JSONB В POSTGRESQL И NOSQL ТРЕНД

сравнение функциональности и производительности

Дмитрий Долгов

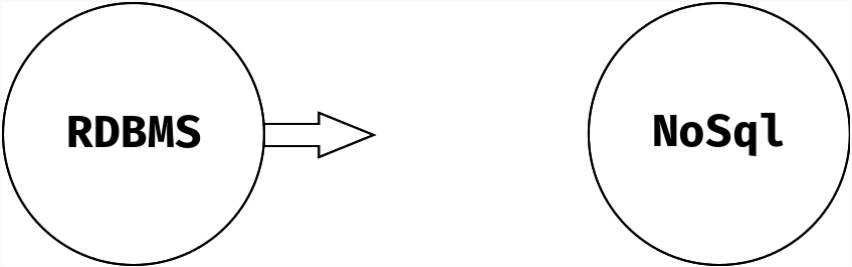
February 2, 2016

ФИЛОСОФСКОЕ ВВЕДЕНИЕ

NoSql популярен и это многим не дает покоя.
Это приводит к тому, что многие реляционные базы данных предлагают поддержку тех или иных возможностей, изначально ассоциирующихся с NoSql.

ФИЛОСОФСКОЕ ВВЕДЕНИЕ

ФИЛОСОФСКОЕ ВВЕДЕНИЕ



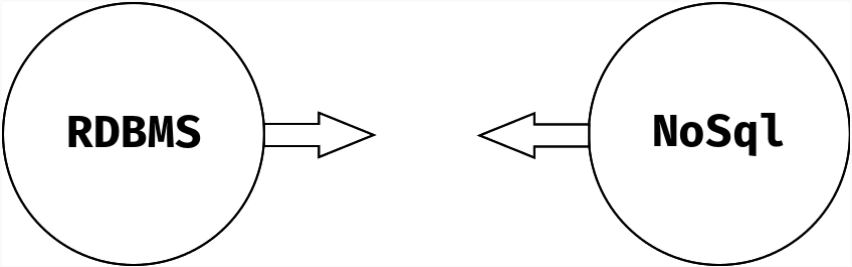
ФИЛОСОФСКОЕ ВВЕДЕНИЕ

Почему это важно? Каков уровень поддержки хранения слабо-структурированных данных в PostgreSQL?

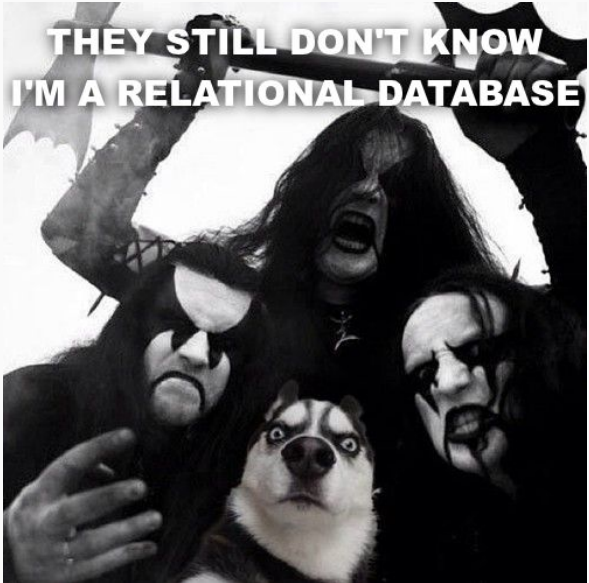
ФИЛОСОФСКОЕ ВВЕДЕНИЕ

ФИЛОСОФСКОЕ ВВЕДЕНИЕ

ФИЛОСОФСКОЕ ВВЕДЕНИЕ



ФИЛОСОФСКОЕ ВВЕДЕНИЕ



СРАВНЕНИЕ ФУНКЦИОНАЛА

ОБЗОР ПО КАТЕГОРИЯМ

DB	Native	Select	Modify	Delete	Attributes	Indexing	Search	Convertion	Syntastic
PG	✓	✓	✓	✓	✓	✓	?	✓	?
Mysql	✓	✓	✓	✓	✓	?	?	✗	?
Oracle	✗	✓	✗	✗	✗	✗	?	?	?
DB2	✓	✗	✗	✗	✗	✗	✗	?	✗
MSSql	✗	✓	✗	✗	✗	✗	?	?	✗

