One step forward true json data type. Nested hstore with arrays support

Oleg Bartunov, Teodor Sigaev
Moscow University, MEPhI

Hstore developers

- Teodor Sigaev, Oleg Bartunov
- Sternberg Astronomical Institute of Moscow University, MEPHI
- Major contributions:
  - PostgreSQL extendability: GiST, GIN, SP-GiST
  - Full-text search, ltree, pg_trgm, hstore, intarray,..

Agenda

- Introduction to hstore
- History of hstore development
- Hstore internals
- Limitations
- Hstore operators and functions
- Performance study
- Summary
- Development plans

Introduction to hstore

- Hstore — key/value storage (inspired by perl hash)

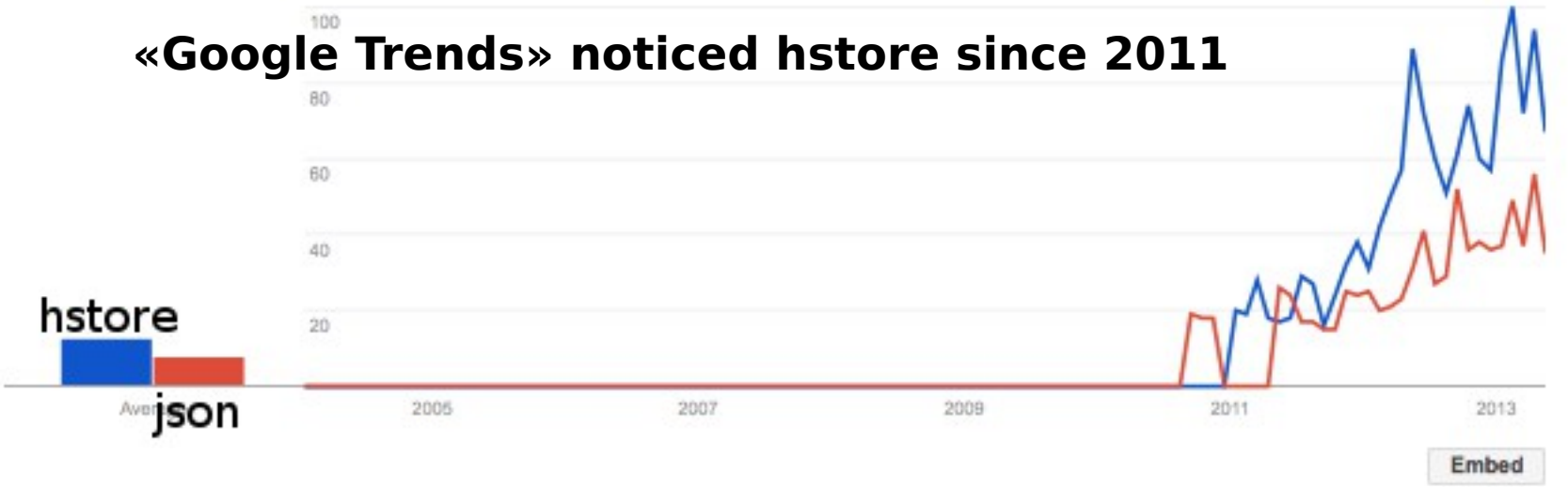
```
'a=>1, b=>2'::hstore
```

  - Key, value — strings
  - Get value for key: hstore -> text
  - Operators with index support (GiST, GIN)

    Check for key: hstore ? text
    Contains: hstore @> hstore
    ……..check documentations for more ……..
  - Functions for hstore manipulations (akeys, avals, skeys, svals, each,……)

Introduction to hstore

«Google Trends» noticed hstore since 2011



History of hstore development

- May 16, 2003 — first version of hstore

```
Date: Fri, 16 May 2003 22:56:14 +0400
From: Teodor Sigaev <teodor@sigaev.ru>
To: Oleg Bartunov <oleg@sai.msu.su>, Alexey Slynko <slynko@tronet.ru>
Cc: E.Rodichev <er@sai.msu.su>
Subject: hash type (hstore)

Готова первайа версия:
zeus:~teodor/hstore.tgz

README написать не успел, поэтому здесь:
1 i/o типа hstore
2 операция hstore->text - извлечение значения по ключу text
select 'a=>q, b=>g'->'a';
 ?
------
 q

3 isexists(hstore), isdefined(hstore), delete(hstore,text) - полный перловый аналог
4 hstore || hstore - конкатенация, аналог в перле %a=( %b, %c );
5 text=>text - возвращает hstore
select 'a'=>'b';
 ?column?
----------
 "a"=>"b"


Все примеры есть в sql/hstore.sql
```

Introduction to hstore

- Hstore benefits
  - Flexible model for storing a semi-structured data in relational database
- Hstore drawbacks
  - Too simple model !
    Hstore key-value model doesn't supports tree-like structures as json (introduced in 2006, 3 years after hstore)

hstore vs json

- PostgreSQL already has json since 9.0, which supports document-based model, but
  - It's slow, since it has no binary representation and needs to be parsed every time
  - Hstore is fast, thanks to binary representation and index support
  - It's possible to convert hstore to json and vice versa, but current hstore is limited to key-value
  - Need hstore with document-based model. Share it's binary representation with json !

