名字 别名 描述

bigserial serial8 自动增长的8字节整数

box 平面上的普通方框

bytea 二进制数据（“字节数组”）

cidr IPv4或IPv6网络地址

circle 平面上的圆

inet IPv4或IPv6主机地址

json 文本 JSON 数据

jsonb 二进制 JSON 数据，已分解

line 面上的无限长的线

lseg 平面上的线段

macaddr MAC（Media Access Control） 地址

macaddr8 MAC (Media Access Control) 地址 (EUI-64 格式)

money 货币数量

path 平面上的几何路径

pg\_lsn PostgreSQL 日志序列号

point 平面上的几何点

polygon 平面上的封闭几何路径

smallserial serial2 自动增长的2字节整数

serial serial4 自动增长的4字节整数

text 变长字符串

tsquery 文本搜索查询

tsvector 文本搜索文档

txid\_snapshot 用户级别事务ID快照

uuid 通用唯一标识码

兼容性

下列类型（或者及其拼写）是SQL指定的：bigint、bit、bit

varying、boolean、char、character varying、character、varchar、date、double

precision、integer、interval、numeric、decimal、real、smallint、time（有时区

或无时区）、timestamp（有时区或无时区）、xml。

名字 别名 描述

xml XML数据

time [ (p) ] [ without time zone ] 一天中的时间（无时区）

time [ (p) ] with time zone

timetz 一天中的时间，包括时区

timestamp [ (p) ] [ without timezone ] 日期和时间（无时区）

timestamp [ (p) ] with time zone

timestamptz 日期和时间，包括时区

smallint int2 有符号2字节整数

real float4 单精度浮点数（4字节）

numeric [ (p, s) ]

decimal [ (p, s) ] 可选择精度的精确数字

integer int, int4 有符号4字节整数

interval [ fields ] [ (p) ] 时间段

date 日历日期（年、月、日）

double precision float8 双精度浮点数（8字节）

character [ (n) ] char [ (n) ] 定长字符串

character varying [ (n) ] varchar [ (n) ] 变长字符串

bit [ (n) ] 定长位串

bit varying [ (n) ] varbit 变长位串

boolean bool 逻辑布尔值（真/假）

Bigint int8 有符号的8字节整数

数字类型

名字 存储尺寸 描述 范围

(整数)

smallint 2字节 小范围整数 -32768 to +32767

integer 4字节 整数的典型选择 -2147483648 to+2147483647

bigint 8字节 大范围整数 -9223372036854775808to

+9223372036854775807

(任意精度)

decimal 可变 用户指定精度，精确 最高小数点前131072位，

以及小数点后1638位

numeric 可变 用户指定精度，精确 最高小数点前131072位，

以及小数点后1638位

(浮点数)

real 4字节 可变精度，不精确 6位十进制精度

double precision

8字节 可变精度，不精确 15位十进制精度

(序列数：用于自增长序列)

smallserial 2字节 自动增加的小整数 1到32767

serial 4字节 自动增加的整数 1到2147483647

bigserial 8字节 自动增长的大整数 1到9223372036854775807

类型smallint、integer和bigint存储各种范围的全部是数字的数，也就是没有小数部分的数字。试图存储超出范围以外的值将导致一个错误。

常用的类型是integer，因为它提供了在范围、存储空间和性能之间的最佳平衡。一般只有在磁盘空间紧张的时候才使用 smallint类型。而只有在integer的范围不够的时候才使用bigint。

SQL只声明了整数类型integer（或int）、smallint和bigint。类型int2、int4和int8都是扩展。

numeric类型可以存储非常大的数字并且准确地进行计算。 我们特别建议将它用于货币金额和其它要求精确计算的场合。

不过， numeric类型上的算术运算比整数类型，浮点数类型要慢很多。

一个numeric类型的标度 (scale)是小数部分的位数，精度(precision) 是全部数据位的数目，也就是小数点两边的位数总和。

因此数字 23.5141 的精度为 6 而标度为 4 。语法：NUMERIC(precision, scale)

如果你关心移植性，那你最好总是显式声明精度和比例

显式指定类型精度时的最大允许精度为 1000，没有指定精度的NUMERIC受到上表描述的限制所控制。

除了普通的数字值之外，numeric类型允许特殊值NaN， 表示“不是一个数字”。任何在NaN上面的操作都生成另外一个NaN。

如果在 SQL 命令里把这些值当作一个常量写，你必须在其周围放上单引号，例如UPDATE table SET x = 'NaN'。

在输入时，字串NaN被识别为大小写无关。

在“不是一个数字”概念的大部分实现中，NaN被认为不等于任何其他数字值（包括NaN）。为了允许numeric值可以被排序和使用基于树的索

引，PostgreSQL把NaN值视为相等，并且比所有非NaN值都要大。

除了普通的数字值之外，浮点类型还有几个特殊值：

Infinity

-Infinity

NaN

这些值分别表示 IEEE 754 特殊值"正无穷大"、"负无穷大"、 "不是一个数字"。在不遵循 IEEE 754 浮点算术的机器上， 这些值的含义可能不是预期的。