POSTGRESQL

POSTGRESQL

- Hstore
- Json
- Jsonb + (jsonbx)

PostgreSQL 9.5

МНОГОЛИКИЙ JSONB

```
select '"string"'::jsonb;
```

МНОГОЛИКИЙ JSONB

```
select '"string"'::jsonb;
```

```
select '1'::jsonb;
```

МНОГОЛИКИЙ JSONB

```
select '"string"'::jsonb;
```

```
select '1'::jsonb;
```

```
select 'true'::jsonb;
```

МНОГОЛИКИЙ JSONB

```
select '"string"'::jsonb;
```

```
select '1'::jsonb;
```

```
select 'true'::jsonb;
```

```
select '[1, 2, 3]'::jsonb;
```

МНОГОЛИКИЙ JSONB

```
select '"string"'::jsonb;
```

```
select '1'::jsonb;
```

```
select 'true'::jsonb;
```

```
select '[1, 2, 3]'::jsonb;
```

```
select '["string", 1]'::jsonb;
```

МНОГОЛИКИЙ JSONB

```
select '"string"'::jsonb;
```

```
select '1'::jsonb;
```

```
select 'true'::jsonb;
```

```
select '[1, 2, 3]'::jsonb;
```

```
select '["string", 1]'::jsonb;
```

```
select '{"key": {"nested": "value"}}'::jsonb;
```

ПОЛУЧЕНИЕ ДАННЫХ

```
select '{"key": "value"}'::jsonb ->> 'key';
```

```
select '["string", 1]'::jsonb -> -1;
```

```
select '{
    "key": {"nested_key": "value"}
}'::jsonb #> '{key, nested_key}'
```

ИЗМЕНЕНИЕ ДАННЫХ

```
select jsonb_set(
    '{"n":null, "a":{"b": 2}}'::jsonb,
    '{n}',
    '[1,2,3]'
);

            jsonb_set
──────────────────────────────────────
{"a": {"b": 2}, "n": [1, 2, 3]}
 (1 row)
```

УДАЛЕНИЕ ДАННЫХ

```
select
    '{"a": {"b": [1, 2, 3]}}'::jsonb
    #-
    '{a, b, -1}';
       ?column?
----------------------
 {"a": {"b": [1, 2]}}
(1 row)
```

АТТРИБУТЫ

```
select jsonb_object_keys(
    '{"key": "value"}'::jsonb
);

 jsonb_object_keys
___________________
 key
(1 row)
```

АТТРИБУТЫ

```
select jsonb_typeof('1'::jsonb);

 jsonb_typeof
──────────────
 number
(1 row)
```

ИНДЕКСИРОВАНИЕ

- GIN индекс для @> <@ ?
- jsonb_path_ops
- jsquery: jsonb_path_value, jsonb_path_key

ПОИСК

- Содержит ли jsonb объект указанных ключ?
- jsquery

КОНВЕРТИРОВАНИЕ

```
select * from test_agg;
 id |   data
----+----------
  1 | value1
  2 | value2
(2 rows)
```

```
select jsonb_pretty(jsonb_agg(test_agg)) from test_agg;
         jsonb_pretty
------------------------------
 [                            +
     {                        +
         "id": 1,             +
         "data": "value1"     +
     },                       +
     {                        +
         "id": 2,             +
         "data": "value2"     +
     }                        +
 ]
(1 row)
```

КОНВЕРТИРОВАНИЕ

```
select array_to_json(
    ARRAY [
        jsonb '{"a":1}',
        jsonb '{"b":[2,3]}'
    ]
);
      array_to_json
_________________________
 [{"a": 1},{"b": [2, 3]}]
(1 row)
```

