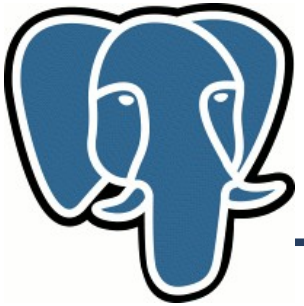
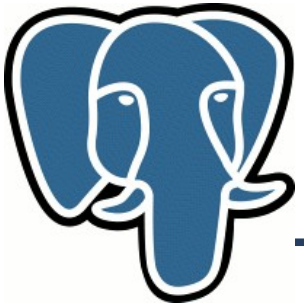


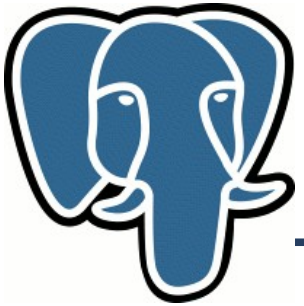
Что нового в PostgreSQL 9.0 ?

Олег Бартунов
ГАИШ МГУ



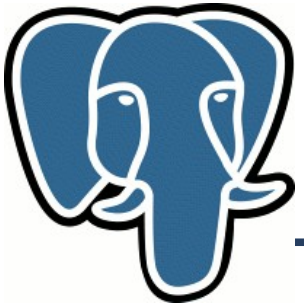
8.5 → 9.0

Чистый маркетинг ?



Pg 9.0: Development

- Commitfest: Июль, Сентябрь, Ноябрь, Январь - **Февраль**
  - Планируемость процесса разработки
    - Протоколирование изменений
    - Обсуждение в mailing lists
  - http://commitfest.postgresql.org/
  - Каждый коммитфест → ALPHA Release
    - Более широкое тестирование
    - Легкая инсталляция (пакеты)
    - Облегчается BETA
    - Инкрементальность обновления приложений

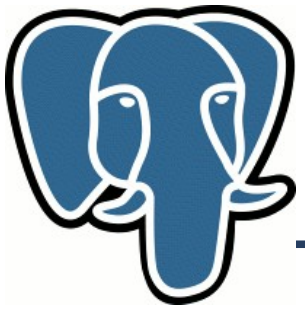


Pg 9.0: Development

- Всего патчей: 204 (bug-fixes не учитываются)
- Авторов патчей: 82
- Коммиттеров: 14

Полезные ссылки:

- http://developer.postgresql.org/pgdocs/postgres/release-9-0.html
- Waiting for 9.0 - http://www.depesz.com/index.php/tag/pg90/

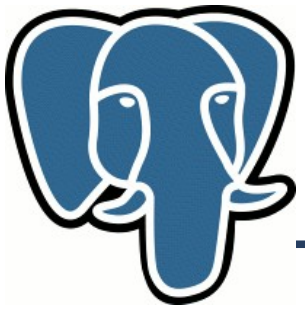


Pg 9.0: Производительность

- **Join (**LEFT**) removal** (Robert Haas) – оптимизация плана

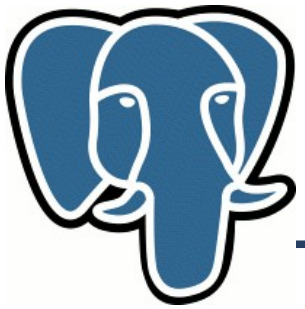
```
SELECT p.id, p.name FROM projects p
   LEFT JOIN person pm
      ON p.project_manager_id = pm.id;
```

- Надо убедиться, что join не меняет множество возвращаемых записей и не добавляет дополнительных атрибутов.
- Если есть уникальный индекс по person(id), то join не нужен !
- Типичная оптимизация для сгенеренных SQL



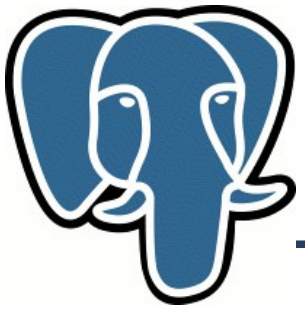
Pg 9.0: Производительность

- Новый VACUUM FULL (Itagaki Takahiro)
  - Раньше использовали CLUSTER вместо вакуума – полностью перезаписывает таблицу, поэтому может работать существенно быстрее, если таблица содержит много 'dead tuples'
  - VACUUM FULL работал in-place – медленно
  - Теперь VACUUM FULL работает как CLUSTER (ctid), но без переупорядочивания записей. Работает быстро ценою дополнительного места.



