《数据库》

openGauss

的数据类型

**目录**

[前 言 3](#_Toc68692898)

[简介 3](#_Toc68692899)

[内容描述 3](#_Toc68692900)

[前置条件 3](#_Toc68692901)

[测试 openGauss DBMS 的数据类型 4](#_Toc68692902)

[连接并登录数据库 4](#_Toc68692903)

[连接数据库 4](#_Toc68692904)

[使用图形化界面 4](#_Toc68692905)

[数值数据类型 4](#_Toc68692906)

[整数类型 4](#_Toc68692907)

[任意精度数字类型 6](#_Toc68692908)

[浮点类型 7](#_Toc68692909)

[序列类型 10](#_Toc68692910)

[货币类型 13](#_Toc68692911)

[字符串类型 13](#_Toc68692912)

[布尔类型 15](#_Toc68692913)

[二进制数据类型 16](#_Toc68692914)

[测试日期类型时间类型 16](#_Toc68692915)

[测试枚举类型 20](#_Toc68692916)

[附录一：openGauss数据库基本操作 21](#_Toc68692917)

[查看数据库对象 21](#_Toc68692918)

[其他操作 22](#_Toc68692919)

**前 言**

简介

本实验指导书主要目的是为了让读者熟悉并掌握openGauss数据库的常用数据类型集合，并通过案例强化学习。

内容描述

本实验指导书从数值数据类型、字符串类型、二进制类型、布尔类型、日期类型和时间类型以及枚举类型入手，结合具体的实践案例入手，进行模拟演练。

前置条件

openGauss数据库开发需要掌握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**附录一**。

**测试 openGauss DBMS 的数据类型**

连接并登录数据库

连接数据库

以操作系统用户omm登录数据库主节点。

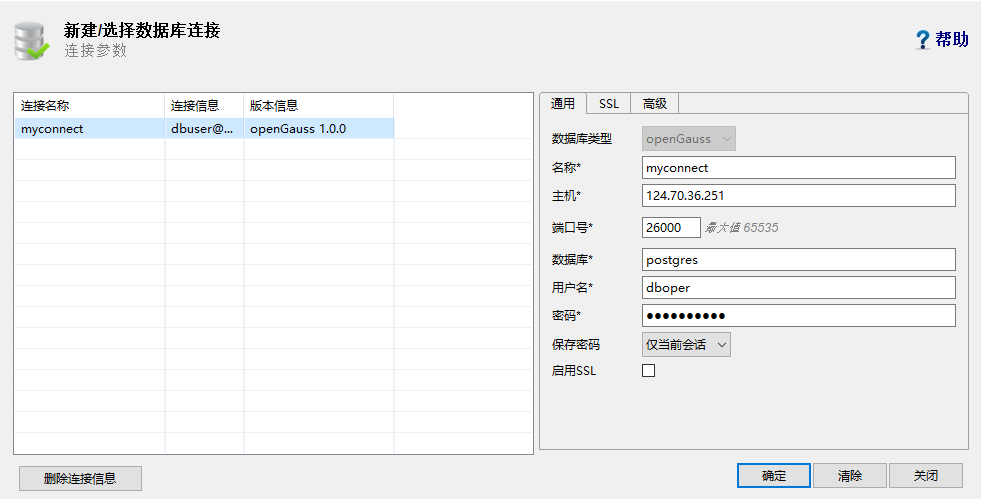
[root@db1 ~]# su - omm

启动服务。启动服务命令：

[omm@db1 ~]$ gs\_om -t start

使用图形化界面

通过Data Studio连接数据库。



数值数据类型

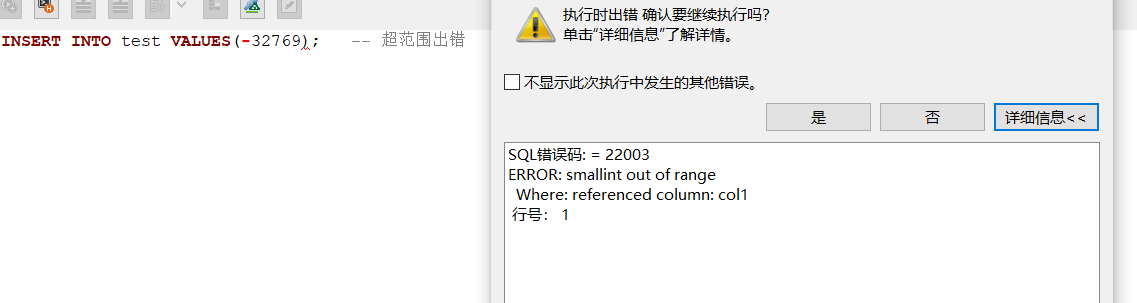
整数类型

smallint

smallint 用 2 个字节存储，最小值为-32768，最大值为 32767

DROP TABLE IF EXISTS test;

