第 3 章

openGauss 开发调试

第 2 章主要介绍了如何将 openGauss 安装在服务器上。本章介绍如何连接到 openGauss 数据库并利用其提供的接口操作和管理数据，包括对存放于 openGauss 中的关系数据库的表格进行增(INSERT)、删(DELETE)、改(UPDATE)、查(SELECT)等操作。本章首先在 3.1 节对 gsql(openGauss structured query language，openGauss 结构化查询语言)客户端连接进行介绍，接着在 3.2 节对通过 GUI 工具 DBeaver 连接数据库进行介绍。实际应用中，在开发者编写程序进行应用程序开发时，往往需要管理大量复杂的数据，调用 openGauss 可以简化自定义数据结构，管理复杂数据的部分工作。最后在 3.3 节和 3.4 节，将会介绍如何在 Java 源码和 C++源码中通过编程语言调用 openGauss 数据库接口对数据进行管理。对于 Java 数据库连接和 C++数据库连接，此处分别使用了前面所述的 JDBC 和 ODBC。

3.1 gsql 客户端连接

通过 gsql 连接 openGauss 主要包含两种方式，分别在 3.1.1 节和 3.1.2 节介绍。而后将会在 3.1.3 节初探通过 gsql 执行 SQL。更多关于 SQL 执行的知识，请参考后续章节。

3.1.1 gsql 本地连接

gsql 是 openGauss 提供的在命令行下运行的数据库连接工具。此工具除了具备操作数据库的基本功能，还提供了若干高级特性，便于用户使用。本书只介绍最基本的 gsql 操作，即如何使用 gsql 连接数据库，关于 gsql 以及 gsql 客户端的更多高级特性，请参考 gsql 用户手册(https://opengauss.org/zh/)。本节先学习如何使用 gsql 本地连接 openGauss。

首先，用户需要确认连接信息。gsql 工具通过数据库主节点连接数据库。因此，连接前，需获取数据库主节点所在服务器的 IP 地址及数据库主节点的端口号信息。在本示例中，以操作系统用户 omm 登录数据库主节点。使用命令如下：

```
gs_om -t status -detail
```

在命令行中将会显示查询得到的 openGauss 各实例，结果如下：

```
[omm@ecs-c32a ~]$ gs_om -t status --detail
[   Cluster State   ]
```

```
cluster_state   : Normal
redistributing  : No
current_az      : AZ_ALL
[   Datanode State   ]
node         node_ip           instance                       state
1   ecs - c32a 192.168.0.40     6001 /gaussdb/data/ecs - c32a P Primary Normal
```

从以上结果可知,部署了数据库主节点实例的服务器 IP 地址为 192.168.0.40,数据库主节点数据路径为"/gaussdb/data/ecs-c32a"。由于在第 2 章中将 openGauss 安装在了单机上,因此此处仅能看见一个主机节点。

接下来用户需要确认数据库主节点的端口号。在上面查询到的数据库主节点数据路径下,保存着配置文件 ***.conf(默认为 postgresql.conf),使用如下命令:

```
cat /gaussdb/data/ecs - c32a/postgresql.conf | grep port
```

查询其中的端口号信息,命令行会显示如下结果:

```
[omm@ecs - c32a ~] $ cat /gaussdb/data/ecs - c32a/testdbql.conf | grep port
port = 26000         # (change requires restart)
#ssl_renegotiation_limit = 0
#amount of data between renegotiations, no longer supported          #tcp_recv_timeout = 0
#SO_RCVTIMEO, specify the receiving timeouts until reporting an error(change requires
restart)
#comm_sctp_port = 1024       # Assigned by installation (change requires restart)
#comm_control_port = 10001   # Assigned by installation (change requires restart)
#supported by the operating system:
#The heartbeat thread will not start if not set localheartbeatport and remoteheartbeatport.
#e.g. 'localhost = xx.xx.xxx.2 localport = 12211 localheartbeatport = 12214 remotehost = xx.
xx.xxx.3 remoteport = 12212 remoteheartbeatport = 12215, localhost = xx.xx.xxx.2 localport =
12213 remotehost = xx.xx.xxx.3 remoteport = 12214'
# %r = remote host and port
alarm_report_interval = 10
support_extended_features = true
```

由此可知端口号为 26000。

默认情况下,客户端连接数据库后处于空闲状态时会根据参数 session_timeout 的默认值自动断开连接。如果要关闭超时设置,设置参数 session_timeout 为 0 即可。以上信息显示成功后,便表示可以正常登录并连接 gsql 了。

接下来需要以操作系统用户 omm 登录数据库主节点连接数据库。如 2.4 节所述,安装时默认生成名称为 testdb 的数据库。连接该数据库的具体命令如下:

```
gsql -d testdb -p 26000
```

其中 testdb 为需要连接的数据库名称，26000 为数据库主节点的端口号。请根据实际情况替换。连接成功后，命令行会显示如下形式的信息：

```
[omm@ecs-c32a ~]$ gsql -d testdb -p 26000
gsql ((openGauss 1.0.1 build 13b34b53) compiled at 2020-10-12 02:00:13 commit 0 last mr  )
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.
testdb=#
```

表示登录成功。omm 用户是管理员用户，因此系统显示 DBNAME=#（此处是 testdb=#）。若使用普通用户身份登录和连接数据库，则系统显示 DBNAME=>。“Non-SSL connection”表示未使用 SSL 方式连接数据库。需要高安全性时，请使用 SSL 连接。首次登录需要修改密码。原始密码为安装 openGauss 数据库时输入的密码，具体请参见在线教程中的相关章节（https://opengauss.org/zh/docs/2.0.0/docs/Quickstart/Quickstart.html/），此处需将原始密码修改为自定义的密码，例如 Mypwd123，命令如下：

```
testdb=# ALTER ROLE omm IDENTIFIED BY 'Mypwd123' REPLACE 'XuanYuan@2012';
```

