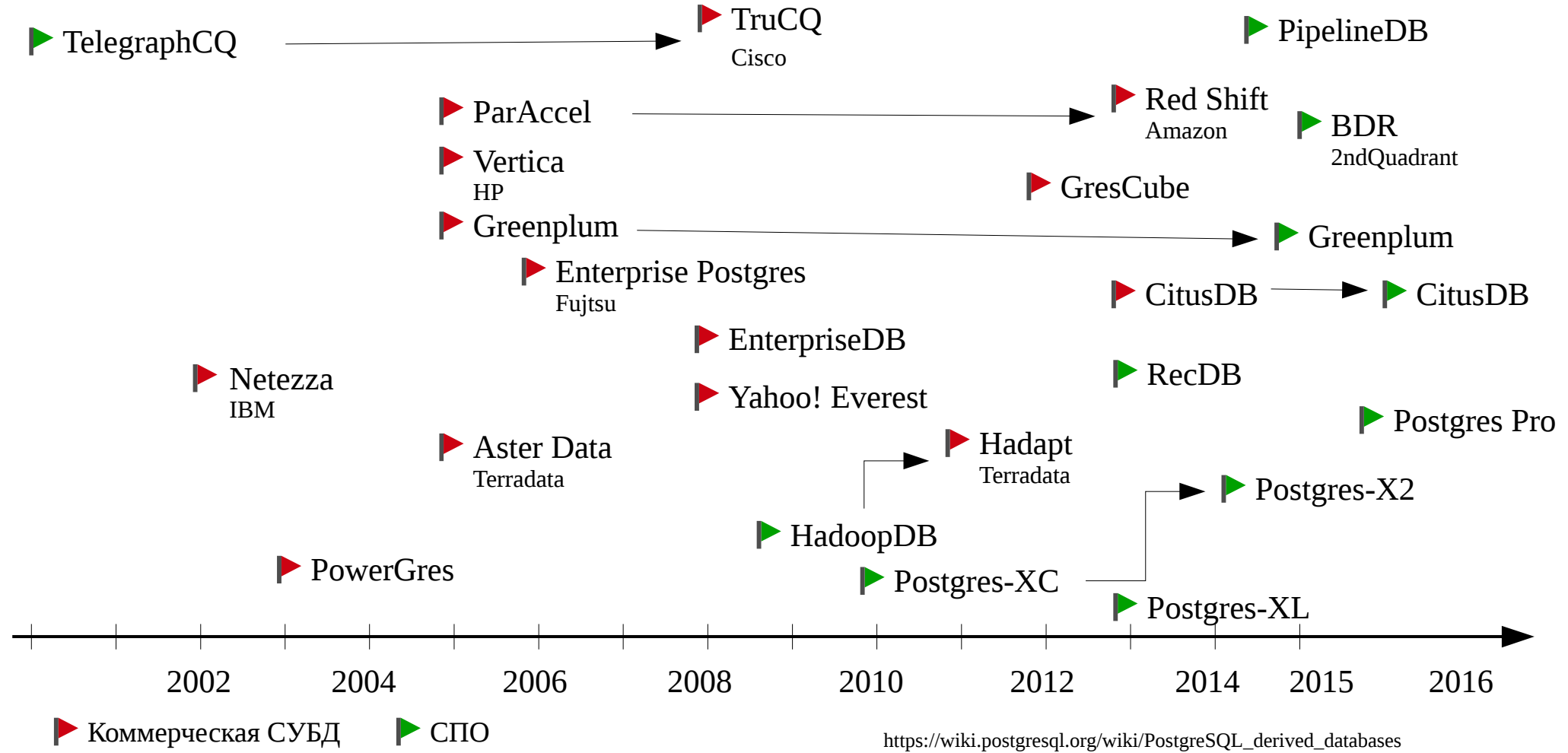
PROFESSIONAL Postgres

Настоящее и будущее PostgreSQL

Олег Бартунов
Иван Панченко

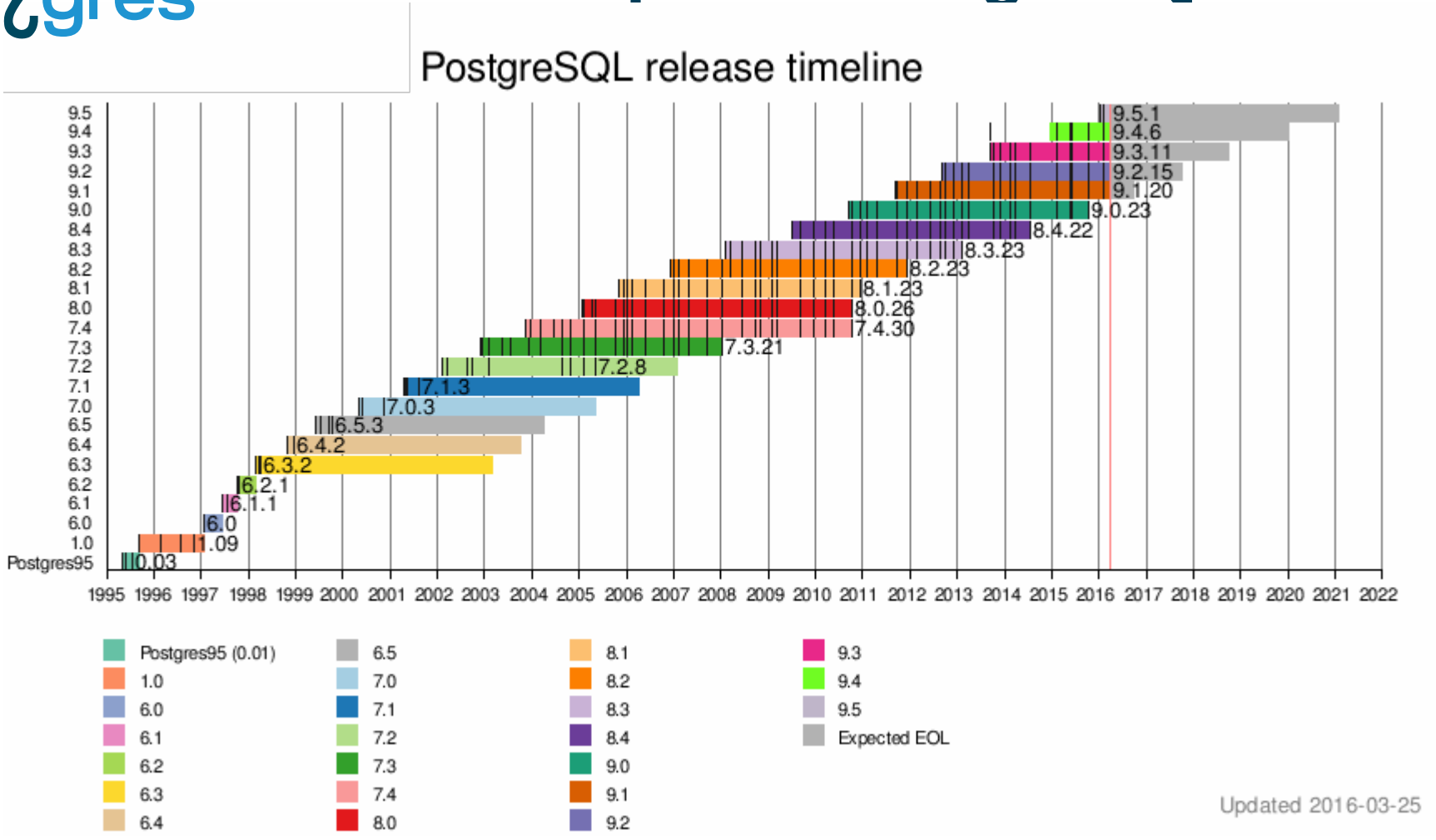
www.postgrespro.ru

Основные форки PostgreSQL



https://wiki.postgresql.org/wiki/PostgreSQL_derived_databases

Версии PostgreSQL



Разработка ядра PostgreSQL: ЖЕСТЬ !

