数值类型介绍

|  |  |  |  |
| --- | --- | --- | --- |
| 类型名称 | 存储空间 | 描述 | 范围 |
| smallint | 2字节 | 小范围整数 | -2^15 ~  2^15-1  /  -32768 to +32767 |
| int | 4字节 | 常用整数 | -2^31 ~ 2^31-1 / -2147483648 to +2147483647 |
| bigint | 8字节 | 大范围整数 | -2^63 ~ 2^63-1  -9223372036854775808 to +9223372036854775807 |
| numeric 或 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16383位 |
| real | 4字节 | 可变精度，不精确 | 6位十进制数字精度 |
| double precision | 8字节 | 可变精度，不精确 | 15位十进制数字精度 |
| smallserial | 2字节 | 小范围自增整数 | 1 ~ 2^15-1 / 1到32767 |
| serial | 4字节 | 自增整数 | 1 ~ 2^31-1 / 1到2147483647 |
| bigserial | 8字节 | 大范围自增整数 | 1 ~ 2^63-1 / 1到9223372036854775807 |

**整数类型**

整数类型有3种： smallint、 int、 bigint， 常用的数据类型是int（或integer） ， 因为它提供了在范围、 存储空间、 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。 通常， 只有integer类型的取值范围不够时才使用bigint类型， 因为前者的执行速度绝对快得多。

**精确的小数类型**

精确的小数类型可用numeric、decimal表示。numeric类型和decimal类型是等效的， 这两种类型都可以存储最多10万位精度的数字， 并且可准确地进行计算。 它们特别适用于货币金额和其他要求精确计算的场合。 不过， 基于numeric类型的算术运算相比于基于整数类型或者下面介绍的浮点数类型的算术运算， 其速度要慢很多。

如果要声明一个字段的类型为numeric， 可以用下面的语句：

numeric：创建一个可以存储任意精度的数值

numeric(m,n)：创建一个m位的数，n指小数位数

numeric(m) ：创建一个m位的数

如果字段声明了标度(n)， 超过小数点位数的标度会自动4舍5入后进行存储。 而对于既没有声明精度也没有声明标度的numeric类型来说， 则会原样存储。对于声明了精度的数值， 如果INSERT语句插入的数值超出声明的精度范围， 则会报错。

示例：

create table t1(id1 numeric(3),id2 numeric(3,0),id3 numeric(3,2),id4 numeric);

insert into t1 values(5.1,5.6,5.123,5.123);--正常执行

insert into t1 values(5.1,5.6,15.123,5.123); -- 报错

**浮点数类型**

数据类型real和double precision是不精确的、 变精度的数字类型。不精确意味着一些值不能准确地转换成内部格式并且是以近似的形式存储的，因此存储和检索一个值可能出现一些缺失。

对于浮点数， 需要注意如下几个方面：

1.如果要求精确地计算（比如计算货币金额） ， 应使用numeric类型。

2.如果想用这些类型做任何重要的复杂计算， 尤其是那些对范围情况（无穷/下溢） 严重依赖的复杂计算， 那么应该仔细评诂你的实现。

3.用两个浮点数值进行等值比较不可能总是按照期望地进行。

除了普通的数字值之外， 浮点类型还有以下几个特殊值：

Infinity：正无穷大

-Infinity：负无穷大

NaN：不是一个数字

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，例如UPDATE table SET x = '-Infinity'。 在输入时，这些字符串是以大小写不敏感的方式识别的

**序列类型**

在PostgreSQL中， smallserial、serial和bigserial类型不是真正的类型，它们只是为了创建唯一标识符列而存在的方便符号（类似其它一些数据库中支持的AUTO\_INCREMENT属性）。

CREATE TABLE t1 (id SERIAL);--使用序列

等价于以下语句：

CREATE SEQUENCE seq1;

CREATE TABLE t1 (id integer NOT NULL DEFAULT nextval('seq1'));

ALTER SEQUENCE seq1 OWNED BY t1.id;

**货币类型**

money类型可以存储固定小数精度的货币数字， 与浮点数不同， 它是完全保证精度的

|  |  |  |  |
| --- | --- | --- | --- |
| 名字 | 存储尺寸 | 描述 | 范围 |
| money | 8 bytes | 货币额 | -92233720368547758.08到+92233720368547758.07 |

其输出格式与参数lc\_monetary的设置有关， 不同的国家其货币输出格式也不相同。如果是中文， 输出的是“￥”， 如果是英文（美国）， 则输出为“$”。

例子：

show lc\_monetary;--查看货币格式

set lc\_monetary = 'en\_US.UTF-8';--设置英文环境

SELECT '12.34'::money;

set lc\_monetary = 'zh\_CN.UTF-8';--设置中文环境

SELECT '12.34'::money;

**数学函数和操作符**

PostgreSQL支持丰富的数学操作符，例如支持加、减、乘、除、取模操作符。

|  |  |  |
| --- | --- | --- |
| 操作符 | 描述 | 例子 |
| + | 加 | 2 + 3 → 5 |
| - | 减 | 2 - 3 → -1 |
| \* | 乘 | 2 \* 3 → 6 |
| / | 除 | 4 / 2 → 2 |
| % | 模（取余） | 5 % 4 → 1 |
| ^ | 指数 | 2 ^ 3 → 8 |
| @ | 绝对值 | @ -5.0 → 5 |

|  |  |  |
| --- | --- | --- |
| 函数 | 描述 | 例子 |
| abs | 绝对值 | abs(-17.4) → 17.4 |
| mod | 模（取余） | mod(9,4) → 1 |
| round | 四舍五入 | round(42.4) → 42 |
| ceil | 返回大于等于给定参数的值 | ceil(42.2) → 43 |
| floor | 返回小于等于给定参数的值 | floor(42.8) → 42 |
| trunc | 截断整数 (向零靠近) | trunc(42.8) → 42 |
| random() | 返回 0.0 <= x < 1.0 之间的随机值 | random() → 0.897124072839091 |

其他数学函数和操作符：http://www.postgres.cn/docs/14/functions-math.html