如需获取帮助信息，可以在命令行中输入 help 命令，得到如下的信息提示：

```
testdb=# help
You are using gsql, the command-line interface to gaussdb.
Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with gsql commands
       \g or terminate with semicolon to execute query
       \q to quit
```

根据提示，\copyright 指令表示版本发布相关信息、\h 指令表示与标准查询语言（SQL）相关的问题、\? 指令表示与 gsql 工具相关的信息、\g 指令用来查询与执行 query 相关的指令，\q 指令用来退出数据库。

在使用 openGauss 遇到问题时，用户要善于通过查询系统文档来获取问题答案。例如，欲查询 openGauss 所包含的 SQL 指令，可以使用\h 得到如下提示：

```
testdb=#  \h
Available help:
ABORT                            ALTER TABLE                      CREATE BARRIER
      CREATE TRIGGER                   DROP ROLE                        PREPARE
 -- MORE --
```

此时按 Enter 键将会显示更多的信息。可以发现，有如上若干条 SQL 语句的帮助说明可供参考，若单纯想了解数据库查询语句 SELECT 的用法，可使用\h SELECT 查看相关文

档,得到此语法的详细使用说明。命令如下:

```
testdb = #   \h SELECT
Command:     SELECT
Description: retrieve rows from a table or view
Syntax:
[ WITH [ RECURSIVE ] with_query [, ...] ]
SELECT [/* + plan_HINT */] [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
    { * | {expression [ [ AS ] output_name ]} [, ...] }
    [ FROM from_item [, ...] ]
    [ WHERE condition ]
    [ GROUP BY grouping_element [, ...] ]
    [ HAVING condition [, ...] ]
    [ WINDOW {window_name AS ( window_definition )} [, ...] ]
    [ { UNION | INTERSECT | EXCEPT | MINUS } [ ALL | DISTINCT ] select ]
    [ ORDER BY {expression [ [ ASC | DESC | USING operator ] | nlssort_expression_clause ]
[ NULLS { FIRST | LAST } ]} [, ...] ]
    [ LIMIT { [offset,] count | ALL } ]
    [ OFFSET start [ ROW | ROWS ] ]
    [ FETCH { FIRST | NEXT } [ count ] { ROW | ROWS } ONLY ]
    [ {FOR { UPDATE | SHARE } [ OF table_name [, ...] ] [ NOWAIT ]} [...] ];
TABLE { ONLY {(table_name)| table_name} | table_name [ * ]};
 -- MORE --
```

3.1.2 gsql 远程连接

本节介绍如何使用 gsql 进行远程连接,使得用户在远程的其他机器上也可以自如地登录和操作服务器上的数据库,而不需要每次都进行 SSH 远程登录,并且不得不把编程工作迁移到安装有 openGauss 的服务器上进行。用户可按照如下顺序操作。

(1) 以操作系统用户 omm 登录数据库主节点。

(2) 配置客户端认证方式,允许客户端以 jack 用户连接到本机,此处远程连接禁止使用 omm 用户(即数据库初始化用户)。例如,配置允许 IP 地址为 10.10.0.30 的客户端访问本机,命令如下:

```
gs_guc set -N all -I all -h "host all jack 10.10.0.30/32 sha256"
```

需要注意的是,在使用 jack 用户前,需先本地连接数据库,并在数据库中使用如下语句创建 jack 用户:

```
testdb = # CREATE USER jack PASSWORD 'Test@123';
```

其中各参数的含义为:

(1) -N all 表示 openGauss 的所有主机。

(2) -I all 表示主机的所有实例。

(3) -h 表示指定需要在 pg_hba.conf 文件中增加的语句。

(4) all 表示允许客户端连接到任意的数据库。

(5) jack 表示连接数据库的用户。

(6) 10.10.0.30/32 表示只允许 IP 地址为 10.10.0.30 的主机连接。此处的 IP 地址不能为 openGauss 内的 IP 地址,在使用过程中,请根据用户的网络配置进行修改。32 表示子网掩码为 1 的位数,即 255.255.255.255。

(7) sha256 表示连接时 jack 用户的密码使用 SHA-256 算法加密。

以上这条命令在数据库主节点实例对应的 pg_hba.conf 文件中添加了一条规则,用于对连接数据库主节点的客户端进行鉴定。pg_hba.conf 文件中的每条记录可以是下面四种格式之一(四种格式的参数说明请参见官网(https://opengauss.org/zh/)中的“管理数据库安全→客户端接入认证→配置文件参考”章节):

(1) local　　DATABASE USER METHOD [OPTIONS]

(2) host　　DATABASE USER ADDRESS METHOD [OPTIONS]

(3) hostssl　　DATABASE USER ADDRESS METHOD [OPTIONS]

(4) hostnossl　DATABASE USER ADDRESS METHOD [OPTIONS]

因为认证时系统将为每个连接请求按照从上到下的顺序检查 pg_hba.conf 中的记录,所以这些记录的顺序是非常关键的。

接下来用户应当安装 gsql 客户端,用于远程连接数据库。在客户端机器上,上传客户端工具包并配置 gsql 的执行环境变量。以 root 用户登录客户端机器,创建“/tmp/tools”目录,并获取软件安装包中的 openGauss-1.0.1-openEuler-64bit-Libpq.tar.gz 上传到/tmp/tools 路径下。该连接工具可以在官网(https://opengauss.org/zh/download.html)下载,如图 3-1 所示。

图 3-1　openGauss 连接程序下载界面

其中,文件“版本 1.0.0 -CentOS_x86_64”针对 Linux 下的 CentOS 发布版。“版本 1.0.0 -openeuler_x86_64”针对 x86_64 架构的操作系统,“版本 1.0.0 -openeuler_aarch64”针对 AArch64 架构的操作系统。不同的操作系统,工具包文件名称会有差异。请根据实际的操

