增删改查

https://blog.csdn.net/qq\_28289405/article/details/80249509?ops\_request\_misc=&request\_id=&biz\_id=102&utm\_term=pgadmin%E4%B8%AD%E5%88%9B%E5%BB%BA%E5%A5%BD%E7%9A%84%E8%A1%A8%E6%A0%BC%E5%A6%82%E4%BD%95%E6%8F%92%E5%85%A5%E6%95%B0%E6%8D%AE&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~sobaiduweb~default-4-80249509.first\_rank\_v2\_pc\_rank\_v29&spm=1018.2226.3001.4187

# **PostgreSQL修改数据库表的列属性(ALTER语句)**

https://blog.csdn.net/u011627980/article/details/51352559?ops\_request\_misc=&request\_id=&biz\_id=102&utm\_term=pgadmin%E4%B8%AD%E5%A6%82%E4%BD%95%E7%BB%99%E8%A1%A8%E6%A0%BC%E4%B8%AD%E7%9A%84%E5%AD%97%E6%AE%B5%E6%B7%BB%E5%8A%A0%E7%BA%A6%E6%9D%9F&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~sobaiduweb~default-5-51352559.first\_rank\_v2\_pc\_rank\_v29&spm=1018.2226.3001.4187

# **pgAdmin添加外键约束**

在pgAdmin 4中,以下是步骤：

>右键单击该表,然后选择“属性”.

>在出现的对话框中,单击“约束/外键”.

>单击外键表右上角的图标.

>单击铅笔图标,该图标位于外键表中现在出现的新行左侧.

>在“常规”选项卡下,为外键指定名称(我遵循{tablename} \_ {columnname} \_fkey的this convention.

>在“列”选项卡下,选择您的引用.

>单击“本地列”右侧的图标.

>单击底部的“保存”按钮.

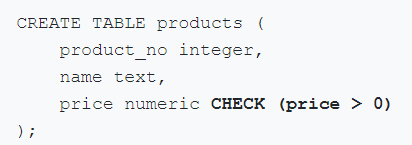
————————————————

# **数据库的创建表时，字段的数据类型和约束条件**

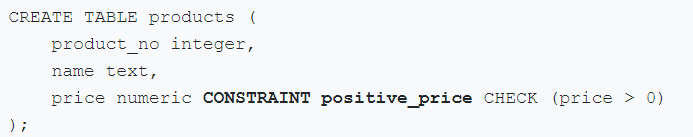
https://blog.csdn.net/Light\_\_1024/article/details/88342893?ops\_request\_misc=%257B%2522request%255Fid%2522%253A%2522164570235416780255244487%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request\_id=164570235416780255244487&biz\_id=0&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~first\_rank\_ecpm\_v1~rank\_v31\_ecpm-4-88342893.pc\_search\_result\_cache&utm\_term=pgadmin%E4%B8%AD%E5%A6%82%E4%BD%95%E7%BB%99%E4%BB%B7%E6%A0%BC%E5%AD%97%E6%AE%B5%E7%BA%A6%E6%9D%9F&spm=1018.2226.3001.4187

## 一：限制

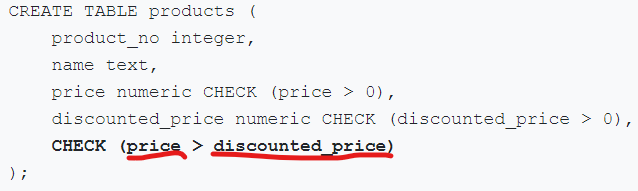
**1.1：检查约束（Ограничения-проверки）**



（1）约束是在数据类型之后，括号中必须包含设置约束的列



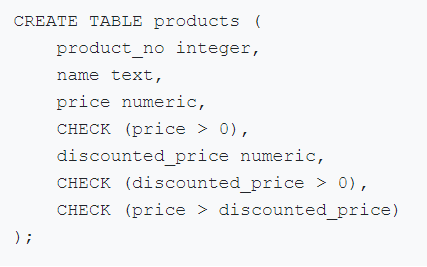
（2）CONSTRAINT +标识符和约束定义本身 （如果不定义 系统将为您选择一个名称）



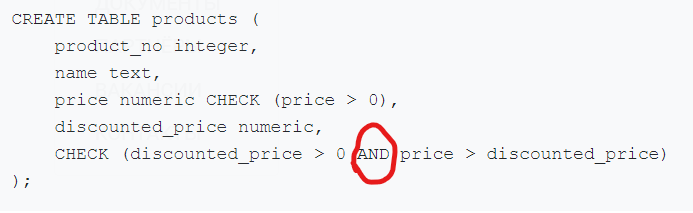
（3）一个检查约束也可以引用多个列，比如打折价格始终低于正常价格。

前面两个是**列约束，**而第三个是**表约束，**因为他的语句是和列定义分开写的。

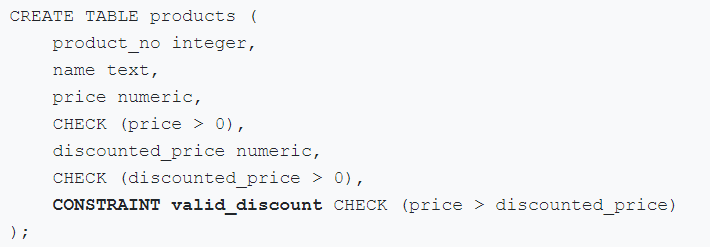
可以合并为如下形式：



也可以这样：



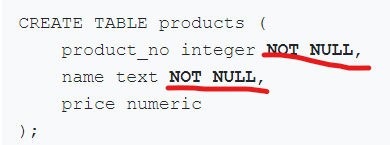
表约束的命名方式与列约束相同：



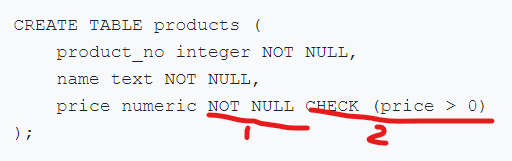
计算结果满足true和NULL，而许多具有NULL操作的表达式将计算为NULL，所以这个检查约束不会阻止NULL被写入关联的列，为了确保不包含控制，需要使用NOT NULL约束

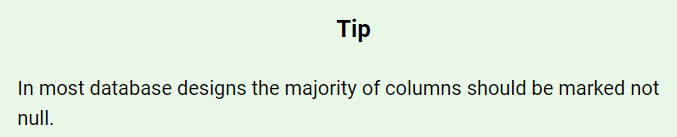
**1.2：Ограничения NOT NULL（NOT NULL约束）**

**指定不能为该列分配空值：**



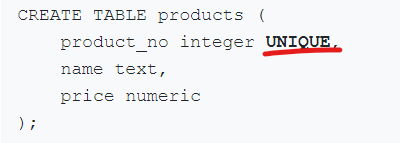
对于某一列，也可以同时限制多个约束，比如**检查约束**+**非空约束**



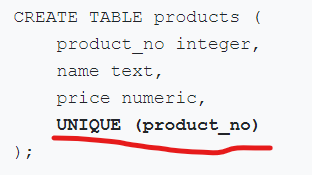


### **1.3： Ограничения уникальности（唯一性约束Unique Constraints）**

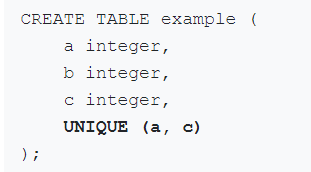
约束某一列column constraint ограничения столбц