History of hstore development

- May 16, 2003 - first (unpublished) version of hstore for PostgreSQL 7.3
- Dec, 05, 2006 - hstore is a part of PostgreSQL 8.2 (thanks, Hubert Depesz Lubaczewski!)
- May 23, 2007 - GIN index for hstore, PostgreSQL 8.3
- Sep, 20, 2010 - Andrew Gierth improved hstore, PostgreSQL 9.0
- May 24, 2013 - Nested hstore with array support, key->value model → document-based model
  PostgreSQL 9.4(?)

Hstore syntax

- Hash-like:

'a=>1' '{a=>1}'
'a=>b, b=>c' '{a=>b, b=>"c"}'

- Array-like:

'a' '{a}' '[a]'
'a,b' '{a,b}' '[a,b]'

- '"a=>b"' — array or hash?

Hstore syntax

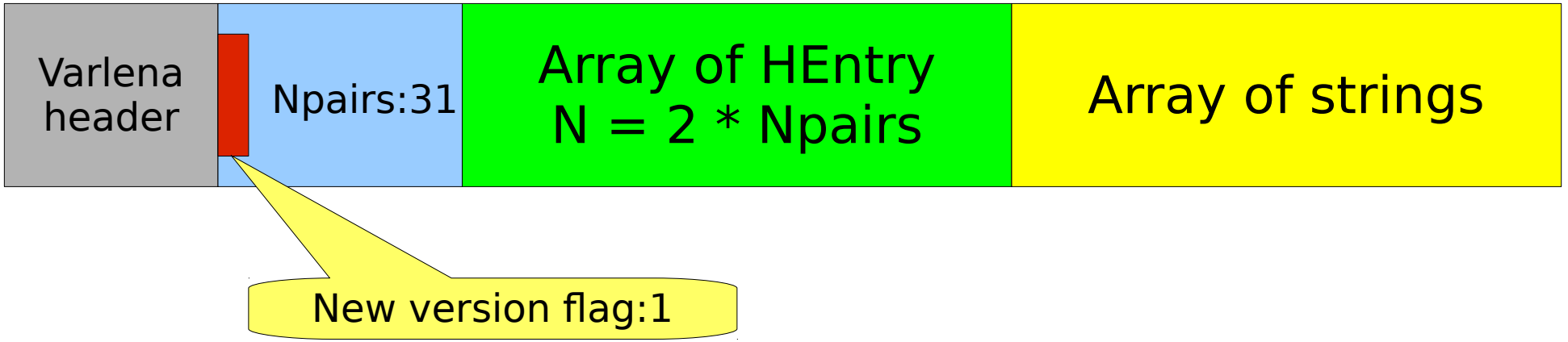
- Combination of hash-like and array-like

  '{a=>1}, {1,2,3}, {c=>{d,f}}'

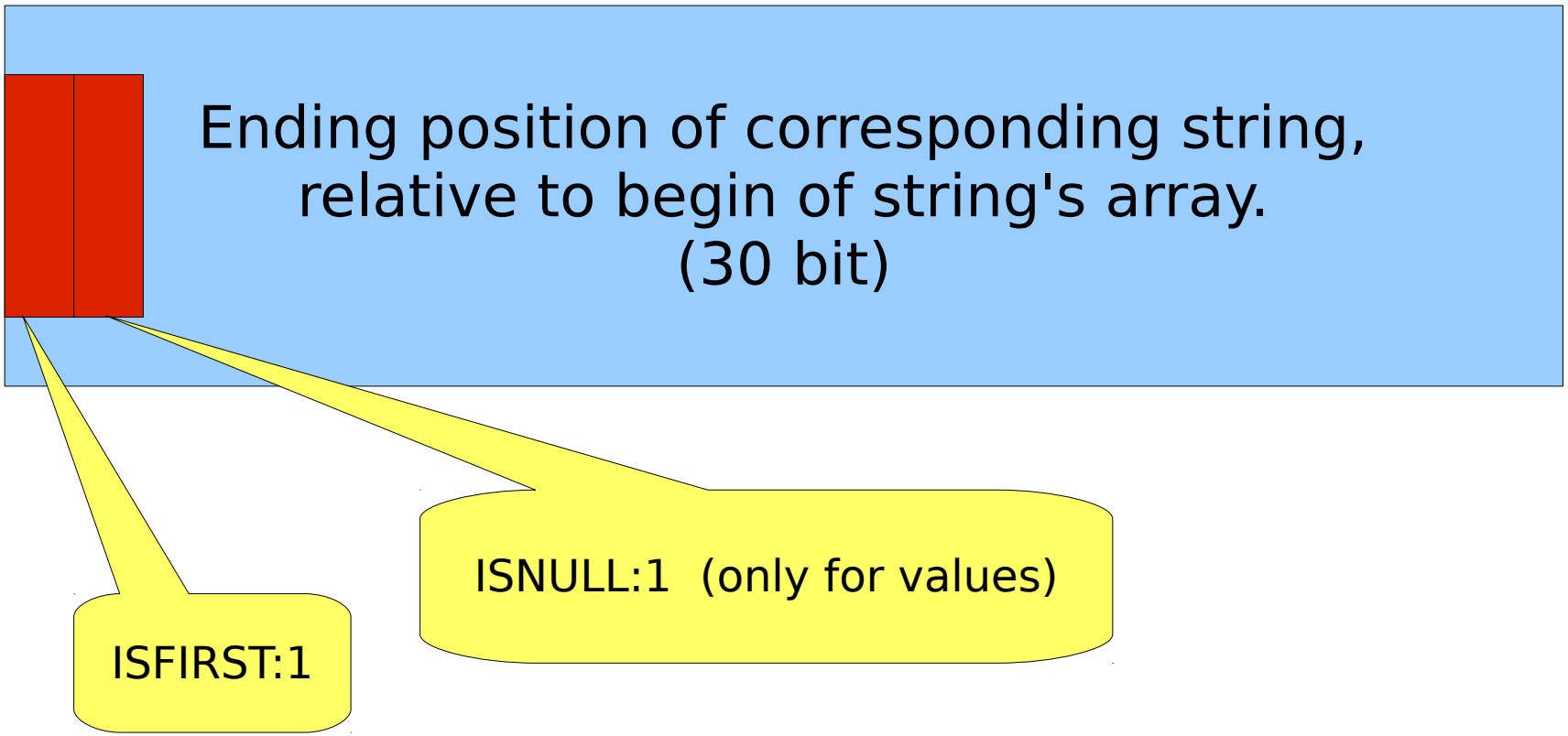
- Nested hstore always requires brackets/braces