作系统类型选择对应的工具包。可以查看您的服务器或 PC 的本机信息来获取。此处以 x86_64 架构为例，下载好的软件安装包 openGauss-1.0.1-openEuler-64bit-Libpq.tar.gz 放置在/direction/Downloads/下，因此使用如下命令：

```
mkdir /tmp/tools/
cp     /direction/openGauss-1.0.1-openEuler-64bit-Libpq.tar.gz /tmp/tools/
```

（1）解压文件，使用如下命令：

```
cd /tmp/tools
tar -zxvf openGauss-1.0.1-openEuler-64bit-Libpq.tar.gz
```

可观察到如下的文件解压过程：

```
user:~ root# cd /tmp/tools
user:tools root# tar -zxvf openGauss-1.0.1-openEuler-64bit-Libpq.tar.gz
x ./include/
x ./include/libpq-fe.h
x ./include/gs_threadlocal.h
x ./include/gs_thread.h
x ./include/testdb_ext.h
x ./include/libpq/
x ./include/libpq/libpq-fs.h
x ./include/libpq-events.h
x ./lib/
x ./lib/libcrypto.so
x ./lib/libcom_err_gauss.so
x ./lib/libpgport_tool.so
x ./lib/libgssrpc_gauss.so
x ./lib/libgssapi_krb5_gauss.so
x ./lib/libcrypto.so.1.1
x ./lib/libkrb5support_gauss.so.0
x ./lib/libpq.so
x ./lib/libpq.so.5.5
x ./lib/libpq.so.5
x ./lib/libkrb5_gauss.so
x ./lib/libssl.so
x ./lib/libssl.so.1.1
x ./lib/libconfig.so
```

（2）登录数据库主节点所在的服务器，复制数据库安装目录下的 bin 目录到客户端主机的/tmp/tools 路径下，随后继续登录客户端主机执行操作，命令如下：

```
scp -r omm@121.36.214.189:/opt/gaussdb/app/bin /tmp/tools
```

稍候片刻，等待文件传输成功，之后可看到远程传输进度提醒，命令如下：

```
User:tools root# scp -r omm@121.36.214.189:/opt/gaussdb/app/bin /tmp/tools
The authenticity of host '121.36.214.189 (121.36.214.189)' can't be established.
ECDSA key fingerprint is SHA-256:JcqTWaMeHRI7eBage+GND0zWrVFAYkbk0aJSpMnj2Bo.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '121.36.xxx.123' (ECDSA) to the list of known hosts.

Authorized users only. All activities may be monitored and reported.
omm@121.36.214.189's password:
kdb5_util                                     100%  138KB 359.5KB/s   00:00
klist                                         100%   41KB 371.4KB/s   00:00
gs_guc                                        100%  329KB 510.8KB/s   00:00
retry_errcodes.conf                           100%  126     1.6KB/s   00:00
gs_basebackup                                 100%  207KB 608.0KB/s   00:00
openssl                                       100%  828KB 256.5KB/s   00:03
openGauss-Package-bak_13b34b53.tar.gz         100%   68MB   1.6MB/s   00:44
gs_probackup                                  100%  571KB   1.2MB/s   00:00
gs_restore                                    100%  327KB   1.3MB/s   00:00
kadmin.local                                  74%   119KB 600.2KB/s   00:00
```

/opt/huawei/install/app 为 clusterconfig.xml 文件中配置的{gaussdbAppPath}路径，121.36.xxx.123 为客户端主机 IP。如果安装顺利，clusterconfig.xml 文件应当在目录 /opt/software/openGauss 下。打开该文件易找到{gaussdbAppPath}对应的路径，本例中为 /opt/gaussdb/app。新版 openGauss 中，bin 文件夹在/opt/gaussdb 下；旧版 openGauss 中，bin 文件夹在/opt/huawei/install 下。

(3) 设置环境变量。登录客户端主机，打开“~/.bashrc”文件，命令如下：

```
vi ~/.bashrc
```

在其中输入如下内容后，使用“:wq!”命令保存并退出。

```
export PATH=/tmp/tools/bin:$PATH
export LD_LIBRARY_PATH=/tmp/tools/lib:$LD_LIBRARY_PATH
```

(4) 使环境变量配置生效。命令如下：

```
source ~/.bashrc
```

连接数据库。数据库安装完成后，默认生成名称为 testdb 的数据库。第一次连接数据库时，可以连接到此数据库。命令如下：

```
gsql -d testdb -h 10.10.0.11 -U jack -p 8000 -W Test@123
```

testdb 为需要连接的数据库名称，10.10.0.11 为数据库主节点所在的服务器 IP 地址，jack 为连接数据库的用户，8000 为数据库主节点的端口号，Test@123 为连接数据库用户 jack 的密码。连接 openGauss 的机器与 openGauss 不在同一网段时，-h 指定的 IP 地址应为 Manager 界面上所设置的 coo.cooListenIp2(应用访问 IP)的取值。此处禁止使用 omm 用户进行远程连接数据库。

3.1.3 通过 gsql 客户端工具执行 SQL 语句

通过客户端连接 openGauss 后，用户可以使用客户端工具连接到 openGauss 执行单条 SQL 语句和批量 SQL 语句。此处着重展示 SQL 的执行，更多的 SQL 语法细节可参见后续章节。

1. 执行单条 SQL 语句

1) 方法一

首先以操作系统用户 omm 登录数据库主节点，然后使用 gsql 连接到 openGauss 服务器。命令如下：

```
gsql -d testdb -h 10.10.0.11 -U jack -p 8000 -W Test@123
```

其中，-d 参数指定目标数据库名、-U 参数指定数据库用户名、-h 参数指定主机名、-p 参数指定端口号信息。最后执行 SQL 语句，以创建数据库 human_staff 为例，语句如下：

```
CREATE DATABASE human_staff;
```