如果在 SQL 命令里把这些数值当作常量写， 你必须在它们周围放上单引号，像这样：UPDATE table SET x = 'Infinity'。

输入时，这些值是以大小写无关的方式识别的。

smallserial,serial和bigserial类型不是真正的类型， 只是为在表中创建唯一标识做的概念上的便利。类似其它一些数据库中的AUTO\_INCREMENT 属性。

货币类型

名字 存储容量 描述 范围

money 8 字节 货币金额 -92233720368547758.08 到 +92233720368547758.07

numeric, int和bigint数据类型的值可以转化为 money类型。从real和double precision 数据类型的转换可以通过先转化为numeric类型

浮点数不应该用来处理货币类型，因为潜在的圆整可能导致错误。

money值可以被转换为numeric而不丢失精度， 转换为其他类型可能丢失精度。

字符类型

名字 描述

character varying(n), varchar(n) 变长，有长度限制

character(n), char(n) 定长，不足补空白

text 变长，无长度限制

character varying(n) 和character(n)，这里的n 是一个正整数。两种类型都可以存储最多n个字符的字符串（没有字节）。

试图存储更长的字符串到这些类型的字段里会产生一个错误， 除非超出长度的字符都是空白，这种情况下该字符串将被截断为最大长度。

如果要存储的字符串比声明的长度短，类型为character的数值将会用空白填满； 而类型为character varying的数值将只是存储短些的字符串。

如果我们明确地把一个数值转换成character varying(n) 或character(n)，那么超长的数值将被截断成n 个字符，且不会抛出错误。

varchar(n)和char(n) 分别是character varying(n) 和character(n)的别名， 没有声明长度的character等于character(1)；

如果不带长度说明词使用character varying， 那么该类型接受任何长度的字符串。

character类型的数值物理上都用空白填充到指定的长度n， 并且以这种方式存储和显示。

不过，在比较两个character 类型的值时，尾随的空白是无关紧要的，不需要理会。

请注意， 在character varying和text数值里， 结尾的空白是有语意的。 并且当使用模式匹配时，如LIKE，使用正则表达式。

这三种类型之间没有性能差别，除了当使用填充空白类型时的增加存储空间， 和当存储长度约束的列时一些检查存入时长度的额外的CPU周期。

虽然在某些其它的数据库系统里，character(n) 有一定的性能优势，但在PostgreSQL里没有。

事实上，character(n)通常是这三个中最慢的， 因为额外存储成本。在大多数情况下，应该使用text 或character varying。

不管怎样，允许存储的最长字符串大概是1GB 。允许在数据类型声明中出现的n 的最大值比这还小。

如果你想存储没有特定上限的长字符串，那么使用text 或没有长度声明的character varying，而不要选择一个任意长度限制。

二进制数据类型

名字 存储空间 描述

bytea 1或4字节加上实际的二进制字符串 变长的二进制字符串

日期/时间类型

名字 存储空间 描述 最低值 最高值 分辨率

timestamp [ (p) ] [ without time zone ] 8 字节 日期和时间(无时区) 4713 BC 294276 AD 1 微秒 / 14 位

timestamp [ (p) ] with time zone 8 字节 日期和时间，有时区 4713 BC 294276 AD 1 微秒 / 14 位

date 4 字节 只用于日期 4713 BC 5874897 AD 1 天

time [ (p) ] [ without time zone ] 8 字节 只用于一日内时间 00:00:00 24:00:00 1 微秒 / 14 位

time [ (p) ] with time zone 12 字节 只用于一日内时间，带时区 00:00:00+1459 24:00:00-1459 1 微秒 / 14 位

interval [ fields ] [ (p) ] 16 字节 时间间隔 -178000000 年 178000000 年 1 微秒 / 14 位

time, timestamp和interval接受一个可选的精度值 p以指明秒域中小数部分的位数。没有明确的缺省精度， p的范围对timestamp和interval类型是从0到6。

对于time类型，如果使用了八字节的整数存储，那么p 允许的范围是从 0 到 6 ，如果使用的是浮点数存储，那么这个范围是 0 到 10 。

布尔数据类型

名称 存储格式 描述

boolean 1 字节 真/假

"真"值的有效文本值是：

TRUE，'t'，'true'，'y'，'yes'，'on'，'1'

对于"假"，你可以使用下面这些：

FALSE，'f'，'false'，'n'，'no'，'off'，'0'

前导或尾随空白将被忽略，大小写无关。使用TRUE和FALSE 这样的字眼比较好(也是SQL兼容的用法)