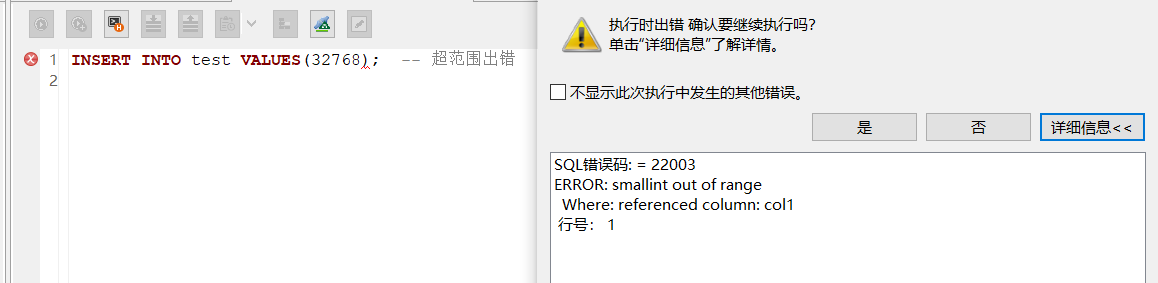
CREATE TABLE test(col1 smallint);

INSERT INTO test VALUES(-32769); -- 超范围出错

INSERT INTO test VALUES(-32768); -- 范围之内的最小值

INSERT INTO test VALUES(32767); -- 范围之内的最大值

INSERT INTO test VALUES(32768); -- 超范围出错



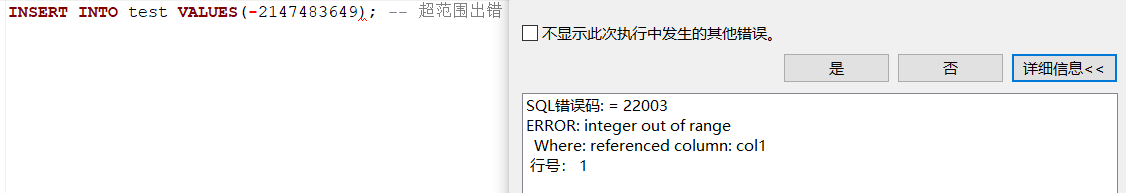
integer

integer 用 4 个字节存储，最小值为-2147483648，最大值为 2147483647

DROP TABLE IF EXISTS test;

CREATE TABLE test(col1 integer);

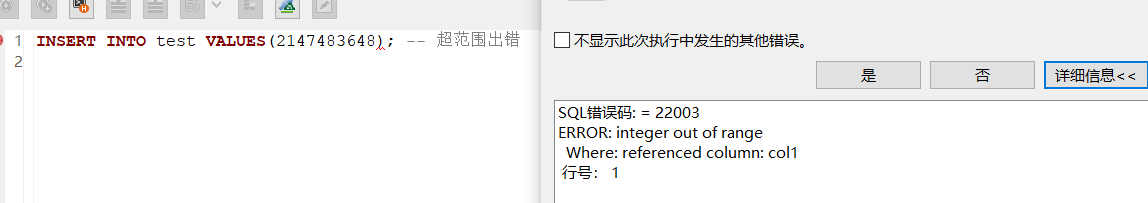
INSERT INTO test VALUES(-2147483649); -- 超范围出错



INSERT INTO test VALUES(-2147483648); -- 范围之内的最小值

INSERT INTO test VALUES(2147483647); -- 范围之内的最大值

INSERT INTO test VALUES(2147483648); -- 超范围出错



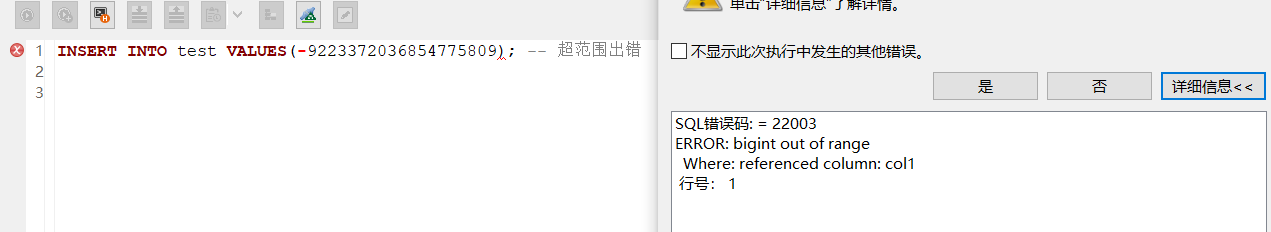
bigint

bigint 用 8 个字节存储，最小值为-9223372036854775808，最大值为 9223372036854775807

DROP TABLE IF EXISTS test;