通常，输入的语句以分号结束。如果输入的语句没有错误，结果就会输出到屏幕上。

2) 方法二

首先以操作系统用户 omm 登录数据库主节点，然后执行如下包含 SQL 语句的命令，可以得到与上文一样的效果：

```
gsql -d testdb -h 10.10.0.11 -U jack -p 8000 -W Test@123 -c 'CREATE DATABASE human_staff';
```

gsql 工具使用-d 参数指定目标数据库名、使用-U 参数指定数据库用户名、使用-h 参数指定主机名、使用-p 参数指定端口号信息，使用-c 参数指定需要执行的 SQL。使用该语句时，用户需要具有相应的权限。

2. 执行批量 SQL 语句

首先以操作系统用户 omm 登录数据库主节点。首先使用 gsql 连接到数据库，然后使用文件作为命令源，而不是交互式输入，gsql 将在处理完文件后结束。

```
gsql -d testdb -h 10.10.0.11 -U jack -p 8000 -f /home/omm/staff.sql
```

gsql 工具使用-d 参数指定目标数据库名、使用-U 参数指定数据库用户名、使用-h 参数指定主机名、使用-p 参数指定端口号信息、使用-f 参数指定文件名(绝对路径或相对路径,且满足操作系统路径命名规则)。本命令中使用 jack 用户连接到远程主机 testdb 数据库的 8000 端口,并采用文件 staff. sql 作为命令源。

3.2　DBeaver 客户端连接

DBeaver 是一款支持多平台的、免费通用的数据库工具,允许查看数据库的结构,执行 SQL 查询和脚本,浏览和导出数据库设计方案(schema),其性能卓越,内存消耗相对较低,支持几乎所有流行的数据库,例如 MySQL、TestdbQL、SQLite、Oracle、DB2、SQL Server 等,同时也支持对 openGauss 的连接。本节将介绍如何通过 DBeaver 连接数据库。

3.2.1　DBeaver 下载

DBeaver 下载的步骤如下:

(1) 从官网(https://dbeaver.io/download/)下载对应操作系统的安装包,下载界面如图 3-2 所示。

图 3-2　DBeaver 客户端下载界面

(2) 以 Mac OS 为例,在下载完 DBeaver Mac 软件后双击 dbeaver-ce-21. 0. 0-macos. dmg,根据操作提示进行安装,如图 3-3 所示。

图 3-3　DBeaver 客户端安装界面

3.2.2　DBeaver 连接

打开 DBeaver 应用软件后，直接单击导航栏上的 File，然后再单击弹出菜单中的新建连接，选择需要连接的数据库类型，如图 3-4 所示。

图 3-4　DBeaver 连接界面

在服务器信息界面中输入服务器地址、端口号、用户名、密码等信息，如图 3-5 所示。

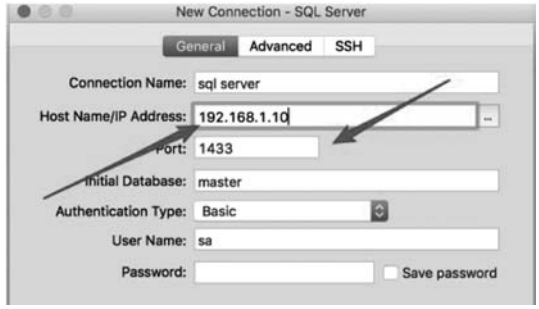

图 3-5　DBeaver 连接信息填写界面

Authentication Type 选择 Basic，单击下方的 test 按钮测试连接是否正常，若连接正常，则显示如图 3-6 所示的结果，此时便可以使用 DBeaver 连接和管理远程服务器的数据库了。

图 3-6　DBeaver 连接成功界面

3.3　openGauss 数据库 JDBC 连接与开发

JDBC(Java database connectivity，Java 数据库连接)是一种用于执行 SQL 语句的 Java API，可以为多种关系数据库提供统一的访问接口，应用程序可基于它操作数据。openGauss 数据库提供了对 JDBC 4.0 特性的支持，需要使用 JDK 1.8 版本编译程序代码，不支持 JDBC 桥接 ODBC 方式。本章只介绍基本概念与一个完整的连接流程，更多示例请参考官网的开发者文档(https://opengauss.org/zh/docs/1.1.0/docs/Developerguide/)。

3.3.1　JDBC 包、驱动类和环境类

(1) JDBC 包：在服务器端源代码目录下执行 build.sh，获得驱动 jar 包 testdbql.jar，包位置在源代码目录下。可以从发布包中获取，包名为 openGauss-x.x.x-操作系统版本号-64bit-Jdbc.tar.gz。

(2) 驱动类：在创建数据库连接之前，需要加载数据库驱动类 org.testdbql.Driver。openGauss 在 JDBC 的使用上与 postgreSQL 的使用方法保持兼容，所以，同时在同一进程内使用两个 JDBC 驱动的时候，可能会出现类名冲突。

(3) 环境类：客户端需配置 JDK 1.8(或更高版本)，首先在命令行窗口输入 java -version，查看 JDK 版本，确认为 JDK 1.8 版本。如果未安装 JDK，请从官方网站下载安装包并安装。如果是在 Linux 或者 Mac OS 上，则在用户根目录的. bash_profile 中加入环境变量信息。在 Windows 中，右击“我的电脑”，选择“属性”。首先在“系统”页面左侧的导航栏中单击“高级系统设置”，然后在“系统属性”页面的“高级”选项卡上单击“环境变量”，最后在“环境变量”页面的“系统变量”区域单击“新建”或“编辑”按钮配置系统变量。