  'a=>1,c=>{b=>2}'
  'a=>1,c=>[b,2]'
  'a=>1,c=>{b,2}'

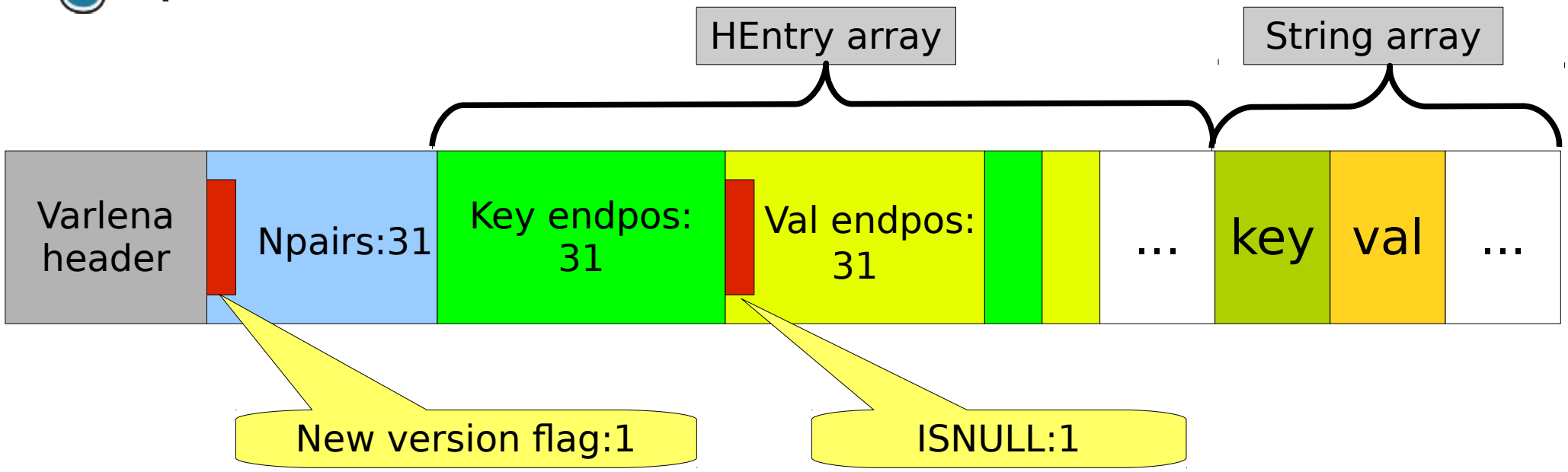
Current: HStore's internals



Current: HEntry



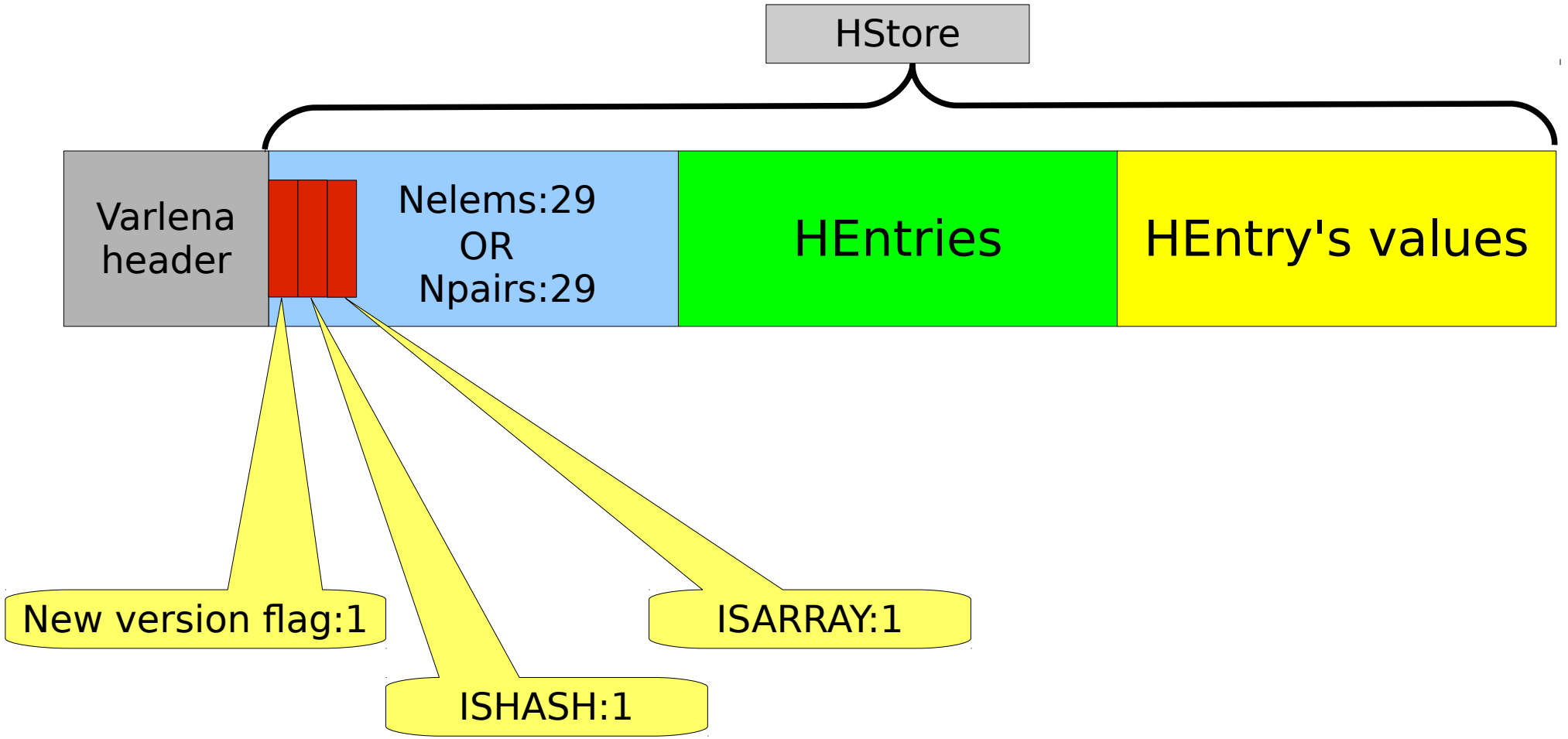
Current: Summary