XML 类型（略）

xml数据类型可以用于存储XML数据。 将XML数据存到text类型中的优势在于它能够为结构良好性来检查输入值，

并且还支持函数对其进行类型安全性检查。

随着默认XML选项的设置，如果字符串中包含一个文档类型声明， 那么你不能直接将其转换成xml类型，因为XML内容片断的定义不支持。

如果非得需要这么做，要么使用XMLPARSE，要么更改XML选项。

(以下属于postgresql有的类型)

枚举类型（略）

枚举类型是一个包含静态和值的有序集合的数据类型。等于某些编程语言中的enum 类型。

一个枚举类型可以是一周中的天，或者一块数据的状态值的集合。

几何类型

名字 存储空间 说明 表现形式

point 16 字节 平面中的点 (x,y)

line 32 字节 (无穷)直线 {A,B,C}

lseg 32 字节 (有限)线段 ((x1,y1),(x2,y2))

box 32 字节 矩形 ((x1,y1),(x2,y2))

path 16+16n 字节 闭合路径(与多边形类似) ((x1,y1),...)

path 16+16n 字节 开放路径 [(x1,y1),...]

polygon 40+16n 字节 多边形(与闭合路径相似) ((x1,y1),...)

circle 24 字节 圆 <(x,y),r> (圆心和半径)

网络地址类型

名字 存储空间 描述

cidr 7 或 19 字节 IPv4 或 IPv6 网络

inet 7 或 19 字节 IPv4 或 IPv6 主机和网络

macaddr 6 字节 MAC 地址

macaddr8 8 字节 MAC 地址 (EUI-64 格式)

inet在一个数据域里保存主机的 IPv4 或 IPv6 地址，以及一个可选的等效子网。

cidr保存一个 IPv4 或 IPv6 网络地址。其输入和输出遵循无类别的互联网域路由习惯。

位串类型（略）

位串就是一串 1 和 0 的字符串。它们可以用于存储和直观化位掩码。 我们有两种 SQL 位类型：bit(n) 和bit varying(n)， 这里的n是一个正整数。

bit类型的数据必须准确匹配长度n， 试图存储短些或者长一些的数据都是错误的。bit varying 类型数据是最长n的变长类型；更长的串会被拒绝。

写一个没有长度的bit等效于bit(1)， 没有长度的bit varying意思是没有长度限制。

文本搜索类型（略）

PostgreSQL提供了两种数据类型用于支持全文检索， 即通过自然语言documents的集合来找到那些匹配一个query 的检索。

tsvector类型产生一个文档（以优化了全文检索的形式）， tsquery类型表示一个文本查询。

UUID 类型（略）

uuid数据类型用来存储RFC 4122，ISO/IEF 9834-8:2005以及相关标准定义的通用唯一标识符（UUID）。

（一些系统认为这个数据类型为全球唯一标识符，或GUID。） 这个标识符是一个由算法产生的128位标识符，

使它不可能在已知使用相同算法的模块中和其他方式产生的标识符相同。

因此，对分布式系统而言，这种标识符比序列能更好的提供唯一性保证，因为序列只能在单一数据库中保证唯一。

JSON 类型（略）

JSON数据类型可以用来存储JSON（JavaScript Object Notation）数据， 就像RFC 7159中指定的那样。

这样的数据也可以存储为text， 但是JSON数据类型更有利于检查每个存储的数值是可用的JSON值。

JSON原始类型 PostgreSQL类型 说明

string text \u0000是不允许的，如果数据库编码不是UTF8， 那么非ASCII Unicode逃逸也一样

number numeric 不允许NaN和infinity值

boolean boolean 只接受小写的true和false

null (none) SQL NULL 是一个不同的概念

Arrays（数组类型【略】）

PostgreSQL允许将字段定义成变长的多维数组。 数组类型可以是任何基本类型或用户定义类型，枚举类型或复合类型。 目前还不支持域的数组。

复合类型(略)

