 



北京海量数据技术股份有限公司

**Vastbase G100 V2.2**

**(Build 10)**

**PostgreSQL兼容性手册**

【版权声明】

©2007-2022 北京海量数据技术股份有限公司 版权所有

本文档著作权归 **北京海量数据技术股份有限公司**（简称“海量数据”）所有，未经海量数据事先书面许可，任何主体不得以任何形式复制、 修改、抄袭、传播全部或部分本文档内容。

北京海量数据技术股份有限公司保留所有的权利。

【服务声明】

本文档意在向客户介绍海量数据全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的产品、服务的种类、服务标准等应由您与海量数据之间的商业合同约定，除非双方另有约定，否则，海量数据对本文档内容不做任何明示或模式的承诺或保证。

**目录**

[1. 函数 3](#_Toc19428)

[1.1. TO\_REGCLASS 3](#_Toc5253)

[2. SQL语法 5](#_Toc1781)

[2.1. 游标（CURSOR）语法 5](#_Toc15513)

[2.2. CREATE SEQUENCE...as data\_type 6](#_Toc24319)

[2.3. 支持update使用where current of 9](#_Toc12106)

[2.4. 字符串连接符+null 10](#_Toc21416)

# 函数

## TO\_REGCLASS

**功能描述**

to\_regclass函数，用以判断表对象是否存在。

**语法格式**

to\_regclass ( text )

**参数说明**

text：支持所有的文本字符、字符数组（字符串）类型，函数入参必须保障需要支持的数据类型有：

| **名称** | **描述** |
| --- | --- |
| CHAR(n)、CHARACTER(n)、 NCHAR(n) | 定长字符串，不足补空格。n是指字节长度，如不带精度n，默认精度为1。 |
| VARCHAR(n) 、CHARACTER VARYING(n) | 变长字符串。n是指字节长度。 |
| VARCHAR2(n) | 变长字符串。是VARCHAR(n)类型的别名。n是指字节长度。 |
| NVARCHAR2(n) | 变长字符串。n是指字节长度。 |
| CLOB | 文本大对象。是TEXT类型的别名。 |
| TEXT | 变长字符串。 |
| VARCHAR | 变长字符串。 |
| CHAR | 单字节内部类型 |
| NAME | 用于对象名的内部类型。 |

注：其他的数据类型，只要能完成向text类型的隐式转换，在Vastbase中就是合法的。

**注意事项**

* 函数输入：类型是text，格式是[[catalogname.]schemaname.]relname；根据search\_path顺序查找，并且查找的是relation。
* 函数返回：类型是regclass。
* 输入输出遵循现有的隐式类型转化规则，隐式类型转化可以在pg\_cast系统表中查看。
* 具备该函数的访问权限，例如to\_regclass函数如果位于pg\_catalog模式下，用户需要具备以下权限：
* pg\_catalog模式的usage权限。
* to\_regclass函数的excute权限。

**示例**

1、创建测试表。

create table table\_exit(id int);

2、验证函数。

select to\_regclass('table\_exit');

结果返回如下：

to\_regclass

-------------

table\_exit

(1 row)

# SQL语法

## 游标（CURSOR）语法

**功能描述**

游标（cursor）是数据库系统在内存中开设的一个数据缓冲区，存放SQL语句的执行结果。

**语法格式**

* Oracle类型和PG类型语法：

name [ [ NO ] SCROLL ] CURSOR [ ( arguments ) ] FOR query;

* Vastbase类型语法：

CURSOR name [ [ NO ] SCROLL ] [ ( arguments ) ] FOR query;

**参数说明**

* name：游标变量的名称。
* [NO] SCROLL：如果指定了SCROLL特性，则表示游标的光标可以向后滚动（回滚）。如果使用 NO SCROLL则游标不能回滚。如果两个关键字都没有出现，则是否允许反向回迁取决于查询。
* CURSOR：关键字，指明变量是游标类型。
* [ ( arguments ) ]：如果指定了arguments，则是一个逗号分隔的对名称数据类型列表，起到类似绑定变量作用。
* FOR：关键字，表示后面的语句为游标定义的内容，可以被IS取代。
* query：查询语句，包含各种关联查询和子查询。