	Start	End
First key	0	HEntry[0]
i-th key	HEntry[i*2 - 1]	HEntry[i*2]
i-th value	HEntry[i*2]	HEntry[i*2 + 1]

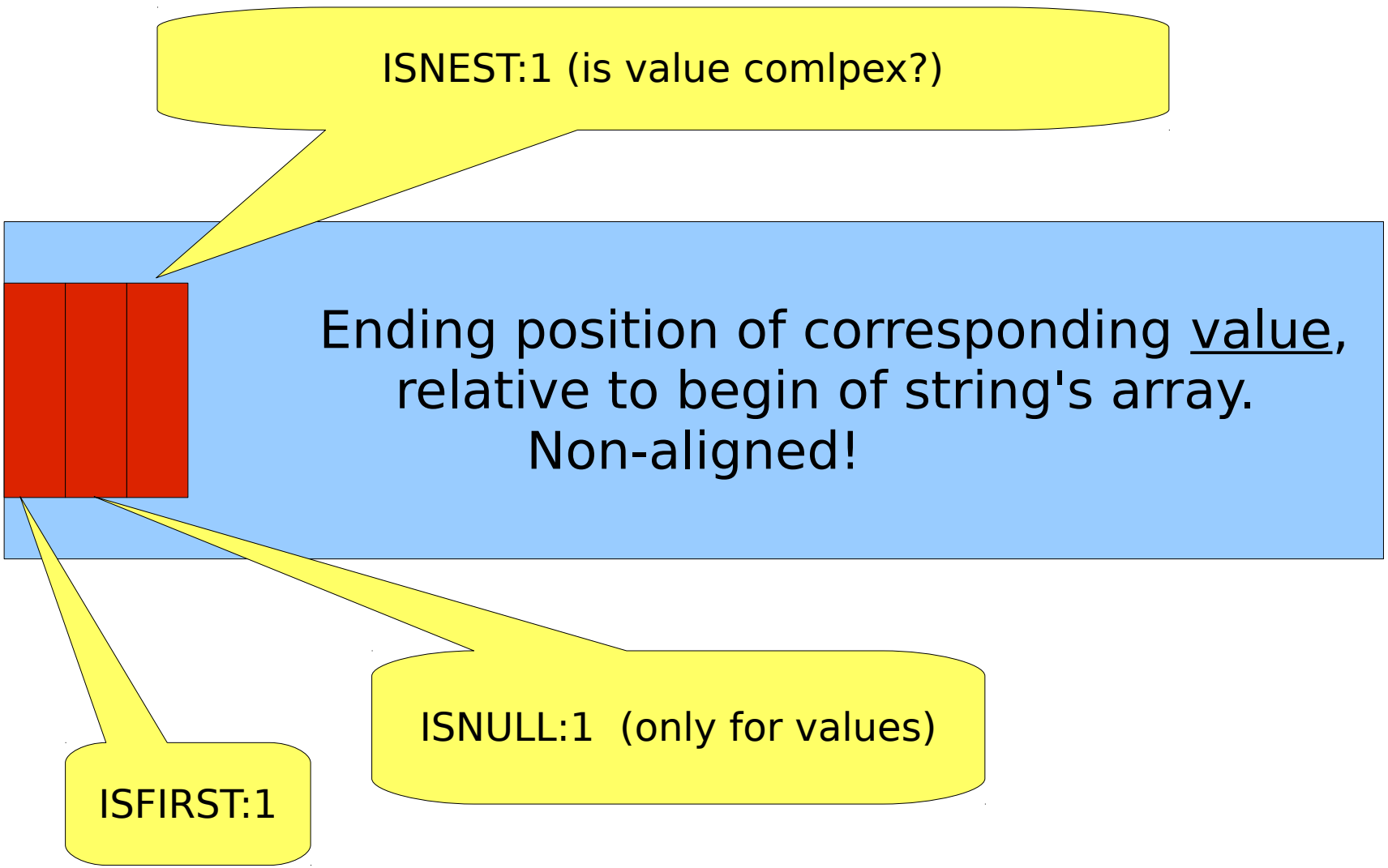
Pairs are lexicographically ordered by key