CREATE TABLE test(col1 bigint);

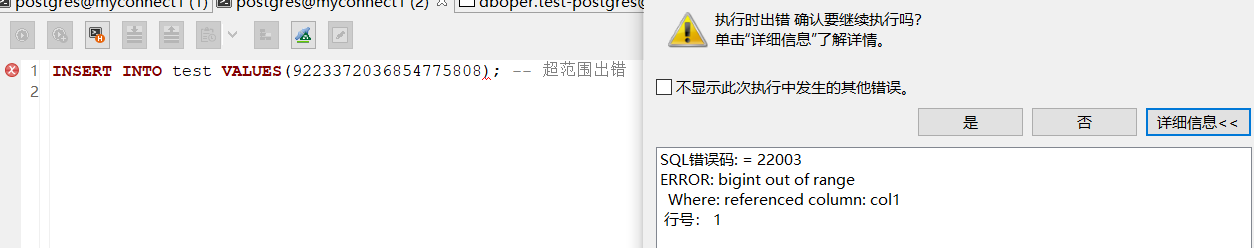
INSERT INTO test VALUES(-9223372036854775809); -- 超范围出错



INSERT INTO test VALUES(-9223372036854775808); -- 范围之内的最小值

INSERT INTO test VALUES(9223372036854775807); -- 范围之内的最大值

INSERT INTO test VALUES(9223372036854775808); -- 超范围出错



任意精度数字类型

numberic(precision,scale)

decimal(precision,scale)

这两种等价！

DROP TABLE IF EXISTS test;

CREATE TABLE test(col1 number(4,2));

INSERT INTO test VALUES (-100); -- 超范围出错

INSERT INTO test VALUES (-99.995); -- 四舍五入后，超范围出错

INSERT INTO test VALUES (-99.994); -- 四舍五入后，在范围之内

INSERT INTO test VALUES (-99.99); -- 在范围之内

INSERT INTO test VALUES (99.99); -- 在范围之内

INSERT INTO test VALUES (99.994); -- 四舍五入后，在范围之内

INSERT INTO test VALUES (99.9949); -- 四舍五入后，在范围之内

INSERT INTO test VALUES (99.995); -- 四舍五入后，超范围出错

INSERT INTO test VALUES (100); -- 超范围出错

SELECT \* FROM test;

浮点类型

real

real 用 4 个字节存储

DROP TABLE IF EXISTS test; CREATE TABLE test(col1 real);

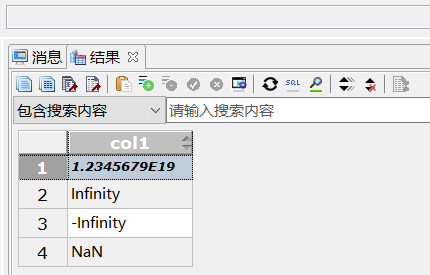
INSERT INTO test VALUES (12345678901234567890.12345678901234567890);

INSERT INTO test VALUES ('Infinity');

INSERT INTO test VALUES ('-Infinity');

INSERT INTO test VALUES ('NaN');

SELECT \* FROM test;



float

DROP TABLE IF EXISTS test; CREATE TABLE test(col1 float); INSERT INTO test

VALUES (12345678901234567890.12345678901234567890);

INSERT INTO test VALUES (92.3);

INSERT INTO test VALUES (92.345678);

INSERT INTO test VALUES (92345678.9);

INSERT INTO test VALUES ('Infinity');

INSERT INTO test VALUES ('-Infinity');

INSERT INTO test VALUES ('NaN');

SELECT \* FROM test;



FLOAT(n)

DROP TABLE IF EXISTS test;

CREATE TABLE test (col1 float(2));

INSERT INTO test VALUES (12345678901234567890.12345678901234567890);

INSERT INTO test VALUES (92.3);

INSERT INTO test VALUES (92.34);

INSERT INTO test VALUES (92.345);

INSERT INTO test VALUES (92.3456);

INSERT INTO test VALUES (92.34567);

INSERT INTO test VALUES (92.345678);

