Олег Бартунов, Александр Коротков

Расширяемость PostgreSQL для хакеров и архитекторов

Российские интернет-технологии

IT-цикл проекта

- Проектирование архитектуры
- Разработка
- Поддержка
- Развитие
  - Увеличение нагрузки, новая функциональность

IT-цикл проекта

- Архитектура системы должна быть расширяемой
- Расширяемость компонентов системы
- Костыли — средство расширяемости

- Костыль — средство добавления недостающей функциональности или исправления серьёзных дыр без должного редизайна системы
- Систем без «костылей» не бывает

Костыли имеют тенденцию накапливаться

Российские интернет-технологии

Система все-таки работает ...

Российские интернет-технологии

Поменять СУБД очень тяжело !

- Frontend-cache-backend-application-cache-файловое хранилище-база данных
- Смена СУБД — мучительный процесс, никакой автомагии
- Выбор СУБД на стадии разработки — важнейшая тема ! 99.99% проектов не требует разработки своей СУБД.

Выбор СУБД

- Архитектор — функциональность, производительность, доступность
- Хакер-дба — удобство, привычка, религия
- Бизнес-маркетинг — сторонние факторы

Выбор СУБД

- **Расширяемость** СУБД — важнейший фактор вашего развития и независимости !
- Расширяемость:
  - Новая функциональность (типы данных, операции)
  - Своими руками
  - Без остановки сервера
  - Без ущерба производительности и надежности

PostgreSQL

- Абсолютно доступная (BSD, нет компании-владельца)
- Хороший научный бэкграунд и история
- Много успешных коммерческих форков (Greenplum, AsterData, JustOneDB, Hadapt...)
- Отличная функциональность и производительность
- Хорошее соответствие стандартам ISO/ANSI SQL 92,99,2003
- Большое и дружественное сообщество разработчиков и пользователей
- Российские разработчики (расширяемость)

PostgreSQL

- Skype - шкалируется до миллиарда польз.
- Hi5.com — 60 млн. пользователей, #8 Alexa traffic rank
- MyYearBook.com — 18,000 req/sec, 300 Gb database
- NASA — обработка спутниковых данных (MODIS)
- Instagram — x100 млн картинок
- Sony (Free Realms) — 10 млн игроков
- Heroku , EngineYard, Saleforce ?

PostgreSQL

- Рамблер
- 1С:Предприятие
- MirTesen, MoiKrug.ru (Yandex)
- Avito.ru — 2000 req/sec
- IRR.ru («Из рук в руки»)
- rabota.ru, price.ru, РБК, МастерХост
- Военные — версия 7.X входит в МСВС, 9.0 - Заря
- Национальная СУБД ?

Расширяемость PostgreSQL

- Функции, операторы, типы данных
- Языки (sql,pl/pgsql,pl/perl,pl/python, pl/tcl, pl/R, pl/java …,pl/v8). Pl/psql от EnterpriseDB!
- Индексный доступ (Btree, Hash, GiST, GIN, SP-GiST)
- Внешние таблицы (oracle, mysql, informix, couchdb, redis, mongodb, twitter, www …)

Индексные методы доступа

- Индекс — это поисковое дерево, в листьях которого содержатся указатели на записи в таблице
- Индекс не содержит информации о видимости записи (MVCC !)
- Индексы только ускоряют выполнение запроса (операторы,операнды)
- Результаты выборки с использованием индекса должны совпадать с последовательным сканом и фильтрацией
- Индексы могут быть **partial** (where price > 0.0), **functional** (to_tsvector(text)), **multicolumn** (timestamp, tsvector)
- Индексные методы доступа (pg_catalog.pg_am) - btree,hash, gist, gin, spgist

Индексные методы доступа

- Btree — операции сравнения
- Hash — равенство
- GiST, GIN, SP-GiST — произвольные операции (похожесть, пересечение множеств, включение,....)

Индексные методы доступа

- +Методы навигации по дереву
- +Добавление и вставка в дерево
- +Конкурентность, восстанавливаемость
- ?Интерфейсы для описания работы с данными (поиск и вставка)

Расширяемость PostgreSQL

- Новая функциональность добавляется **без остановки сервера**
- Не требуются «магические» знания core-девелоперов
- Новые типы данных - «first class citizens». Все свойства встроенных типов данных — конкурентность, восстанавливаемость, индексный доступ, версионность
- Примеры: ltree — иерархические данные, hstore — слабо-структурированные данные, pg_trgm — очепятки, postgis — GIS, полнотекстовый поиск, ...