СИНТАКСИС

```
update some_table set jsonb_data =
    jsonb_set(jsonb_data, '{a, a1, a2}', '42');
```

VS

```
update some_table
    set jsonb_data['a']['a1']['a2'] = 42;
```

MYSQL

MYSQL

MySql 5.7.7, тип данных JSON

ВОЗМОЖНЫЕ ВИДЫ

```
select cast('"string"' as json);
```

ВОЗМОЖНЫЕ ВИДЫ

```
select cast('"string"' as json);
```

```
select cast('["string", 1]' as json);
```

ВОЗМОЖНЫЕ ВИДЫ

```
select cast('"string"' as json);

select cast('["string", 1]' as json);

select cast('{"key": {"nested": "value"}}' as json);
```

ПОЛУЧЕНИЕ ДАННЫХ

```
select json_extract('{"key": "value"}', '$.*');
```

```
select cast('{"key": "value"}' as json) -> 'key';
```

ИЗМЕНЕНИЕ ДАННЫХ

```
select json_set(
    '{"n":null, "a":{"b": 2}}',
    '$.n',
    '[1,2,3]',
    '$.a',
    1
);

____________________________

{"a": 1, "n": "[1,2,3]"}
1 row in set (0.01 sec)
```

УДАЛЕНИЕ ДАННЫХ

```
select json_remove(
    '{"a": {"b": [1, 2, 3]}}',
    '$.a.b[2]'
)

_____________________
{"a": {"b": [1, 2]}}
1 row in set (0.01 sec)
```

АТТРИБУТЫ

```
select json_type('1');

_______________
INTEGER
1 row in set (0.01 sec)
```

АТТРИБУТЫ

Нет методов для получения ключей, значений и проч.

Есть методы для получения длинны или глубины json.

ИНДЕКСИРОВАНИЕ

Тип json на прямую не индексируется, в качестве workaround предлагается создавать generated поля с json_extract.

ПОИСК

- Поиск в пути $, *
- Поиск по значению json_search

```
select json_search(
    '{"a": "test", "b": [1, 2, "test2"]}',
    'all', 'test%'
);

_______________
["$.a", "$.b[2]"]
1 row in set (0.00 sec)
```

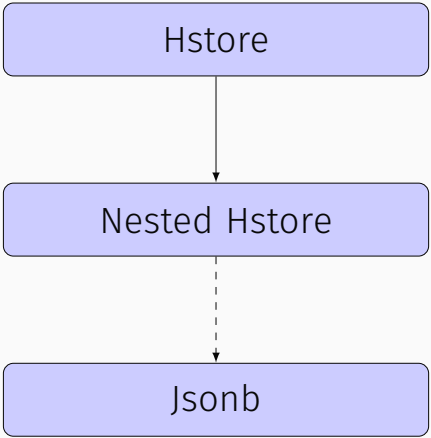
ORACLE

ORACLE

Oracle 12.1.0.2, тип данных JSON

Требует **WITH UNIQUE KEYS**

ТЕСТ



MANY FACES OF JSONB

```
select '"string"'::jsonb;
```

MANY FACES OF JSONB

```
select '"string"'::jsonb;
```

```
select '1'::jsonb;
```

MANY FACES OF JSONB

```
select '"string"'::jsonb;
```

```
select '1'::jsonb;
```

```
select 'true'::jsonb;
```

MANY FACES OF JSONB

```
select '"string"'::jsonb;

select '1'::jsonb;

select 'true'::jsonb;

select '[1, 2, 3]'::jsonb;
```

MANY FACES OF JSONB

```
select '"string"'::jsonb;
```

```
select '1'::jsonb;
```

```
select 'true'::jsonb;
```

```
select '[1, 2, 3]'::jsonb;
```

```
select '["string", 1]'::jsonb;
```

MANY FACES OF JSONB

```
select '"string"'::jsonb;

select '1'::jsonb;

select 'true'::jsonb;

select '[1, 2, 3]'::jsonb;

select '["string", 1]'::jsonb;

select '{"key": {"nested": "value"}}'::jsonb;
```

LACK OF SOME FUNCTIONALITY

Тест тест

- Get an element at arbitrary path (#>)
- Delete an element at arbitrary path (?)
- Update an element at arbitrary path (?)
- Add a new element to arbitrary path (?)