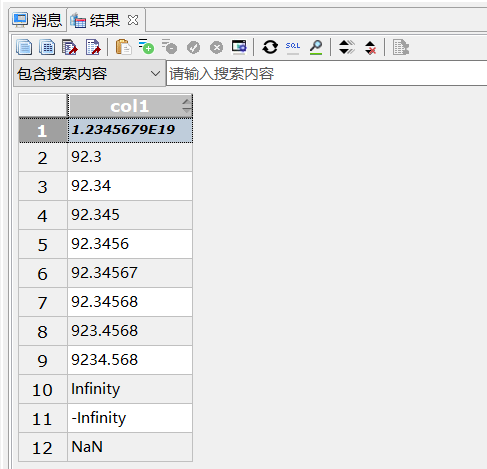
INSERT INTO test VALUES (923.456789);

INSERT INTO test VALUES (9234.56789);

INSERT INTO test VALUES ('Infinity');

INSERT INTO test VALUES ('-Infinity');

INSERT INTO test VALUES ('NaN'); SELECT \* FROM test;



double precision

DROP TABLE IF EXISTS test;

CREATE TABLE test (col1 double precision);

INSERT INTO test VALUES (12345678901234567890.12345678901234567890);

INSERT INTO test VALUES (92.3);

INSERT INTO test VALUES (92.34);

INSERT INTO test VALUES (92.345);

INSERT INTO test VALUES (92.3456);

INSERT INTO test VALUES (92.34567);

INSERT INTO test VALUES (92.345678);

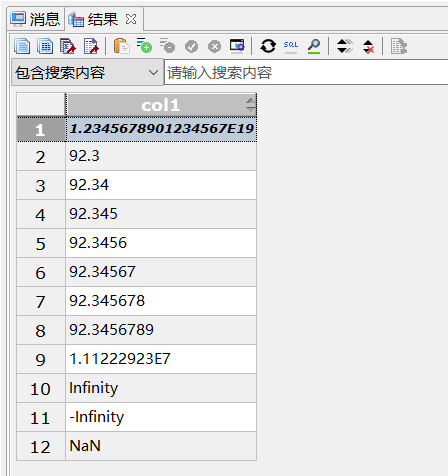
INSERT INTO test VALUES (92.3456789);

INSERT INTO test VALUES (11122292.3);

INSERT INTO test VALUES ('Infinity');

INSERT INTO test VALUES ('-Infinity');

INSERT INTO test VALUES ('NaN'); SELECT \* FROM test;



序列类型

smallserial

用 2 个字节存储，范围是 1-32767

DROP TABLE IF EXISTS test;

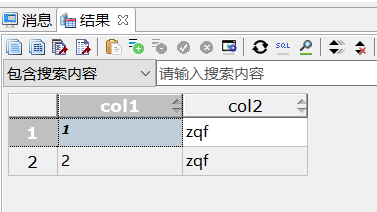
CREATE TABLE test (col1 smallserial,col2 char(10));

-- 使用序列插入 2 条记录

INSERT INTO test(col2) VALUES('zqf');

INSERT INTO test(col2) VALUES('zqf');

SELECT \* FROM test;



-- 将序列的当前值设置为 32766

SELECT setval('test\_col1\_seq', 32766, false);

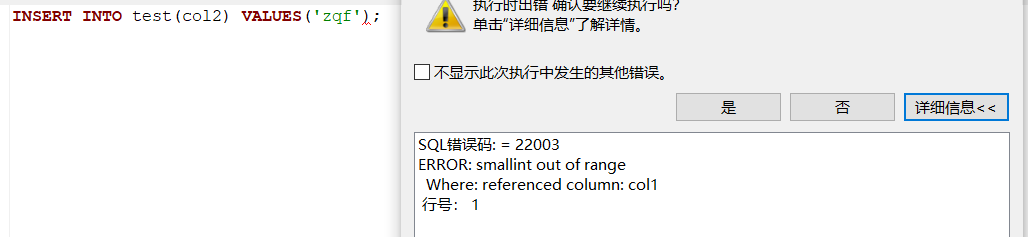


-- 使用序列插入 3 条记录（插入第 3 条的时候会出错！）

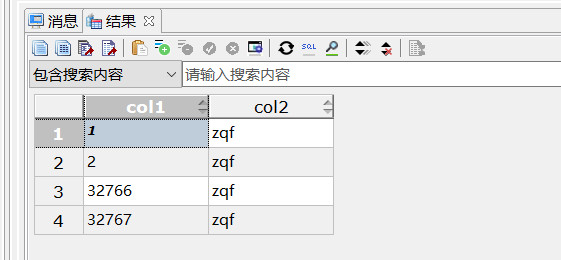
INSERT INTO test(col2) VALUES('zqf'); --正常

INSERT INTO test(col2) VALUES('zqf'); --正常

INSERT INTO test(col2) VALUES('zqf'); --出错，系列超过了最大值



SELECT \* FROM test;



serial

用 4 个字节存储，范围是 1-2147483647

DROP TABLE IF EXISTS test;