Расширяемость PostgreSQL

Как это работает ?

Разоблачение магии

План

1. Системный каталог
2. Бинарное представление типа на примере HSTORE и JSON
3. Функции
4. Операторы

План

5. Cast’ы
6. Оценки селективности
7. Индексы
8. Extension'ы

Системный каталог

- Все метаданные объектов базы (таблицы, типы данных, индексы, операторы, функции, расширения...) хранятся в системном каталоге (pg_catalog.*)
- 2 интерфейса - SQL (psql -E), функциональный
- Syscache — кэш каталога, используется executor, planner, optimizer
- Bootstrap — специальный режим initdb, создает системный каталог из *.bki файлов

Системный каталог

```
postgres=# \dt pg_catalog.*
                  List of relations
   Schema   |          Name           | Type  |  Owner
------------+-------------------------+-------+----------
 pg_catalog | pg_aggregate            | table | postgres
 pg_catalog | pg_am                   | table | postgres
 pg_catalog | pg_amop                 | table | postgres
 pg_catalog | pg_amproc               | table | postgres
 pg_catalog | pg_attrdef              | table | postgres
 pg_catalog | pg_attribute            | table | postgres
 pg_catalog | pg_auth_members         | table | postgres
 pg_catalog | pg_authid               | table | postgres
 pg_catalog | pg_cast                 | table | postgres
 pg_catalog | pg_class                | table | postgres
 pg_catalog | pg_collation            | table | postgres
 pg_catalog | pg_constraint           | table | postgres
 pg_catalog | pg_conversion           | table | postgres
 pg_catalog | pg_database             | table | postgres
 pg_catalog | pg_db_role_setting      | table | postgres
 ...........................................................
 pg_catalog | pg_user_mapping         | table | postgres
(51 rows)
```

```
postgres=# \d pg_cast
    Table "pg_catalog.pg_cast"
   Column    |  Type  | Modifiers
-------------+--------+-----------
 castsource  | oid    | not null
 casttarget  | oid    | not null
 castfunc    | oid    | not null
 castcontext | "char" | not null
 castmethod  | "char" | not null
Indexes:
    "pg_cast_oid_index" UNIQUE, btree (oid)
    "pg_cast_source_target_index" UNIQUE,
btree (castsource, casttarget)
```

Российские интернет-технологии

Системный каталог

позволяет добавлять новые объекты:

- типы данных
- операторы
- классы операторов
- …

онлайн

Российские интернет-технологии

JSON

- 9.2: native -> json

  array_to_json(), row_to_json()
- 9.3: json -> native

  ->, ->>, #>, #>>, etc

Бинарное представление JSON

```
SELECT '{"a":1,"b":2}'::json;

0000: 7B 22 61 22 | 3A 31 2C 22  {"a":1,"
0008: 62 22 3A 32 | 7D           b":2}
```

Бинарное представление = текстовое!

HSTORE

- Эффективное бинарное представление
- Вложенность, поддержка массивов (WIP)
- JSON – как представление: hstore_to_json()

Бинарное представление HSTORE

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

```
0000: 02 00 00 80 | 01 00 00 80 ☻  A☺  A
0008: 02 00 00 00 | 03 00 00 00 ☻   ♥
0010: 04 00 00 00 | 61 31 62 32 ♦   a1b2
```

Эффективное бинарное представление!

Российские интернет-технологии

Сравнение

```
CREATE TABLE hstore_test AS (SELECT
'a=>1, b=>2, c=>3, d=>4, e=>5'::hstore AS v
FROM generate_series(1,1000000));

CREATE TABLE json_test AS (SELECT
'{"a":1, "b":2, "c":3, "d":4, "e":5}'::json
AS v FROM generate_series(1,1000000));
```

Сравнение

```
SELECT sum((v->'a')::text::int)
FROM json_test;
```

1291,060 ms

```
SELECT sum((v->'a')::int)
FROM hstore_test;
```

303,267 ms

Сравнение

```
"a"=>{"9840", "8818", "1613", "2793",
"6633", "1356", "3578", "6939", "8876",
"8848", "8150", "9889", "5905", "1023",
"6154", "6708", "1430", "9521", "4566",
"6001", "7807", "9140"}
```

Сравнение

```
SELECT SUM((h%>'a'->0)::int)
FROM aa;
```

457.706 ms

```
SELECT SUM((j->'a'->>0)::int)
FROM jj;
```

7462.802 ms

Как это работает?