Pg 9.0: Производительность

- Индексная поддержка IS NOT NULL
- GUC переменные для Tablespace – настройка под разные носители (быстрые/медленные hdd, ssd)
  - seq_page_cost
  - random_page_cost (=seq_page_cost для ssd)
- ALTER TABLESPACE … SET seq_page_cost=1.0
  ALTER TABLESPACE … RESET seq_page_cost;

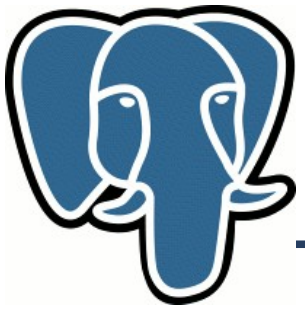


Pg 9.0: Производительность

- Переопределение оценки количества уникальных значений n_distinct атрибута для последующей команды ANALYZE.

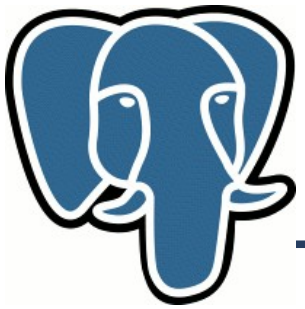
```
ALTER TABLE … ALTER COLUMN … SET (n_distinct = ND)
ALTER TABLE … ALTER COLUMN … SET (n_distinct_inherited = ND)
```

  - ND>0 – n_distinct = ND
  - -1<ND<0 – n_distinct линейно пропорционально размеру таблицы
    n_distinct = |ND|*estimated_rows
  - ND=0 – n_distinct определяется ANALYZE



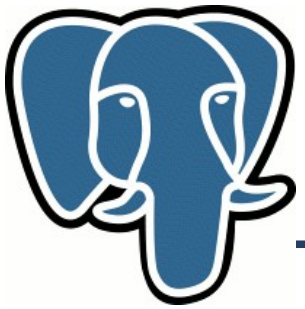
Pg 9.0: Производительность

```
postgres=# alter table aa alter column id set (n_distinct=10);
ALTER TABLE
postgres=# analyze aa;
ANALYZE
postgres=# SELECT attname As colname, n_distinct FROM pg_stats
  WHERE schemaname = 'public' and tablename = 'aa';
 colname | n_distinct
---------+------------
 id      |          10
(1 row)
postgres=# alter table aa alter column id set (n_distinct=0);
ALTER TABLE
postgres=# analyze aa;
ANALYZE
postgres=# SELECT attname As colname, n_distinct FROM pg_stats
  WHERE schemaname = 'public' and tablename = 'aa';
 colname | n_distinct
---------+------------
 id      |  -0.454545
(1 row)
```



Pg 9.0: Производительность

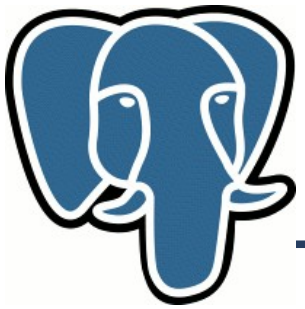
- Rbtree – red-black tree для GIN индекса: нормальная производительность для упорядоченных данных
- Оптимизация
  - foo <> true как foo=false
  - Foo <>false как foo=true
- Улучшена поддержка паралеллизма pg_restore (-j)
- Поддержка Windows 64



Pg 9.0: Производительность

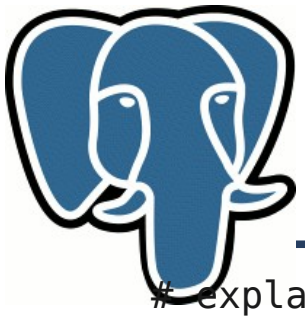
- **Buffers in EXPLAIN** (Robert Haas)

```
=# explain (analyze on, buffers on) select 1 from pg_attribute;
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Seq Scan on pg_attribute  (cost=0.00..59.29 rows=2129 width=0)
 (actual time=0.005..11.028 rows=2129 loops=1)
   Buffers: shared hit=23 read=15
 Total runtime: 11.301 ms
(3 rows)

-------------------------------------------------------------------------------------------------------------------
 Seq Scan on pg_attribute  (cost=0.00..59.29 rows=2129 width=0)
 (actual time=0.006..0.399 rows=2129 loops=1)
   Buffers: shared hit=38
 Total runtime: 0.626 ms
(3 rows)
```



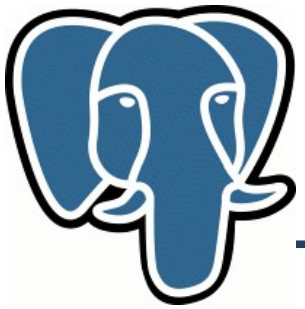
Pg 9.0: Производительность

- **EXPLAIN output в машинном формате** (Robert Haas et al.)
  - Визуализация, мониторинг, больше информации
  - EXPLAIN (option value, option value, ...) query
    - analyze on
    - verbose on
    - buffers on
    - format {xml ,json,yaml}