**注意事项**

* 该功能仅在数据库兼容模式为Oracle类型和PG类型时能够使用（即创建DB时DBCOMPATIBILITY='A'或者'PG'），其他类型数据库兼容模式保持原有语法不做改动。
* 在Vastbase使用游标（CURSOR）语法时，仅需要在Oracle类型和PG类型语法基础上调换变量名与类型关键字的顺序即可在Vastbase中实现该功能。

**示例**

* Oracle兼容模式的数据库中。

CREATE OR REPLACE procedure a() as

CURSOR c\_id FOR select \* from tt;

BEGIN

select 2\*3 from dual;

END;

/

* PG兼容模式的数据库中。

CREATE OR REPLACE procedure a() as

CURSOR c\_id FOR select \* from tt;

BEGIN

select 2\*3 from dual;

END;

/

## CREATE SEQUENCE...as data\_type

**功能描述**

CREATE SEQUENCE是向当前数据库里增加一个新的序列。在PG兼容模式下，可以使用as datatype语法可选项，指定序列的数据类型。

**语法格式**

CREATE [TEMPORARY|TEMP] SEQUENCE [IF NOT EXISTS] name [as data\_type]

[INCREMENT [BY] increment]

[MINVALUE minvalue|NO MINVALUE][MAXVALUE maxvalue|NO MAXVALUE]

[START[WITH]start][CACHE cache][[NO] CYCLE]

[OWNED BY{tablename.column\_name|NONE}]

**参数说明**

* TEMPOPARY/TEMP：如果声明了这个修饰词，那么该序列对象只为这个会话创建，并且在会话结束的时候自动删除。
* name：序列的名称。
* data\_type：序列的数据类型，Vastbase G100支持的数据类型有smallint、integer和bright，其中bright是默认值。
* increment：指定序列的步长。一个正数将生成一个递增的序列，一个负数将生成一个递减的序列。缺省值为1。
* MINVALUE minvalue | NO MINVALUE| NOMINVALUE：
* 执行序列的最小值。如果没有声明minvalue或者声明了NO MINVALUE，则递增序列的缺省值为1，递减序列的缺省值为-263-1。NOMINVALUE等价于NO MINVALUE
* MAXVALUE maxvalue | NO MAXVALUE| NOMAXVALUE：
* 执行序列的最大值。如果没有声明maxvalue或者声明了NO MAXVALUE，则递增序列的缺省值为263-1，递减序列的缺省值为-1。NOMAXVALUE等价于NO MAXVALUE
* start
* 指定序列的起始值。缺省值：对于递增序列为minvalue，递减序列为maxvalue。
* cache
* 为了快速访问，而在内存中预先存储序列号的个数。
* 缺省值为1，表示一次只能生成一个值，也就是没有缓存。
* 不建议同时定义cache和maxvalue或minvalue。因为定义cache后不能保证序列的连续性，可能会产生空洞，造成序列号段浪费。
* CYCLE
* 用于使序列达到maxvalue或者minvalue后可循环并继续下去。
* 如果声明了NO CYCLE，则在序列达到其最大值后任何对nextval的调用都会返回一个错误。
* NOCYCLE的作用等价于NO CYCLE。
* 缺省值为NO CYCLE。
* 若定义序列为CYCLE，则不能保证序列的唯一性。
* OWNED BY
* 将序列和一个表的指定字段进行关联。这样，在删除那个字段或其所在表的时候会自动删除已关联的序列。关联的表和序列的所有者必须是同一个用户，并且在同一个模式中。需要注意的是，通过指定OWNED BY，仅仅是建立了表的对应列和sequence之间关联关系，并不会在插入数据时在该列上产生自增序列。
* 缺省值为OWNED BY NONE，表示不存在这样的关联。
* 通过OWNED BY创建的Sequence不建议用于其他表，如果希望多个表共享Sequence，该Sequence不应该从属于特定表。

**注意事项**

* 使用此子句仅在PG兼容模式下有效。
* 用户需要具备创建序列的权限。

**示例**

1、创建数据库，检查兼容性。

CREATE DATABASE test\_10379 DBCOMPATIBILITY 'PG';

\c test\_10379

show sql\_compatibility;

2、创建一个名为serial1的递增序列，从101开始，并指定data\_type类型为smallint。

create sequence serial1 as smallint start 101;

