件下所有的文件，包括以＂.＂和＂..＂开头的隐藏文件 （Linux下文件隐藏文件是以 .开头的，如果存在 .. 代表存在着父目录）；
* -d --列出目录本身而非目录内的文件，通常要与-l一起使用；
* -R --同时列出所有子目录层，与-l相似，只是不显示出文件的所有者，相当于编程中的＂递归＂实现；
* -t --按照时间进行文件的排序，Time（时间）；
* -s --在每个文件的后面打印出文件的大小，size（大小）；
* -S --以文件的大小进行排序。

参数说明：

* 目录或文件。

命令示例：

* 以长格式列出当前目录中的文件及目录：

ls -l

## cp

复制文件或者目录。

命令语法：

cp [选项] 参数1 参数2

常用选项：

* -f --如果目标文件无法打开则将其移除并重试(当 -n 选项存在时则不需再选此项)；
* -n --不要覆盖已存在的文件(使前面的 -i 选项失效)；
* -I --覆盖前询问(使前面的 -n 选项失效)；
* -p --保持指定的属性(默认：模式,所有权,时间戳)，如果可能保持附加属性：环境、链接、xattr 等；
* -R,-r --复制目录及目录内的所有项目。

参数说明：

* 参数1：源文件。
* 参数2：目标文件。

命令示例：

* 将home目录中的abc文件复制到opt目录下：

cp /home/abc /opt

注：目标文件存在时，会询问是否覆盖。这是因为cp是cp -i的别名。目标文件存在时，即使加了-f标志，也还会询问是否覆盖。

## rm

删除一个目录中的一个或多个文件或目录，它也可以将某个目录及其下的所有文件及子目录均删除。对于链接文件，只是删除了链接，原有文件均保持不变。

rm是一个危险的命令，使用的时候要特别当心，否则整个系统就会毁在这个命令（比如在/（根目录）下执行rm \* rf）。所以，我们在执行rm之前最好先确认一下在哪个目录，到底要删除什么东西，操作时保持高度清醒的头脑。

命令语法：

rm [选项] 文件

常用选项：

* -f --忽略不存在的文件，从不给出提示；
* -r --指示rm将参数中列出的全部目录和子目录均递归地删除。

参数说明：

* 需要删除的文件或目录。

命令示例：

* 删除文件：

rm qwe

注：输入rm qwe命令后，系统会询问是否删除，输入y后就会删除文件，不想删除文件则输入n。

* 强制删除某个文件：

rm-rf clusterconfig.log

## cat

连接文件并在标准输出上输出。这个命令常用来显示文件内容，或者将几个文件连接起来显示，或者从标准输入读取内容并显示，它常与重定向符号配合使用。

命令语法：

cat [选项] [参数]

常用选项：

* -E --在每行结束显示$；
* -n –由1开始对给所有输出行编号；
* -b 或 --number-nonblank：和 -n 相似，只不过对于空白行不编号；
* -v --使用 ^ 和 M- 符号，除了 LFD 和 TAB 之外。

参数说明：

* 可操作的文件名。

命令示例:

* 显示testfile文件的内容：

cat textfile

* 把 textfile1 和 textfile2 的文档内容加上行号（空白行不加）之后将内容追加到 textfile3 文档里：

cat -b textfile1 textfile2 >> textfile3

* 向/etc/profile中追加内容（输入EOF表示结束追加）：

cat >>/etc/profile<<EOF

>export LD\_LIBRARY\_PATH=$packagePath/script/gspylib/clib:$LD\_LIBRARY\_PATH

>EOF

注：

* EOF是end of file的缩写，表示＂文字流＂（stream）的结尾。＂文字流＂可以是文件（file），也可以是标准输入（stdin）。在Linux系统之中，EOF是当系统读取到文件结尾，所返回的一个信号值（也就是-1）。

# 附录二：openGauss数据库基本操作

## 查看数据库对象

* 查看帮助信息。

postgres=# \?

* 切换数据库。

postgres=# \c dbname

* 列举数据库。

使用\l元命令查看数据库系统的数据库列表。

postgres=# \l

使用如下命令通过系统表pg\_database查询数据库列表。

postgres=# SELECT datname FROM pg\_database;

* 列举表。

postgres=# \dt

* 列举所有表、视图和索引。

postgres=# \d+

使用gsql的\d+命令查询表的属性。

postgres=# \d+ tablename

* 查看表结构。

postgres=# \d tablename

* 列举schema。

postgres=# \dn

* 查看索引。

postgres=# \di

* 查询表空间。

使用gsql程序的元命令查询表空间。

postgres=# \db

检查pg\_tablespace系统表。如下命令可查到系统和用户定义的全部表空间。

postgres=# SELECT spcname FROM pg\_tablespace;

* 查看数据库用户列表。

postgres=# SELECT \* FROM pg\_user;

* 查看用户属性。

postgres=# SELECT \* FROM pg\_authid;

* 查看所有角色。

postgres=# SELECT \* FROM PG\_ROLES;

## 其他操作

* 查看openGauss支持的所有SQL语句。

postgres=#\h

* 切换用户。

postgres=# \c – username

* 退出数据库。

postgres=# \q