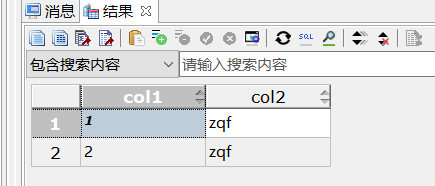
CREATE TABLE test (col1 serial,col2 char(10));

-- 使用序列插入 2 条记录

INSERT INTO test(col2) VALUES('zqf');

INSERT INTO test(col2) VALUES('zqf');

SELECT \* FROM test;



-- 将序列的当前值设置为 2147483646

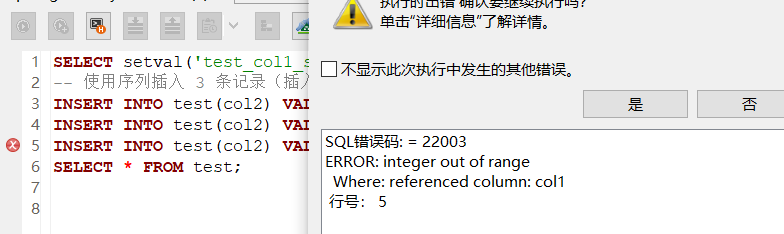
SELECT setval('test\_col1\_seq',2147483646, false);

-- 使用序列插入 3 条记录（插入第 3 条的时候会出错！）

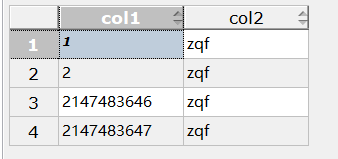
INSERT INTO test(col2) VALUES('zqf'); --正常

INSERT INTO test(col2) VALUES('zqf'); --正常

INSERT INTO test(col2) VALUES('zqf'); --出错，系列超过了最大值



SELECT \* FROM test;



bigserial

用 8 个字节存储，范围是 1-9223372036854775807

DROP TABLE IF EXISTS test;

CREATE TABLE test (col1 bigserial,col2 char(10));

-- 使用序列插入 2 条记录

INSERT INTO test(col2) VALUES('zqf'); INSERT INTO test(col2) VALUES('zqf');

SELECT \* FROM test;



-- 将序列的当前值设置为 9223372036854775806

SELECT setval('test\_col1\_seq', 9223372036854775806, false);

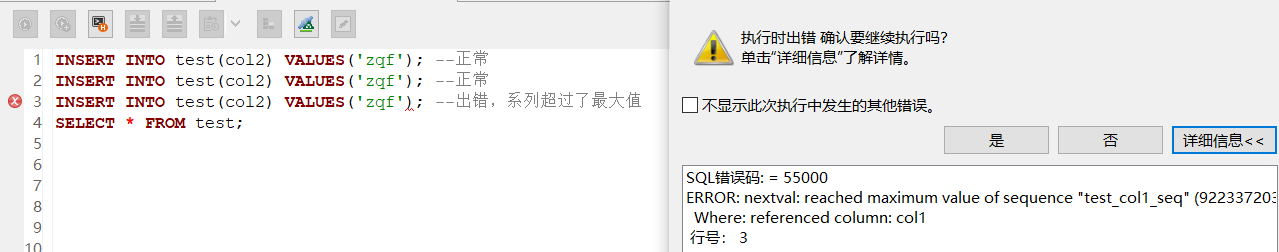


-- 使用序列插入 3 条记录（插入第 3 条的时候会出错！）

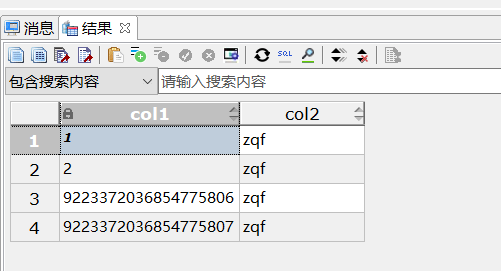
INSERT INTO test(col2) VALUES('zqf'); --正常

INSERT INTO test(col2) VALUES('zqf'); --正常

INSERT INTO test(col2) VALUES('zqf'); --出错，系列超过了最大值



SELECT \* FROM test;



货币类型

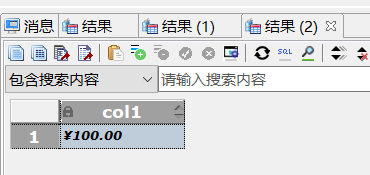
money

DROP TABLE IF EXISTS test; CREATE TABLE test(col1 money); SHOW lc\_monetary;

