Авторы: Егор Рогов, Павел Лузанов

Обратная связь
Отзывы, замечания и предложения направляйте по адресу:
edu@postgrespro.ru

Темы

Массивы и работа с ними

Функции с переменным числом параметров

Полиморфные функции с массивами

Поддержка массивов в PL/pgSQL

Тип массива

PROFESSIONAL Postgres

Массив

набор пронумерованных элементов одного типа

Объявление

тип_элемента[]

тип_элемента[][]

в скобках можно указать размер, но он игнорируется

Литералы и конструкторы

'{*значение* [, *значение*...]}'

ARRAY[*значение* [, *значение*...]]

ARRAY(*подзапрос_с_одним_столбцом*) или array_agg(*выражение*)

Массив, как и составной тип (запись), не является скалярным и состоит из нескольких элементов другого типа. Но, в отличие от записей, а) все эти элементы имеют одинаковый тип и б) обращение к ним происходит не по имени, а по целочисленному индексу (здесь *индекс* понимается в математическом смысле, а не как индекс БД).

Тип массива не надо специально объявлять. Достаточно просто добавить квадратные скобки к имени типа элементов.

Массивы могут быть как одномерными, так и многомерными. Одномерные массивы автоматически расширяются при необходимости; многомерные — нет. Но в любом случае указанный в скобках размер массива игнорируется.

https://postgrespro.ru/docs/postgresql/9.6/arrays.html

Значение массива создается либо строковым литералом (набор значений в фигурных скобках), либо с помощью конструктора ARRAY.

https://postgrespro.ru/docs/postgresql/9.6/sql-expressions.html

Также массив можно сконструировать из SQL-запроса, возвращающего один столбец, с помощью конструкции ARRAY или агрегатной функции array_agg.

https://postgrespro.ru/docs/postgresql/9.6/functions-aggregate

Использование

Массив — полноценный тип SQL

Элемент массива

массив[*индекс_элемента*]

по умолчанию нумерация с единицы

элемент за пределами — неопределенное значение (не ошибка)

Срез массива

массив[*нижний_индекс*:*верхний_индекс*]

Операции

сравнение, вхождение, пересечение, конкатенация

использование с ANY и ALL вместо подзапроса

...

Массив является обычным типом SQL, так что его можно использовать так же, как и любой другой тип: создавать столбцы таблиц этого типа, использовать его для параметров функций и т. п.

Классический реляционный подход позволяет обойтись без массивов в базе данных: элементы массива можно поместить в отдельную таблицу и связать ее с основной внешним ключом. Тем не менее, использование массивов бывает удобно; поиск элемента в массиве может быть ускорен специальными индексами. В частности, массивы довольно активно используются в системном каталоге PostgreSQL.

Элементы массива могут использоваться как обычные скалярные значения. Для доступа к элементу указывается его индекс в квадратных скобках.

Можно использовать и срезы (slice) массивов: для этого в скобках указывается не одиночный индекс, а диапазон индексов.

Массивы можно сравнивать между собой, проверять на неопределенность, искать вхождение элемента и находить пересечение с другими массивами, конкатенировать и пр.

Также массивы можно использовать по аналогии с подзапросами в конструкциях ANY/SOME и ALL (вообще массив можно «превратить» в таблицу с помощью функции unnest).

https://postgrespro.ru/docs/postgresql/9.6/functions-comparisons.html

Функции

С переменным числом параметров

- все необязательные параметры должны иметь одинаковый тип
- передаются в функцию в виде массива
- последний параметр-массив объявляется как VARIADIC

Полиморфные

- работают со значениями разных типов;
тип конкретизируется во время выполнения
- дополнительные полиморфные псевдотипы `anyarray` и `anynonarray`
- полиморфные функции могут иметь переменное число параметров

Массивы позволяют создавать функции с переменным числом параметров.

В отличие от параметров со значениями по умолчанию, которые при объявлении функции надо явно перечислить, необязательных параметров может быть сколько угодно и все они передаются функции в виде массива. Но, как следствие, все они должны иметь один и тот же тип.