CREATE TYPE

```
CREATE TYPE hstore (
    INTERNALLENGTH = -1,
    INPUT = hstore_in,
    OUTPUT = hstore_out,
    RECEIVE = hstore_recv,
    SEND = hstore_send,
    STORAGE = extended
);
```

pg_type

typname	hstore
typnamespace	2200
typowner	16384
typlen	-1
typbyval	f
typinput	hstore_in
typoutput	hstore_out
typreceive	hstore_recv
typsend	hstore_send
typstorage	x (extended)
.....	

hstore_in

```
CREATE FUNCTION
hstore_in(cstring)
RETURNS hstore
AS '$libdir/hstore',
'hstore_in'
LANGUAGE C STRICT
IMMUTABLE;
```

```
                pg_proc
----------------+---------------
proname         | hstore_in
pronamespace    | 2200
proowner        | 16384
prolang         | 13
procost         | 1
prorows         | 0
provariadic     | 0
prorettype      | 16385 (hstore)
proargtypes     | 2275  (cstring)
prosrc          | hstore_in
probin          | $libdir/hstore
......
```

Российские интернет-технологии

Type input

```
Datum
hstore_in(PG_FUNCTION_ARGS)
{
   HSParser      state;
   int4          buflen;
   HStore       *out;
......
```

Type input

```
test=# SELECT 'a => 1, b => 2'::hstore;
 hstore
--------------------
 "a"=>"1", "b"=>"2"
```

Как это работает?

Type input

```
test=# SELECT typinput FROM pg_type
       WHERE typname = 'hstore';
 typinput
-----------
 hstore_in
```

Type input

```
test=# SELECT prolang, prosrc, probin
       FROM pg_proc
       WHERE proname = 'hstore_in' AND
             Proargtypes = '2275';
                                (cstring)
 prolang |  prosrc   |     probin
---------+-----------+----------------
      13 | hstore_in | $libdir/hstore
(1 row)
```

Российские интернет-технологии

CREATE OPERATOR

```
CREATE OPERATOR -> (
    LEFTARG = hstore,
    RIGHTARG = text,
    PROCEDURE = fetchval
);
```

```
                pg_operator
-------------+------------
oprname      | ->
oprnamespace | 2200
oprowner     | 16384
oprkind      | b
oprleft      | 16385(hstore)
oprright     | 25    (text)
oprresult    | 25
oprcom       | 0
oprnegate    | 0
oprcode      | fetchval
......
```

Российские интернет-технологии

CREATE FUNCTION

```
CREATE FUNCTION
fetchval(hstore,text)
RETURNS text
AS '$libdir/hstore',
'hstore_fetchval'
LANGUAGE C STRICT
IMMUTABLE;
```

pg_proc

```
---------------+---------------
proname        | fetchval
pronamespace   | 2200
proowner       | 16384
prolang        | 13
procost        | 1
prosrc         | hstore_fetchval
probin         | $libdir/hstore
proconfig      |
proacl         |
......
```

Operator

```
Datum
hstore_fetchval(PG_FUNCTION_ARGS)
{
   HStore  *hs = PG_GETARG_HS(0);
   text    *key = PG_GETARG_TEXT_PP(1);
......
```

Российские интернет-технологии

Operator

```
test=# SELECT ('a => 1, b => 2'::hstore)
        ->'a';
 ?column?
----------
 1
(1 row)
```

Как это работает?

Operator

```
test # SELECT oprcode FROM pg_operator
WHERE oprname = '->' AND oprleft = 16385
AND oprright = 25 ;                    (hstore)
 oprcode       (text)
----------
 fetchval
(1 row)
```

CREATE CAST

```
CREATE CAST
(hstore AS json)
WITH FUNCTION
hstore_to_json(hstore);
```

```
                pg_cast
-------------+------
castsource  | 16385 (hstore)
casttarget  | 114   (json)
castfunc    | 16425 (hstore_to_json)
castcontext | e
castmethod  | f
```

Cast

```
Datum
hstore_to_json(PG_FUNCTION_ARGS)
{
   HStore     *in = PG_GETARG_HS(0);
   int        buflen,
              i;
   int        count = HS_COUNT(in);
......
```

Российские интернет-технологии

Cast

```
test=# SELECT 'a=>1, b=>2'::hstore::json;
          json
-----------------------
 {"a": "1", "b": "2"}
```

Как это работает?

