Специальные типы данных

Специальные типы данных

- Массивы
- JSON

Массивы

Массивы

- Многомерные массивы **переменной** длины
- Элементами массивов могут быть:
  - любые встроенные или определённые пользователями базовые типы
  - перечисления, составные типы, типы-диапазоны или домены
- Для объявления типа массива:
  - к названию типа элементов добавляются квадратные скобки (**[]**)
  - запись с ключевым словом **ARRAY**

```
integer[3]    integer ARRAY[3]    integer ARRAY
```

Определение массива

- 2 способа:
  - В виде символьной константы в формате:

```
'{значение1, значение2, ... }'
```

  - С помощью функции **ARRAY**

```
ARRAY[значение1, значение2, ... ]
```

- Значение:
  - либо константа
  - либо вложенный массив

```
SELECT  '{{1,2},{3,4}}'::integer[],
        ARRAY[[1,2], [3,4]] ::integer[];
```

int4	array
{{1,2},{3,4}}	{{1,2},{3,4}}

Список значений

Приведение типа

```
SELECT    '{1,2,3,4,5}'::integer[],
          ARRAY[1,2,3]::integer[];
```

int4	array
{1,2,3,4,5}	{1,2,3}

Добавление элементов в массив

- Добавление элементов в массив
  - Оператор **||** (конкатенация)
  - Функция **array_append**(массив, <эл-нт>)
  - Функция **array_prepend**(<эл-нт>, массив)

Добавление в конец массива

Добавление в начало массива

```
SELECT '{1,2,3,4}'::integer[] || 5,
       array_append('{1,2,3,4}'::integer[],5),
       array_prepend(5,'{1,2,3,4}'::integer[]);
```

?column?	array_append	array_prepend
{1,2,3,4,5}	{1,2,3,4,5}	{5,1,2,3,4}

https://postgrespro.ru/docs/postgresql/12/functions-array

Удаление и изменение элементов массива

- Удаление элементов из массива
  - Функция **array_remove**(массив, <значение эл-нта>)
- Изменение элемента массива
  - Функция **array_replace**(массив, <старое значение>, <новое значение>)

```
SELECT array_remove('{1,2,3,4}'::integer[],3),
       array_replace('{1,2,3,4}'::integer[],2,5);
```

array_remove	array_replace
{1,2,4}	{1,5,3,4}

https://postgrespro.ru/docs/postgresql/12/functions-array

Извлечение элементов массива

- Индексация элементов массива начинается с 1
- Для извлечения:
  - Одного элемента массива – необходимо указать его номер в []
  - Нескольких элементов из массива – необходимо в [] скобках указать номер первого элемента и номер последнего извлекаемого элемента разделенные символом двоеточие

```
select  ('{ "(921)-745-8965","(908)-567-78234",
           "(911)-745-8512", "(812)-750-8965"}'::text[])[2];
```

text
(908)-567-78234

```
select  ('{ "(921)-745-8965","(908)-567-78234",
           "(911)-745-8512", "(812)-750-8965"}'::text[])[3:4];
```

text
{(911)-745-8512,(812)-750-8965}

Проверка на вхождение в массив

- Для проверки вложенности элементов одного массива в состав элементов другого массива используются операторы вложенности массивов (<@ и @>)
  - Один массив считается вложенным в другой, если каждый элемент первого встречается во втором
  - Повторяющиеся элементы рассматриваются на общих основаниях

```
select '{(408)-567-78234}'::text[] <@ '{"(408)-745-8965","(408)-567-78234"}'::text[];
```

- Для проверки вхождения литерала в массив используется оператор **ANY** (**SOME**):

```
SELECT 10 = SOME (ARRAY[192, 168, 10, 10]) ;
```

Разворачивание массива в набор записей

- Чтобы представить элементы массива в виде значений некоторого столбца необходимо воспользоваться функцией **UNNEST**(**ANYARRAY**)

```
SELECT
       UNNEST(ARRAY[100, 110, 153, 500]) as prodid,
       50000::money as price ,
       '2022-10-20' as change_date;
```

	prodid	price	change_date
1	100	$50,000.00	2022-10-20
2	110	$50,000.00	2022-10-20
3	153	$50,000.00	2022-10-20
4	500	$50,000.00	2022-10-20

Пример использования массива

```
select contactname,
       split_part(contactname,',',1) as "Lname",
       trim((string_to_array(contactname,','))[2]) as "Fname"
from "Sales"."Customers";
```

	contactname	Lname	Fname
1	Bassols, Pilar Colome	Bassols	Pilar Colome
2	Richardson, Shawn	Richardson	Shawn
3	Russo, Giuseppe	Russo	Giuseppe
4	Cheng, Yao-Qiang	Cheng	Yao-Qiang

Типы JSON

Типы JSON

- Предназначены для сохранения и обработки данных в формате JSON (JavaScript Object Notation)
- 2 типа:
  - **json**
    - значения сохраняются в исходном виде – **высокая скорость сохранения**
    - при использовании выполняется полный разбор – **низкая скорость обработки**
    - сохраняет порядок следования ключей и повторяющиеся значения ключей, при этом функции обработки будут считать действительной последнюю пару
  - **jsonb**
    - разбор производится однократно при сохранении – **низкая скорость сохранения**
    - при использовании разбор не выполняется – **высокая скорость обработки** (при выборке по ключу в 1000 раз!)
    - ключи не дублируются
    - отсортированы по длине и ключу
    - поддерживается индексирование

https://postgrespro.ru/docs/postgresql/12/datatype-json

ВАЖНО!

