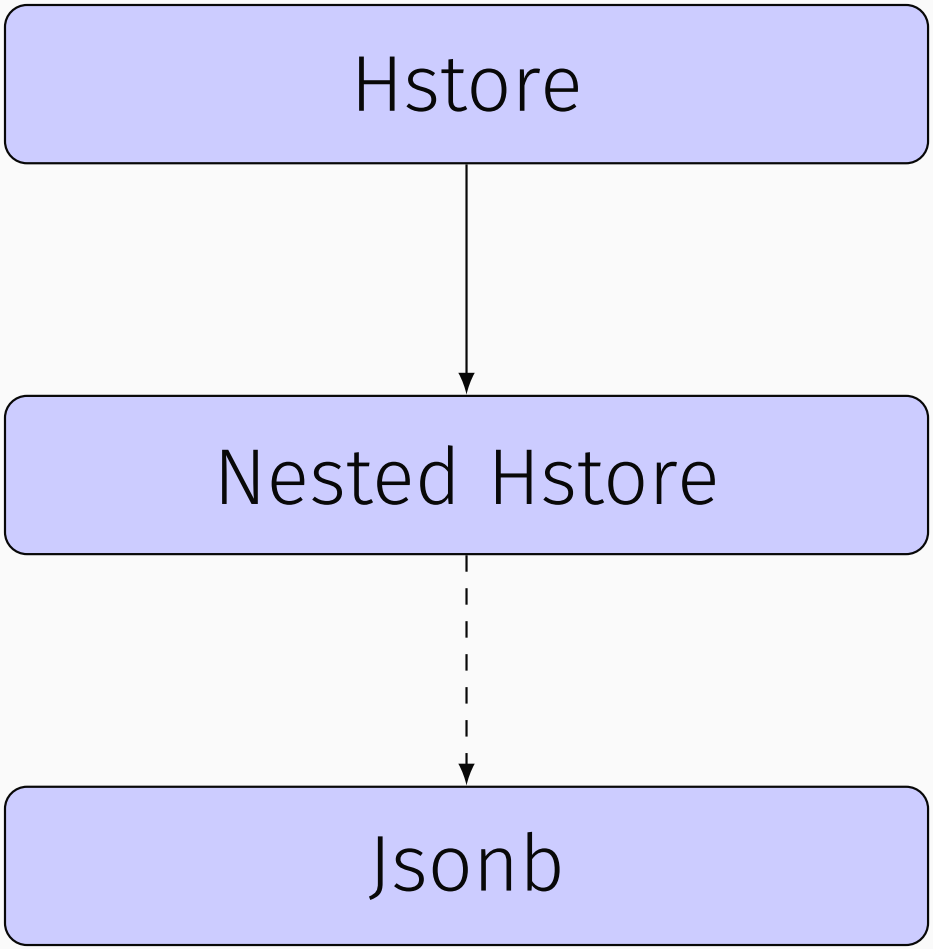
ВСЕ, ЧТО ВЫ ХОТЕЛИ ЗНАТЬ О JSONB, НО БОЯЛИСЬ СПРОСИТЬ.

Справочник по синтаксису.

Дмитрий Долгов

October 1, 2015

HISTORY



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;
```

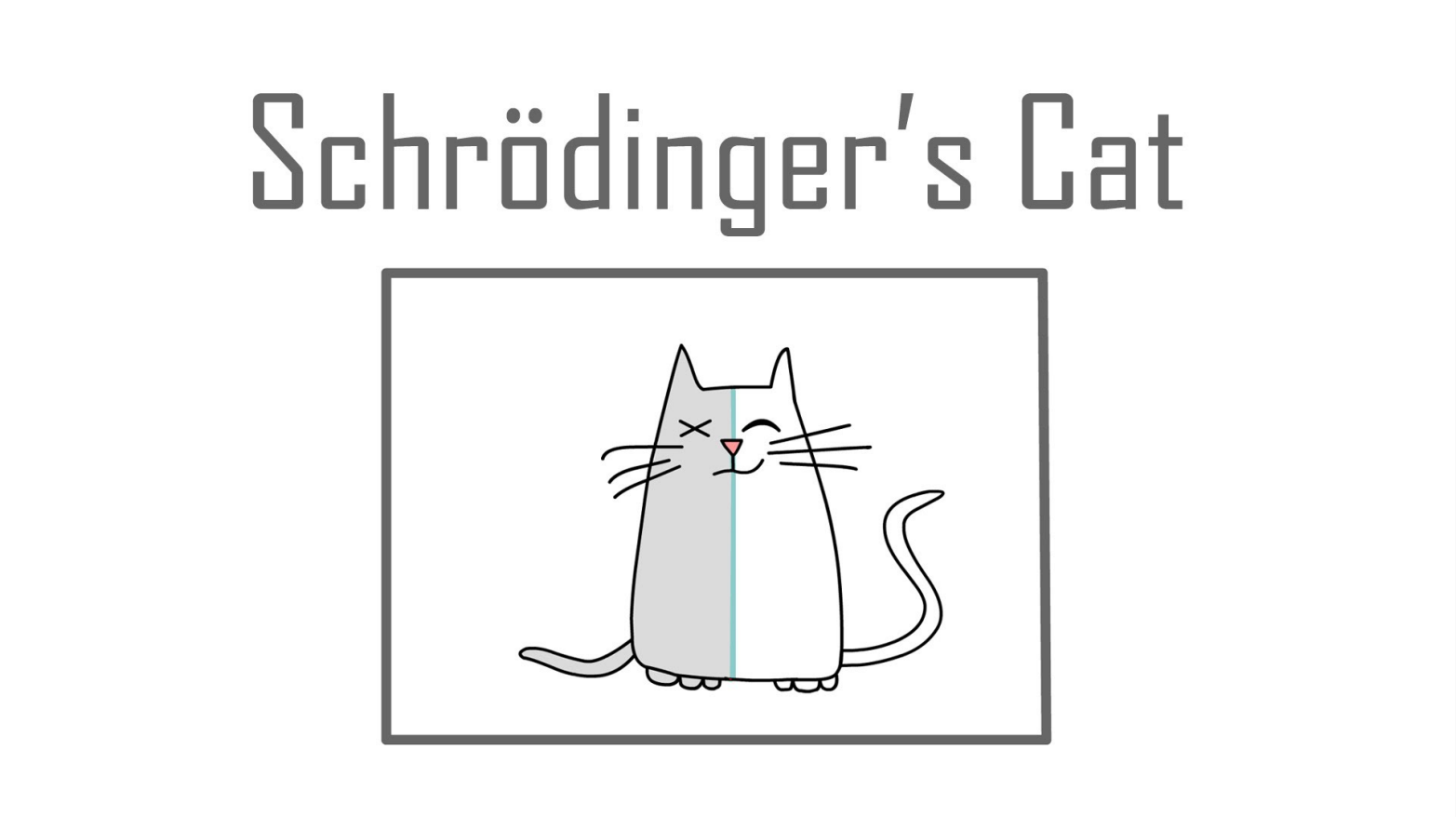
```
select '[1, 2, 3]'::jsonb;
```

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

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

JSONB IN 9.4

Is jsonb fully functional in 9.4?



Yes and no.

LACK OF SOME FUNCTIONALITY

- 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?