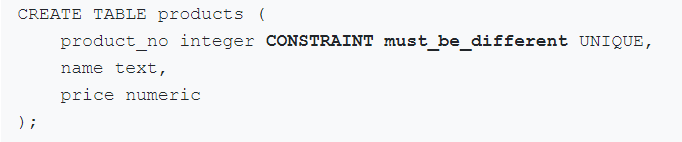


表约束table constraint ограничения таблицы



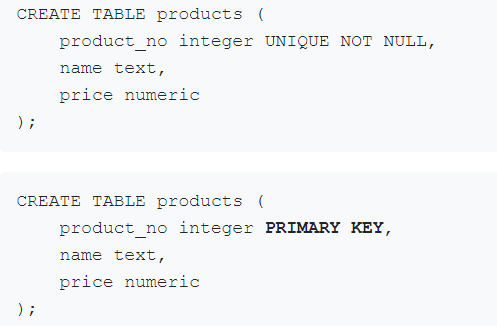
为**一组列定义唯一约束**，需要写成表约束的形式



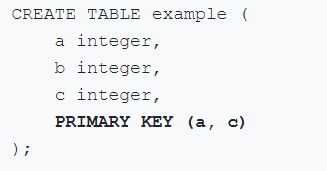


**1.4： Первичные ключи（主键Primary Keys）**

**PRIMARY=UNIQUE+NOT NULL**



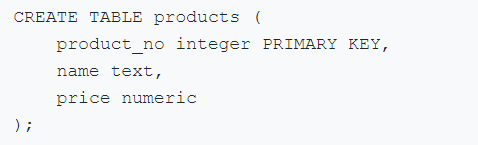
**主键也可以包含多个列：例子如下 和唯一性约束类似 每张表必须有一个主键**



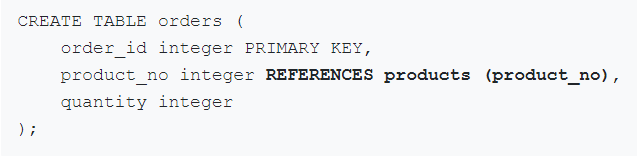
**1.5： Внешние ключи（外键 Foreign Keys）**

外键约束指定一个列（或一组列）的值必须与另一个表的某行中的值匹配，两个相关表的参照完整性。

如有一个产品表如下：**主表**



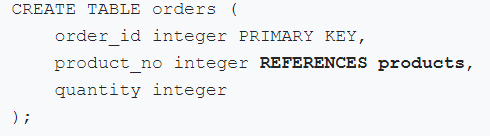
订单中只包含现有的产品，因此，可以给products表一个**外键约束** 有了这个约束，就不可能创建一个值products\_no不在products表中（并且不为NULL）的订单 如下：子表



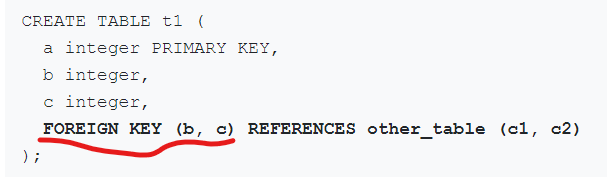
Order表：**子表**，**从属**

products表：**主表 主要**

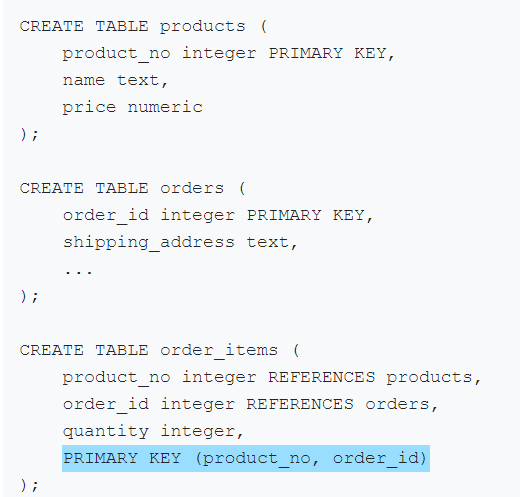
**也可省略主表中的列名，外键将隐式链接到主表的主键，简写如下：**



**外键也可以引用一组列，但必须写成表约束的形式，例如：外键中有几列就一一对应主表中的几列，**



**一个表也可以包含多个外键约束，这对于多对多关系链接表很有用，比如订单表能够包含多个产品（前面的外键约束没法实现），因此，可使用如下方案：**



**在最后一个表中，主键覆盖了外键**

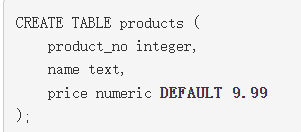
**Ограничения на связь таблиц表关系限制**

**ON UPDATE CASCADE ON DELETE SET NULL**

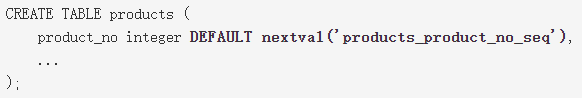


## 二：默认值

**创建表格时，一些列没有值，可以为其填充初始值。在表定义中，默认值列跟在列数据类型之后，形式如下：**



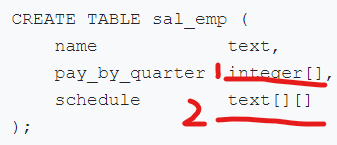
默认值可以是一个表达式，只要插入默认值（而不是创建表时），就会对其进行评估。 一个常见的示例是时间戳列的默认值为 CURRENT\_TIMESTAMP，以便将其设置为插入行的时间。 另一个常见的例子是为每一行生成一个“序列号”。 在 PostgreSQL 中，这通常通过以下方式完成：