Nested: Layout

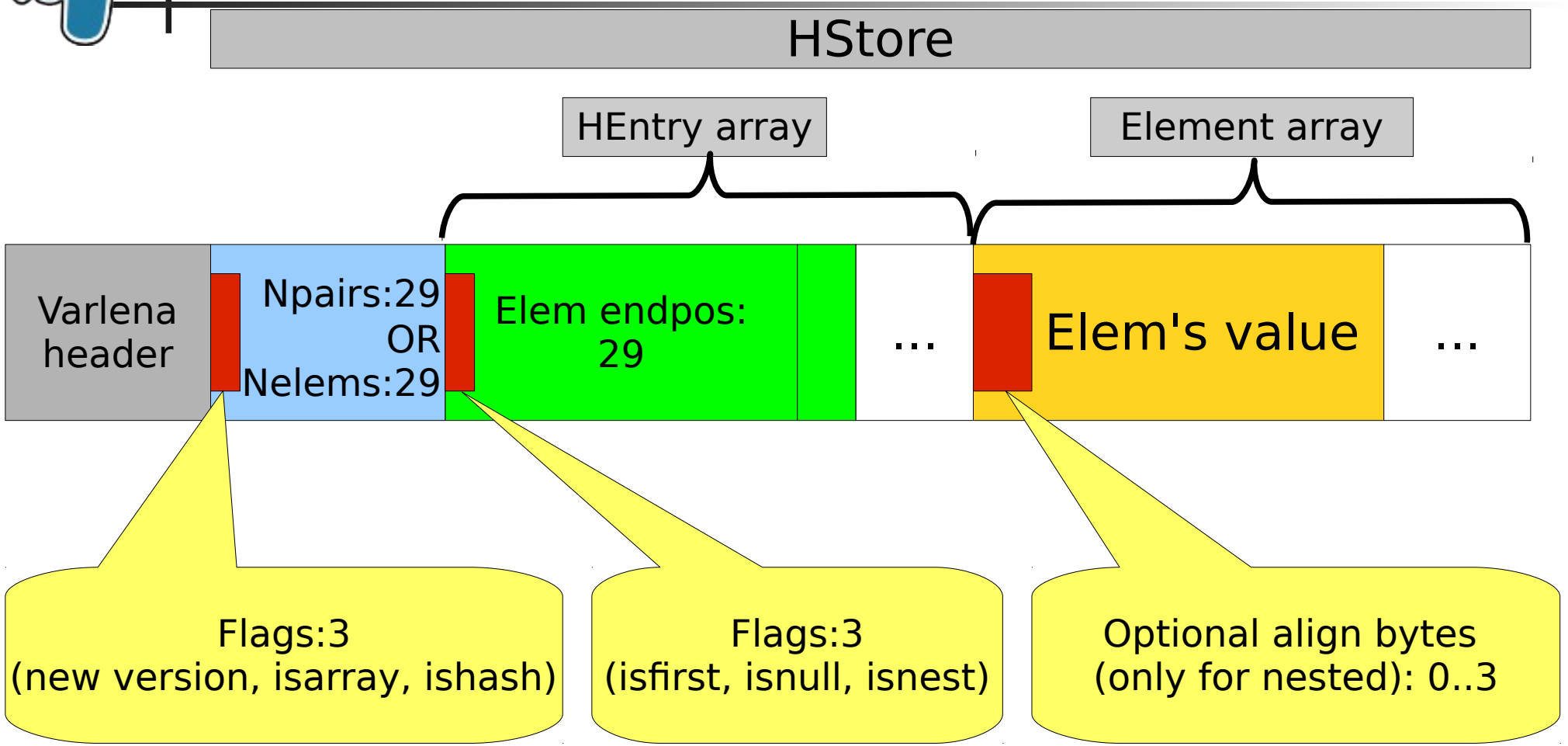


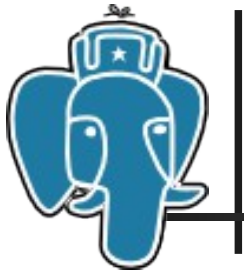
HEntry value could be an hstore itself

Nested: HEntry



Nested: Summary





Nested: Access

For complex value start = INTALING(start)