- Идея должна быть понятной сообществу и одобрена («правильный» use-case)
- Методы и подходы должны быть обсуждены (PR, поиск спонсоров)
- Вы должны успеть подать на коммитфест до “feature freeze”
- Реализация должна пройти коммитфест
- Всегда найдется “умник”, которому не понравится
  - Вид вашего кода – отступы, trailing white-spaces
  - Названия переменных и функций
  - Отсутствие должного количества комментариев, документации
  - Ваша медленная реакция на замечания

История разработки KNN

- Начало проекта - **Sep 8, 2007 at 7:54 PM**

```
date    Sat, Sep 8, 2007 at 7:54 PM
subject Chat with Sergey V. Karpov

7:36 PM me: я тут knn-search занимаюсь, масса интересного. Все
думаю, как в постгресе это поиметь
 Sergey: а что это такое?
7:37 PM me: k-nearest соседей - супер важная задача
  найти 5 ближайших точек
7:38 PM Sergey: ближайших к чему?
 me: к заданной точке
7:39 PM Sergey: в какой системе координат?
 me: в любой, в n-мерном пространстве. В простом варианте - хотя
бы на земле/небе
7:40 PM это нужно для поиска похожих картинок, например.
  навиный вариант повторять запросы - не катит
```

История разработки KNN

- TODO (http://www.sai.msu.su/~megera/wiki/TODO)
  начало 2008 года, уже есть понимание что делать
- 25 июня 2009 года – письмо Paul Ramsey (POSTGIS)
- 10 июля 2009 года – контракт Open Planning Project Inc.
- 20 ноября 2009 года – патч KNNGiST v.0.1 (модуль расш)
- Commitfest nightmare
  - 22 июля 2010 – KNNGiST (v.0.8), commitfest
  - 13 сентября 2010 – KNNGiST (v.0.9)
  - 03 декабря 2010 – Tom Lane committed for 9.1 !
  - 21 января 2011 – contrib/btree_gist committed !

История разработки KNN

- Итого: На проект ушло больше 3 лет !
- Реальное программирование заняло несколько месяцев
- Основные причины:
  - Отсутствие поддержки
    - Слабая информационная связь разработчиков и заказчиков
  - Занятость разработчиков
  - Усложнение процедуры рассмотрения проектов в сообществе

Расписание CommitFest'ов и релизов

- 1-30 сентября 2015 : CF 1
- 1-30 ноября 2015 : CF 2
- 2 января — 8 февраля 2016 : CF 3
- 7 января 2016 : 9.5 Release
- 1-31 марта 2016 : Финальный CF для 9.6
- 8 апреля : Feature Freeze
- Июнь 2016 : 9.6 beta
- Сентябрь 2016 : 9.6 release
- 1-30 сентября 2016 : CF 1 для 9.7 ….

https://commitfest.postgresql.org/

Postgres PROFESSIONAL

Что в PostgreSQL 9.5 ?

- Table Sample (SQL:2003)
- Grouping sets (многоуровневая группировка)
- BRIN-индексы
- UPSERT
- Row-Level Security
- Функции для JSONB
- SKIP LOCKED
- Параллельный VACUUM
- GiST Index Only Scans
- Abbreviated keys (ускорение сортировки)

Это не всё

Table Sample

- Задача: получить выборку, достаточную для статистики.

```
SELECT avg(salary) FROM emp TABLESAMPLE SYSTEM (50);
SELECT avg(salary) FROM emp TABLESAMPLE BERNOULLI (50);
SELECT avg(salary) FROM emp TABLESAMPLE BERNOULLI (50)
    REPEATABLE(100);
```

Агрегация выполняется не по всей таблице, а по 50%.
SYSTEM- постранично, Бернулли — по записям.

```
DELETE FROM emp TABLESAMPLE BERNOULLI (50);
```

Grouping Sets

```
SELECT brand, size, sum(sales)
   FROM items_sold
   GROUP BY GROUPING SETS
   ((brand), (size), ());
```

```
 brand          | size | sum
----------------+------+-----
 Большевичка    |      |  30
 Трехгорная М   |      |  20
                | L    |  15
                | M    |  35
                |      |  50
(5 rows)
```

Несколько группировок за один запрос.

ROLLUP, CUBE

```
ROLLUP (x,y,z)
```

```
GROUPING SETS (
    ( x,y,z ),
    ( x,y),
    ( x ),
    ( )
)
```

```
CUBE (x,y,z)
```

```
GROUPING SETS (
    (x,y,z),
    (x,y),
    (x,  z),
    (  y,z),
    (x), (y), (z),
    ()
)
```

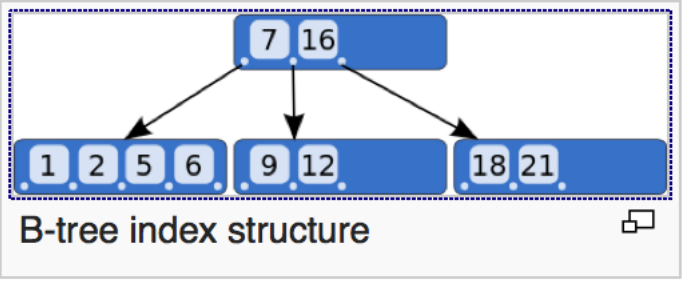
Удобные сокращения для Grouping Sets

BRIN (Block Range) - индексы

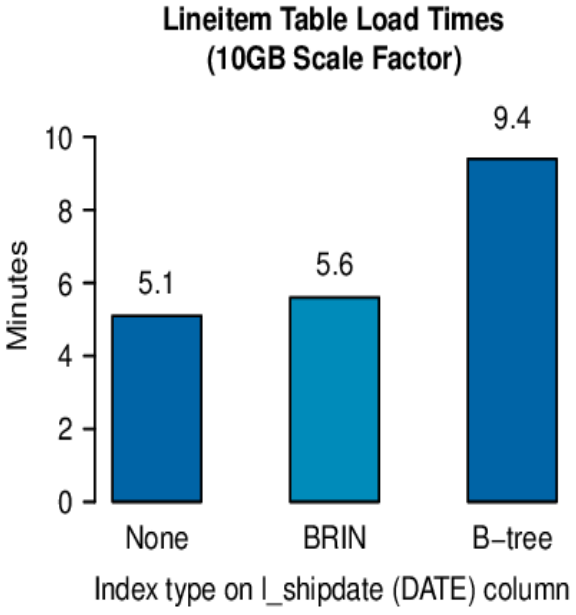
```
CREATE INDEX brin_index_test  ON test_data
   USING brin(id)
   WITH (pages_per_range = 128);
```