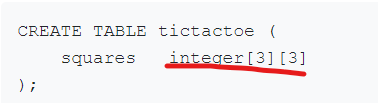
## 三：数组

**3.1：数组类型声明**

**格式如下：**



**一维数组和二维数组**



**可在创建表格时就可以定义数组的大小，在声明一维数组时，也可以使用ARRAY定义数组大小 如下所示：**



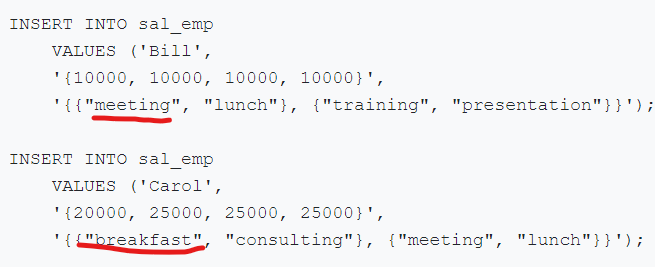
**或者也可以不指定数组的大小，这种情况下，不会对数组的实际大小施加限制：**



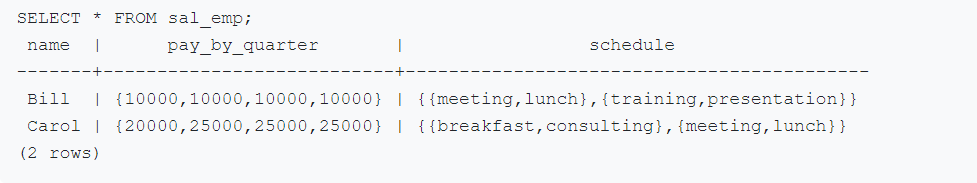
**3.2：输入数组值**

**数组输入值的形式如下，比如一个3\*3的数组输入值：**



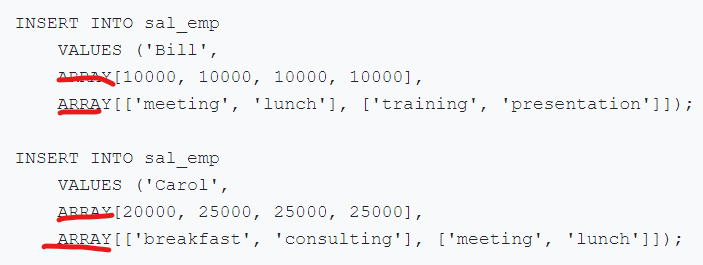


**输出结果：添加包含NULL的字符串时，需要用双引号引起来**



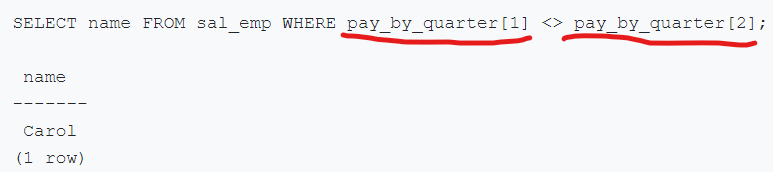
**在多维数组中，每一维的元素个数必须相同，否则会发生错误**

**也可以使用ARRAY构造函数语法：需要注意的是，字符串元素是用单引号而不是双引号**

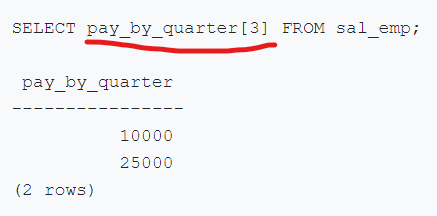


**3.3：访问数组**

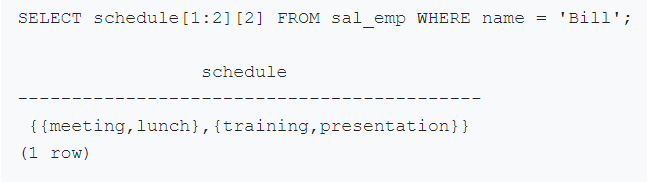
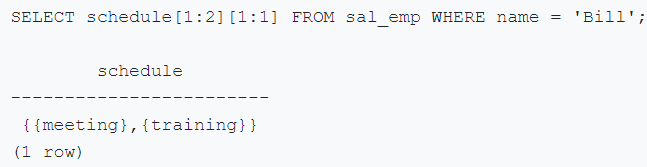
**访问数组中确定的元素，初始编号《》末尾编号 第一列第二行**



**查询某一字段中的所有元素，语句如下：查询第三季度的员工工资**

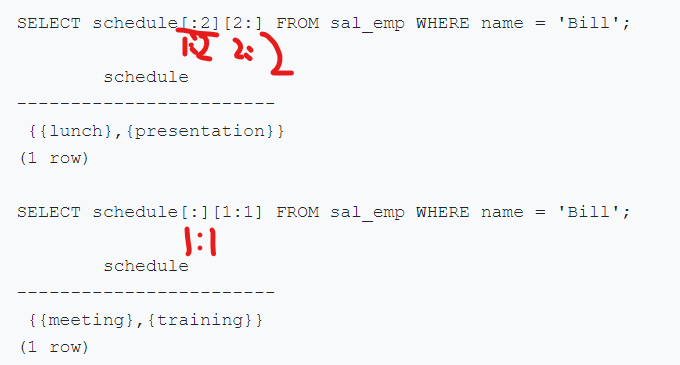


**也可以获取一个数组中的子数组：**

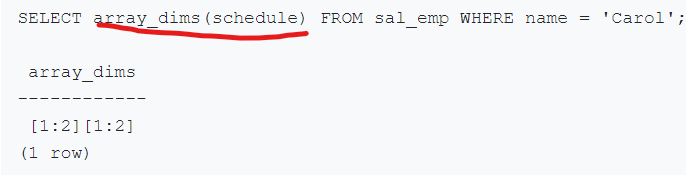


1. **隐式切片 等效于[1:2] 显式切片 如果某一个维度被切片则包含一个冒号： 如果只有一个数字，那么将包含从1到指定数字的元素 所有维度。**

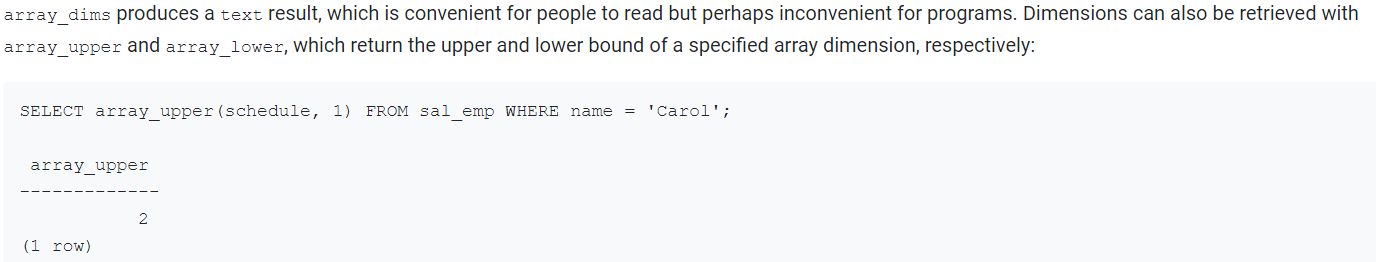
**如果切片形式中省略不写，那么下限就是到最小的数字，上限就是到最大的数字，例子如下：**



