Слабо-структурированные данные в СУБД PostgreSQL (мастер-класс)

Александр Коротков, Иван Панченко

Postgres Professional

2017

1. Материалы

Docker

```
docker run --rm -p 127.0.0.1:9000:5432 waaeer/semitutorial
```

VirtualBox

```
https://pgconf.ru/media/2017/03/14/json-tutorial96.ova
```

Руки

```
https://pgconf.ru/2017/93658
```

Там же ссылки на документацию.

2. Jsonb

Исходные данные: company

```
# \d company
  Table "public.company"
 Column | Type   | Modifiers
--------+-------+-----------
 js     | jsonb |

# SELECT js FROM company limit 5;

-----------------------------------------------------------------
 {"_id": {"$oid": "52cdef7c4bab8bd675297d8a"}, "name": "Wetpaint
 {"_id": {"$oid": "52cdef7c4bab8bd675297d8b"}, "name": "AdventNe
 {"_id": {"$oid": "52cdef7c4bab8bd675297d8c"}, "name": "Zoho",
 {"_id": {"$oid": "52cdef7c4bab8bd675297d8d"}, "ipo": null, "nam
 {"_id": {"$oid": "52cdef7c4bab8bd675297d8e"}, "ipo": {"pub_day"
(5 rows)
```

Jsonb pretty print

```
# SELECT jsonb_pretty(js) FROM company LIMIT 1;
-[ RECORD 1 ]+-----------------------------------
jsonb_pretty | {
             |     "_id": {
             |         "$oid": "52cdef7c4bab8bd675297d8a"
             |     },
             |     "name": "Wetpaint",
             |     "image": {
             |         "available_sizes": [
             |             [
             |                 [
             |                     150,
             |                     75
.....................................................
```

Вытаскивание полей

```
# SELECT js -> 'name',
         js -> 'acquisition' -> 'price_amount',
         js #> '{acquisition,price_amount}'
  FROM company
  LIMIT 5;
        ?column?          | ?column?  | ?column?
--------------------------+-----------+-----------
 "Wetpaint"               | 30000000  | 30000000
 "AdventNet"              | NULL      | NULL
 "Zoho"                   | NULL      | NULL
 "Digg"                   | 500000    | 500000
 "Facebook"               | NULL      | NULL
(5 rows)
```

Вытаскивание полей и типы (1/2)

-> вытаскивает тип jsonb

```
# SELECT js -> 'name' FROM company
  WHERE js -> 'name' ILIKE '%paint%';
ERROR:  operator does not exist: jsonb ~~* unknown
```

Приведение значения поля к строке

```
# SELECT (js -> 'name')::text, js ->> 'name'
  FROM company LIMIT 5;
    text      | ?column?
--------------+-----------
 "Wetpaint"   | Wetpaint
 "AdventNet"  | AdventNet
 "Zoho"       | Zoho
 "Digg"       | Digg
 "Facebook"   | Facebook
(5 rows)
```

Вытаскивание полей и типы (2/2)

```
# SELECT
      js ->> 'name',
      js ->> 'number_of_employees'
  FROM company
  WHERE
      js ->> 'name' ILIKE '%paint%' AND
      (js ->> 'number_of_employees')::int >= 10;
 ?column? | ?column?
----------+----------
 Wetpaint | 47
(1 row)
```

Вытаскивание элементов массивов

```
# SELECT
      js -> 'offices' -> 0 ->> 'city',
      js #>> '{offices,0,city}'
  FROM company LIMIT 5;
   ?column?    |   ?column?
---------------+---------------
 Seattle       | Seattle
 Pleasanton    | Pleasanton
 Pleasanton    | Pleasanton
 San Francisco | San Francisco
 Menlo Park    | Menlo Park
(5 rows)
```