При объявлении функции последним указывается один параметр, помеченный как VARIADIC, имеющий тип массива.

https://postgrespro.ru/docs/postgresql/9.6/xfunc-sql.html

Раньше (в теме «SQL. Функции») мы говорили про полиморфные функции — такие функции могут работать с параметрами разных типов. При объявлении функции указывается специальный полиморфный псевдотип, а конкретный тип уточняется во время выполнения по фактическому типу переданных параметров.

Для массивов есть отдельный полиморфный тип anyarray (и anynonarray для не-массивов).

Этот тип можно использовать совместно с передачей переменного числа аргументов при объявлении VARIADIC-параметра.

https://postgrespro.ru/docs/postgresql/9.6/extend-type-system.html

Циклы PL/pgSQL

Цикл по элементам массива

```
FOREACH цель IN ARRAY массив LOOP
    операторы
END LOOP;
```

цель — переменная того же типа, что и элементы массива
(может быть списком переменных, если это составной тип)

можно перебрать обычным циклом, используя array_lower, array_upper

Цикл по срезам массива

```
FOREACH цель SLICE размерность_среза IN ARRAY массив LOOP
    операторы
END LOOP;
```

цель — массив

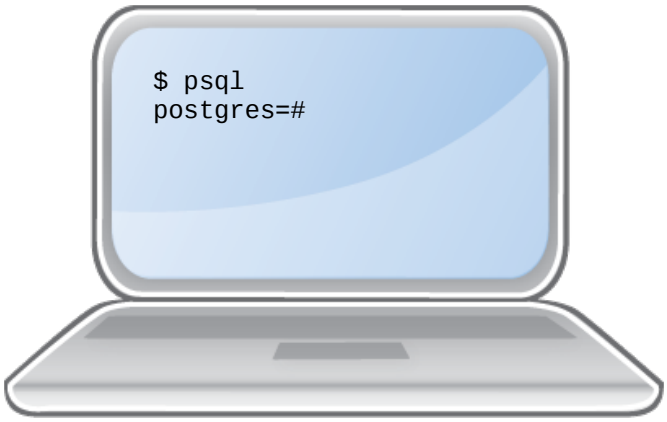
Для итерации по элементам массива вполне можно использовать обычный целочисленный цикл FOR, используя функции, возвращающие минимальный и максимальный индексы массива.

Однако есть и специализированный вариант цикла: FOREACH. В таком варианте переменная цикла пробегает не индексы элементов, а сами элементы. Поэтому переменная должна иметь тот же тип, что и элементы массива (как обычно, если элементами являются записи, то одну переменную составного типа можно заменить несколькими скалярными переменными).

Тот же цикл с фразой SLICE позволяет итерировать срезы массива. Например, для двумерного массива одномерными срезами будут его строки.

https://postgrespro.ru/docs/postgresql/9.6/plpgsql-control-structures.html

Демонстрация



Итоги

Массив состоит из пронумерованных элементов одного и того же типа данных

Столбец с массивом как альтернатива отдельной таблице: удобные операции и индексная поддержка

Позволяет создавать функции с переменным числом параметров

Практика

1. Создайте функцию add_book для добавления новой книги. Функция принимает два параметра — название книги и массив идентификаторов авторов — и возвращает идентификатор новой книги.
2. Проверьте, что в приложении появилась возможность добавлять книги.

1.

```
FUNCTION add_book(title text, authors integer[])
RETURNS integer
```

Практика

1. Реализуйте функцию map, принимающую два параметра: массив вещественных чисел и название вспомогательной функции, принимающей один параметр вещественного типа. Функция должна возвращать исходный массив, в котором к каждому элементу применена вспомогательная функция.
2. Реализуйте функцию reduce, принимающую два параметра: массив вещественных чисел и название вспомогательной функции,
принимающей два параметра вещественного типа.
Функция должна возвращать вещественное число, полученное последовательной сверткой массива слева направо.
3. Сделайте функции map и reduce полиморфными.

1. Например:

```
map(ARRAY[4.0,9.0],'sqrt') → ARRAY[2.0,3.0]
```

2. Например:

```
reduce(ARRAY[1.0,3.0,2.0,0.5],'greatest') → 3.0
```

В этом случае значение вычисляется как

```
greatest( greatest( greatest(1.0,3.0), 2.0 ), 0.5 )
```