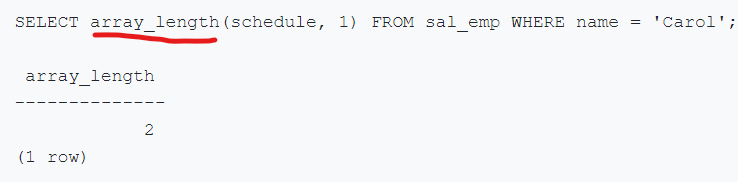
**获取数组维度的语句如下：**



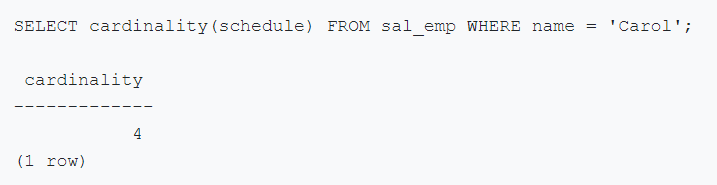
**也可以获得数组指定维度的下限和上限：语句如下：**



**返回指定数组维度中的元素数：**

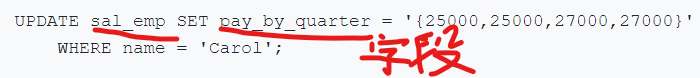


**Cardinality返回所有维度的数组元素总数。**

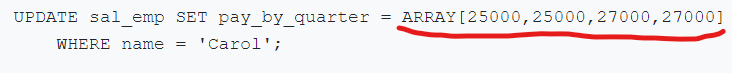


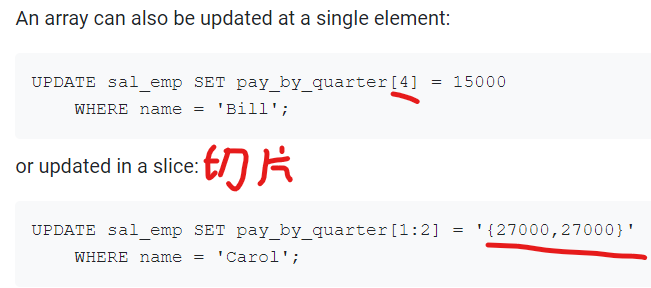
**3.4：更改数组**

**完全替换数组值：**

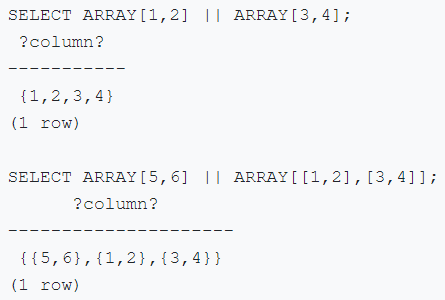


**使用ARRAY：**



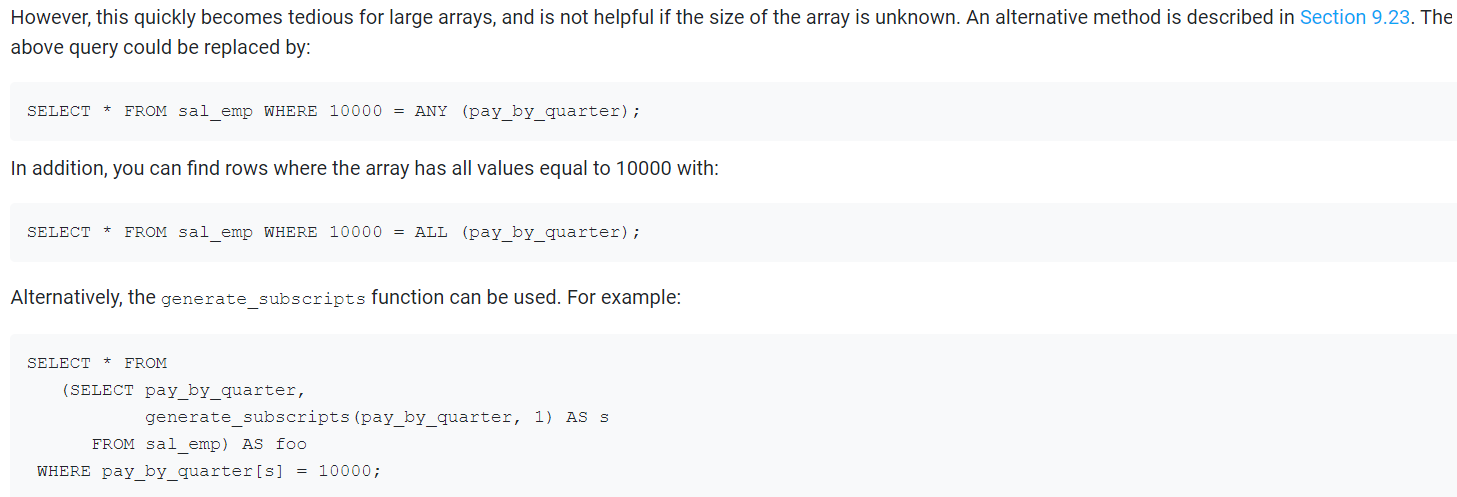


数组值也可以使用连接运算符||构造：数组的连接，不同维度的数组连接要用中括号括起来



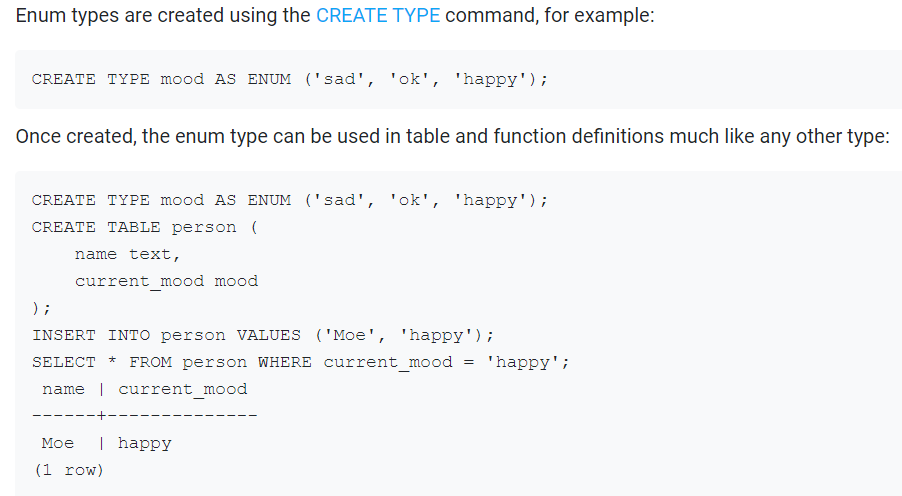
https://www.cnblogs.com/panpanwelcome/p/14899488.html