Индексы по выражениям

```
# CREATE UNIQUE INDEX company_id_idx
ON company ((js -> '_id' ->> '$oid'));
CREATE INDEX

# EXPLAIN ANALYZE SELECT js -> 'name' FROM company
  WHERE js -> '_id' ->> '$oid' = '52cdef7c4bab8bd675297d8a';
                                                        QUERY PL
-------------------------------------------------------------------
 Index Scan using company_id_idx on company  (cost=0.29..8.31 ro
   Index Cond: (((js -> '_id'::text) ->> '$oid'::text) = '52cdef
 Planning time: 0.084 ms
 Execution time: 0.156 ms
(4 rows)
```

Разворачивание JSON-массивов в rowset

```
# SELECT jsonb_array_elements(js -> 'competitions')
  FROM company
  WHERE js -> '_id' ->> '$oid' = '52cdef7c4bab8bd675297d8a';
                        jsonb_array_elements
-----------------------------------------------------------------------
 {"competitor": {"name": "Wikia", "permalink": "wikia"}}
 {"competitor": {"name": "JotSpot", "permalink": "jotspot"}}
 {"competitor": {"name": "Socialtext", "permalink": "socialtext"}}
 {"competitor": {"name": "Ning by Glam Media", "permalink": "ning"}}
 {"competitor": {"name": "Soceeo", "permalink": "soceeo"}}
 {"competitor": {"name": "Yola", "permalink": "yola"}}
 {"competitor": {"name": "SocialGO", "permalink": "socialgo"}}
 {"competitor": {"name": "IslamNor", "permalink": "islamnor"}}
(8 rows)
```

Разворачивание JSON-объекта в key-value rowset

```
# SELECT (jsonb_each(js #> '{offices, 0}')).*
  FROM company
  WHERE js -> '_id' ->> '$oid' = '52cdef7c4bab8bd675297d8a';
     key      |        value
--------------+----------------------
 city         | "Seattle"
 address1     | "710 - 2nd Avenue"
 address2     | "Suite 1100"
 latitude     | 47.603122
 zip_code     | "98104"
 longitude    | -122.333253
 state_code   | "WA"
 description  | ""
 country_code | "USA"
(9 rows)
```

Оператор @>

- Для скалярных a и b, a @> b тогда и только тогда, когда a = b.
- Если a и b объекты, то для каждого ключа в b должен существовать ключ в a, содержащий соответствующее значение.
- Если a и b массивы, то для каждого элемента в b должен существовать содержащий его элемент в a.

Примеры.

arg1	arg2	result
1	1	true
1	"1"	false
{"a": 1, "b": 2}	{"a": 1}	true
{"a": 1, "b": 2}	{"a": 2}	false
[1, 2]	[1]	true
[{"a": 1, "b": 2}]	[{"a": 1}, {"b": 2}]	true
[{"a": 1}, {"b": 2}]	[{"a": 1, "b": 2}]	false

Установка plv8

```
$ git clone https://github.com/plv8/plv8.git
$ cd plv8
$ make USE_PGXS=1
$ sudo make USE_PGXS=1 install
$ make USE_PGXS=1 installcheck
$ psql DB -c ”CREATE EXTENSION plv8;”
```

Задача #1

Написать на plv8 функцию, которая для компании выводит список стран, где у неё есть офисы.

```
offices: [
 { country_code : ”RUS” }
]
```

Задача #1: решение

```
CREATE FUNCTION extract_countries(company jsonb)
RETURNS text[]
AS $$
    var tmp = {}, result = [];
    for(var i = 0; i < company.offices.length; i++)
        tmp[company.offices[i].country_code] = true;
    for(var i in tmp)
    result.push(i)
    return result;
$$ LANGUAGE plv8 IMMUTABLE STRICT;
```

Индекс по странам

```
CREATE INDEX company_countries_idx ON company
USING gin(extract_countries(js));
# EXPLAIN ANALYZE SELECT * FROM company
  WHERE extract_countries(js) @> '{RUS}';
                                                                QUERY
-----------------------------------------------------------------------
 Bitmap Heap Scan on company  (cost=8.73..348.48 rows=94 width=8
     (actual time=0.038..0.089 rows=50 loops=1)
   Recheck Cond: (extract_countries(js) @> '{RUS}'::text[])
   Heap Blocks: exact=48
   ->  Bitmap Index Scan on company_countries_idx  (cost=0.00..8
         Index Cond: (extract_countries(js) @> '{RUS}'::text[])
 Planning time: 0.057 ms
 Execution time: 0.113 ms
(7 rows)
```