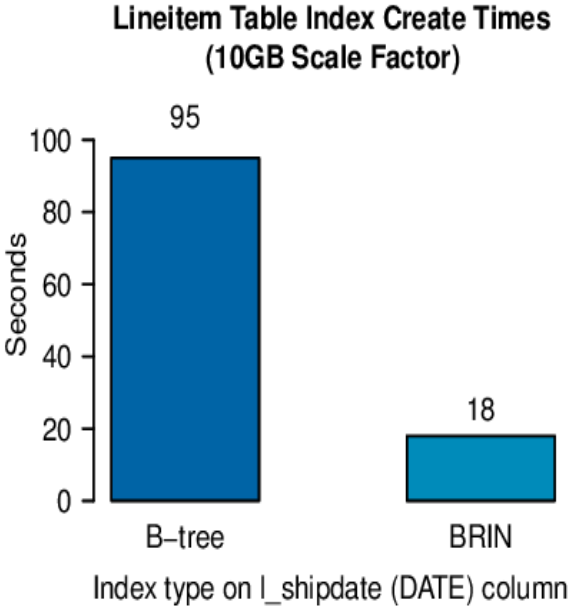
Lossy!

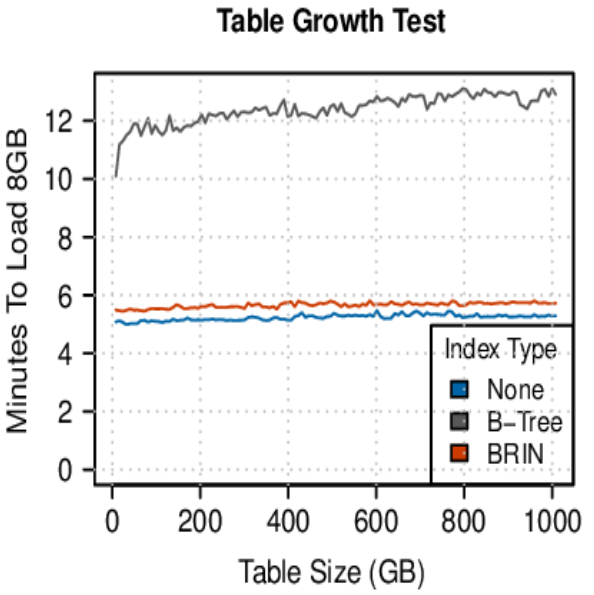
- Индексируются не записи, а страницы. Внутри страницы — sequence scan
- Хорошо подходит для append-only таблиц
- Быстро строится и занимает очень мало места
- Ищет медленнее, чем B-Tree



BRIN (Block Range) - индексы







UPSERT (MERGE)

INSERT … ON CONFLICT ….

```
CREATE TABLE x (y INT PRIMARY KEY);

INSERT INTO x VALUES (1);

INSERT INTO x VALUES (1)
   ON CONFLICT DO NOTHING;

INSERT INTO x VALUES (1)
   ON CONFLICT DO UPDATE SET y=2;
```

UPSERT (MERGE)

```
CREATE TABLE x (id INT PRIMARY KEY, name TEXT);

INSERT INTO x VALUES (1, 'Веники');

INSERT INTO x VALUES (1, 'Вареники')
   ON CONFLICT DO UPDATE SET name=EXCLUDED.name;
```

Row-Level Security

В postgresql.conf

```
row_security = on
```

```
CREATE TABLE orders (id INT, data TEXT, owner TEXT);
ALTER TABLE orders ENABLE ROW LEVEL SECURITY;

CREATE POLICY order_access ON orders FOR ALL TO PUBLIC
   USING (owner = CURRENT_USER);

GRANT ALL ON TABLE orders TO PUBLIC;
```

Каждый пользователь увидит только свои записи

Операторы и функции для JSONB

```
SELECT '{"x":2, "y":3}'::JSONB || '{"x":10}'::JSONB;
```

```
?column?
-------------------
 {"x": 10, "y": 3}
(1 row)
```

```
UPDATE t SET data = data || '{"key":"value"}'::JSONB;
```

SKIP LOCKED

```
SELECT * FROM tasks FOR UPDATE;
      --- Ждать
```

```
SELECT * FROM tasks FOR UPDATE NOWAIT;
     --- Не ждать; ошибка, если есть блокировки
```

```
SELECT * FROM tasks FOR UPDATE SKIP LOCKED;
     --- Не ждать; получить только незаблокированные
```

Ускорение сортировки

- Проблема: Сравнение в постгресе для типа text очень медленное, используется strcoll(), которая учитывает локаль, но в 1000 раз медленнее, чем сравнение целых чисел.
- Abbreviated keys (укороченные ключи) для сортировки character(n), bytea, uuid, numeric (Peter Geoghegan):
  - ГРУБО: Строки преобразуются в бинарные ключи strxfrm()),сравниваются первые 8 байт как целые числа, если эти 8 байт одинаковые, то вызываем strcoll()
- Великолепный доклад Greg Stark «Sorting through the ages» https://pgconf.ru/media/2016/02/19/9%20%D0%A1%D1%82%D0%B0%D1%80%D0%BA.pdf

Ускорение сортировки

```
strcoll("Олег Бартунов", "Магнус Хагандер") == 3
 => "Олег Бартунов" > "Магнус Хагандер"
```

- strcoll can be very slow because comparison rules can be very complicated
- Even when comparing ASCII it is much slower than integer comparisons

```
strxfrm("Олег Бартунов")
 => C393C38EC382C2BEC2BCC2BBC395C397C39AC391C393C2BD01...
strxfrm("Магнус Хагандер")
 => C390C2BBC2BEC391C39AC396C39DC2BBC2BEC2BBC391C2BFC3…
```

- Result from strxfrm can be compared quickly using memcmp
- But result can be very large which requires more memory and disk

Ускорение сортировки

- Use first 8 bytes (4 bytes on 32-bit platforms) as integer for fast comparisons

```
strxfrm("Олег Бартунов")
 => C393C38EC382C2BEC2BCC2BBC395C397C39AC391C393C2BD01...
strxfrm("Магнус Хагандер")
 => C390C2BBC2BEC391C39AC396C39DC2BBC2BEC2BBC391C2BFC3…
0xC393C38EC382C2BE > 0xC390C2BBC2BEC391
 => TRUE
```

- When two strings start with similar prefixes then the integers may be equal
- Call strcoll to disambiguate these cases

Ускорение сортировки

- Однако, не все так просто. В некоторых версиях glibc strcoll() не «матчит» strxfrm(), поэтому использование abbreviated keys может приводить к проблемам, например, к «битым» индексам.
- Например, проблема есть в
  - Debian 6.0.1
  - CentOS release 6.7 (Final)
- Описание проблемы http://www.postgresql.org/message-id/111D0E27-A8F3-4A84-A4E0-B0FB703863DF@s24.com

commit 8aa6e97805a79eb30ac9c36acb1126280c2ffbdfESC[m
Author: Robert Haas <rhaas@postgresql.org>
Date: Wed Mar 23 15:58:34 2016 -0400

    Disable abbreviated keys for string-sorting in non-C locales.