**3.5：在数组中查找值**

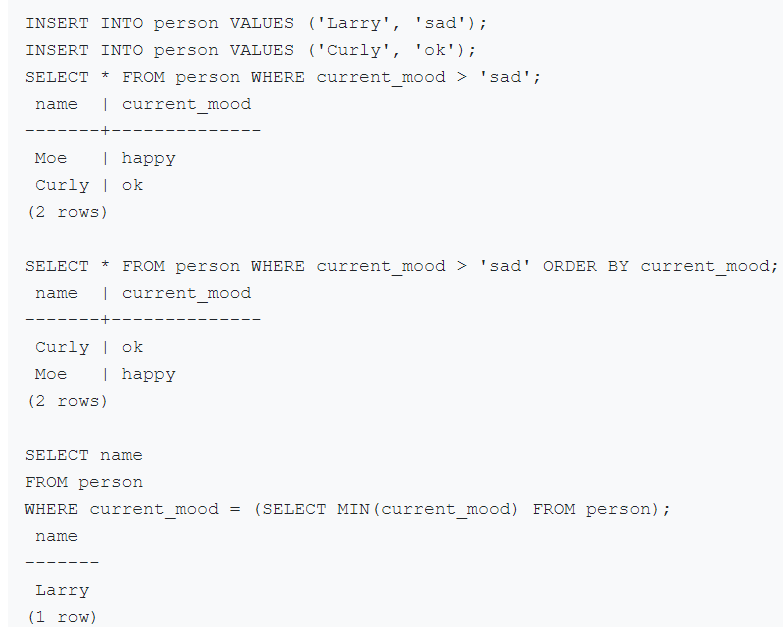


**3.6：输入和输出数组值的语法**

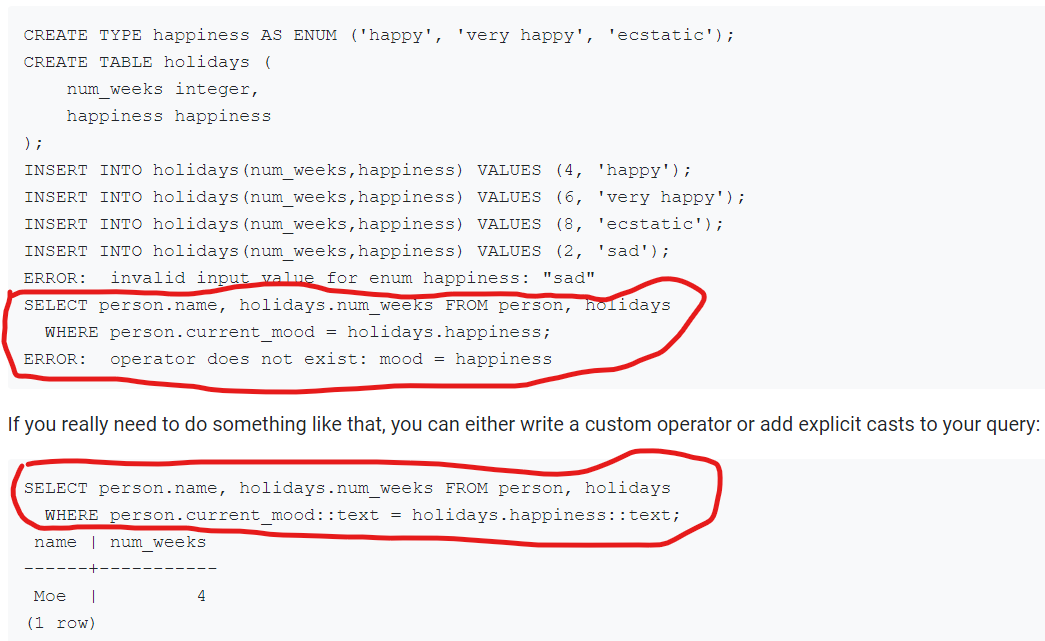
**四 枚举类型Типы перечислений**



比较运算在枚举中也适用，



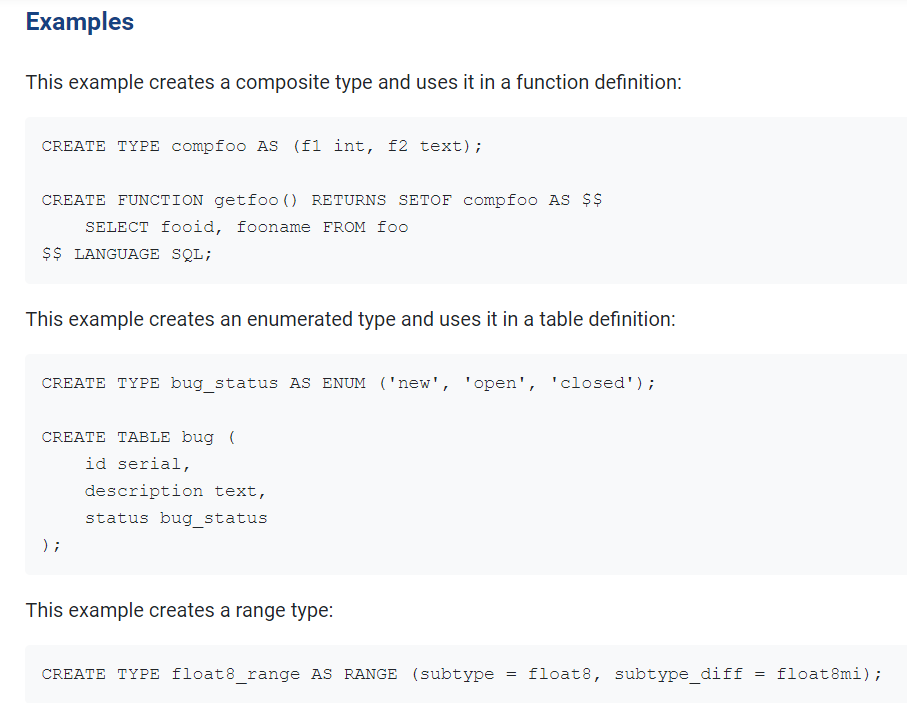
注意点：

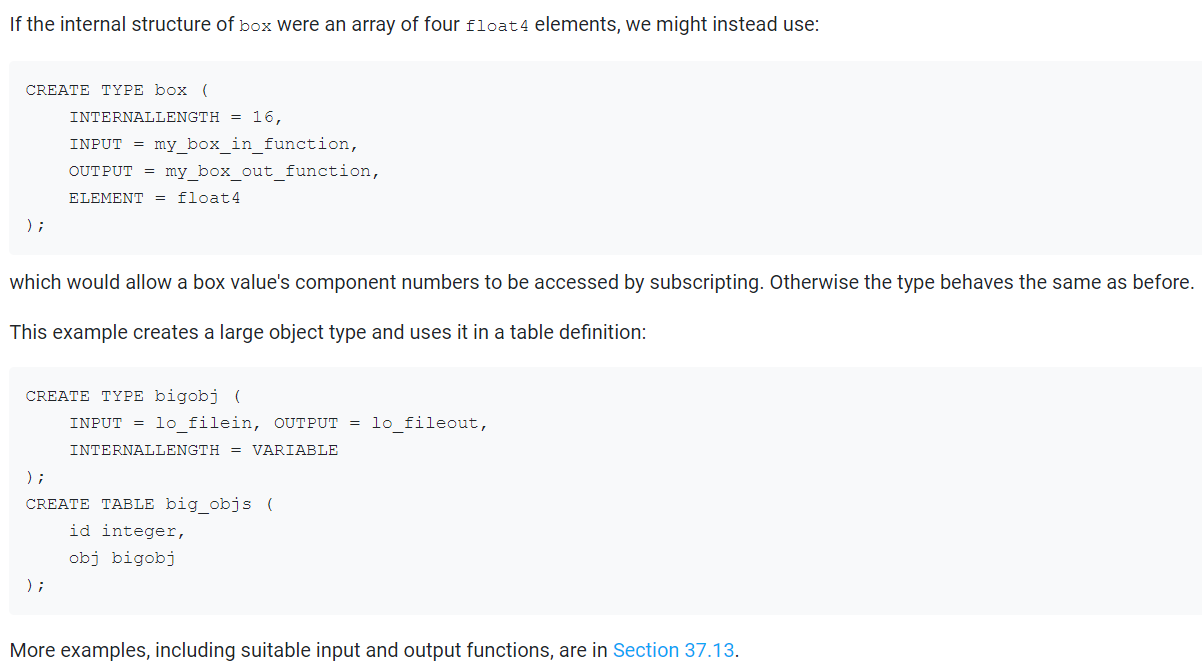
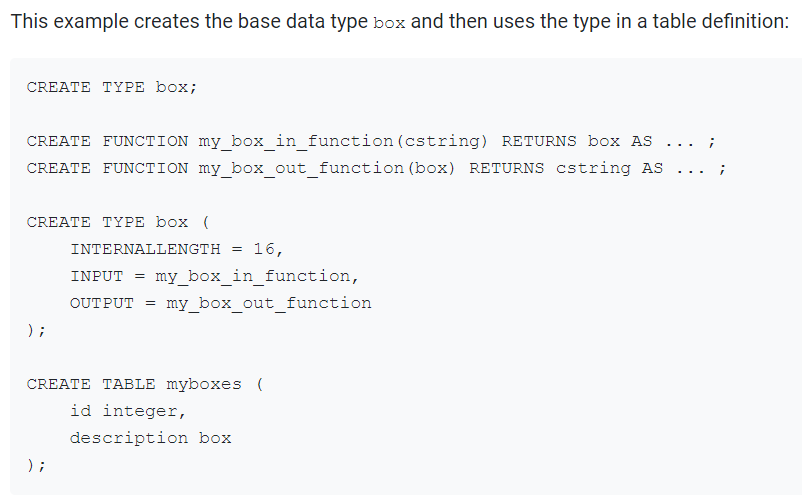