Задача #2

Написать на plv8 агрегат, который который посчитает частоты различных размеров картинок.

```
image: { available_sizes : [
  [800,600], [1024,768]
]}
```

Задача #2: решение (1/2)

```
CREATE FUNCTION image_size_agg_sfunc(state jsonb,
                    company jsonb) RETURNS jsonb
AS $$
    if (!company.image) return state;
    var sizes = company.image.available_sizes
    for (var i = 0; i < sizes.length; i++)
    {
        var sizeKey = sizes[i][0][0] + 'x' +
                      sizes[i][0][1];
        if (sizeKey in state) state[sizeKey]++;
        else state[sizeKey] = 1;
    }
    return state;
$$ LANGUAGE plv8 IMMUTABLE STRICT;
```

Задача #2: решение (2/2)

```
CREATE AGGREGATE image_size_agg (jsonb) (
    SFUNC = image_size_agg_sfunc,
    STYPE = jsonb,
    INITCOND = '{}');

# SELECT image_size_agg(js) FROM company;
-[ RECORD 1 ]--+-----------------------------------------
image_size_agg | {"1x1": 15, "32x8": 3, "150x7": 1, "150x8": 5,
```

Установка jsquery

```
$ git clone https://github.com/postgrespro/jsquery.git
$ cd jsquery
$ make USE_PGXS=1
$ sudo make USE_PGXS=1 install
$ make USE_PGXS=1 installcheck
$ psql DB -c ”CREATE EXTENSION jsquery;”
```

Индексы для jsonb

- GIN jsonb_ops: индексируем ключи отдельно, значения отдельно (операторы @>, ?, ?&, ?|).
- GIN jsonb_path_ops: индексируем hash(path + value) (операторы @>).
- jsonb_path_value_ops: индексируем hash(path) + hash(value) (операторы @>, jsonb @@ jsquery).
- jsonb_value_path_ops: индексируем hash(value) + bloom(path) (операторы @>, jsonb @@ jsquery).
- Индексы по отдельным ключам (js -> 'key').

Пример: найти комании с офисами в России

```
-- Оператор @>
# SELECT js->'name' FROM company
  WHERE js @> '{"offices":[{"country_code": "RUS"}]}';

-- Оператор jsquery
# SELECT js->'name' FROM company
  WHERE js @@ 'offices.#.country_code = "RUS"';

-- Разворачивание jsonb в rowset
# SELECT js->'name' FROM company
  WHERE EXISTS (
      SELECT 1
      FROM jsonb_array_elements(js->'offices') el
      WHERE el ->> 'country_code' = 'RUS');
```

Задача #3

Найти компании, конкуренты Wikia, у которых есть офис в США.

Реализовать решение с помощью:

- оператора @>,
- jsquery,
- разворачивания jsonb в rowset.

```
{ competitions: [ { competitor : { name : "EE" }}, ...] }
```

Задача #3: решение (1/3)

```
-- Оператор @>
# SELECT js->'name' FROM company
  WHERE
    js @>
     '{"competitions":[{"competitor":
        {"name": "Wikia"}
      }]}' AND
    js @>
     '{"offices":[{"country_code": "USA"}]}';
```

Задача #3: решение (2/3)

```
-- jsquery
# SELECT js->'name' FROM company
  WHERE js @@
   'competitions.#.competitor.name = "Wikia" AND
    offices.#.country_code = "USA"';
```

Задача #3: решение (3/3)

```
-- Разворачивание jsonb в rowset
SELECT js->'name' FROM company
WHERE
  EXISTS (
    SELECT 1
    FROM jsonb_array_elements(js->'competitions') el
    WHERE el -> 'competitor' ->> 'name' = 'Wikia'
  ) AND EXISTS (
    SELECT 1
    FROM jsonb_array_elements(js->'offices') el
    WHERE el ->> 'country_code' = 'USA'
  );
```

Задача #4

Найти компании, в которые Trinity Ventures инвестировала в 2007 или 2008 году.