Cast

```
test=# SELECT castfunc
       FROM pg_cast
       WHERE castsource = 16385 (hstore)
       AND casttarget = 114;(json)
 castfunc
----------
    16425
(1 row)
```

Российские интернет-технологии

Cast

```
test=# SELECT prolang, prosrc, probin
       FROM pg_proc
       WHERE oid = 16425; (castfunc)
 prolang |     prosrc      |     probin
---------+------------------+----------------
      13 | hstore_to_json | $libdir/hstore
(1 row)
```

Оценки селективности

```
test= # EXPLAIN SELECT * FROM hstore_test
        WHERE v @> 'a => 1'::hstore;

                                QUERY PLAN

-----------------------------------------------------------
Seq Scan on hstore_test  (cost=0.00..22810.00 rows=1000
width=55)
   Filter: (v @> '"a"=>"1"'::hstore)
```

Откуда это взялось?

Российские интернет-технологии

Оценки селективности

```
test=# SELECT oprrest FROM
pg_operator WHERE oprname = '@>' AND
oprleft = 16385 AND oprright = 16385;
 oprrest
---------
 contsel
(1 row)
```

(hstore) (hstore)

contsel

```
Datum
contsel(PG_FUNCTION_ARGS)
{
    PG_RETURN_FLOAT8(0.001);
}
```

CREATE OPERATOR CLASS

```
CREATE OPERATOR CLASS gin_hstore_ops
DEFAULT FOR TYPE hstore USING gin AS
   OPERATOR        7       @>,
   OPERATOR        9       ?(hstore,text),
   OPERATOR        10      ?|(hstore,text[]),
   OPERATOR        11      ?&(hstore,text[]),
   FUNCTION        1       bttextcmp(text,text),
   FUNCTION        2       gin_extract_hstore(internal, internal),
   FUNCTION        3       gin_extract_hstore_query(internal,
internal, int2, internal, internal),
   FUNCTION        4       gin_consistent_hstore(internal, int2,
internal, int4, internal, internal),
   STORAGE         text;
```

CREATE OPERATOR CLASS

pg_opfamily

```
-------------+---------------
oid          | 16494
opfmethod    | 2742 (gin)
opfname      | gin_hstore_ops
opfnamespace | 2200
opfowner     | 16384
```

pg_opclass

```
-------------+---------------
oid          | 16495
opcmethod    | 2742(gin)
opcname      | gin_hstore_ops
opcnamespace | 2200
opcowner     | 10
opcfamily    | 16494
opcintype    | 16385 (hstore)
opcdefault   | t
opckeytype   | 25    (text)
```

Российские интернет-технологии

CREATE OPERATOR CLASS

pg_amproc

amproclefttype	amprocrighttype	amprocnum	amproc
16385	16385	1	bttextcmp
16385	16385	2	gin_extract_hstore
16385	16385	3	gin_extract_hstore_query
16385	16385	4	gin_consistent_hstore

CREATE OPERATOR CLASS

pg_amop

amopfamily	16494	16494	16494	16494
amoplefttype	16494	16494	16494	16494
amoprighttype	25	1009	1009	16494
amopstrategy	9	10	11	7
amoppurpose	s	s	s	s
amopopr	16417	16399	16401	16403
amopmethod	2742	2742	2742	2742
amopsortfamily	0	0	0	0
	(@>)	(?)	(?|)	(?&)

Российские интернет-технологии

HSTORE index

```
test=# CREATE INDEX hstore_idx ON hstore_test
       USING gin (v);

test=# SELECT * FROM hstore_test
       WHERE v @> 'a=>10'::hstore;
```