    Unfortunately, every version of glibc thus far tested has bugs whereby
    strcoll() ordering does not match strxfrm() ordering as required by
    the standard. This can result in, for example, corrupted indexes.
    Disabling abbreviated keys in these cases slows down non-C-collation
    string sorting considerably, but there seems to be no practical
    alternative. Users who are confident that their libc implementations
    are solid in this regard can re-enable the optimization by compiling
    with TRUST_STRXFRM.

    Users who have built indexes using PostgreSQL 9.5 or PostgreSQL
9.5.1
    should REINDEX if there is a possibility that they may have been
    affected by this problem.
    Report by Marc-Olaf Jaschke. Investigation mostly by Tom Lane, with
    help from Peter Geoghegan, Noah Misch, Stephen Frost, and me. Patch
    by me, reviewed by Peter Geoghegan and Tom Lane.

Ускорение сортировки

- Что же делать нам с нашей локалью !!??
- Postgres Pro имеет поддержку ICU (--with-icu) для обеспечения системо-независимой локали (важно для 1С, в частности). ICU быстрее и обеспечивает корректную работу abbreviated keys.
- Пример Роберта Хааса из http://www.postgresql.org/message-id/CA+TgmoaTiYy9aaMRe7m71Z=mrNZ_aPqepspQHtSNHq8Wiafjow@mail.gmail.com

Ускорение сортировки

```
create table stuff as select random()::text as a, 'filler
filler filler'::text as b, g as c from generate_series(1, 1000000) g;
create index on stuff (a);


Pgpro 9.5.0.1 (--with icu)          Vanilla REL9_5_STABLE (no abbrev.)
 2.3 sec                              30 sec

Эффект abbreviated keys не должен быть заметен:

create table stuff as select 'aaaaaaaa' || random()::text as
a, 'filler filler filler'::text as b, g as c from generate_series(1,
1000000) g;

Pgpro 9.5.0.1 (--with icu)          Vanilla REL9_5_STABLE (no abbrev.)
 11.4 sec                             32 sec
```

Используйте Postgres Pro !

Ускорение 15x

Что такое Postgres Pro ?

1. Российский форк PostgreSQL, вошедший в Реестр Российского ПО
2. Наши разработки ядра PostgreSQL, которые ещё не успели войти в релиз, но закоммичены в апстрим
3. Бэкпорты из 9.6, которые мы считаем полезными
4. Полезные расширения (не только наши)
5. Возможность оперативно реагировать на запросы клиентов

Postgres Pro предоставляет доступ к новой функциональности и улучшениям раньше, и позволяет быстрее реагировать на запросы клиентов

Что еще есть в Postgres Pro ?

- Увеличение производительности на многоядерных системах:
- Улучшение полнотекстового поиска:
  поиск фраз, словарь=extension,
  словари в shared memory
- Покрывающие индексы. Поддержка конструкции INCLUDES в CREATE INDEX. https://pgconf.ru/2016/89847
- Переносимость: ICU
- Модуль pg_trgm: нечеткий поиск подстроки
- Модуль pageinspect: доступ к внутреннему представлению данных
- Модуль sr_plan: сохранение планов запросов
- Модуль dump_stat: дамп и восстановление статистики
- Модуль JSQuery
- Мониторинг ожиданий (WAIT-мониторинг с сэмплингом)
- Libpq с поддержкой failover

Скоро в Postgres Pro (9.5.*)

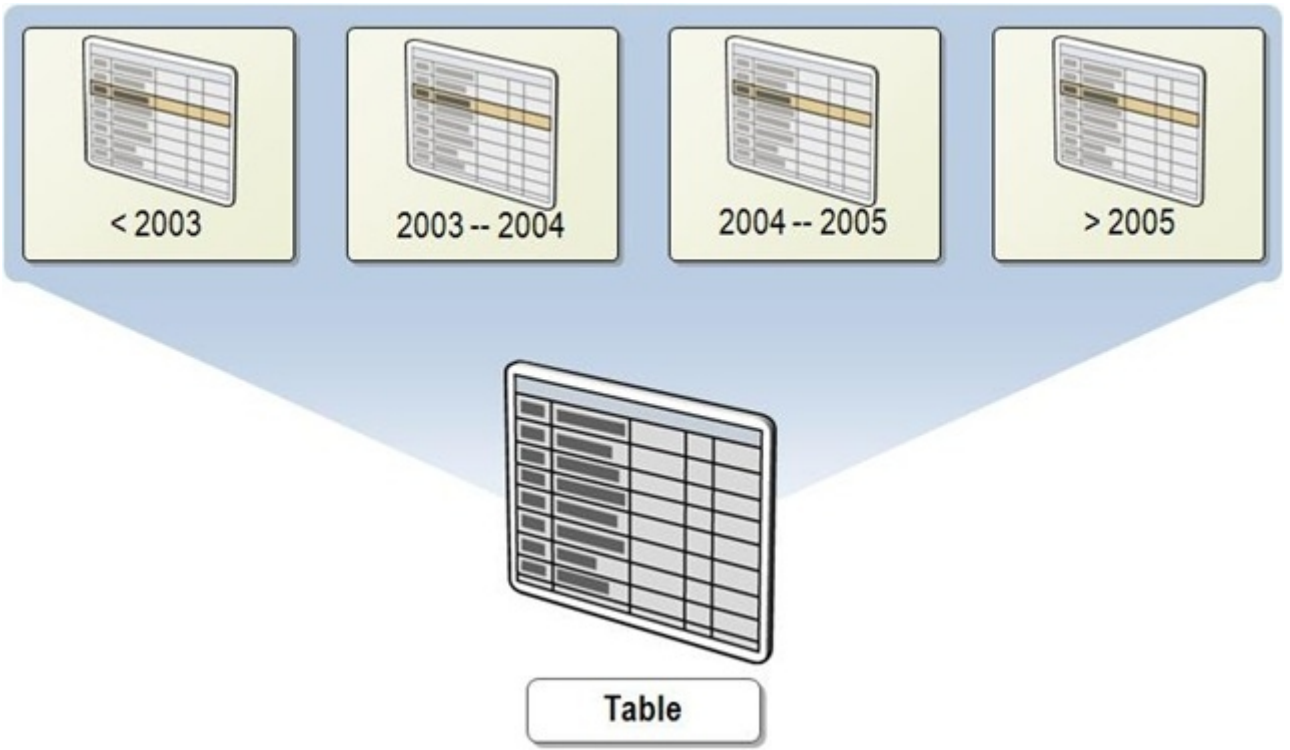
- Секционирование таблиц без накладных расходов
- Инкрементальный бэкап на уровне блоков
- Компрессия B-Tree индексов
- Миллисекундный полнотекстовый поиск (на основе CREATE ACCESS METHOD и Custom WAL)

https://wiki.postgresql.org/images/2/25/Full-text_search_in_PostgreSQL_in_milliseconds-extended-version.pdf

- Мониторинг ожиданий

http://akorotkov.github.io/blog/2016/08/26/wait_monitoring_9_6/ ,
http://www.postgresql.org/docs/devel/static/monitoring-stats.html#WAIT-EVENT-TABLE

