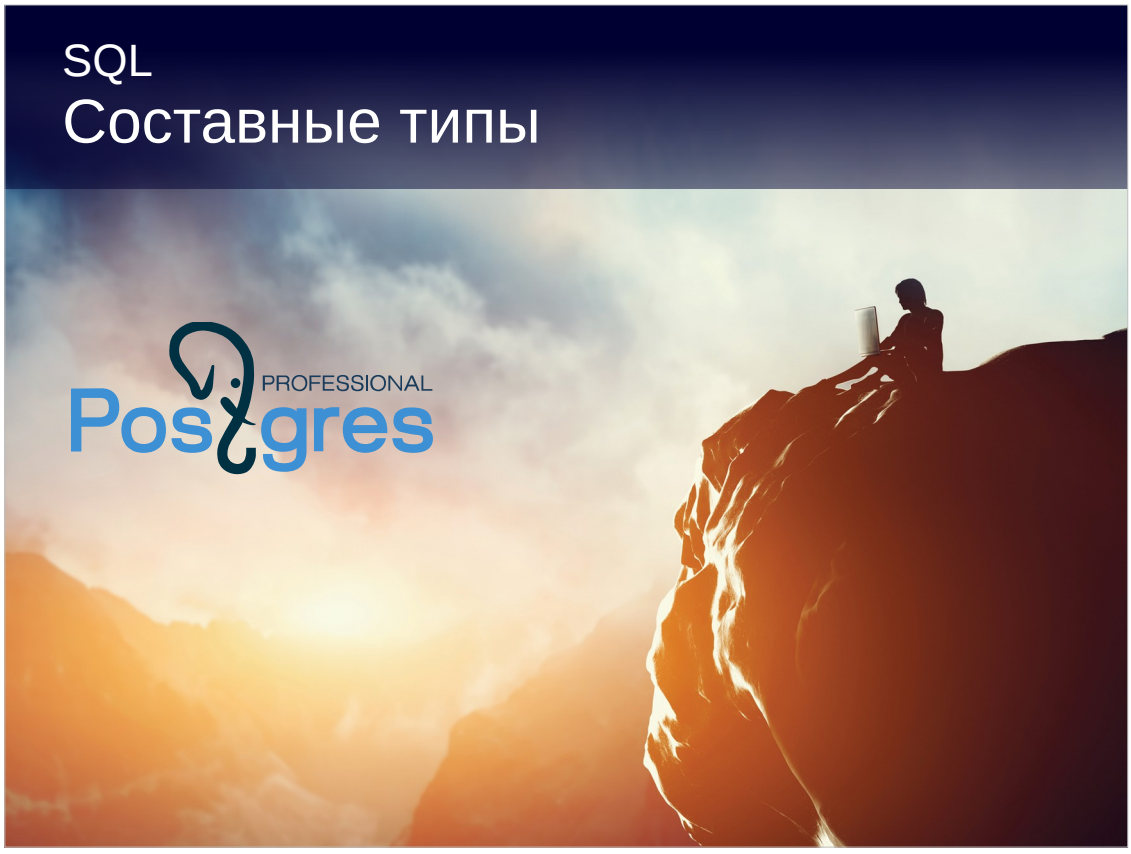


Авторы: Егор Рогов, Павел Лузанов

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

Темы

Составные типы и работа с ними

Параметры составных типов

Функции, возвращающие составной тип

Функции, возвращающие множества

Способы вызова функций

Составные типы

Составной тип

набор именованных атрибутов (полей)

то же, что «строка» или «запись» (в некоторых языках — «структура»)

Объявление

```
CREATE TYPE имя AS ( атрибут [, атрибут...] );
```

создание таблицы вводит одноименный составной тип

ограничения целостности не поддерживаются

Литералы и конструкторы

```
'(значение [, значение...])'
ROW(значение [, значение...])
```


Составной тип — это набор атрибутов, каждый из которых имеет свое имя и свой тип. Типы атрибутов могут отличаться (в отличие от массивов, которые мы будем рассматривать позже). Составной тип можно рассматривать как табличную строку. Часто он называется «записью» (а в Си-подобных языках такой тип называется «структурой»).

https://postgrespro.ru/docs/postgresql/9.6/rowtypes

Составной тип — объект базы данных, его объявление регистрирует новый тип в системном каталоге, после чего он становится полноценным типом SQL. При создании таблицы автоматически создается и одноименный составной тип, представляющий строку этой таблицы. Важное отличие состоит в том, что в составном типе нет ограничений целостности.

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

Значение составного типа создается либо строковым литералом (набор значений в круглых скобках), либо с помощью *табличного конструктора* ROW.

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

Использование

Составной тип — полноценный тип SQL

- поля таблиц
- использование в выражениях, параметрах функций и т. п.

Атрибут как скалярное значение

- *значение_составного_типа*.*атрибут*
- иногда нужны скобки, чтобы отличать от обращения к столбцу таблицы (*значение_составного_типа*).*атрибут*

Значения составного типа

- сравнение
- проверка на null
- использование с подзапросами

Составной тип можно использовать точно так же, как и любой другой тип SQL, например, создавать столбцы таблиц этого типа, использовать его для параметров функций и т. п.

В традиционных языках записи используют для логического объединения связанных данных. Однако в SQL при желании использовать составной тип для столбца, следует задуматься о вынесении соответствующих полей в отдельную таблицу. Это позволить избежать избыточности данных (нормализация), упростить индексирование (в случае составного типа потребуется индекс по выражению).

Атрибуты составного типа могут использоваться как обычные скалярные значения (хотя атрибут, в свою очередь, тоже может иметь составной тип). Для доступа к атрибуту значения составного типа, имя атрибута указывается после значения через точку. Но поскольку в SQL конструкция с точкой означает обращение к столбцу таблицы, значение составного типа нужно брать в скобки.