**五：创建类型：CREATE TYPE**

基本类型 枚举类型 数组类型 范围类型 复合类型

例子：





**六：几何类型Geometric Types**

**[8.8.1. Points](https://postgrespro.ru/docs/postgresql/12/datatype-geometric?lang=en" \l "id-1.5.7.16.5)**

**[8.8.2. Lines](https://postgrespro.ru/docs/postgresql/12/datatype-geometric?lang=en" \l "DATATYPE-LINE)**

**[8.8.3. Line Segments](https://postgrespro.ru/docs/postgresql/12/datatype-geometric?lang=en" \l "DATATYPE-LSEG)**

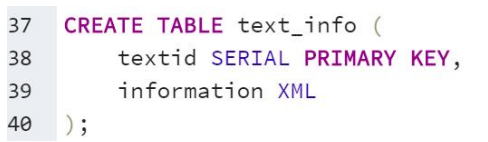
**[8.8.4. Boxes](https://postgrespro.ru/docs/postgresql/12/datatype-geometric?lang=en" \l "id-1.5.7.16.8)**

**[8.8.5. Paths](https://postgrespro.ru/docs/postgresql/12/datatype-geometric?lang=en" \l "id-1.5.7.16.9)**

**[8.8.6. Polygons](https://postgrespro.ru/docs/postgresql/12/datatype-geometric?lang=en" \l "DATATYPE-POLYGON)**

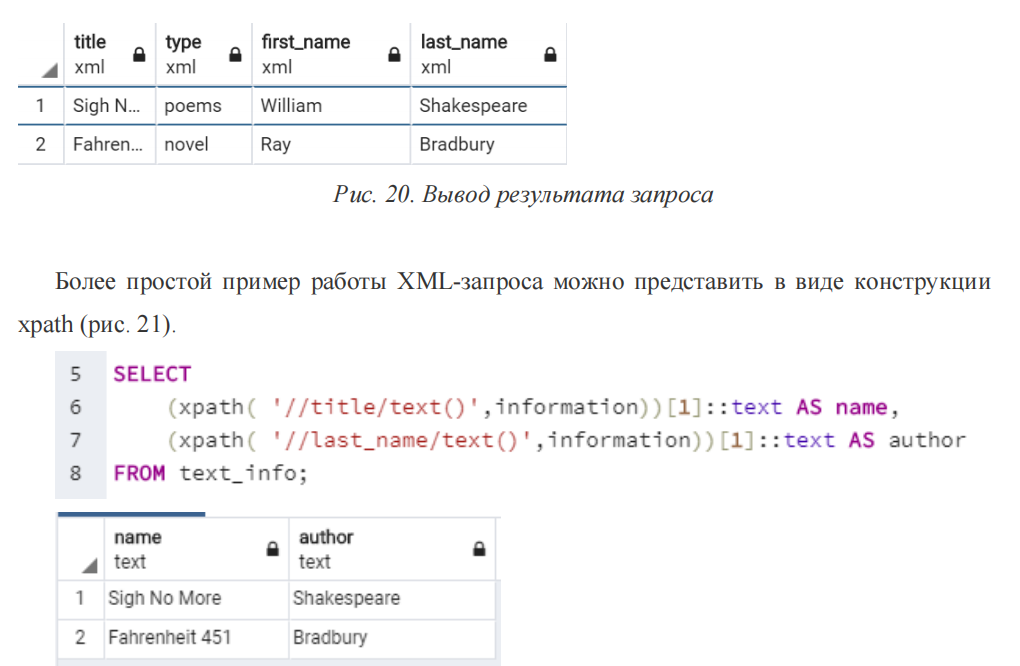
**[8.8.7. Circles](https://postgrespro.ru/docs/postgresql/12/datatype-geometric?lang=en" \l "DATATYPE-CIRCLE)**