Секционирование



Секционирование

До сих пор:

- Делается через наследование таблиц (мега-костыль ?)
- Работает неэффективно

Уже сделано в Postgres Pro (расширение pathman):

- Хранение метаданных секций
- Бинарный поиск по ним при планировании запроса
- RANGE partitioning

Секционирование (планы на 9.6)

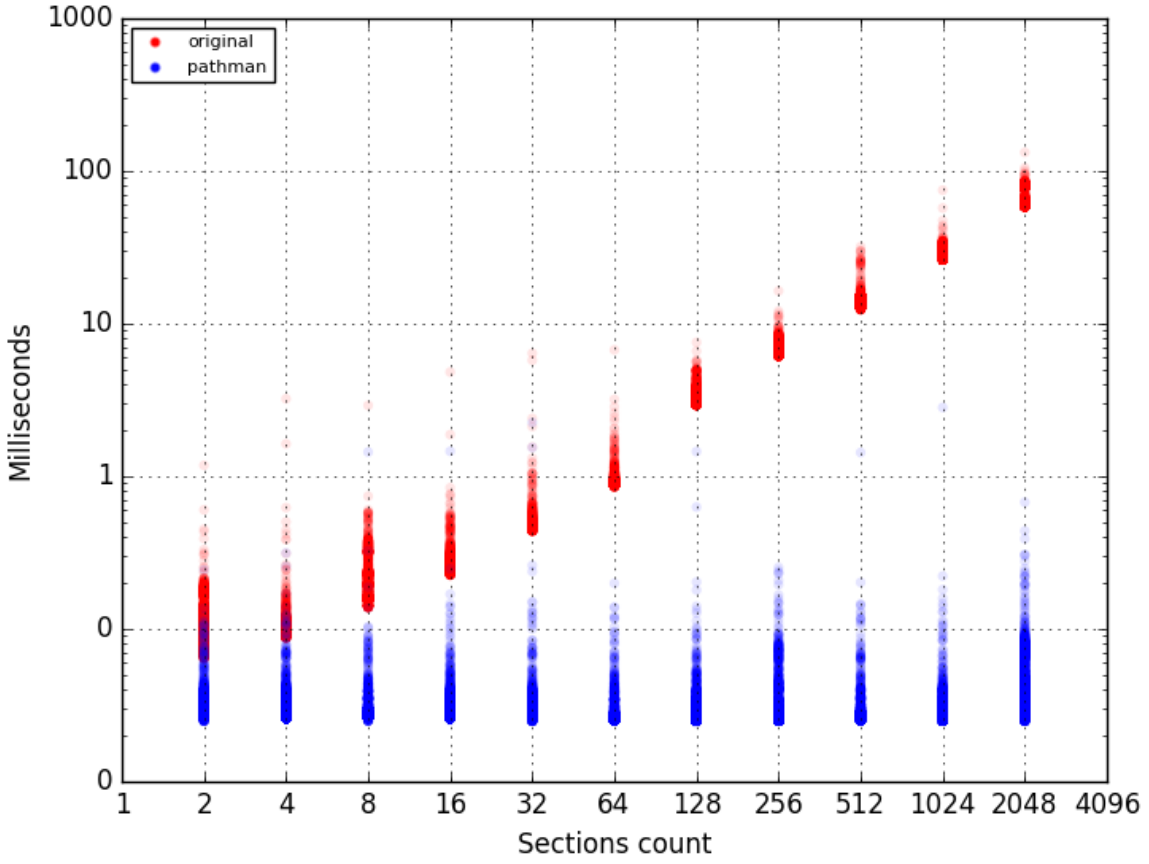
- Выбор секций во время выполнения запроса
- Упорядоченная выдача выборок из секций для ускорения Merge join и сортировки.
- Оптимизация Hash join при секционировании по ключу JOIN
- LIST-partitioning
- HASH-partitioning
- Поддержка декларативного синтаксиса

Секционирование

Время планирования при попадании в план одной секции.

Обычная реализация на основе наследования таблиц страдает большими накладными расходами.

Pathman — свободен от них.



Секционирование

Производительность SELECT, INSERT, UPDATE

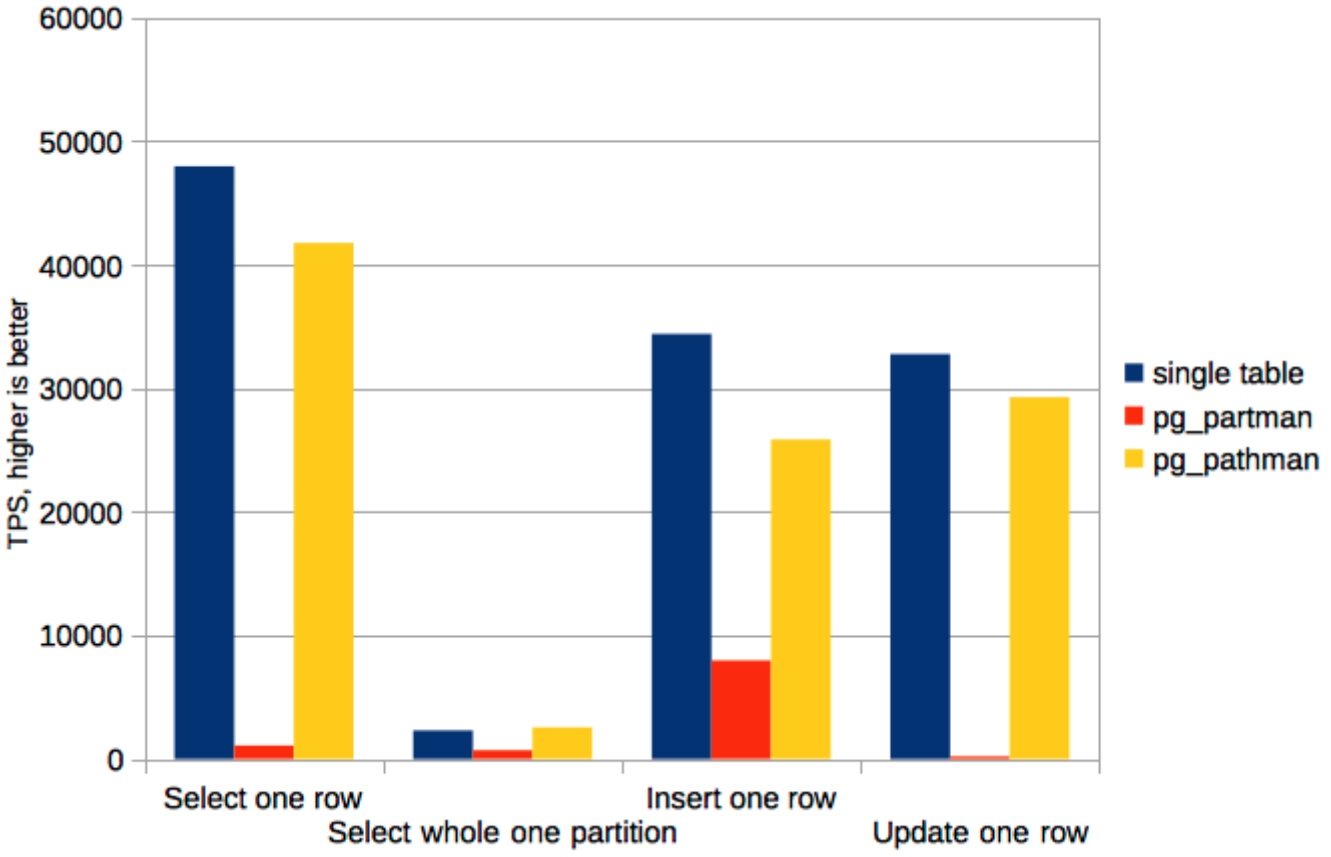
pg_partman — основан на наследовании таблиц