HASH	Start	End
First key	0	HEntry[0]
i-th key	HEntry[i*2 - 1]	HEntry[i*2]
i-th value	align(HEntry[i*2])	HEntry[i*2 + 1]

Pairs are lexicographically ordered by key

ARRAY	Start	End
First elem	0	HEntry[0]
i-th elem	align(HEntry[i - 1])	HEntry[i]

Elements are not ordered

Hstore limitations

- Levels: unlimited
- Number of elements in array: 2^29
- Number of pairs in hash: 2^29
- Length of string: 2^29 bytes
- Length of nested hash or array: 2^29 bytes

2^29 bytes = 512 MB

Compatibility

- HStore as type is absolutely [pg_]upgrade-friendly
  (ISHASH bit could be set automatically,
  current version will always contains zeros)
- It's also true for GIN indexes: instead of KV notation it uses KVE
- It's not true for GiST: old version doesn't uses KV notation, now it uses KVE. Indexes should be recreated.

Hstore syntax cont.

```
=# select '{a=>1}, {1,2,3}, {c=>{d,f}}'::hstore;
                    hstore
-----------------------------------------------
 {{"a"=>"1"}, {"1", "2", "3"}, {"c"=>{"d", "f"}}}
```

- hstore.array_square_brackets [false],true

```
=# set hstore.array_square_brackets=true;
=# select '{a=>1}, {1,2,3}, {c=>{d,f}}'::hstore;
                    hstore
-----------------------------------------------
 [{"a"=>"1"}, ["1", "2", "3"], {"c"=>["d", "f"]}]
```

Hstore syntax cont.

- hstore.root_array_decorated [true],false

```
=# set hstore.root_array_decorated=false;
    postgres=# select '{a=>1}, {1,2,3}, {c=>{d,f}}'::hstore;
                        hstore
    ------------------------------------------------
    {"a"=>"1"}, ["1", "2", "3"], {"c"=>["d", "f"]}
```

- hstore.root_hash_decorated true,[false]

```
=# set hstore.root_hash_decorated=true;
postgres=# select 'a=>1'::hstore;
   hstore
-----------
 {"a"=>"1"}
```

Hstore syntax cont.

```
=# set hstore.pretty_print=true;
=# select '{a=>1}, {1,2,3}, {c=>{d,f}}'::hstore;
      hstore
-------------------
 {                 +
     {             +
         "a"=>"1"  +
     },            +
     {             +
         "1",      +
         "2",      +
         "3"       +
     },            +
     {             +
         "c"=>     +
         {         +
             "d",  +
             "f"   +
         }         +
     }             +
 }
(1 row)
```

Operators and functions

- Get value by key
  - text hstore -> text

```
=# select 'a=>1,b=>{c=>3,d=>{4,5,6}},1=>f'::hstore -> 'b';
          ?column?
-----------------------------------
 "c"=>"3", "d"=>{"4", "5", "6"}
```

  - hstore hstore %> text

```
=# select 'a=>1,b=>{c=>3,d=>{4,5,6}},1=>f'::hstore %> 'a';
 ?column?
----------
 {"1"}
```

Operators and functions

- Get value by path
  - text hstore #> path

```
=# select 'a=>1,b=>{c=>3,d=>{4,5,6}},1=>f'::hstore #> '{b,d,0}';
?column?
----------
 4
```

  - hstore hstore #%> path

```
=# select 'a=>1,b=>{c=>3,d=>{4,5,6}},1=>f'::hstore #%>'{b,d}';
     ?column?
-----------------
 {"4", "5", "6"}
```

Operators and functions

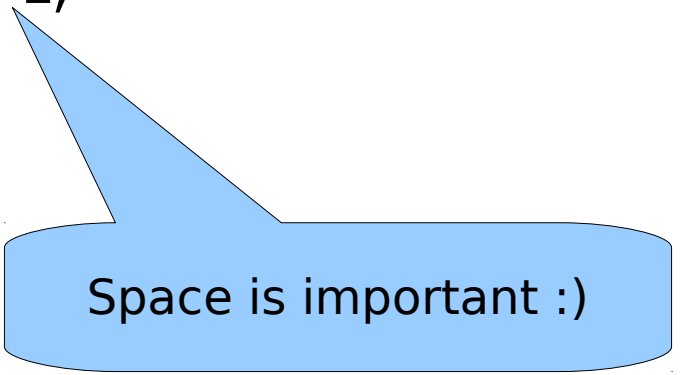
- Get array element by index
  - text hstore->integer

```
=# select '{a,b,3,4,5}'::hstore->1;
 ?column?
----------
 b                                  – negative index starts from the end
=# select '{a,b,3,4,5}'::hstore-> -2;
 ?column?
----------
 4
```