图 3-7 JDBC 的开发流程

3.3.2 JDBC 连接 openGauss 的开发流程

从逻辑层面讲，JDBC 的开发流程如图 3-7 所示。

接下来分别介绍具体的连接数据库操作和关闭连接操作。

在创建数据库连接之前，需要先加载数据库驱动程序。加载驱动程序有两种方法：①在代码中创建连接之前的任意位置隐含装载：Class. forName(org. testdbql. Driver)；②在 JVM 启动时传递参数：java -Djdbc. drivers＝org. testdbql. Driver jdbctest。在创建数据库连接之后，才能使用它执行 SQL 语句操作数据。JDBC 提供了 3 种方法，用于创建数据库连接，代码如下：

```
DriverManager.getConnection(String url);
DriverManager.getConnection(String url, Properties info);
DriverManager.getConnection(String url, String user, String password);
```

下面的示例演示了如何连接 JDBC：

```
public static Connection getConnect(String username, String passwd)
    {
        //驱动类
        String driver = "org.testdbql.Driver";
        //数据库连接描述符
        String sourceURL = "jdbc:testdbql://10.10.0.13:8000/testdb";
        Connection conn = null;
        try
        {
            //加载驱动
            Class.forName(driver);
        }
        catch( Exception e )
        {
            e.printStackTrace();
            return null;
        }
```

```
        try
        {
             //创建连接
            conn = DriverManager.getConnection(sourceURL, username, passwd);
            System.out.println("Connection succeed!");
        }
        catch(Exception e)
        {
            e.printStackTrace();
            return null;
        }

        return conn;
    };
// 以下代码将使用 Properties 对象作为参数建立连接
public static Connection getConnectUseProp(String username, String passwd)
    {
        //驱动类
        String driver = "org.testdbql.Driver";
        //数据库连接描述符
        String sourceURL = "jdbc:testdbql://10.10.0.13:8000/testdb?";
        Connection conn = null;
        Properties info = new Properties();

        try
        {
            //加载驱动
            Class.forName(driver);
        }
        catch( Exception e )
        {
            e.printStackTrace();
            return null;
        }
        try
        {
             info.setProperty("user", username);
             info.setProperty("password", passwd);
             //创建连接
             conn = DriverManager.getConnection(sourceURL, info);
             System.out.println("Connection succeed!");
        }
        catch(Exception e)
        {
            e.printStackTrace();
            return null;
        }
        return conn;
    };
```

在使用数据库连接完成相应的数据操作后,需要关闭数据库连接。关闭数据库连接,直接调用 close()方法即可,代码如下:

```
Connection conn = null;
conn.close();
```

3.3.3 JDBC 连接 openGauss 执行 SQL 语句示例

1. 执行普通的 SQL 语句

应用程序通过执行 SQL 语句操作数据库中的数据(不用传递参数的语句),需要首先调用 Connection 的 createStatement()方法创建语句对象,代码如下:

```
Connection conn = null;
Statement stmt = conn.createStatement();
```

然后调用 Statement 的 executeUpdate()方法执行 SQL 语句,代码如下:

```
int rc = stmt.executeUpdate("CREATE TABLE customer_t1(c_customer_sk INTEGER, c_customer_
name VARCHAR(32));");
```

最后关闭语句对象,代码如下:

```
stmt.close();
```

2. 执行预编译 SQL 语句

预编译语句是只编译和优化一次,然后通过设置不同的参数值多次使用。由于已经预先编译好了,因此后续使用会减少执行时间。如果要多次执行一条语句,首先需要调用 Connection 的 prepareStatement()方法创建预编译语句对象,代码如下:

```
PreparedStatement pstmt = con.prepareStatement("UPDATE customer_t1 SET c_customer_name = ?
WHERE c_customer_sk = 1");
```

接着调用 PreparedStatement 的 setShort 设置参数,代码如下:

```
pstmt.setShort(1, (short)2);
```

然后调用 PreparedStatement 的 executeUpdate()方法执行预编译 SQL 语句,代码如下:

```
int rowcount = pstmt.executeUpdate();
```

最后调用 PreparedStatement 的 close()方法关闭预编译语句对象，代码如下：

```
pstmt.close();
```

3. 调用存储过程

openGauss 支持通过 JDBC 直接调用事先创建的存储过程，首先调用 Connection 的 prepareCall()方法创建调用语句对象，代码如下：

```
Connection myConn = null;
CallableStatement cstmt = myConn.prepareCall("{? = CALL TESTPROC(?,?,?)}");
```

接着调用 CallableStatement 的 setInt()方法设置参数，代码如下：

```
cstmt.setInt(2, 50);
cstmt.setInt(1, 20);
cstmt.setInt(3, 90);
```

调用 CallableStatement 的 registerOutParameter()方法注册输出参数，代码如下：

```
cstmt.registerOutParameter(4, Types.INTEGER);        //注册 out 类型的参数,类型为整型
```

然后调用 CallableStatement 的 execute()方法，代码如下：

```
cstmt.execute();
```

调用 CallableStatement 的 getInt()方法获取输出参数，代码如下：

```
int out = cstmt.getInt(4);                          //获取 out 参数
```

最后调用 CallableStatement 的 close()方法关闭调用语句，代码如下：

```
cstmt.close();
```

4. 执行批处理

用一条预处理语句处理多条相似的数据。数据库只创建一次执行计划，节省了语句的编译和优化时间。首先调用 Connection 的 prepareStatement()方法创建预编译语句对象，代码如下：

```
Connection conn = null;
PreparedStatement pstmt = conn.prepareStatement("INSERT INTO customer_t1 VALUES (?)");
```

接着针对每条数据调用 setShort 设置参数，以及调用 addBatch 确认该条数据设置完

毕，代码如下：

```
pstmt.setShort(1, (short)2);
pstmt.addBatch();
```

然后调用 PreparedStatement 的 executeBatch()方法执行批处理，代码如下：

```
int[] rowcount = pstmt.executeBatch();
```