pg_pathman — разработка Postgres Pro

https://pgconf.ru/2016/89633

http://akorotkov.github.io/blog/categories/partitioning/

http://postgrespro.ru/blog/pgsql/pg_pathman



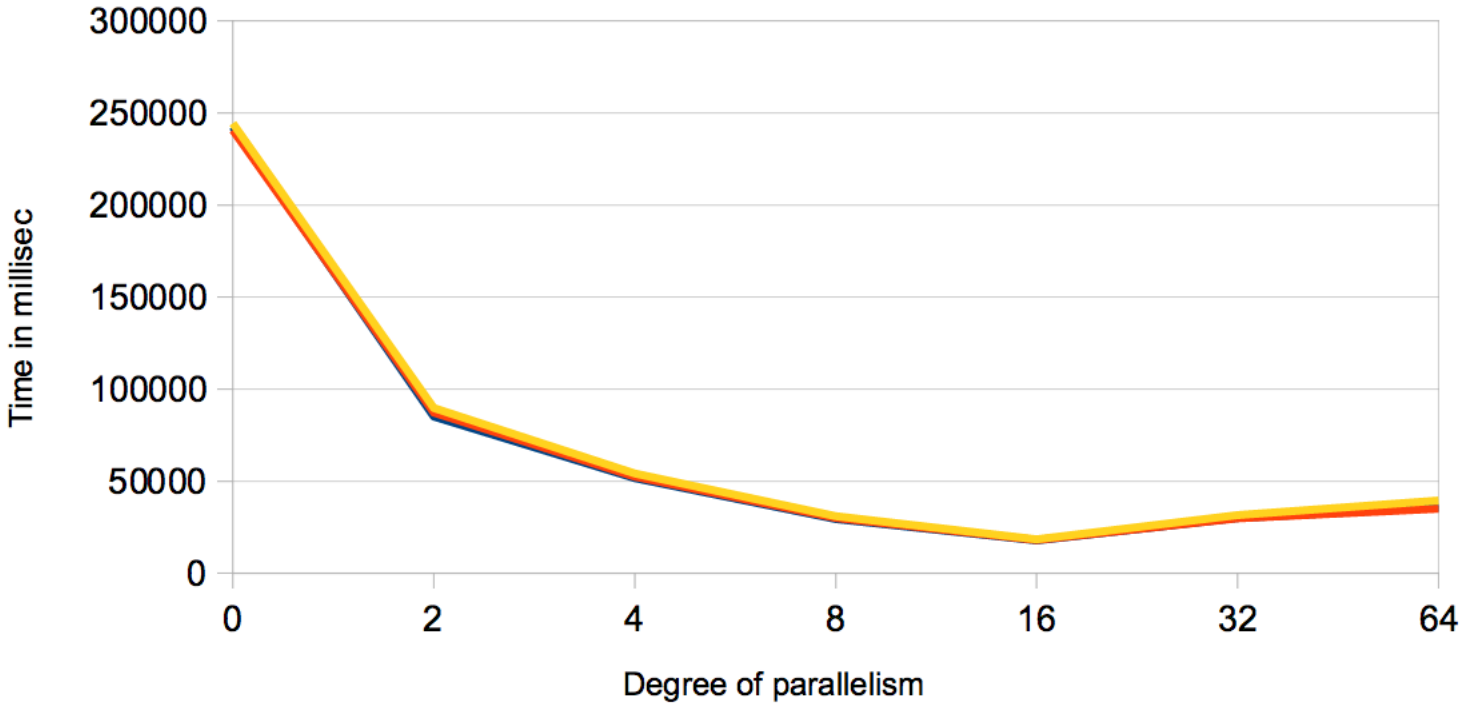
Планы Postgres Pro

- Компрессия данных
- Система управления кластером СУБД
- Упрощение миграции с Oracle
- Пакеты, автономные транзакции, глобальные переменные
- Улучшение мониторинга
- Частичный бэкап и восстановление
- JIT-компиляция SQL-запросов. Использование GPU.
- InMemory и колоночные хранилища
- Совершенствование кластера

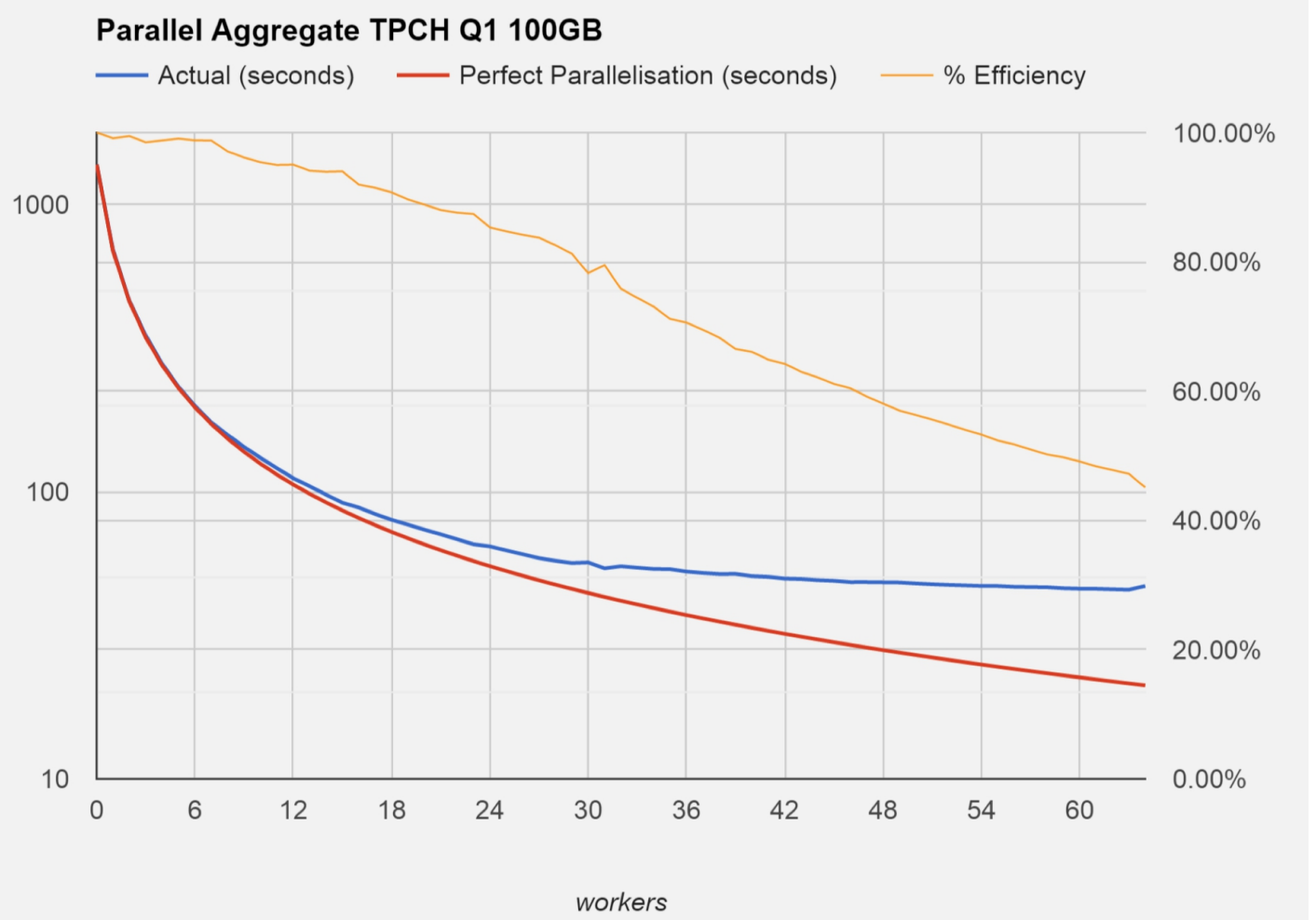
Что ждёт нас «завтра» (9.6)?