**七：XML**

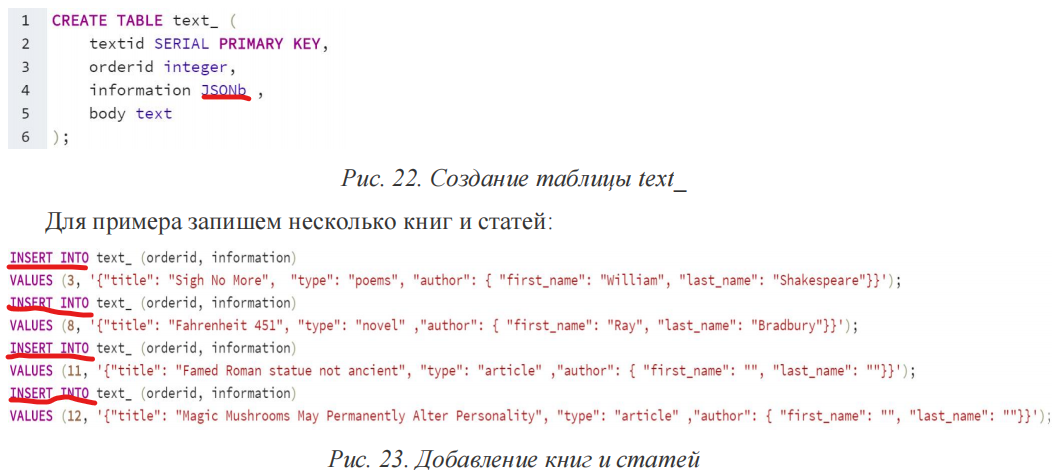


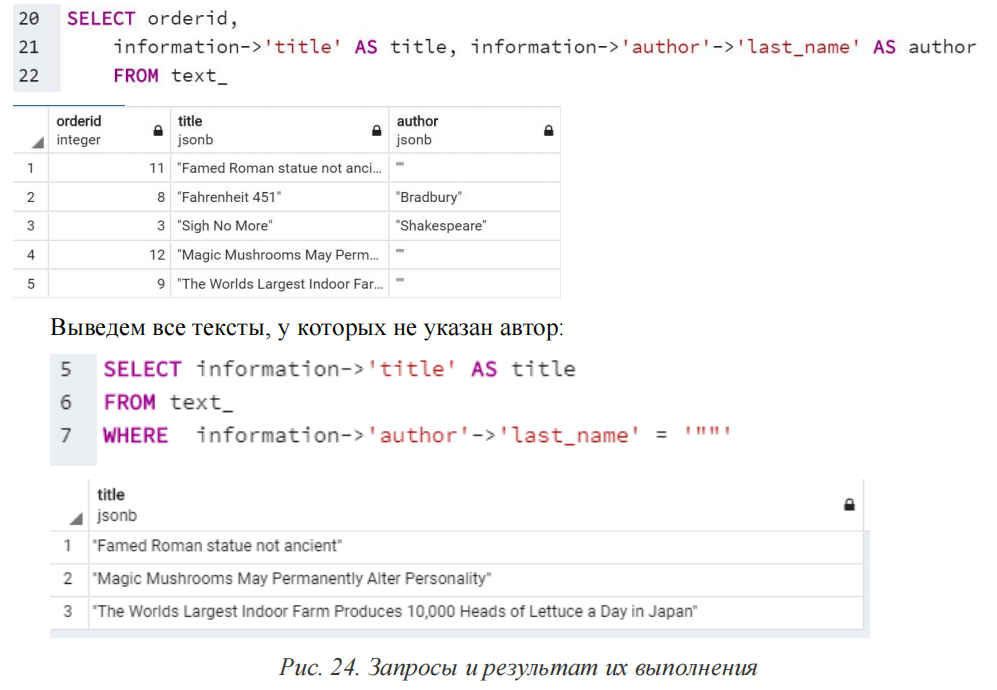
输入信息：





八：JOSN





**九：分区表*Секционирование таблиц***

PostgreSQL 支持简单的表分区。切片

表被称为将一个大的逻辑表拆分为多个

小的物理部分。分区可以带来以下好处：

a) 在某些情况下会显着提高性能；

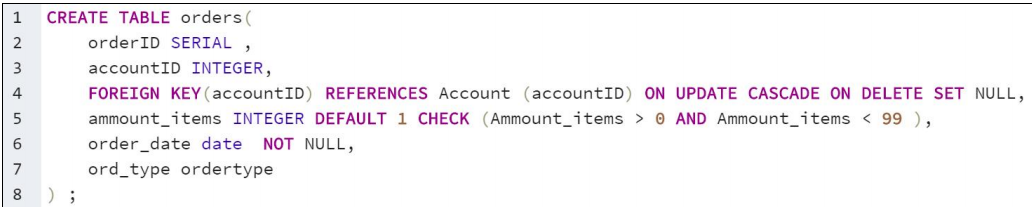
b) 删除单个部分要快得多。

有两种类型的分区：声明式分区和

使用继承进行分区。让我们考虑第二个选项。

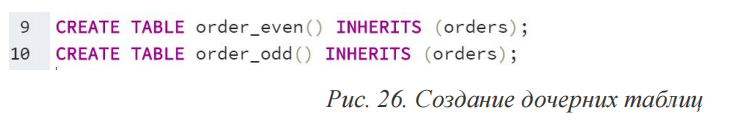
该方法分五个步骤实现。

1. 创建一个主表，子表将从中继承：



1. 创建几个子表。

例如，让我们创建两个表偶/奇序号原则。 INHERITS 语句用于从父表订单继承：

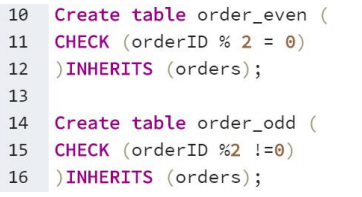


3. 使用构造向表添加非重叠约束：

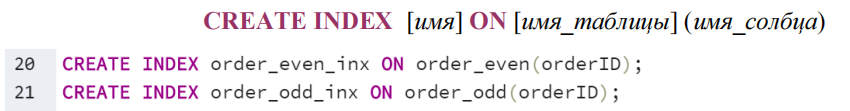
检查（表达式）

通过这种方式，我们将确保每个偶数或奇数

记录 ID 将转到正确的表：

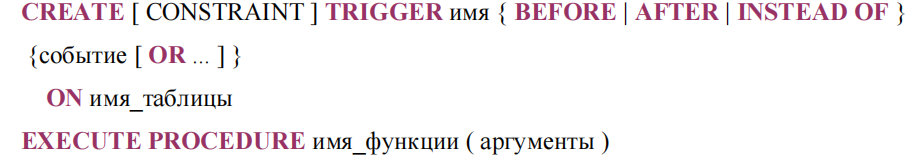


4.需要在key列上为每个子表建立索引，以及您自行决定的任何其他索引。 指数是手段之一提高数据库性能。 使用索引，数据库服务器可以更快地找到并提取所需的行。 索引创建构造看起来像这样：

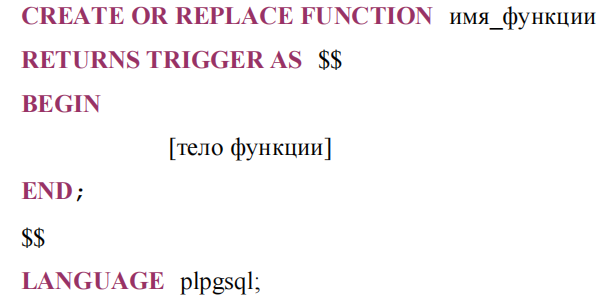


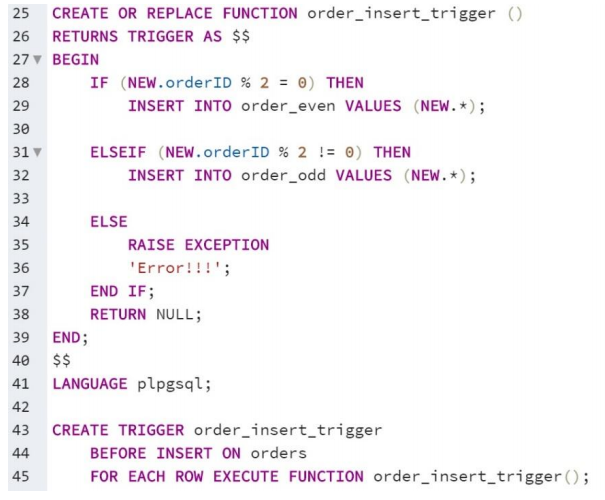
5.这样在执行查询INSERT INTO orders(value)之后数据出现在所需的子表中，需要执行相应的

触发组。让我们使用触发器函数来创建它，它将是当数据库中的数据发生变化时调用。触发器将与指定的表、视图或第三方相关联表并将在某些事件中执行指定的功能。触发器可以配置为在行操作之前触发（之前约束检查并尝试插入、更新或删除），或完成后（检查约束并执行 INSERT、UPDATE或 DELETE），或代替操作（在添加、更改和删除行介绍）。二十触发器功能可防止实际上不影响一行中的数据，而通常无论是否数据是否已更改。一般来说，触发器是这样声明的：

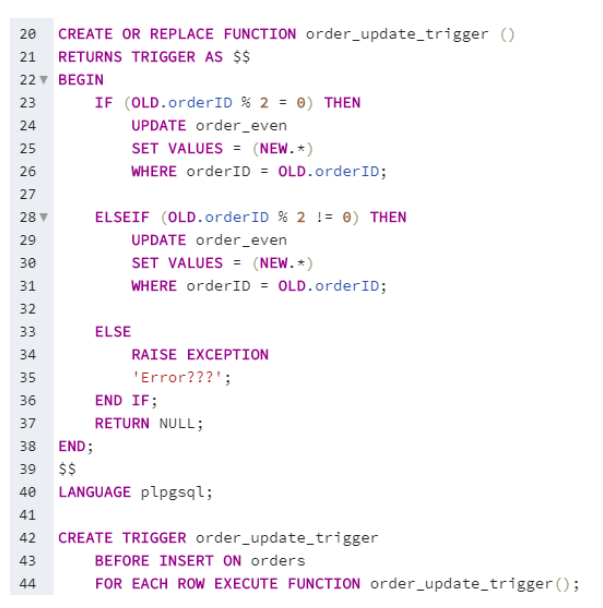


触发器必须写在触发器函数中。 一般语法触发函数可以表示为：





同样，我们将编写一个更新数据的触发函数



让我们评论一下代码的工作：

在第 25 行，我们将声明一个触发函数order\_insert\_trigger() ，

在第 28 和 31 行，我们创建了两个不重叠的条件

NEW.orderID % 2 = 0 和 NEW.orderID % 2 != 0

NEW 变量包含用于 INSERT/UPDATE 命令的新数据库行。

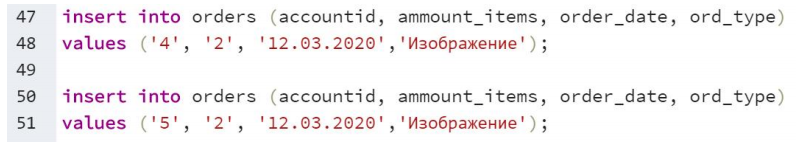
OLD 变量包含旧的命令数据库字符串

更新/删除。

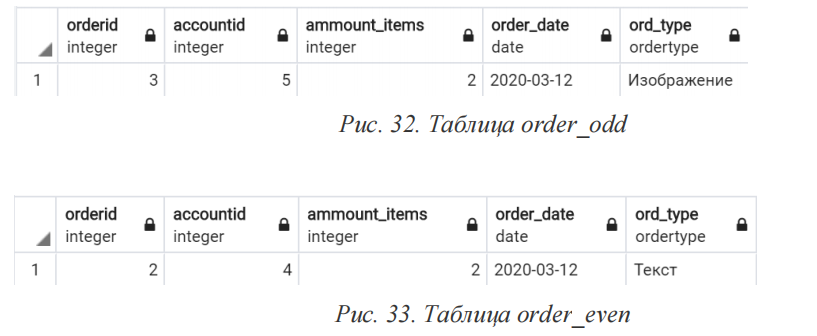
所以我们检查新orderID的模2，如果它等于零，将整行 (VALUES ( NEW .\* )) 放入“偶数”表中。同样对于奇数标识符。