返回结果为：

CREATE SEQUENCE

3、查看这个序列。

select \* from serial1;

返回结果为：

sequence\_name | last\_value | start\_value | increment\_by | max\_value | min\_value

| cache\_value | log\_cnt | is\_cycled | is\_called | uuid | type\_id

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

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

serial1 | 101 | 101 | 1 | 32767 | 1

| 1 | 0 | f | f | 0 | 21

(1 row)

## 支持update使用where current of

**功能描述**

在PG兼容模式下，Vastbase G100支持update命令使用where current of可选子句。使用此子句可以更新游标对应的字段值。

**语法格式**

UPDATE[ONLY]table\_name[\*][[AS]alias]

SET{column\_name={expression|DEFAULT}|

()

()

}[,...]

[FROM from\_item[,...]]

[WHERE cindition|WHERE CURRENT OF cursor\_name]

[RETURNING \*|output\_expression[[AS]output\_name][,...]]

**参数说明**

cursor\_name：要在WHERE CURRENT OF条件中使用的游标名。要被更新的是从这个游标中最近取出的行。该游标必须是一个在UPDATE目标表上的非分组查询。

**注意事项**

* 仅在PG兼容模式下有效。
* WHERE CURRENT OF不能和一个布尔条件一起指定。

**示例**

1、创建数据库，检查兼容性。

CREATE DATABASE my\_test DBCOMPATIBILITY 'PG';

\c my\_test

show sql\_compatibility;

2、创建测试表。

create table t1(col1 int,col2 text,col3 varchar,col4 timestamp,col5 boolean);

3、创建用户并授权。

create user user1 with password 'Bigdata@123';

grant all on t1 to user1;

4、插入数据。

insert into t1 values(1,'testtest1','test1','20220127',true);

insert into t1 values(2,'testtest2','test2','20220227',false);

insert into t1 values(3,'testtest3','test3','20220327',false);

5、切换到用户user1，开始事务。

\c - user1;

start transaction;

declare cursor1 cursor for select \* from t1 where col1=1;//定义游标

fetch forward 1 from cursor1;//抓取头一行到游标cursor1里

update t1 set col1=0 where current of cursor1;//更新游标中对应行的字段值

select \* from t1;//查看结果

close cursor1;//关闭游标并提交事务

end;//结束事务

返回结果为：

col1 | col2 | col3 | col4 | col5

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

2 | testtest2 | test2 | 2022-02-27 00:00:00 | f

3 | testtest3 | test3 | 2022-03-27 00:00:00 | f

0 | testtest1 | test1 | 2022-01-27 00:00:00 | t

(3 rows)

col1=1的数据改为col1=0，并插在表尾。

## 字符串连接符+null

**功能描述**

字符串连接符+null的执行结果对标PG执行结果。

**语法格式**

select type '123'||'null'

**参数说明**

* type ：数据类型，指定后接内容的数据类型，如int、number等。
* '123' ：由type指定类型的内容，由用户需求和数据类型决定内容。

**注意事项**

type指定内容和null必须由单引号' '包围，否则执行结果为空或出错。错误示范如：select int '123'||null; 或 select int 123||'null';

**示例**

**示例1** ：PG兼容模式下查看执行结果。

1、创建数据库，确定兼容性。

CREATE DATABASE db1 DBCOMPATIBILITY='PG';

\c db1

2、执行命令查看int数据类型+null结合的执行结果。

select int '123'||'null';

执行结果：

?column?

----------

123null

(1 row)

**示例2** ：其他兼容模式下查看执行结果。

1、创建数据库，确定兼容性。

CREATE DATABASE db1 DBCOMPATIBILITY='A';

\c db1

2、执行命令查看int数据类型+null结合的执行结果。

select int '123'||'null';

执行结果：

?column?