最后调用 PreparedStatement 的 close()方法关闭预编译语句对象，代码如下：

```
pstmt.close();
```

3.3.4 JDBC 连接 openGauss 结果集处理

不同数据类型的结果集(ResultSet)有各自的应用场景，应用程序需要根据实际情况选择相应的结果集类型。在执行 SQL 语句过程中，需要先创建相应的语句对象，而部分创建语句对象的方法提供了设置结果集类型的功能。涉及的 Connection 的方法如下：

```
//创建一个 Statement 对象，该对象将生成具有给定类型和并发性的 ResultSet 对象
createStatement(int resultSetType, int resultSetConcurrency);

//创建一个 PreparedStatement 对象，该对象将生成具有给定类型和并发性的 ResultSet 对象
prepareStatement(String sql, int resultSetType, int resultSetConcurrency);

//创建一个 CallableStatement 对象，该对象将生成具有给定类型和并发性的 ResultSet 对象
prepareCall(String sql, int resultSetType, int resultSetConcurrency);
```

结果集类型见表 3-1。

表 3-1 结果集类型

参　数	描　述
resultSetType	表示结果集的类型有三种： • ResultSet.TYPE_FORWARD_ONLY：ResultSet 只能向前移动，此变量是默认值； • ResultSet.TYPE_SCROLL_SENSITIVE：修改后重新滚动到修改所在行，可以看到修改后的结果； • ResultSet.TYPE_SCROLL_INSENSITIVE：对可修改例程所做的编辑不进行显示。 说明： 结果集从数据库中读取数据之后，即使类型是 ResultSet.TYPE_SCROLL_SENSITIVE，也不会看到由其他事务在这之后引起的改变。调用 ResultSet 的 refreshRow()方法，可进入数据库并从其中取得当前光标所指记录的最新数据

续表

参　数	描　述
resultSetConcurrency	表示结果集的并发有两种类型： • ResultSet.CONCUR_READ_ONLY：如果不从结果集中的数据建立一个新的更新语句，就不能对结果集中的数据进行更新； • ResultSet.CONCUR_UPDATEABLE：可改变的结果集。对于可滚动的结果集，可对结果集进行适当的改变

ResultSet 对象具有指向其当前数据行的光标。最初，光标被置于第一行之前。next()方法将光标移动到下一行；因为该方法在 ResultSet 对象没有下一行时返回 false，所以可以在 while 循环中使用它迭代结果集。但对于可滚动的 ResultSet，JDBC 驱动程序提供了更多的定位方法，使 ResultSet 指向特定的行。结果定位方法见表 3-2。

表 3-2　结果定位方法

方　法	描　述	方　法	描　述
next()	把 ResultSet 向下移动一行	afterLast()	把 ResultSet 定位到最后一行之后
previous()	把 ResultSet 向上移动一行	first()	把 ResultSet 定位到第一行
beforeFirst()	把 ResultSet 定位到第一行之前	last()	把 ResultSet 定位到最后一行

对于可滚动的结果集，会调用定位方法来改变光标的位置。JDBC 驱动程序提供了获取结果集中光标所处位置的方法。获取光标位置的方法见表 3-3。

表 3-3　获取光标位置的方法

方　法	描　述	方　法	描　述
isFirst()	光标是否在第一行	isAfterLast()	光标是否在最后一行之后
isLast()	光标是否在最后一行	getRow()	获取当前光标在第几行
isBeforeFirst()	光标是否在第一行之前		

ResultSet 对象提供了丰富的方法，以获取结果集中的数据。获取数据常用的方法见表 3-4，其他方法请参考 JDK 官方文档。

表 3-4　获取数据常用的方法

方　法	描　述
int getInt(int columnIndex)	按列标获取 Int 型数据
int getInt(String columnLabel)	按列名获取 Int 型数据
String getString(int columnIndex)	按列标获取 String 型数据
String getString(String columnLabel)	按列名获取 String 型数据
Date getDate(int columnIndex)	按列标获取 Date 型数据
Date getDate(String columnLabel)	按列名获取 Date 型数据

下面的示例将演示如何基于 openGauss 提供的 JDBC 接口开发应用程序。其中包含的 CREATE TABLE 语句表示表的创建，INSERT 语句表示向表中插入数据，UPDATE 语句表示更新数据。更多关于 SQL 的介绍请参阅后续章节。

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.CallableStatement;

public class DBTest {

  //创建数据库连接
  public static Connection GetConnection(String username, String passwd) {
    String driver = "org.testdbql.Driver";
    String sourceURL = "jdbc:testdbql://localhost:8000/testdb";
    Connection conn = null;
    try {
      //加载数据库驱动
      Class.forName(driver).newInstance();
    } catch (Exception e) {
      e.printStackTrace();
      return null;
    }

    try {
      //创建数据库连接
      conn = DriverManager.getConnection(sourceURL, username, passwd);
      System.out.println("Connection succeed!");
    } catch (Exception e) {
      e.printStackTrace();
      return null;
    }
    return conn;
  };
  //执行普通的 SQL 语句，创建 customer_t1 表
  public static void CreateTable(Connection conn) {
    Statement stmt = null;
    try {
      stmt = conn.createStatement();

      //执行普通 SQL 语句
      int rc = stmt
          .executeUpdate("CREATE TABLE customer_t1(c_customer_sk INTEGER, c_customer_name
VARCHAR(32));");
```

```
            stmt.close();
        } catch (SQLException e) {
            if (stmt != null) {
                try {
                    stmt.close();
                } catch (SQLException e1) {
                    e1.printStackTrace();
                }
            }
            e.printStackTrace();
        }
    }

    //执行预处理语句,批量插入数据
    public static void BatchInsertData(Connection conn) {
        PreparedStatement pst = null;

        try {
            //生成预处理语句
            pst = conn.prepareStatement("INSERT INTO customer_t1 VALUES (?,?)");
            for (int i = 0; i < 3; i++) {
                //添加参数
                pst.setInt(1, i);
                pst.setString(2, "data " + i);
                pst.addBatch();
            }
            //执行批处理
            pst.executeBatch();
            pst.close();
        } catch (SQLException e) {
            if (pst != null) {
                try {
                    pst.close();
                } catch (SQLException e1) {
                    e1.printStackTrace();
                }
            }
            e.printStackTrace();
        }
    }