JSONBX

Pgxn extension for PostgreSQL 9.4, which contains implementation of some missing functionality. It based on nested version of hstore and provided this functions for the corresponding patch for 9.5

GET DATA

BY KEY

```
select '{"key": "value"}'::jsonb -> 'key';
```

```
select '{"key": "value"}'::jsonb ->> 'key';
```

BY INDEX

```
select '["string", 1]'::jsonb -> 0;
```

```
select '["string", 1]'::jsonb -> -1;
```

BY PATH

- Point out an element inside a jsonb field
- Technically, it's just an array of items
- Each element is an index (include negative values) or a key

```
'{first_key, second_key, 1, -1}'
```

BY PATH

```
select '{
    "key": {"nested_key": "value"}
}'::jsonb #> '{key, nested_key}'
```

```
select '{
    "key": [1, 2, "string"]
}'::jsonb #> '{key, -1}'
```

KNOW YOUR DATA

CONTAINS OR CONTAINED?

```
select
    '{"a": 1, "b": 1}'::jsonb
    @>
    '{"a": 1}'::jsonb
```

```
select '[1]'::jsonb <@ '[1, 2]'::jsonb
```

EQUALITY

```
select
    '{"key":"value"}'::jsonb
    <>
    '{"key":"another_value"}'::jsonb;
```

```
select
    '{"key":"value"}'::jsonb
    =
    '{"key":"another_value"}'::jsonb;
```

JSONB_PRETTY

```
select jsonb_pretty('{"a":"test","b":[1,2,3],"c":"test3","d":{"dd":"test4","dd2":{"ddd":"test5"}}}'::jsonb);

        jsonb_pretty
-----------------------------
 {                           +
     "a": "test",            +
     "b": [                  +
         1,                  +
         2,                  +
         3                   +
     ],                      +
     "c": "test3",           +
     "d": {                  +
         "dd": "test4",      +
         "dd2": {            +
             "ddd": "test5"+
         }                   +
     }                       +
 }
(1 row)
```

AGGREGATION

```
select * from test_agg;
 id |   data
----+----------
  1 | value1
  2 | value2
(2 rows)
```

```
select jsonb_pretty(jsonb_agg(test_agg)) from test_agg;
         jsonb_pretty
------------------------------
 [                            +
     {                        +
         "id": 1,             +
         "data": "value1"     +
     },                       +
     {                        +
         "id": 2,             +
         "data": "value2"     +
     }                        +
 ]
(1 row)
```

GET KEYS

```
select jsonb_object_keys(
    '{"key": "value"}'::jsonb
);

 jsonb_object_keys
___________________
 key
(1 row)
```

BUILD & TRANSFORM

BUILD

```
select jsonb_build_object('key', 'value');
 jsonb_build_object
____________________
 {"key": "value"}
(1 row)
```

```
select jsonb_object('{key, value}');
   jsonb_object
_________________
 {"key": "value"}
(1 row)
```

BUILD

```
select jsonb_build_array('string', 1);
 jsonb_build_array
___________________
 ["string", 1]
(1 row)
```

POPULATE RECORD

```
create type data_type as (key text, value text);

select * from jsonb_populate_record(
    null::data_type,
    '{"key": "some_key", "value": "some_data"}'
);

   key    |   value
----------+-----------
 some_key | some_data
(1 row)
```

MANIPULATION WITH JSONB

JSONB_SET: REPLACE VALUE

```
select jsonb_set(
    '{"n":null, "a":{"b": 2}}'::jsonb,
    '{n}',
    '[1,2,3]'
);

            jsonb_set
──────────────────────────────────────
{"a": {"b": 2}, "n": [1, 2, 3]}
 (1 row)
```