- PostgreSQL позволяет использовать только одну кодировку символов в базе данных. Если кодировка базы данных не UTF-8:
  - данные JSON не будут полностью соответствовать спецификации
  - нельзя будет вставить символы, непредставимые в кодировке сервера
  - допустимыми будут символы, представимые в кодировке сервера, но не в UTF-8

Формат представления

- В качестве json значений могут выступать:
  - Числа, строки в двойных кавычках, true и false (в нижнем регистре) или null

```
SELECT '5'::json, '"Post"'::json, 'true'::json, null::json;
```

  - Массив из нуля и более элементов (элементы могут быть разных типов)

```
SELECT '[1, 2, "foo", null]'::json;
```

  - Объект, содержащий пары ключей и значений
    - **Ключи** объектов — это всегда строки в двойных кавычках
    - Массивы и объекты могут вкладываться произвольным образом

```
SELECT '{"cheef":"Ivan", "Empl":["Svetlana","Eugen"]}'::jsonb;
```

Функции-конструкторы

- Для формирования json можно использовать специализированные функции:
  - to_json(anyelement) и to_jsonb(anyelement)
  - jsonb_build_object(VARIADIC "any") и jsonb_build_object(VARIADIC "any")
  - array_to_json(anyarray [, pretty_bool])
  - row_to_json(record [, pretty_bool])
  - json_object(keys text[], values text[]) и jsonb_object(keys text[], values text[])

```
SELECT json_object('{product_name, Apple, price, 229, category, 1}'),
       json_object('{{product_name, Apple},{price, 229},{category, 1}}'),
       json_object('{product_name, price, category}', '{Apple,229, 1}')
```

	json_object
1	{"product_name" : "Apple", "price" : "229", "category" : "1"}

https://postgrespro.ru/docs/postgresql/12/functions-json

Извлечение элемента json и jsonb

- Извлечение элемента
  - по индексу элемента массива JSON - индексация элементов начинается с 0. Отрицательные числа задают позиции с конца

```
SELECT  '[{"a":"foo"},{"b":"bar"},{"c":"baz"}]'::json -> 2;
SELECT  '[{"a":"foo"},{"b":"bar"},{"c":"baz"}]'::json -> -1;
```

	?column?
1	{"c":"baz"}

  - по ключу

```
SELECT  json_object('{product_name, Apple, price, 229}') -> 'price';
```

	?column?
1	"229"

  - по заданному пути

```
SELECT '{"boss":"Ivan","emp":["Sveta","Egor","Eugen"]}'::json #>>'{emp,2}';
SELECT '{"boss":"Ivan","emp":["Sveta","Egor","Eugen"]}'::json #>>'{emp,-1}';
```

	?column?
1	Eugen

https://postgrespro.ru/docs/postgresql/12/functions-json

Проверка вхождения

- Типы данных json или jsonb не требуют задавать структуру объектов, т. е. конкретные имена ключей и типы значений
- Может потребоваться выполнить проверку **jsonb**:
  - На наличие ключа

```
SELECT  '{"a":"foo","b":"bar","c":"baz"}'::jsonb ? 'b';
```

	?column?
1	[v]

  - На наличие пути/значения

```
SELECT  '{"a":1, "b":2}'::jsonb @> '{"b":2}'::jsonb;
```

	?column?
1	[v]

https://postgrespro.ru/docs/postgresql/12/functions-json#FUNCTIONS-JSONB-OP-TABLE

Изменение json и jsonb

- Добавление/изменение

```
select jsonb_set('{"cheef":"Ivan", "Emp":["Svetlana","Eugen"]}',
                 '{Empl,2}', '"Egor"'); --Добавляем новый элемент
```

	jsonb_set
1	{"Empl": ["Svetlana", "Eugen", "Egor"], "cheef": "Ivan"}

```
select jsonb_set('{"cheef":"Ivan", "Emp":["Svetlana","Eugen"]}',
                 '{Empl,0}', '"Egor"'); --Заменяем 0 элемент
```

	jsonb_set
1	{"Empl": ["Egor", "Eugen"], "cheef": "Ivan"}

- Удаление

```
select '{"cheef":"Ivan", "Emp":["Svetlana","Eugen"]}'::jsonb - 'Emp';
```

	?column?
1	{"cheef": "Ivan"}

https://postgrespro.ru/docs/postgresql/12/functions-json