    //执行预编译语句,更新数据
    public static void ExecPreparedSQL(Connection conn) {
        PreparedStatement pstmt = null;
        try {
            pstmt = conn
```

```
            .prepareStatement("UPDATE customer_t1 SET c_customer_name = ? WHERE c_customer_sk = 1");
        pstmt.setString(1, "new Data");
        int rowcount = pstmt.executeUpdate();
        pstmt.close();
      } catch (SQLException e) {
        if (pstmt != null) {
          try {
            pstmt.close();
          } catch (SQLException e1) {
            e1.printStackTrace();
          }
        }
        e.printStackTrace();
      }
    }
  //执行存储过程
    public static void ExecCallableSQL(Connection conn) {
      CallableStatement cstmt = null;
      try {
        cstmt = conn.prepareCall("{? = CALL TESTPROC(?,?,?)}");
        cstmt.setInt(2, 50);
        cstmt.setInt(1, 20);
        cstmt.setInt(3, 90);
        cstmt.registerOutParameter(4, Types.INTEGER);  //注册 out 参数,类型为整型
        cstmt.execute();
        int out = cstmt.getInt(4);                     //获取 out 参数
        System.out.println("The CallableStatment TESTPROC returns:" + out);
        cstmt.close();
      } catch (SQLException e) {
        if (cstmt != null) {
          try {
            cstmt.close();
          } catch (SQLException e1) {
            e1.printStackTrace();
          }
        }
        e.printStackTrace();
      }
    }
    /**
     * 主程序,逐步调用各静态方法
     * @param args
     */
    public static void main(String[] args) {
      //创建数据库连接
      Connection conn = GetConnection("tester", "Password1234");
```

```
        //创建表
        CreateTable(conn);
        //批量插入数据
        BatchInsertData(conn);
        //执行预编译语句,更新数据
        ExecPreparedSQL(conn);
        //执行存储过程
        ExecCallableSQL(conn);
        //关闭数据库连接
        try {
          conn.close();
        } catch (SQLException e) {
          e.printStackTrace();
        }
      }
    }
```

3.4　openGauss 数据库 ODBC 连接

ODBC(open database connectivity,开放数据库互连)是由微软公司基于 X/OPEN CLI 提出的用于访问数据库的应用程序编程接口。应用程序通过 ODBC 提供的应用程序接口与数据库进行交互,增强了应用程序的可移植性、扩展性和可维护性。通过获取 ODBC 进行 C++调用 openGauss 编程,步骤如下:

(1) 获取 unixODBC 驱动,获取 unixODBC 源码包后进行编译安装(http://sourceforge.net/projects/unixodbc/files/unixODBC/2.3.0/unixODBC-2.3.0.tar.gz/download),命令如下:

```
[omm@db1 ~]# cd /soft
[omm@db1 soft]# tar -zxvf unixODBC-2.3.0.tar.gz
[omm@db1 soft]# vi unixODBC-2.3.0/configure
------------------------------------------------------------ #
[root@db1 ~]# cd /soft/unixODBC-2.3.0
[root@db1 unixODBC-2.3.0]#  ./configure --enable-gui=no
#鲲鹏服务器上编译需追加参数: --build=aarch64-unknown-linux-gnu
[root@db1 unixODBC-2.3.0]# make[root@db1 unixODBC-2.3.0]# make install
```

(2) 下载 openGauss 的 ODBC 驱动,解压并替换部分客户端原有的 lib 库(https://opengauss.org/zh/download.html),替换客户端 openGauss 驱动程序,命令如下:

```
[root@db1 ~]# vi /usr/local/etc/odbcinst.ini
-----------------------------------------------------------
```

```
[openGauss]
Driver64 = /usr/local/lib/psqlodbcw.so setup = /usr/local/lib/psqlodbcw.so
-----------------------------------------------------------
[root@db1 ~]# vi /usr/local/etc/odbc.ini
------------------------------------------------------------ [openGaussODBC]
Driver = openGauss
Servername = 192.168.0.225
Database = testdb
Username = jack
Password = gauss@123
Port = 26000
Sslmode = allow
-----------------------------------------------------------
```

（3）数据库服务器授予客户端权限，命令如下：

```
$ gs_guc reload -N all -I all -h "host all jack 127.0.0.1/32 SHA-256"
```

（4）配置环境变量，命令如下：

```
# vi ~/.bashrc
-------------------------------------------
export LD_LIBRARY_PATH = /usr/local/lib/:$LD_LIBRARY_PATH
export ODBCSYSINI = /usr/local/etc
export ODBCINI = /usr/local/etc/odbc.ini
-------------------------------------------
# source ~/.bashrc
```

（5）使用 isql 进行登录测试，命令如下：

```
[root@db1 soft]# isql -v openGaussODBC
```

具体开发流程与 JDBC 相同，如图 3-7 所示。ODBC 常用功能示例代码如下：

```
// 此示例演示如何通过 ODBC 方式获取 openGauss 中的数据
// DBtest.c (compile with: libodbc.so)
#include <stdlib.h>
#include <stdio.h>
#include <sqlext.h>
#ifdef WIN32
#endif
SQLHENV       V_OD_Env;         // Handle ODBC environment
SQLHSTMT      V_OD_hstmt;       // Handle statement
SQLHDBC       V_OD_hdbc;        // Handle connection
char          typename[100];
SQLINTEGER    value = 100;
SQLINTEGER    V_OD_erg,V_OD_buffer,V_OD_err,V_OD_id;
```