JSONB_SET: CREATE MODE BY DEFAULT

```
select jsonb_set(
    '{"a":{"b": 2}}'::jsonb,
    '{c}',
    '[1,2,3]'
);

            jsonb_set
-----------------------------------------
{"a": {"b": 2}, "c": [1, 2, 3]}
 (1 row)
```

JSONB_SET: TURN OFF THE CREATE MODE

```
select jsonb_set(
    '{"a":{"b": 2}}'::jsonb,
    '{c}',
    '[1,2,3]',
    false
);

    jsonb_set
_________________
 {"a": {"b": 2}}
(1 row)
```

JSONB_DELETE: BY KEY

```
select '{"a":1 , "b":2, "c":3}'::jsonb - 'a';

     ?column?
-----------------
 {"b": 2, "c": 3}
(1 row)
```

JSONB_DELETE: BY INDEX

```
select '["a", "b", "c"]'::jsonb - 2;

  ?column?
------------
 ["a", "b"]
(1 row)
```

JSONB_DELETE: BY PATH

```
select
    '{"a": {"b": [1, 2, 3]}}'::jsonb
    #-
    '{a, b, -1}';
       ?column?
----------------------
 {"a": {"b": [1, 2]}}
(1 row)
```

JSONB_CONCAT ("SHALLOW CONCATENATION")

```
select
    '{"a": 1, "b": [2, 3]}'::jsonb
    ||
    '{"a": 4, "c": [5, 6]}'::jsonb

              ?column?
──────────────────────────────────────
 {"a": 4, "b": [2, 3], "c": [5, 6]}
(1 row)
```

JSONB_CONCAT ("SHALLOW CONCATENATION")

```
select
    '{"key": {"nested_key": "value"}}'::jsonb
    ||
    '{"key": {"another_key": "another_value"}}'::jsonb

                    ?column?
──────────────────────────────────────────────
 {"key": {"another_key": "another_value"}}
(1 row)
```

JSONB_CONCAT ("SHALLOW CONCATENATION")

Array will consume an object

```
select
    '{"key": "value"}'::jsonb
    ||
    '[1, 2, 3]'::jsonb

          ?column?
──────────────────────────────
 [{"key": "value"}, 1, 2, 3]
(1 row)
```

JSONB_CONCAT ("SHALLOW CONCATENATION")

Scalars are acting like arrays

```
select '"b"'::jsonb || '"a"'::jsonb

  ?column?
____________
 ["b", "a"]
```

PERFORMANCE

COMPARISON

PLANS FOR THE FUTURE

ARRAY-STYLE SUBSCRIPTING

```
update some_table
    set jsonb_data['a']['a1']['a2'] = 42;
```

```
select jsonb_data['a']['a1']['a2']
    from some_table;
```

JSONB POINTER & PATCH

- Json pointer [rfc6901]
- Json patch [rfc6901]

NEW FUNCTIONS AND IMPROVEMENTS

There are still missing functionality and improvements, that can be useful for JSONB. Some of them will be implented as parts of jsonbx extension (for 9.5), and will be proposed for 9.6

JSONB_DELETE

```
select jsonb_delete_jsonb(
    '{"a": 1, "b": {"c": 2}, "f": [4, 5]}'::jsonb,
    '{"a": 4, "f": [4, 5], "c": 2}'::jsonb
);

      jsonb_delete
_____________________________
  {"a": 1, "b": {"c": 2}}
(1 row)
```

JSONB_INTERSECTION

```
select jsonb_intersection(
    '{"a":2, "d": {"f": 3}, "g":[4, 5]}'::jsonb,
    '{"a":2, "f": 3, "g":[4, 5]}'::jsonb
);

   jsonb_intersection
──────────────────────────
  {"a": 2, "g": [4, 5]}
(1 row)
```

JSONB_DEEP_MERGE

```
select jsonb_deep_merge(
    '{"a": {"b":1}}'::jsonb,
    '{"a": {"c":2}}'::jsonb
);

     jsonb_deep_merge
____________________________
   {"a": {"b":1, "c":2}}
(1 row)
```

FINISH

CONTACTS

github.com/erthalion/jsonbx

@erthalion

9erthalion6 at gmail dot com

Questions?