Реализовать решение с помощью:

- оператора @>,
- jsquery,
- разворачивания jsonb в rowset.

```
funding_rounds: [
    { funded_year : 2007, investiments: [
        {financial_org: { name: ”ООО”}}
    ]},...
]
```

Задача #4: решение (1/3)

```
-- Оператор @>
# SELECT js->'name' FROM company
  WHERE
    js @>
     '{"funding_rounds":[{"funded_year": 2007,
       "investments":[{"financial_org":
         {"name":"Trinity Ventures"}}
       ]}
     ]}' OR
    js @>
     '{"funding_rounds":[{"funded_year": 2008,
       "investments":[{"financial_org":
         {"name":"Trinity Ventures"}}
       ]}
     ]}';
```

Задача #4: решение (2/3)

```
-- jsquery
# SELECT js->'name' FROM company
  WHERE
    js @@
     'funding_rounds.#(
        funded_year IN (2007,2008) AND
        investments.#.
          financial_org.name = "Trinity Ventures")';
```

Задача #4: решение (3/3)

```
-- Разворачивание jsonb в rowset
# SELECT js->'name' FROM company
  WHERE EXISTS (
    SELECT 1
    FROM jsonb_array_elements(js->'funding_rounds') el
    WHERE el -> 'funded_year' IN ('2007','2008') AND EXISTS
     (SELECT 1
      FROM jsonb_array_elements(el -> 'investments') el2
      WHERE el2 -> 'financial_org' ->>
                   'name' = 'Trinity Ventures'));
```

Задача #5

Найти компании, которые в 2008 году привлекли более 100 000 000 USD инвестиций.

Реализовать решение с помощью:

- jsquery,
- разворачивания jsonb в rowset.

```
funding_rounds: [
    { funded_year : 2007,  raised_amount : 1000000,
      raised_currency_code:"USD"
    }, ....
]
```

Задача #5: решение (1/2)

```
-- jsquery
# SELECT js -> 'name'
  FROM company
  WHERE js @@
   'funding_rounds.# (funded_year = 2008 AND
                      raised_currency_code = "USD" AND
                      raised_amount > 100000000)';
```

Задача #5: решение (2/2)

```
-- Разворачивание jsonb в rowset
# SELECT js->'name' FROM company
  WHERE EXISTS (
    SELECT 1
    FROM jsonb_array_elements(js->'funding_rounds') el
    WHERE
      el -> 'funded_year' = '2008' AND
      el ->> 'raised_currency_code' = 'USD' AND
      el -> 'raised_amount' > '100000000');
```

Задача #6

Найти компании, у которых суммарный объём инвестиций был более 1 000 000 000 USD.

Реализовать решение с помощью разворачивания jsonb в rowset.

```
funding_rounds: [
    { funded_year : 2007,  raised_amount : 1000000,
      raised_currency_code:”USD”
    }, ....
]
```

Задача #6: решение

```
-- Разворачивание jsonb в rowset
SELECT js->'name' FROM company
  WHERE (
    SELECT SUM((el ->> 'raised_amount')::numeric)
    FROM jsonb_array_elements(js->'funding_rounds') el
    WHERE
      el ->> 'raised_currency_code' = 'USD') > 1000000000;
```

Задача #7

Выдать названия компаний в порядке количества упомянутых лиц.

Реализовать решение с помощью разворачивания jsonb в rowset.

```
relationships: [
    { person : { last_name : ”Ivanov” } },
     ....
]
```

Задача #7: решение

```
SELECT js->>'name',
       (SELECT COUNT    (DISTINCT r->'person'->>'last_name')
           FROM jsonb_array_elements(js->'relationships') r
       ) n
FROM company ORDER BY n DESC;
```

Домашнее задание

Конкурс Выдать названия компаний в порядке количества упомянутых лиц, и сам список этих лиц для каждой компании в виде массива.

Реализовать решение с помощью разворачивания jsonb в rowset.

Postgres PROFESSIONAL

Спасибо за внимание!