----------

123null

(1 row)

## 表增加自增列

**功能描述**

对已存在的表，支持修改时对表增加自增列。

**语法格式**

ALTER TABLE test ADD COLUMN column\_id serial|bigserial|smallserial;

**参数说明**

**column\_id**

自增列名。

**注意事项**

* 该功能仅在数据库兼容模式为Postgresql时能够使用（即创建DB时DBCOMPATIBILITY='PG'），在其他数据库兼容模式下不能使用该特性。
* 增加一列自增列的前提是表存在。
* 一次修改操作能增加一列自增列。
* 本功能不支持在列存表、临时表中修改。

**示例**

1、创建数据库，设置兼容性为PG。

CREATE DATABASE my\_test DBCOMPATIBILITY='PG';

\c my\_test

show sql\_compatibility;

查看兼容性结果为：

sql\_compatibility

-------------------

PG

(1 row)

2、创建表。

CREATE TABLE test(a int);

3、修改表增加一列自增列。

ALTER TABLE test ADD COLUMN id1 serial;

4、往表中插入数据后查看表内容。

insert into test values(1);

insert into test values(0);

insert into test values(2);

select \* from test;

返回结果为：

a | id1

---+-----

1 | 1

0 | 2

2 | 3

(3 rows)

## 创建临时序列

**功能描述**

在PG兼容模式下，Vastbase G100支持创建临时序列功能。创建的临时序列会在会话退出时删除。

**语法格式**

CREATE [ TEMPORARY | TEMP ] SEQUENCE name [as data\_type]

[ INCREMENT [ BY ] increment ]

[ MINVALUE minvalue | NO MINVALUE | NOMINVALUE ] [ MAXVALUE maxvalue | NO MAXVALUE | NOMAXVALUE]

[ START [ WITH ] start ] [ CACHE cache ] [ [ NO ] CYCLE | NOCYCLE ]

[ OWNED BY { table\_name.column\_name | NONE } ];

**参数说明**

* **TEMPOPARY/TEMP**

如果声明了这个修饰词，那么该序列对象只为这个会话创建，并且在会 话结束的时候自动删除。

* **name**

序列的名称。

* **data\_type**

序列的数据类型，Vastbase G100支持的数据类型有smallint、 integer和bright，其中bright是默认值。

* **increment**

指定序列的步长。一个正数将生成一个递增的序列，一个负数将生成一 个递减的序列。缺省值为1。

* **MINVALUE minvalue | NO MINVALUE| NOMINVALUE**

执行序列的最小值。如果没有声明minvalue或者声明NOMINVALUE， 则递增序列的缺省值为1，递减序列的缺省值为-263-1。 NOMINVALUE等价于NO MINVALUE。

* **MAXVALUE maxvalue | NO MAXVALUE| NOMAXVALUE**

执行序列的最大值。如果没有声明maxvalue或者声明了NO MAXVALUE，则递增序列的缺省值为263-1，递减序列的缺省值为-1。 NOMAXVALUE等价于NO MAXVALUE。

* **start**

指定序列的起始值。缺省值：对于递增序列为minvalue，递减序列为 maxvalue。

* **cache**

为了快速访问，而在内存中预先存储序列号的个数。缺省值为1，表示 一次只能生成一个值，也就是没有缓存。

* **CYCLE**
* 用于使序列达到maxvalue或者minvalue后可循环并继续下去。
* 如果声明了NO CYCLE，则在序列达到其最大值后任何对nextval的调用都会返回一个错误。
* NOCYCLE的作用等价于NO CYCLE。
* 缺省值为NO CYCLE。
* 若定义序列为CYCLE，则不能保证序列的唯一性。
* **OWNED BY**
* 将序列和一个表的指定字段进行关联。这样，在删除那个字段或其所在表的时候会自动删除已关联的序列。关联的表和序列的所有者必须 是 同一个用户，并且在同一个模式中。需要注意的是，通过指定OWNEDBY，仅仅是建立了表的对应列和sequence之间关联关系，并不会在插入数据时在该列上产生自增序列。
* 缺省值为OWNED BY NONE，表示不存在这样的关联。

**注意事项**

* 该功能仅在数据库兼容模式为PG时能够使用（即创建DB时DBCOMPATIBILITY='PG'），在其他数据库兼容模式下不能使用该特性。
* 用户需要具备创建序列的权限。
* 该功能仅支持行存表。
* 该功能不支持在高可用场景下使用。
* 不支持在事务块中修改序列的maxvalue。

**示例**

1、创建数据库my\_test，检查兼容性。

CREATE DATABASE my\_test DBCOMPATIBILITY 'PG';

\c my\_test

show sql\_compatibility;

2、创建一个名为seq的临时序列，序列的起始值为10。

create temp sequence seq start 10;