HSTORE index

```
 QUERY PLAN
-------------------------------------------------------------------------
Bitmap Heap Scan on hstore_test  (cost=39.75..3049.43 rows=1000
width=57) (actual time=81.436..81.459 rows=100 loops=1)
   Recheck Cond: (v @> '"a"=>"10"'::hstore)
   ->  Bitmap Index Scan on hstore_idx  (cost=0.00..39.50
rows=1000 width=0) (actual time=81.414..81.414 rows=100 loops=1)
         Index Cond: (v @> '"a"=>"10"'::hstore)
 Total runtime: 81.522 ms
```

Российские интернет-технологии

HSTORE index

```
test=# SELECT i.indexrelid
FROM pg_index I
JOIN pg_opclass opc ON opc.oid = i.indclass[0]
JOIN pg_amop amop ON opc.opcfamily = amop.amopfamily
JOIN pg_operator op ON amop.amopopr = op.oid
WHERE indrelid = 16530 AND op.oprname = '@>'
                 (hstore_test)
      AND oprleft = 16385 AND oprright = 16385;
                    (hstore)              (hstore)
indexrelid
------------
      16536
```

Российские интернет-технологии

Expression index

```
test=# CREATE INDEX hstore_idx ON hstore_test
       USING gin (v);

test=# SELECT * FROM hstore_test WHERE (v->'a')::int = 10;

-------------------------------------------------------------
Seq Scan on hstore_test  (cost=0.00..31354.00 rows=5000
width=57) (actual time=1.284..327.407 rows=100 loops=1)
   Filter: (((v -> 'a'::text))::integer = 10)
   Rows Removed by Filter: 999900
 Total runtime: 327.470 ms
```

Российские интернет-технологии

Expression index

```
test=# CREATE INDEX hstore_a_idx
       ON hstore_test (((v->'a')::int));

test=# SELECT * FROM hstore_test WHERE (v->'a')::int = 10;

-------------------------------------------------------------
Index Scan using hstore_a_idx on hstore_test
(cost=0.43..16535.93 rows=5000 width=57) (actual
time=0.018..0.033 rows=100 loops=1)
   Index Cond: (((v -> 'a'::text))::integer = 10)
 Total runtime: 0.057 ms
```

Российские интернет-технологии

Expression index

```
test=# CREATE INDEX hstore_a_idx
       ON hstore_test (((v->'a')::int));

test=# SELECT * FROM hstore_test WHERE v->'a' = '10';

------------------------------------------------------------
Seq Scan on hstore_test  (cost=0.00..26354.00 rows=5000
width=57) (actual time=0.747..231.017 rows=100 loops=1)
   Filter: ((v -> 'a'::text) = '10'::text)
   Rows Removed by Filter: 999900
 Total runtime: 231.077 ms
(4 rows)
```

Российские интернет-технологии

Extension

Решает

- Проблему pg_dump/pg_restore
- Проблему версий

CREATE EXTENSION

```
CREATE EXTENSION
hstore;
```

pg_extension

```
---------------+--------
oid            | 316474
extname        | hstore
extowner       | 16384
extnamespace   | 2200
extrelocatable | t
extversion     | 1.0
extconfig      |
extcondition   |
```

CREATE EXTENSION

pg_depend

classid	objid	objsubid	refclassid	refobjid	refobjsubid	deptype
1247 (pg_type)	316475 (hstore)	0	3079 (pg_extension)	316474 (hstore)	0	e (extension)
1255 (pg_proc)	316476 (hstore_in)	0	3079 (pg_extension)	316474 (hstore)	0	e (extension)
1255 (pg_proc)	316477 (hstore_out)	0	3079 (pg_extension)	316474 (hstore)	0	e (extension)
1255 (pg_proc)	316478 (hstore_recv)	0	3079 (pg_extension)	316474 (hstore)	0	e (extension)

Российские интернет-технологии

hstore.control

```
# hstore extension
comment = 'data type for storing sets of (key, value) pairs'
default_version = '1.1'
module_pathname = '$libdir/hstore'
relocatable = true
```

hstore--1.1.sql

```
/* contrib/hstore/hstore--1.1.sql */

-- complain if script is sourced in psql, rather than via CREATE
EXTENSION
\echo Use "CREATE EXTENSION hstore" to load this file. \quit

CREATE TYPE hstore;

CREATE FUNCTION hstore_in(cstring)
RETURNS hstore
AS 'MODULE_PATHNAME'
LANGUAGE C STRICT IMMUTABLE;
.....................................................................
```

Спасибо за внимание !

Российские интернет-технологии