INSERT INTO test VALUES (100); SELECT \* FROM test;

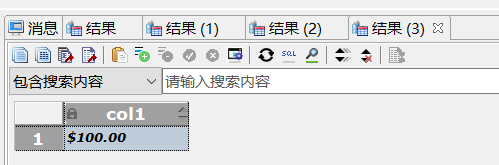
SET lc\_monetary="zh\_CN.utf8";

SELECT \* FROM test;



SET lc\_monetary="en\_US.utf8";

SELECT \* FROM test;



字符串类型

character 或者 char

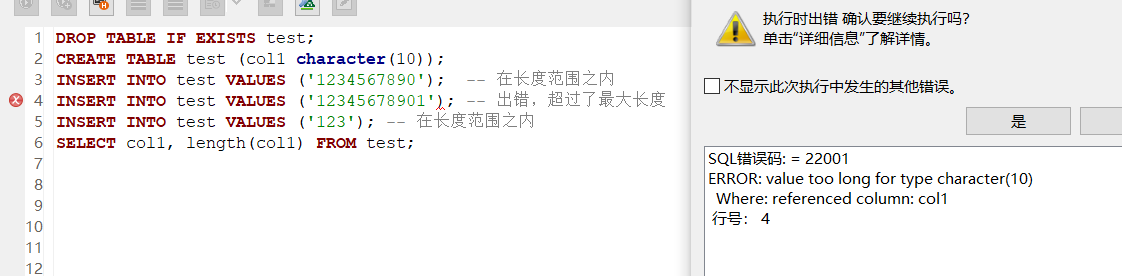
定长字符串，长度不足会使用空格填充

DROP TABLE IF EXISTS test;

CREATE TABLE test (col1 character(10));

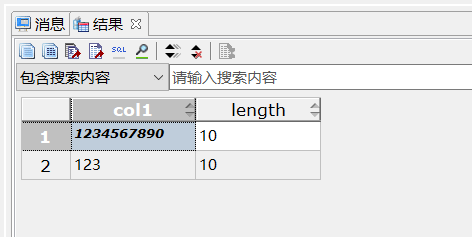
INSERT INTO test VALUES ('1234567890'); -- 在长度范围之内

INSERT INTO test VALUES ('12345678901'); -- 出错，超过了最大长度



INSERT INTO test VALUES ('123'); -- 在长度范围之内

SELECT col1, length(col1) FROM test;



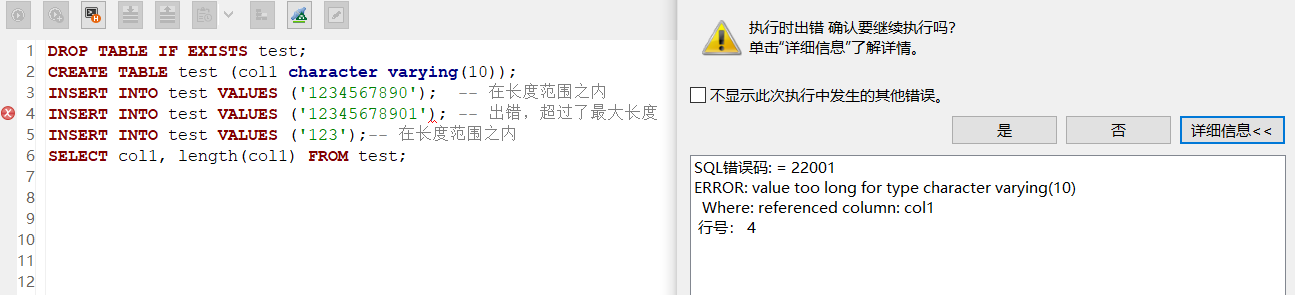
character varying 或者 varchar

变长字符串，按实际的字符进行存储，不能超过最大的存储

DROP TABLE IF EXISTS test;

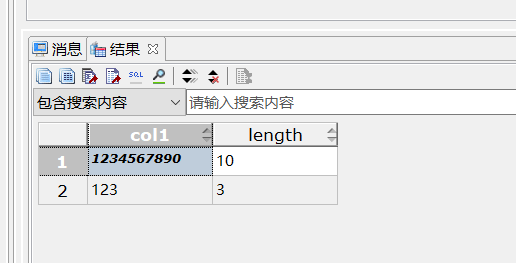
CREATE TABLE test (col1 character varying(10));

INSERT INTO test VALUES ('1234567890'); -- 在长度范围之内

INSERT INTO test VALUES ('12345678901'); -- 出错，超过了最大长度

INSERT INTO test VALUES ('123');-- 在长度范围之内

SELECT col1, length(col1) FROM test;