3、查看这个序列。

\d seq

返回结果如下，表示可以查询到临时序列的信息。

Sequence "pg\_temp\_\_1\_2\_140135787915008.seq"

Column | Type | Value

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

sequence\_name | name | seq

last\_value | bigint | 10

start\_value | bigint | 10

increment\_by | bigint | 1

max\_value | bigint | 9223372036854775807

min\_value | bigint | 1

cache\_value | bigint | 1

log\_cnt | bigint | 0

is\_cycled | boolean | f

is\_called | boolean | f

uuid | bigint | 0

type\_id | oid | 20

4、退出当前会话，进入新会话，查询seq序列。

\d seq

返回结果如下，显示没有序列seq。

Did not find any relation named "seq".

## 支持COPY语法中format值不带引号

**功能描述**

在PG兼容模式下，copy语法中的format值可以不带引号。

**语法格式**

* 从一个文件拷贝至一个表：

COPY table\_name [ ( column\_name [, ...] ) ]

FROM { 'filename' | STDIN }

WITH (FORMAT binary);

* 把一个表拷贝到一个文件：

COPY table\_name [ ( column\_name [, ...] ) ]

TO { 'filename' | STDOUT }

WITH (FORMAT binary);

COPY query

TO { 'filename' | STDOUT }

WITH (FORMAT binary);

fig: 说明：

完整copy语法内容及其参数说明参见：开发者指南 -> SQL语法参考 ->SQL语法 -> copy章节。

**参数说明**

* **column\_name**

可选的待拷贝字段列表。

取值范围：如果没有声明字段列表，将使用所有字段。

* **table\_name**

表的名称（可以有模式修饰）。

取值范围：已存在的表名。

* **STDIN**

声明输入是来自标准输入。

* **STDOUT**

声明输出打印到标准输出。

* **FORMAT**

​ 数据源文件的格式。

​ 取值范围：CSV、TEXT、FIXED、BINARY。

* **binary**

​ 数据源文件的格式之一，binary形式的选项会使得所有的数据被存储/ 读作二进制格式而不是文本。 这比TEXT和CSV格式的要快一些，但 是一个BINARY格式文件可移植性比较差。

* **query**

查询语句。

**注意事项**

导入内容不能超出数值范围。

**示例**

1、创建数据库，并指定兼容模式。

CREATE DATABASE my\_test DBCOMPATIBILITY='PG';

\c my\_test

show sql\_compatibility;

查看兼容性：

sql\_compatibility

-------------------

PG

(1 row)

2、创建csv文件，添加内容后保存退出。

vi /tmp/import\_test.csv

1,zhangsan,-92254.08,9月花销,2022/09/28,20:50:45

2,lisi,87758.07,8月收入,2022/09/05,14:20:15

3,wangwu,7758.07,8月报销,2022/09/01,16:43:05

3、创建表。

create table test(

id integer,

name varchar(100),

cash money,

info text,

date date,

time time

);

create table test\_bak(

id integer,

name varchar(100),

cash money,

info text,

date date,

time time

);

4、从文件导入数据并查询。

copy test from '/tmp/import\_test.csv' DELIMITERS ',';

select \* from test;

结果显示为：

id | name | cash | info | date | time

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

1 | zhangsan | -$92,254.08 | 9月花销 | 2022-09-28 | 20:50:45

2 | lisi | $87,758.07 | 8月收入 | 2022-09-05 | 14:20:15

3 | wangwu | $7,758.07 | 8月报销 | 2022-09-01 | 16:43:05

(3 rows)

5、binary导出数据到csv文件。

copy test to '/tmp/test\_out.csv' with (format BINARY);

结果显示为：

COPY 3

6、binary导入数据并查询。

copy test\_bak from '/tmp/test\_out.csv' with (format BINARY);

select \* from test\_bak;

显示结果为：

id | name | cash | info | date | time

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

1 | zhangsan | -$92,254.08 | 9月花销 | 2022-09-28 | 20:50:45

2 | lisi | $87,758.07 | 8月收入 | 2022-09-05 | 14:20:15

3 | wangwu | $7,758.07 | 8月报销 | 2022-09-01 | 16:43:05

(3 rows)



电话：010-82838118

地址：北京市海淀区学院路30号科大天工大厦B座6层

官网：www.vastdata.com.cn