Значения составного типа «в целом» можно сравнивать между собой, проверять на неопределенность (null), использовать с подзапросами в таких конструкциях, как IN, ANY/SOME, ALL.

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

Функции

Обычные функции

возвращают одно значение

обычно вызываются в списке выборки запроса

при вызове во фразе FROM возвращают одну строку

синтаксически `таблица.столбец` = `столбец(таблица)`,
что делает функции и столбцы «взаимозаменяемыми»

Функции, возвращающие множество строк

```
RETURNS SETOF тип
RETURNS TABLE( атрибут [,атрибут...] )
```

могут возвращать несколько строк

обычно вызываются в предложении FROM

можно использовать как представление с параметрами,
особенно в сочетании с подстановкой тела функции в запрос

Функции могут принимать параметры составного типа, равно как и возвращать значение составного типа.

Обычно функции вызываются в списке выборки запроса (предложение SELECT). Если вызвать функцию в предложении FROM, ее возвращаемое значение будет «обернуто» в строку.

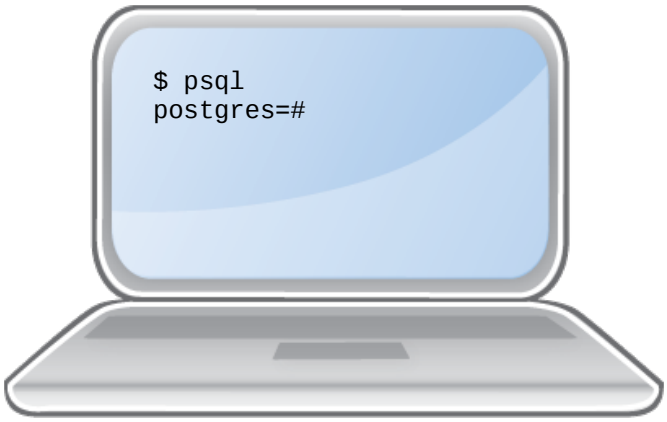
Интересно, что для доступа к столбцу таблицы можно использовать не только привычную форму `таблица.столбец`, но и функциональную: `столбец(таблица)`. Это позволяет создавать вычислимые поля, определяя функцию, принимающую на вход составной тип.

Но можно определить функцию, возвращающую множество строк. Такие функции естественно вызывать в предложении FROM и можно рассматривать как своеобразное представление. (Формально PostgreSQL позволяет вызвать такую функцию и в списке выборки, но так лучше не делать.)

https://postgrespro.ru/docs/postgresql/9.6/xfunc-sql.html
Как и с обычными функциями, в ряде случаев планировщик может подставить тело функции в основной запрос. Это позволяет создавать «представления с параметрами».

https://wiki.postgresql.org/wiki/Inlining_of_SQL_functions

Демонстрация



Итоги

Составной тип объединяет значения других типов

Упрощает и обогащает работу функций с таблицами

Позволяет создавать вычислимые поля
и представления с параметрами

Практика

1. Создайте функцию onhand_qty для подсчета имеющихся в наличии книг. Функция принимает параметр составного типа books и возвращает целое число.
2. Используйте эту функцию в представлении catalog_v в качестве «вычисляемого поля» onhand_qty.
3. Проверьте, что приложение начало отображать количество книг.
4. Создайте табличную функцию get_catalog для поиска книг. Функция принимает значения полей формы поиска «Магазина» («имя автора», «название книги», «есть на складе») и возвращает подходящие книги в формате catalog_v.
5. Проверьте, что в «Магазине» начал работать поиск и просмотр.

1.

```
FUNCTION onhand_qty(book books) RETURNS integer
```

4.

```
FUNCTION get_catalog(
    author_name text, book_title text, in_stock boolean
)
RETURNS TABLE(
    book_id integer, display_name text, onhand_qty integer
)
```

При решении хотелось бы воспользоваться уже готовым представлением catalog_v, просто наложив ограничения на строки. Но в этом представлении и название книги, и авторы находятся в одном поле, к тому же в сокращенном виде. Очевидно, что поиск автора «Лев» по полю «Л .Н. Толстой» не даст результата.

Можно было бы повторить в функции get_catalog запрос из catalog_v, но это дублирование кода, что плохо. Поэтому расширьте представление catalog_v, добавив в него дополнительные поля: заголовок книги и полный список авторов.

5.

Проверьте, что корректно обрабатываются пустые поля на форме. Когда клиент вызывает функцию get_catalog, передает ли он в этом случае пустые строки или неопределенные значения?

Практика

1. Напишите функцию, переводящую строку, содержащую число в шестнадцатеричной системе, в обычное целое число. Например: `convert('FF')` → `255`
2. Добавьте в функцию второй необязательный параметр — основание системы счисления (по умолчанию — 16). Например: `convert('0110',2)` → `6`
3. Табличная функция generate_series не работает со строковыми типами. Предложите свою функцию для генерации последовательностей строк из заглавных английских букв.

1. Например:

```
convert('FF') → 255
```

Для решения пригодятся: табличная функция regexp_split_to_table, функции uppper и reverse, конструкция WITH ORDINALITY.

Другое решение возможно с помощью рекурсивного запроса.

Проверить реализацию можно, используя шестнадцатеричные константы: `SELECT X'FF'::integer;`

2. Например:

```
convert('0110',2) → 6
```

3. Считайте, что на вход подаются строки равной длины. Например:

```
generate_series('AA','ZZ') →
 → 'AA'
   'AB'
   'AC'
   ...
   'ZY'
   'ZZ'
```