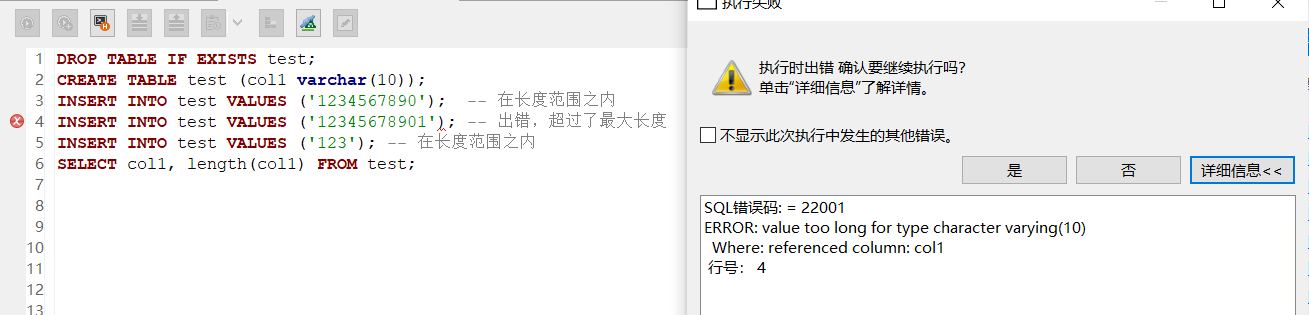


DROP TABLE IF EXISTS test;

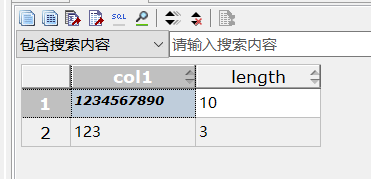
CREATE TABLE test (col1 varchar(10));

INSERT INTO test VALUES ('1234567890'); -- 在长度范围之内

INSERT INTO test VALUES ('12345678901'); -- 出错，超过了最大长度

INSERT INTO test VALUES ('123'); -- 在长度范围之内

SELECT col1, length(col1) FROM test;



text

变长字符串，最大存储 1GB

text 等价于 varchar（这里 varchar 没有限制最大的值 n）

DROP TABLE IF EXISTS test;

CREATE TABLE test (col1 text);

INSERT INTO test VALUES ('1234567890');

SELECT col1, length(col1) FROM test;



布尔类型

boolean

DROP TABLE IF EXISTS test;

CREATE TABLE test (col1 boolean);

INSERT INTO test VALUES ('t');

INSERT INTO test VALUES ('f');

INSERT INTO test VALUES ('y');

INSERT INTO test VALUES ('n');

INSERT INTO test VALUES ('1');

INSERT INTO test VALUES ('0');

INSERT INTO test VALUES ('on');

INSERT INTO test VALUES ('off');

INSERT INTO test VALUES ('yes');

INSERT INTO test VALUES ('no');

INSERT INTO test VALUES ('true');

INSERT INTO test VALUES ('false');

SELECT col1 FROM test;



二进制数据类型

BLOB

适用于存储图像、视频、音频等。大对象类型，最大存储 4GB

RAW

存储的为可变长二进制数据，是一种较老的数据类型，用这种格式来保存的图形文件或带格式的文本文件，如Miceosoft Word文档。

测试日期类型时间类型

DATE

4 个字节存储，日期，不包括一天中的时间。支持日期运算。

DROP TABLE IF EXISTS test;

CREATE TABLE test ( col1 DATE );

INSERT INTO test VALUES (date '08-09-2017');

set datestyle='YMD'; --设置日期的格式

INSERT INTO test VALUES (date '2018-09-10');

SELECT \* FROM test;



另外的一个方法：

DROP TABLE test;

CREATE TABLE test ( col1 DATE );

INSERT INTO test VALUES (to\_date('08-09-2017','MM-DD-YYYY'));

INSERT INTO test VALUES (to\_date('08-09-2017 01:02:03','MM-DD-YYYY HH:MI:SS'));

SELECT col1,length(col1) FROM test;

SELECT (to\_char(col1,'HH24:MI:SS')) col1\_TIME FROM test ;

SELECT (to\_char(col1,'MM-DD-YYYY')) col1\_Date FROM test;

SELECT (to\_char(col1,'MM-DD-YYYY HH:MI:SS')) col1\_DateAndTime FROM test;



TIMESTAMP(p) with time zone

TIMESTAMPTZ(p)

TIMESTAMP(p) with time zone

带有时区信息

DROP TABLE test;

CREATE TABLE test ( col1 TIMESTAMPTZ(3));