- Параллельное исполнение запроса
  - Seqscan
  - Join
  - Aggregates
  - Explain analyze показывает информацию по каждому воркеру
  - Parallel create GIN index ?
  - FDW pushdown

Завтра. Параллельный seqscan



Завтра. Параллельная агрегация



Завтра (9.6). Расширяемость

- Create access method
- Custom WAL records ?

Можно будет создавать полноценные методы доступа как расширения !

Завтра (9.6). DBA

- Немного счастья для DBA
  - VACUUM FREEZE не будет трогать уже «замороженные» блоки — сильное облегчение для больших и нагруженных проектов
  - idle_in_transaction_session_timeout - решает проблему «idle in transaction"
  - a generic command progress reporting facility
  - simple VACUUM progress reporting — можно следить за выполнением вакуума pg_stat_progress_vacuum.

Завтра (9.6). DBA

Представление pg_config

```
[local]:5432 postgres@postgres=# select * from pg_config;
       name        |
           setting

-------------------+-----------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
---------
 BINDIR            | /usr/local/pgsql-head/bin
 DOCDIR            | /usr/local/pgsql-head/share/doc
 HTMLDIR           | /usr/local/pgsql-head/share/doc
 INCLUDEDIR        | /usr/local/pgsql-head/include
 PKGINCLUDEDIR     | /usr/local/pgsql-head/include
 INCLUDEDIR-SERVER | /usr/local/pgsql-head/include/server
 LIBDIR            | /usr/local/pgsql-head/lib
 PKGLIBDIR         | /usr/local/pgsql-head/lib
 LOCALEDIR         | /usr/local/pgsql-head/share/locale
 MANDIR            | /usr/local/pgsql-head/share/man
 SHAREDIR          | /usr/local/pgsql-head/share
 SYSCONFDIR        | /usr/local/pgsql-head/etc
 PGXS              | /usr/local/pgsql-head/lib/pgxs/src/makefiles/pgxs.mk
 CONFIGURE         | '--prefix=/usr/local/pgsql-head' '--enable-depend' '--with-libs=/usr/local/lib'
 CC                | gcc
 CPPFLAGS          | -DFRONTEND
 CFLAGS            | -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labe
ls -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -Wno-unused-command-line-argu
ment -O2
 CFLAGS_SL         |
 LDFLAGS           | -L../../src/common -L/usr/local/lib -Wl,-dead_strip_dylibs
 LDFLAGS_EX        |
 LDFLAGS_SL        |
 LIBS              | -lpgcommon -lpgport -lz -lreadline -lm
 VERSION           | PostgreSQL 9.6devel
(23 rows)
```

- Wait monitoring

```
[local]:5432 postgres@postgres=# \d pg_stat_activity
          View "pg_catalog.pg_stat_activity"
      Column      |           Type           | Modifiers
------------------+--------------------------+-----------
 datid            | oid                      |
 datname          | name                     |
 pid              | integer                  |
 usesysid         | oid                      |
 usename          | name                     |
 application_name | text                     |
 client_addr      | inet                     |
 client_hostname  | text                     |
 client_port      | integer                  |
 backend_start    | timestamp with time zone |
 xact_start       | timestamp with time zone |
 query_start      | timestamp with time zone |
 state_change     | timestamp with time zone |
 wait_event_type  | text                     |
 wait_event       | text                     |
 state            | text                     |
 backend_xid      | xid                      |
 backend_xmin     | xid                      |
 query            | text                     |
```

- pg_wait_sampling (http://akorotkov.github.io/blog/2016/08/26/wait_monitoring_9_6/)

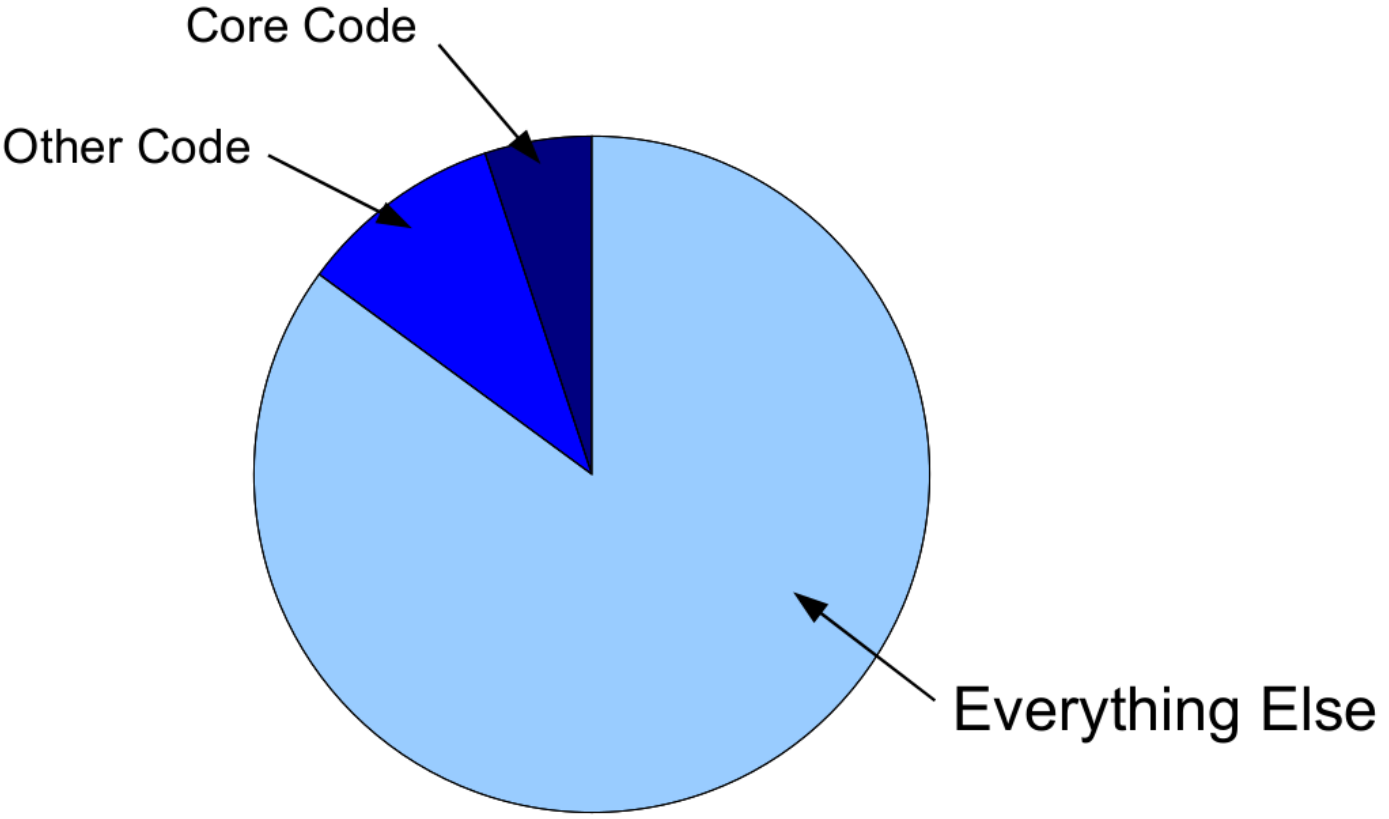
Завтра (9.6) ….

