字符类型分为变长和定长类型。

|  |  |
| --- | --- |
| 类型名称 | 描述 |
| varchar(n)  character varying(n) | 变长，最大为1GB。未声明长度的character varying， 那么该类型接受任何长度的字符串。 |
| char(n)  character(n) | 定长，不足补空白，未声明长度的character等价于character(1) |
| text | 变长，无长度限制。 |

varchar (n)和char(n)， 其中n是一个正整数，如果存储的字符长度超过n，则会报错。如果存储的字符串位数比n小，varchar仅存储实际的位数，而char会用空白填充。text类型可以存储任何长度的串。

长的字符串会被系统自动压缩，也可能会存储在toast表中， 这里只放一个指针， 这样它们就不会干扰对短字段值的快速访问了。 不管怎样， 允许存储的最长字符串大概是1GB。

这三种类型之间没有性能差别，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)通常是这三种类型之中最慢的一个，因为它需要额外的存储开销。在大多数情况下，应该使用text或者character varying。

CREATE TABLE t1 (a varchar(4),b char (4));

INSERT INTO t1 VALUES ('ok',’ok’);

SELECT char\_length(a),octet\_length(b) FROM t1;--查看字符串位数和字节数

CREATE TABLE t1 (b varchar(5));

INSERT INTO t1 VALUES ('ok');

INSERT INTO t1 VALUES ('good ');

INSERT INTO t1 VALUES (‘too long’);--报错

INSERT INTO t1 VALUES ('too long'::varchar(5)); -- explicit truncation

SELECT b, char\_length(b) FROM t1;

**字符串函数和操作符**

|  |  |  |
| --- | --- | --- |
| 函数 | 描述 | 例子 |
| || | 连接两个字符串 | 'Post' || 'greSQL' → PostgreSQL |
| char\_length | 返回字符串中的字符数 | char\_length('josé') → 4 |
| lower | 将字符串转换为全部小写 | lower('TOM') → tom |
| octet\_length | 返回字符串的字节数 | octet\_length('josé') → 4 |
| overlay | 替换子串 | overlay('hello world','yes',2) → hyeso world |
| position | 返回字符串的位置 | position('ll' in 'hello world') → 3 |
| substring | 提前字符串中的子串 | substring('Thomas',2,3) → hom |
| upper | 将字符串转换为全部大写 | lower('tom') → TOM |
| initcap | 将单词的第一个字母转换为大写 | initcap('hi THOMAS') → Hi Thomas |
| left | 返回左边的 n 字符 | left('abcde', 2) → ab |
| length | 返回字符串中的字符数 | length('jose') → 4 |
| md5 | 计算参数的 MD5 值 | md5('abc') → 900150983cd24fb0​d6963f7d28e17f72 |
| repeat | 重复字符串多少次 | repeat('Pg', 4) → PgPgPgPg |
| replace | 替换子串 | replace('abcdefabcdef', 'cd', 'XX') → abXXefabXXef |
| reverse | 颠倒字符串的顺序 | reverse('abcde') → edcba |
| right | 返回右边的 n 字符 | right('abcde', 2) → de |

其他字符串函数和操作符：http://www.postgres.cn/docs/14/functions-string.html