Operators and functions

- Get array element by index
  - hstore hstore%>integer

```
=# select '{a,b,3,4,5}'::hstore%>1;
 ?column?
----------
 {"b"}                                  – negative index starts from the end
=# select '{a,b,3,4,5}'::hstore%> -2;
 ?column?
----------
 {"4"}
```



Operators and functions

- Chaining operators to go deep

```
=# select 'a=>1,b=>{c=>3,d=>{4,5,6}},1=>f'::hstore %> 'b'->'c';
 ?column?
----------
 3
=# select 'a=>1,b=>{c=>3,d=>{4,5,6}},1=>f'::hstore #%> '{b,d}'->0;
 ?column?
----------
 4
```

Operators and functions

- hstore hstore || hstore

```
=# select 'a=>1,b=>{c=>3,d=>{4,5,6}}'::hstore || 'b=>{c=>4}'::hstore;
          ?column?
--------------------------
 "a"=>"1", "b"=>{"c"=>"4"}
```

- Concatenation with path

  hstore concat_path(hstore,text[],hstore)

```
=# select concat_path('a=>1,b=>{c=>3,d=>{4,5,6}}'::hstore,'{b,d}', '1');
                    concat_path
-----------------------------------------------
 "a"=>"1", "b"=>{"c"=>"3", "d"=>{"4", "5", "6", "1"}}
```

Operators and functions

- Concatenation with path

  hstore concat_path(hstore,text[],hstore)

  With empty path it works exactly as old || operator

```
=# select concat_path('a=>1,b=>{c=>3,d=>{4,5,6}}'::hstore,'{}', 'a=>2');
                    concat_path
-----------------------------------------------------
 "a"=>"2", "b"=>{"c"=>"3", "d"=>{"4", "5", "6"}}
```

Operators and functions

- Contains operators @>, <@ goes deep

```
=# SELECT 'a=>{1,2,{c=>3, x=>4}}, c=>b'::hstore @> 'a=>{{c=>3}}';
?column?
----------
 t
=# SELECT 'a=>{{c=>3}}' <@ 'a=>{1,2,{c=>3, x=>4}}, c=>b'::hstore;
?column?
----------
 t
```

Operators and functions

- setof hstore hvals(hstore)

```
=# SELECT *  FROM
     hvals('{{tags=>1, sh=>2}, {tags=>3, sh=>4}}'::hstore) AS q;
          q
----------------------
 "sh"=>"2", "tags"=>"1"
 "sh"=>"4", "tags"=>"3"
=# SELECT q->'tags'  FROM
     hvals('{{tags=>1, sh=>2}, {tags=>3, sh=>4}}'::hstore) AS q;
 ?column?
----------
    1
    3
```

Operators and functions

- setof hstore hvals(hstore, text[])

```
=# SELECT *  FROM
     hvals('{{tags=>1, sh=>2}, {tags=>3,sh=>4}}'::hstore,'{null,tags}');
 hvals
-------
 {"1"}
 {"3"}
```

- setof text svals(hstore,text[])

Operators and functions

- Replace with path
  hstore replace(hstore,text[],hstore)

```
=# select replace('a=>1,b=>{c=>3,d=>{4,5,6}}'::hstore,'{b,d}', '1');
                 replace
-----------------------------------------------
 "a"=>"1", "b"=>{"c"=>"3", "d"=>{"1"}}
```

Operators and functions

- hstore <-> json conversion
  - json hstore_to_json(hstore)

```
=# select hstore_to_json('a=>1,b=>{c=>3,d=>{4,5,6}}'::hstore);
                  hstore_to_json
-----------------------------------------------
 {"a": "1", "b": {"c": "3", "d": ["4", "5", "6"]}}
```

  - hstore json_to_hstore(json)

```
=# select json_to_hstore('{"a": "1", "b": {"c": "3", "d": ["4", "5",
"6"]}}'::json);
                  json_to_hstore
-----------------------------------------------
 "a"=>"1", "b"=>{"c"=>"3", "d"=>["4", "5", "6"]}
```

Operators and functions

- hstore <-> json cast
  - hstore::json

```
=# select 'a=>1'::hstore::json;
    json
------------
 {"a": "1"}
```

  - json::hstore

```
=# select '{"a": "1"}'::json::hstore;
 hstore
----------
 "a"=>"1"
```

Operators and functions

- hstore <-> json cast
  - Hstore has no types support as json, so :(

```
=# select '{"a":3.14}'::json::hstore::json;
      json
--------------
 {"a": "3.14"}

=# select '3.14'::json::hstore::json;
   json
----------
 ["3.14"]
```

Operators and functions

```
=# set hstore.pretty_print=true;
=# select hstore_to_json('{a=>1}, {1,2,3}, {c=>{d,f}}'::hstore);
  hstore_to_json
--------------------
 [                  +
     {              +
         "a": "1"   +
     },             +
     [              +
         "1",       +
         "2",       +
         "3"        +
     ],             +
     {              +
         "c":       +
         [          +
             "d",   +
             "f"    +
         ]          +
     }              +
 ]
(1 row)
```

Operators and functions

- There are more operators and functions available !

