UPDATE AND DELETE OPERATIONS FOR JSONB

providing some needed functions and operators for jsonb

Andrew Dunstan, Dmitry Dolgov
June 3, 2015

INTRODUCTION

HSTORE

Hstore is a key-value binary storage. Doesn't support tree-like nested structures, but there is a nested version of hstore.

JSONB

Binary JSON storage. JSONB was introduced in PostgreSQL 9.4 and supports fast lookups and simple expression search queries using Generalized Inverted Indexes (GIN). It supposed to be document-oriented and was designed for the schema-less data.

LACK OF SOME JSONB FUNCTIONALITY

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

NEW FEATURES

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

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)
```


JSONB_SET: REPLACE

jsonb_set(jsonb, text[ ], jsonb)

```
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

jsonb_set(jsonb, text[ ], jsonb)

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

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


PATH FORMAT FOR JSONB UPDATE FUNCTIONS

- Path is an array of element for each nesting level
- Each element is a key (if target is an object) or index (if target is an array)
- Negative idx value is supported (countdown from the last key/element)

PATH FORMAT FOR JSONB UPDATE FUNCTIONS
- Path is an array of element for each nesting level
- Each element is a key (if target is an object) or index (if target is an array)
- Negative idx value is supported (countdown from the last key/element)

probably jsquery can be more useful for this

PATH FORMAT: EXAMPLES

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

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


PATH FORMAT: EXAMPLES

```
select jsonb_set(
    '{"a":{"b": [1, 2, 3]}}'::jsonb,
    '{a, b, -1}'
    '4'
);

             jsonb_set
-------------------------------------
{"a": {"b": [1, 2, 4]}}
 (1 row)
```


JSONB_DELETE

jsonb_delete(jsonb, text)

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

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

jsonb_delete(jsonb, text) allow to delete only top level key from object

JSONB_DELETE

jsonb_delete(jsonb, int)

```
select jsonb_delete(
    '["a", "b", "c"]'::jsonb,
    2
);

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


JSONB_DELETE

jsonb_delete(jsonb, text[])

```
select jsonb_delete(
    '{"a":1 , "b":{"f": [2, 3, 4] "c":5}'::jsonb,
    {b, f, -1}
);

            jsonb_delete
---------------------------------------
 {"a":1, "b": {"f": [2, 3]}, "c": 5}
(1 row)
```

JSONB_DELETE

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

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


JSONB_CONCAT

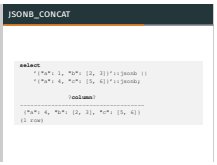
jsonb_concat(jsonb, jsonb)

(aka "shallow concatenation")

```
select jsonb_concat(
    '{"a": 1, "b": [2, 3]}'::jsonb,
    '{"a": 4, "c": [5, 6]}'::jsonb
);
             jsonb_concat
-----------------------------------
 {"a": 4, "b": [2, 3], "c": [5, 6]}
(1 row)
```


JSONB_CONCAT

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

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



This operator is questionable, and I'm not sure how it will be in 9.5, but I think it will be present in this form in jsonbx extension

PLANS FOR THE FUTURE

GENERAL THOUGHTS

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

jsonb_delete(jsonb, jsonb)

```
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

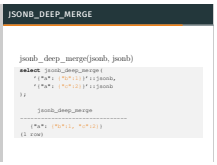
jsonb_intersection(jsonb, jsonb)

```
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

jsonb_deep_merge(jsonb, jsonb)

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

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



Form of this function is questionable, but I think it will be present in this form in jsonbx extension

SHORTLIST

- Operations with keys and values?
- Integration with Jsquery?
- More elegant syntax?

CONCLUSION

SUMMARY

Jsonbx extension provide minimum required functionality for the purpose of updating JSONB in PostgreSQL 9.4
Corresponding patch provide the same amount of new functions for PostgreSQL 9.5
Work is in progress, and development of jsonb will be continued (for 9.4 and 9.5 separately)

QUESTIONS?