```
int main(int argc,char *argv[])
{
      // 1. 申请环境句柄
      V_OD_erg = SQLAllocHandle(SQL_HANDLE_ENV,SQL_NULL_HANDLE,&V_OD_Env);
      if ((V_OD_erg != SQL_SUCCESS) && (V_OD_erg != SQL_SUCCESS_WITH_INFO))
      {
           printf("Error AllocHandle\n");
           exit(0);
      }
      // 2. 设置环境属性(版本信息)
      SQLSetEnvAttr(V_OD_Env, SQL_ATTR_ODBC_VERSION, (void*)SQL_OV_ODBC3, 0);
      // 3. 申请连接句柄
      V_OD_erg = SQLAllocHandle(SQL_HANDLE_DBC, V_OD_Env, &V_OD_hdbc);
      if ((V_OD_erg != SQL_SUCCESS) && (V_OD_erg != SQL_SUCCESS_WITH_INFO))
      {
           SQLFreeHandle(SQL_HANDLE_ENV, V_OD_Env);
           exit(0);
      }
      // 4. 设置连接属性
      SQLSetConnectAttr(V_OD_hdbc, SQL_ATTR_AUTOCOMMIT, SQL_AUTOCOMMIT_ON, 0);
      // 5. 连接数据源,这里的 userName 与 password 分别表示连接数据库的用户名和用户密码,
      // 请根据实际情况修改
      // 如果 odbc.ini 文件中已经配置了用户名和用户密码,那么这里可以留空(""); 但是不建议
      // 这么做,因为一旦 odbc.ini 权限管理不善,将导致数据库用户密码泄露
      V_OD_erg = SQLConnect(V_OD_hdbc, (SQLCHAR*) "gaussdb", SQL_NTS,
                        (SQLCHAR*) "userName", SQL_NTS,  (SQLCHAR*) "password", SQL_NTS);
      if ((V_OD_erg != SQL_SUCCESS) && (V_OD_erg != SQL_SUCCESS_WITH_INFO))
      {
          printf("Error SQLConnect %d\n",V_OD_erg);
          SQLFreeHandle(SQL_HANDLE_ENV, V_OD_Env);
          exit(0);
      }
      printf("Connected !\n");
      // 6. 设置语句属性
      SQLSetStmtAttr(V_OD_hstmt,SQL_ATTR_QUERY_TIMEOUT,(SQLPOINTER *)3,0);
      // 7. 申请语句句柄
      SQLAllocHandle(SQL_HANDLE_STMT, V_OD_hdbc, &V_OD_hstmt);
      // 8. 直接执行 SQL 语句
      SQLExecDirect(V_OD_hstmt,"drop table IF EXISTS customer_t1",SQL_NTS);
      SQLExecDirect(V_OD_hstmt,"CREATE TABLE customer_t1(c_customer_sk INTEGER, c_customer_
name VARCHAR(32));",SQL_NTS);
      SQLExecDirect(V_OD_hstmt,"insert into customer_t1 values(25,li)",SQL_NTS);
      // 9. 准备执行
      SQLPrepare(V_OD_hstmt,"insert into customer_t1 values(?)",SQL_NTS);
```

```
        // 10. 绑定参数
        SQLBindParameter(V_OD_hstmt,1,SQL_PARAM_INPUT,SQL_C_SLONG,SQL_INTEGER,0,0,
                         &value,0,NULL);
        // 11. 执行准备好的语句
        SQLExecute(V_OD_hstmt);
        SQLExecDirect(V_OD_hstmt,"select id from testtable",SQL_NTS);
        // 12. 获取结果集某一列的属性
        SQLColAttribute(V_OD_hstmt,1,SQL_DESC_TYPE,typename,100,NULL,NULL);
        printf("SQLColAtrribute %s\n",typename);
        // 13. 绑定结果集
        SQLBindCol(V_OD_hstmt,1,SQL_C_SLONG, (SQLPOINTER)&V_OD_buffer,150,
                   (SQLLEN *)&V_OD_err);
        // 14. 通过 SQLFetch 获取结果集中的数据
        V_OD_erg = SQLFetch(V_OD_hstmt);
        // 15. 通过 SQLGetData()获取并返回数据
        while(V_OD_erg != SQL_NO_DATA)
        {
            SQLGetData(V_OD_hstmt,1,SQL_C_SLONG,(SQLPOINTER)&V_OD_id,0,NULL);
            printf("SQLGetData ---- ID = %d\n",V_OD_id);
            V_OD_erg = SQLFetch(V_OD_hstmt);
        };
        printf("Done !\n");
        // 16. 断开数据源连接并释放句柄资源
        SQLFreeHandle(SQL_HANDLE_STMT,V_OD_hstmt);
        SQLDisconnect(V_OD_hdbc);
        SQLFreeHandle(SQL_HANDLE_DBC,V_OD_hdbc);
        SQLFreeHandle(SQL_HANDLE_ENV, V_OD_Env);
        return(0);
    }
```

3.5 小结

本章首先介绍了 gsql 本地连接和远程连接，然后介绍了通过 gsql 和 DBeaver 客户端连接数据库，最后介绍了如何在 Java 源码和 C++源码中通过编程语言调用 openGauss 数据库接口对数据进行管理。

3.6 习题

1. openGauss 数据库连接方式分为哪几类？
2. 尝试使用 DBeaver 连接 MySQL 和 postgreSQL。

3. TPC-H 是数据库领域的知名基准测试数据集(http://www.tpc.org/tpch/)。下载 TPC-H 生成器并结合 C++和 ODBC 驱动完成如下操作：

(1) 新建数据库,将其命名为 TPC-H；

(2) 新建 TPC-H 下的 8 个表；

(3) 将生成的 1GB 的 TPC-H 数据批量导入数据库；

(4) 计算 TPC-H 数据库的 order 表中 totalprice 列的平均值；

(5) 删除 8 个表；

(6) 删除 TPC-H 数据库。