- \ev — редактирование представлений
- \sv — показ
- KNN для CUBE
- TSVECTOR EDITING FUNCTIONS
  convert tsvector to/from text array, set weight for given lexemes, delete lexeme(s), unnest, filter lexemes with given weights
- Улучшена поддержка словарей
- Поиск фраз ?
- Covering indexes ?
- Btree-compression ?
- Масштабирование на большое количество ядер ?

Что будет в «10.0» (в работе)

- BDR — двунаправленная репликация
  http://2ndquadrant.com/en/resources/bdr/
- Pglogical (5x быстрее slony, londiste3)
  http://2ndquadrant.com/en/resources/pglogical/
- Declarative partitioning (+pg_pathman)
- Highly Available multi-master
- Инкрементальный бэкап
- Миллисекундный полнотекстовый поиск
- In-memory

50 способов помочь сообществу



50 способов помочь сообществу

Ядро
Разработка, review, тестирование, reporting bugs

Экосистема
Расширения, драйверы, ORM, средства мониторинга… поддержка Pg в прикладном ПО
Создание дистрибутивов, пакетирование

Документация
Улучшение, перевод, публикация статей, книг, учебных, маркетинговых материалов…блоггинг!

Расскажите о своей истории с PostgreSQL!

Общение, образование
Создание локальных сообществ
Проведение конференций, митапов, семинаров, учебных курсов.

Внедрите PostgreSQL!
В Вашей компании. Запустите учебный курс в Вашем ВУЗе

Спонсорство
Спонсируйте разработку нужной Вам функциональности.
Спонсируйте мероприятие.

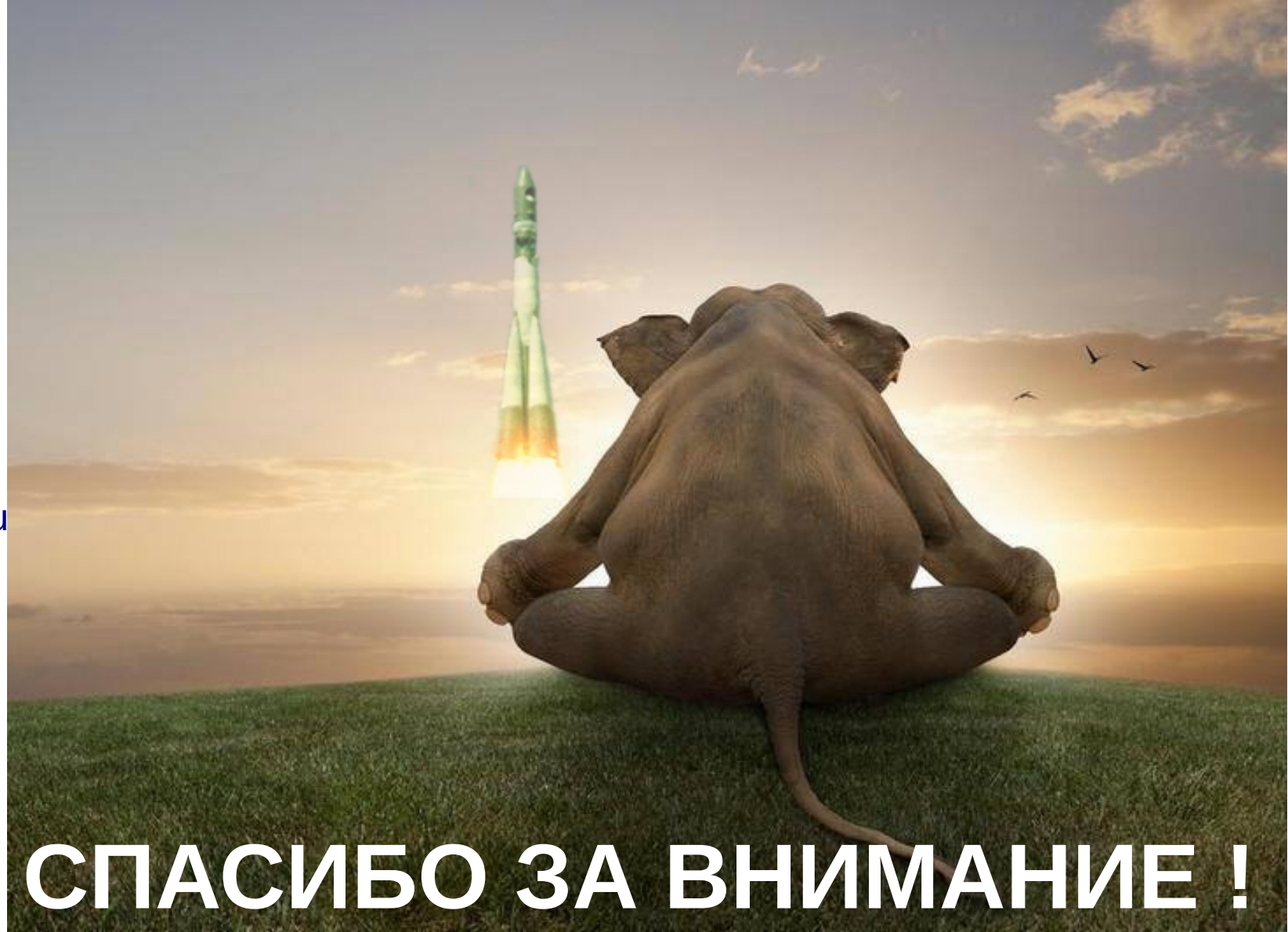
Postgres PROFESSIONAL

www.postgrespro.ru

info@postgrespro.ru

jobs@postgrespro.ru

sales@postgrespro.ru



СПАСИБО ЗА ВНИМАНИЕ !