第 43 行创建 order\_insert\_trigger 触发器。其运行条件正在向表中添加一条记录（在 INSERT ON 订单之前）。FOR EACH ROW EXECUTE FUNCTION order\_insert\_trigger() 的条目表示

触发器函数将为每个添加的行触发。之后，我们在表格中输入数据。



现在，要查看结果，我们显示子表并确保在每个子表都得到了对应的结果



**存储函数**

存储过程是一组指令的函数作为一个整体执行，存储在服务器上。编写存储过程允许您限制对数据和表的访问，从而保护它们免受

无意的数据修改。 https://postgrespro.ru/docs/postgrespro/12/sql 创建过程。



**动态查询**

通常需要在 PL / pgSQL 中的函数内部动态形成命令，然后有些命令每次执行时都可以使用不同的命令。表或数据类型。对于动态命令的执行，

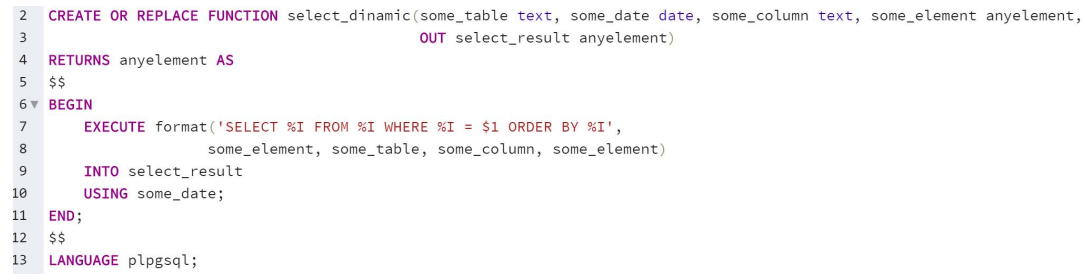
执行语句：



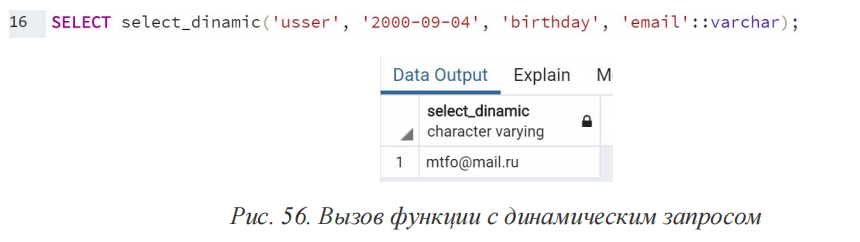
其中 command-string 是一个表达式，它与命令文本本身形成一个字符串，

这是需要履行的。目标是将放置结果的变量命令。 USING 中的表达式形成了将被插入到命令中的值。在命令文本中，可以使用参数值、参数引用表示为$1、$2等。这些符号表示在使用子句。参数符号只能用于代替值数据。如果要动态生成表或列的名称，它们必须作为文本输入。除了表名或列名，您可以使用带有 format() 函数的 %I 格式提示（字符分隔的文本换行符，连接在一起）。

例如，让我们创建一个带有动态查询的“select\_dinamic”函数将输入表的名称（some\_table），一些日期，名称此表中具有日期类型的列 (some\_date) 和一些任意类型的字段这个表（some\_element），以及字段 some\_element 的值将被返回，对应于 some\_date 字段中等于输入日期的日期。



例如，可以调用这样的函数来查找具有所需内容的用户电子邮件出生日期



周期和条件

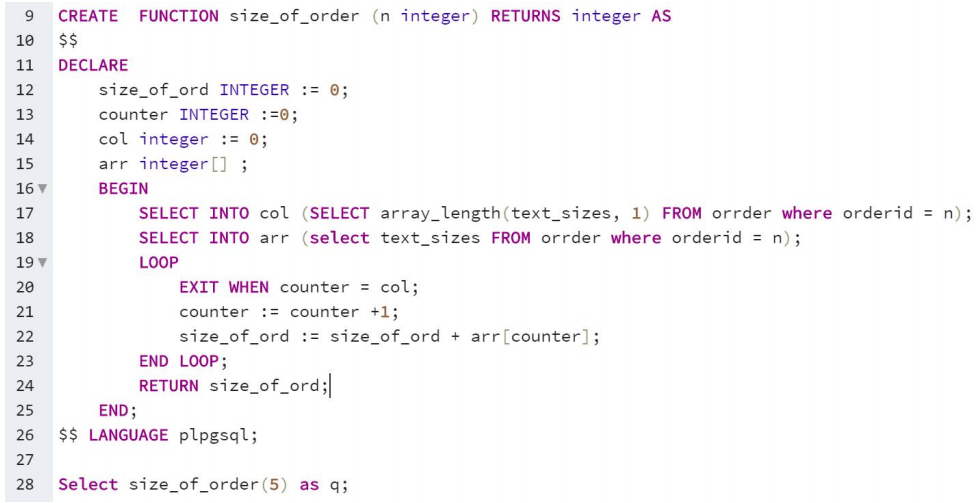
PostgreSQL 使用多种条件语句。

https://postgrespro.ru/docs/postgrespro/12/plpgsql-control-structures：

例如，考虑一个包含循环的存储过程

数组元素。 这样的函数将计算文本文件的总大小

根据输入的订单 ID 进行订单（参见图 57）。



让我们解释一下这个函数是如何工作的：

在第 11-15 行，我们将声明变量在存储过程中使用。

在第 17、18 行中，我们分配给变量 col、arr

查询结果。 在变量 col 我们写入元素的数量数组文本大小。 让我们将 text\_sizes 数组复制到 arr 变量。 让我们在一行中声明一个循环

1. 退出循环的条件将是完整枚举数组的所有元素。我们在循环的每次迭代中增加计数器计数器，添加大小值数组的当前元素到总数。 我们返回数组的总大小。

**使用文件系统**

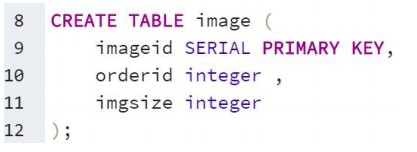
考虑使用存储过程来读取存储在

本地目录。 正在读取的文件将被输入到表中。 这样的方法

适用于信息以文本文件形式呈现的情况，例如，

.csv。 https://postgrespro.ru/docs/postgrespro/12/sql-copy 。

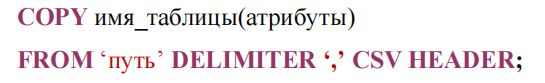
为此，首先创建图像表（图 58）。



准备好 CSV 格式的数据后，我们将编写一个接受的存储过程

用于执行读/写操作的参数。 从光盘复制信息

您需要执行该功能：



要从数据库中卸载信息，请使用以下函数：



现在，根据参数，信息将被加载或从数据库中卸载（见图 59）