Performance

- Data
  - 1,252,973 bookmarks from Delicious in json format
  - The same bookmarks in hstore format
  - The same bookmarks as text
- Server
  - desktop Linux, 8 GB RAM, 4-cores Xeon 3.2 GHz,
- Test
  - Input performance - copy data to table
  - Access performance - get value by key
  - Search performance contains @> operator

Performance

- Data
  - 1,252,973 bookmarks from Delicious in json format
  - The same bookmarks in hstore format
  - The same bookmarks as text

```
=# \dt+
                      List of relations
 Schema | Name | Type  |  Owner   |  Size   | Description
--------+------+-------+----------+---------+-------------
 public | hs   | table | postgres | 1379 MB |
 public | js   | table | postgres | 1322 MB |
 public | tx   | table | postgres | 1322 MB |
```

Performance

```
=# select h  from hs limit 1;
                                              h
-----------------------------------------------------------------------------------------------
     "id"=>"http://delicious.com/url/b5b3cbf9a9176fe43c27d7b4af94a422#mcasas1",             +
     "link"=>"http://www.theatermania.com/broadway/",                                       +
     "tags"=>                                                                               +
     {                                                                                      +
         {                                                                                  +
             "term"=>"NYC",                                                                 +
             "label"=>NULL,                                                                 +
             "scheme"=>"http://delicious.com/mcasas1/"                                      +
         },                                                                                 +
         {                                                                                  +
             "term"=>"english",                                                             +
             "label"=>NULL,                                                                 +
             "scheme"=>"http://delicious.com/mcacas1/"                                      +
         },                                                                                 +
     },                                                                                     +
     "links"=>                                                                              +
     {                                                                                      +
         {                                                                                  +
             "rel"=>"alternate",                                                            +
             "href"=>"http://www.theatermania.com/broadway/",                               +
             "type"=>"text/html"                                                            +
         }                                                                                  +
     },                                                                                     +
     "title"=>"TheaterMania",                                                               +
     "author"=>"mcasas1",                                                                   +
     "source"=>NULL,                                                                        +
     "updated"=>"Tue, 08 Sep 2009 23:28:55 +0000",                                          +
     "comments"=>"http://delicious.com/url/b5b3cbf9a9176fe43c27d7b4af94a422",               +
     "guidislink"=>"false",                                                                 +
     "title_detail"=>                                                                       +
     {                                                                                      +
         "base"=>"http://feeds.delicious.com/v2/rss/recent?min=1&count=100",                +
         "type"=>"text/plain",                                                              +
         "value"=>"TheaterMania",                                                           +
         "language"=>NULL                                                                   +
     },                                                                                     +
     "wfw_commentrss"=>"http://feeds.delicious.com/v2/rss/url/b5b3cbf9a9176fe43c27d7b4af94a422"+
```

Performance

- Input performance
  - Copy data (1,252,973 rows) as text, json,hstore

```
copy tt from  '/path/to/test.dump'
```

Text: 57 s

Json: 61 s

Hstore: 76 s – there is some room to speedup

Performance

- Access performance — get value by key
  - Base: select h from hs;
  - Hstore: select h->'updated' from hs;
  - Json: select j->>'updated' from js;
  - Regexp: select (regexp_matches(t, '"updated":"([^"]*)"'))[1] from tx;

  Base: 0.3 s

  hstore: 0.5 s

  Json: 11. s

  regexp: 18.8 s

Performance

- Access performance — get value by key

  Base: 0.3 s
  hstore: 0.5 s
  Json: 11. s
  regexp: 18.8 s

- Hstore is ~ 50x faster json
  thanks to binary representation !

Performance

- Search performance — contains @> operator
  - Hstore - seqscan, GiST, GIN

```
select count(*) from hs where h @> 'tags=>{{term=>NYC}}';
```

  - Json — estimation, GiST, GIN (functional indexes)
    exact time > estimation (there are may be many tags)

```
select count(*) from js where j#>>'{tags,0,term}' = 'NYC';
```

Performance

- Search performance — contains @> operator
  - Hstore - seqscan, GiST, GIN

    100s 400s - create index

    64MB 815MB

    0.98s 0.3s 0.1s

    3x 10x

  - Json — estimation, GiST, GIN (functional indexes)

    130s 500s - create index

    12s 2s 0.1s

    6x 120x

Recheck (GiST) calls json_to_hstore()

Summary

- Hstore is now nested and supports arrays

  Document-based model !
- Hstore access to specified field is fast (thanks to binary representation)
- Hstore operators can use GiST and GIN indexes
- Json users can use functional GIN index and get considerable speedup
- Hstore's binary representation can be used by json

Development plans

- Speedup hstore input
- Hstore query language - hpath, hquery ?
- Better indexing - SP-GiST-GIN hybrid index
- Statistics support (challenging task)
- Types support (?)
- Documentation
- Submit patch for 9.4
- Add binary representation to json
- Add index support for json

Availability

- Patch to master branch is available

http://www.sigaev.ru/misc/nested_hstore-0.15.patch.gz

Thanks !