一个复合类型表示一行或一个记录的结构，它本质上就是一个域名和它们数据类型的列

表。PostgreSQL允许把复合类型用在很多能用简单类型的地方。例如，一个表的一列可以被

声明为一种复合类型。

范围类型（略）

范围数据类型代表着某一元素类型在一定范围内的值。 (此元素类型称为该范围的 子类型)。 例如，

timestamp 范围可能被用于代表一间会议室被预定的时间范围。 这种情况下数据 类型为tsrange ("timestamp range"的简写)，

并且timestamp 是子类型。子类型必须具备完整的排序，这样清晰定义了元素值在范围之内，之前，或者之后。

范围类型是有用的。因为他们代表了在单一范围内的许多元素值，并且清晰表达了诸如重叠范围等概念。

出于计划目的的时间和日期范围的使用是一个最清晰的例子；价格范围，仪器测量的范围等也有用。

PostgreSQL 提供下列内嵌范围类型：

int4range — integer的范围

int8range —bigint的范围

numrange —numeric的范围

tsrange —timestamp without time zone的范围

tstzrange —timestamp with time zone的范围

daterange —date的范围

对象标识符类型（略）

PostgreSQL在内部使用对象标识符(OID)作为各种系统表的主键。 同时，系统不会给用户创建的表增加一个 OID 系统字段

(除非在建表时声明了WITH OIDS 或者配置参数default\_with\_oids设置为开启)。oid 类型代表一个对象标识符。

除此以外oid还有几个别名：regproc, regprocedure, regoper, regoperator, regclass, regtype, regconfig, 和regdictionary。

目前oid类型用一个四字节的无符号整数实现。因此， 它不够提供大数据库范围内的唯一性保证，甚至在单个的大表中也不行。

因此，我们不鼓励在用户创建的表中使用 OID 字段做主键。OID 最好只是用于系统表。

pg\_lsn 类型（略）

pg\_lsn数据类型可以用来存储LSN（Log Sequencce Number日志序列数字） 数据，

它是一个指向XLOG中的位置的指针。这个类型是XLogRecPtr 的表示，是PostgreSQL的一种内部系统类型。

伪类型

PostgreSQL类型系统包含一系列特殊用途的条目， 它们按照类别来说叫做伪类型。伪类型不能作为字段的数据类型，

但是它可以用于声明一个函数的参数或者结果类型。伪类型在一个函数不只是简单地接受并返回某种SQL 数据类型的情况下很有用。

名字 描述

any 表示一个函数接受任何输入数据类型。

anyelement 表示一个函数接受任何数据类型。

anyarray 表示一个函数接受任意数组数据类型。

anynonarray 表示一个函数接受任意非数组数据类型。

anyenum 表示一个函数接受任意枚举数据类型。

anyrange 表示一个函数接受任意范围数据类型。

cstring 表示一个函数接受或者返回一个空结尾的 C 字符串。

internal 表示一个函数接受或者返回一种服务器内部的数据类型。

language\_handler 一个过程语言调用处理器声明为返回language\_handler。

fdw\_handler 一个外部数据封装器声明为返回fdw\_handler。

record 标识一个函数返回一个未声明的行类型。

trigger 一个触发器函数声明为返回trigger。

void 表示一个函数不返回数值。

opaque 一个已经过时的类型，以前用于所有上面这些用途。

注：以上文本选自于PostgreSQl中文文档：第 **8** 章 数据类型

SQL语句建立到在数据库的执行：

1. 一个由应用程序到PostgreSQL服务器的连接必须被建立。应用程序传递一个查询给服务

器并等待接收由服务器传回的结果。

2. 分析阶段对由应用程序传递的查询进行语法检查，并创建一个查询树。

3. 重写系统得到分析阶段创建的查询树，并查找可以应用到该查询树的任何规则（存储

在系统目录中）。对找到的规则，它会执行规则体中给定的转换。

重写系统的一个应用是实现视图。不论何时发生一个视图（即一个虚拟表）上的查询，

重写系统将用户查询重写为一个访问视图定义中给定的基本表的查询来替代。

4. 规划器/优化器接手（重写过的）查询树并创建一个将被作为执行器输入的查询计划。

它会先创建所有可能导向相同结果的路径。例如，如果在一个被扫描的关系上有一个索

引，则有两条可供扫描的路径。其中之一是一个简单的顺序扫描，而另一个则是使用索

引。接下来执行每条路径所需的代价被估算出来并且代价最低的路径将被选中。代价最

低的路径将被扩展成一个完整的计划可供执行器使用。

5. 执行器递归地逐步通过计划树并按照计划表述的方式获取行。执行器在扫描关系时会使

用存储系统、执行排序和连接、估算条件并最后归还得到的行。

PostgreSQL的引擎：

引擎分为三部分：

查询引擎：最优访问路径（分析器{分析器[语法分析器][词法分析器]，转换器（创建查询树）}，PostgreSQL规则系统（视图），规划器/优化器（选择最优路径））

执行引擎：文件访问操作接口（执行器,返回结果给客户端）

存储引擎：数据和文件的访问

上述文档：选自于PostgreSQL中文社区的**PostgreSQL执行引擎简介 -- 李浩**

网址：<http://www.postgres.cn/news/viewone/1/203>

部分节选自PostgreSQl中文文档：第 **50** 章 **PostgreSQL**内部概述

开源社区中国：https://www.oschina.net/question/96003\_2140840?sort=time

1： RDBMS ， 毫无疑问， 只有postgresql可选

2: Key ->value ， postgresql 的jsonb数据类型完美支持.

3：全文搜索 ， postgresql 完美支持

4：big data , Olap , greenplum