Atitit 常用数据类型有哪些

目录

[1.1. 数值类型 整数 小数 自增整数 1](#_Toc12719)

[1.2. 货币类型 1](#_Toc23623)

[1.3. 字符类型 字符串 2](#_Toc8208)

[1.4. 日期/时间类型 时间戳 2](#_Toc19635)

[1.5. Url类型 2](#_Toc22212)

[1.6. 几何类型 2](#_Toc24790)

[1.7. 尔类型 2](#_Toc316)

[1.8. 枚举类型 3](#_Toc32235)

[1.9. Eml类型 3](#_Toc24877)

[1.10. Cellphone numb 3](#_Toc4773)

[1.11. 3](#_Toc30446)

[1.12. 串类型 3](#_Toc177)

[1.13. 文本搜索类型 3](#_Toc7548)

[1.14. UUID 类型 3](#_Toc16144)

[1.15. XML 类型 4](#_Toc16358)

[1.15.1. 创建XML值 4](#_Toc25694)

[1.16. JSON 类型 4](#_Toc15100)

[1.17. 数组类型 4](#_Toc71)

[1.17.1. 声明数组 4](#_Toc21860)

[1.18. 复合类型 5](#_Toc9661)

[1.18.1. 声明复合类型 5](#_Toc18835)

[1.19. 范围类型 5](#_Toc21409)

[1.20. 对象标识符类型 5](#_Toc30769)

[1.21. 伪类型 5](#_Toc6150)

[2. Rf 6](#_Toc20205)

## 数值类型 整数 小数 自增整数

数值类型由 2 字节、4 字节或 8 字节的整数以及 4 字节或 8 字节的浮点数和可选精度的十进制数组成。

下表列出了可用的数值类型。

## 货币类型

money 类型存储带有固定小数精度的货币金额。

numeric、int 和 bigint 类型的值可以转换为 money，不建议使用浮点数来处理处理货币类型，因为存在舍入错误的可能性。

| **名字** | **存储容量** | **描述** | **范围** |
| --- | --- | --- | --- |
| money | 8 字节 | 货币金额 | -92233720368547758.08 到 +92233720368547758.07 |

## 字符类型 字符串

下表列出了 PostgreSQL 所支持的字符类型：

|  |  |
| --- | --- |
| **序号** | **名字 & 描述** |
| 1 | **character varying(n), varchar(n)**  变长，有长度限制 |
| 2 | **character(n), char(n)**  f定长,不足补空白 |
| 3 | **text**  变长，无长度限制 |

## 日期/时间类型 时间戳

下表列出了 PostgreSQL 支持的日期和时间类型。

## Url类型

## 几何类型

几何数据类型表示二维的平面物体。

下表列出了 PostgreSQL 支持的几何类型。

最基本的类型：点。它是其它类型的基础。

## 尔类型

PostgreSQL 支持标准的 boolean 数据类型。

boolean 有"true"(真)或"false"(假)两个状态， 第三种"unknown"(未知)状态，用 NULL 表示。

| **名称** | **存储格式** | **描述** |
| --- | --- | --- |
| boolean | 1 字节 | true/false |

## 枚举类型

枚举类型是一个包含静态和值的有序集合的数据类型。

PostgtesSQL中的枚举类型类似于 C 语言中的 enum 类型。

## Eml类型

## Cellphone numb

## 

## 串类型

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

全文检索即通过自然语言文档的集合来找到那些匹配一个查询的检索。

PostgreSQL 提供了两种数据类型用于支持全文检索：

|  |  |
| --- | --- |
| **序号** | **名字 & 描述** |
| 1 | **tsvector**  tsvector 的值是一个无重复值的 lexemes 排序列表， 即一些同一个词的不同变种的标准化。 |
| 2 | **tsquery**  tsquery 存储用于检索的词汇，并且使用布尔操作符 &(AND)，|(OR)和!(NOT) 来组合它们，括号用来强调操作符的分组。 |

## UUID 类型

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。

UUID 被写成一个小写十六进制数字的序列，由分字符分成几组， 特别是一组8位数字+3组4位数字+一组12位数字，总共 32 个数字代表 128 位， 一个这种标准的 UUID 例子如下：

a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11

## XML 类型

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。 要使用这个数据类型，编译时必须使用 **configure --with-libxml**。

xml 可以存储由XML标准定义的格式良好的"文档"， 以及由 XML 标准中的 **XMLDecl? content** 定义的"内容"片段， 大致上，这意味着内容片段可以有多个顶级元素或字符节点。 xmlvalue IS DOCUMENT 表达式可以用来判断一个特定的 xml 值是一个完整的文件还是内容片段。

### **创建XML值**

使用函数 xmlparse: 来从字符数据产生 xml 类型的值：

XMLPARSE (DOCUMENT '<?xml version="1.0"?><book><title>Manual</title><chapter>...</chapter></book>')

XMLPARSE (CONTENT 'abc<foo>bar</foo><bar>foo</bar>')

## JSON 类型

json 数据类型可以用来存储 JSON（JavaScript Object Notation）数据， 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。

此外还有相关的函数来处理 json 数据：

|  |  |
| --- | --- |
| **实例** | **实例结果** |
| array\_to\_json('{{1,5},{99,100}}'::int[]) | [[1,5],[99,100]] |
| row\_to\_json(row(1,'foo')) | {"f1":1,"f2":"foo"} |

## 数组类型

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。

### **声明数组**

创建表的时候，我们可以声明数组，方式如下：

CREATE TABLE sal\_emp (

name text,

pay\_by\_quarter integer[],

schedule text[][]);

pay\_by\_quarter 为一位整型数组、schedule 为二维文本

## 复合类型

复合类型表示一行或者一条记录的结构； 它实际上只是一个字段名和它们的数据类型的列表。PostgreSQL 允许像简单数据类型那样使用复合类型。比如，一个表的某个字段可以声明为一个复合类型。

### **声明复合类型**

下面是两个定义复合类型的简单例子：

## 范围类型

范围数据类型代表着某一元素类型在一定范围内的值。

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。

PostgreSQL 内置的范围类型有：

int4range — integer的范围

int8range —bigint的范围

## 对象标识符类型

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。

## 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。

下表列出了所有的伪类型：

# Rf

H5数据表单效验

Atitit mysql 数据类型 5.7.9

目录

[1.1. 数值类型 1](#_Toc29217)

[2. 字符串 2](#_Toc18277)

[3. 时间日期 2](#_Toc19263)

[4. 地理位置 2](#_Toc12274)

[5. 几何数据的存储，生成，分析，优化。 空间数据类型（存储） 2](#_Toc32647)

[6. 复合json 3](#_Toc27760)

[6.1. Enum 枚举类型 set 3](#_Toc21678)

[7. 问题，， 3](#_Toc2361)

[7.1. navacite 不支持查询json问题，可能是版本太低，使用java直接查询 3](#_Toc30005)

[7.2. Json索引问题 3](#_Toc24595)

[8. JSON数据查询： 4](#_Toc28658)

[8.1. 2）插入数据 4](#_Toc30747)

[8.2. 首先通过json对象中的某个键值对来做条件查询 5](#_Toc31934)

[8.2.1. 其他JSON函数 5](#_Toc2608)

[9. ref 6](#_Toc30652)