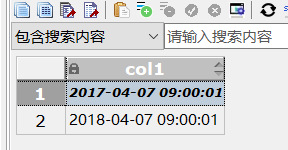
show datestyle;



INSERT INTO test VALUES (timestamp'2017-04-07 9:00:01.234');

INSERT INTO test VALUES (timestamp'2018-04-07 9:00:01.2345');

SELECT col1 FROM test;



select now(); -- now()函数返回一个带时区的时间戳

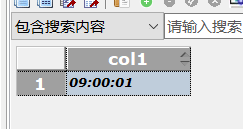
TIME(p) without time zone

DROP TABLE test;

CREATE TABLE test ( col1 TIME(3) without time zone);

INSERT INTO test VALUES ('9:00:01');

SELECT col1 FROM test;



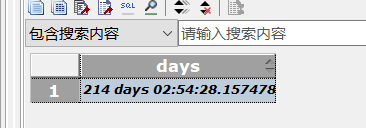
interval 时间间隔

两个时间戳计算时间间隔

DROP TABLE test;

CREATE TABLE test ( col1 TIMESTAMPTZ,col2 TIMESTAMPTZ);

INSERT INTO test VALUES (to\_timestamp('2020-08-09 9:00:01','MM-DD-YYYY hh24:mi:ss'),now()); SELECT col2-col1 days FROM test;



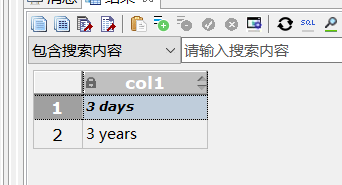
DROP TABLE test;

CREATE TABLE test ( col1 interval day(3) to second(4));

INSERT INTO test VALUES (interval '3' DAY);

INSERT INTO test VALUES (interval '3' YEAR);

SELECT \* FROM test;



**时间值函数**

ADD\_MONTHS

两个参数：DATE 和一个数（正负表示时间的前后）

DROP TABLE test;

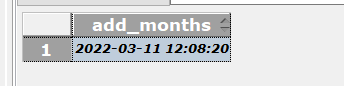
CREATE TABLE test (col1 DATE);

insert into test values(now());

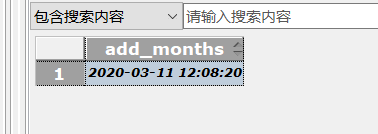
select \* from test;



select ADD\_MONTHS(col1,12) from test;



select ADD\_MONTHS(col1,-12) from test;

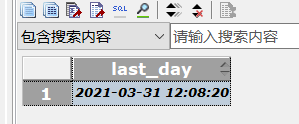


LAST\_DAY

返回本月的最后一天的时间

select \* from test;

select LAST\_DAY(col1) from test;



NEXT\_DAY

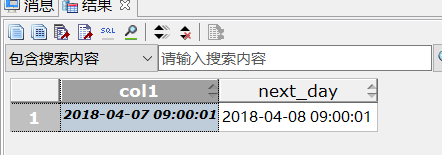
1 表示星期天，7 表示周六

DROP TABLE test;

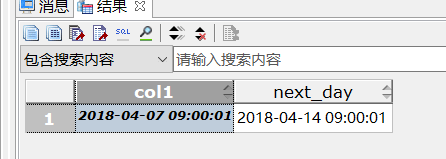
CREATE TABLE test (col1 DATE );

insert into test values('2018-04-07 9:00:01');

select col1,NEXT\_DAY(col1,1) from test;



select col1,NEXT\_DAY(col1,7) from test;



测试枚举类型

ENUM

对应于一些编程语言中的 enum 类型

CREATE TYPE weekday as ENUM('Sun','Mon','Tues','Wed','Thur','Fri','Sat');

DROP TABLE test;

CREATE TABLE test(col1 weekday);

INSERT INTO test VALUES ('Sun');

INSERT INTO test VALUES ('Mon');

INSERT INTO test VALUES ('Tues');

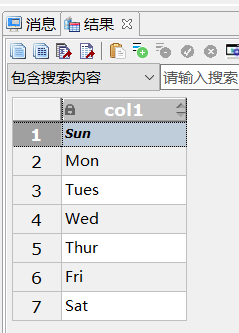
INSERT INTO test VALUES ('Wed');

INSERT INTO test VALUES ('Thur');

INSERT INTO test VALUES ('Fri');

INSERT INTO test VALUES ('Sat');

SELECT \* FROM test;



# 附录一：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;

* 要查看用户属性：

*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**

* 切换数据库：

*postgres*=# \c dbname

* 切换用户：

*postgres*=# \c – username

* 退出数据库：

postgres=# \q