Pg 9.0: Производительность

```
# explain ( analyze on, format json ) select * from pg_class where relname = 'xxx';
QUERY PLAN
------------------------------------------------------
[
{
"Plan": {
"Node Type": "Index Scan",
"Scan Direction": "Forward",
"Index Name": "pg_class_relname_nsp_index",
"Relation Name": "pg_class",
"Alias": "pg_class",
"Startup Cost": 0.00,
"Total Cost": 8.27,
"Plan Rows": 1,
"Plan Width": 185,
"Actual Startup Time": 0.023,
"Actual Total Time": 0.023,
"Actual Rows": 0,
"Actual Loops": 1,
"Index Cond": "(relname = 'xxx'::name)"
},
"Triggers": [
],
"Total Runtime": 0.102
}
]
```



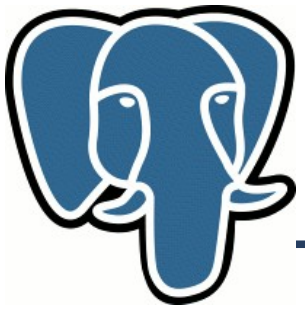
Pg 9.0: Admin

- **Application Name** (Dave Page)
  Новый параметр - *application_name* (logs (%a), pg_stat_activity)

  libpq:

  ```
  PGAPPNAME=qq psql -c 'show application_name'
   application_name
  -----------------
   qq
  ```

  SQL: SET application_name='qq'



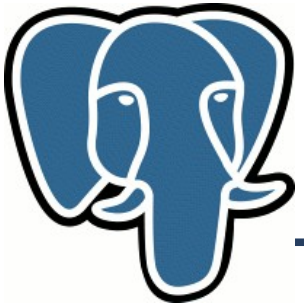
Pg 9.0: Admin

- **Application Name** (Dave Page)

```
PGAPPNAME=qq psql 1c
psql (9.0devel)
Type "help" for help.

1c=# select datname, procpid, usename, application_name
from pg_stat_activity;
  datname | procpid | usename  | application_name
  --------+---------+----------+-----------------
  1c      |   19189 | postgres | qq
```

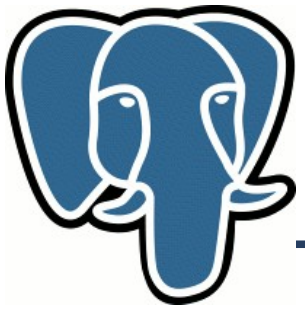
- **pg_ctl -D data initdb -o '--locale=ru_RU.UTF-8'**
- **Улучшенная диагностика нарушения уникальности:**

```
ERROR:  duplicate key value violates unique constraint "uu_id_key"
DETAIL:  Key (id)=(1) already exists.
```



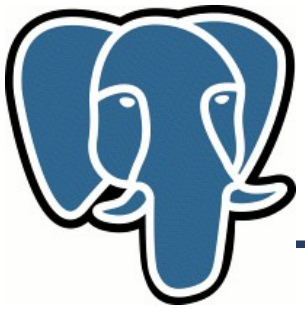
Pg 9.0: Admin

- **Per user, per database GUCs** (Alvaro Herrera)
  - 8.4
  - ALTER DATABASE <database> SET <guc> TO <value>;
    ALTER ROLE <role> SET <guc> TO <value>;
  - 9.0
  - ALTER ROLE <role> IN DATABASE <database> SET <guc> TO <value>;



Pg 9.0: Admin

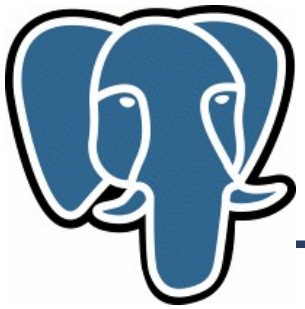
- **GRANT ALL** (Petr Jelinek)
  - GRANT <privileges> ON ALL <object type>S IN SCHEMA <schema> TO <role>;
    # grant all on all tables in schema public to postgres;
  - REVOKE <privileges> ON ALL <object type>S IN SCHEMA <schema> FROM <role>;
  - Привилегии по-умолчанию:
    ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT ALL ON TABLES TO web_user, postgres;



Pg 9.0: Admin

- **\d show child tables** (Damien Clochard)

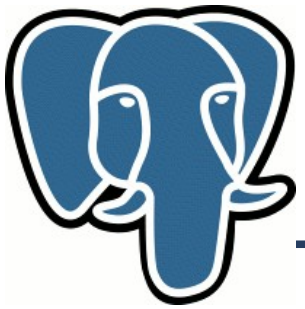
```
# create table x (id int4);
# create table x2 () INHERITS (x);
# create table x3 () INHERITS (x);
# \d x
Table "public.x"
Column | Type | Modifiers
--------+---------+-----------
id | integer |
Number of child tables: 2 (Use \d+ to list them.

# \d+ x
                    Table "public.x"
 Column |  Type   | Modifiers | Storage | Description
--------+---------+-----------+---------+-------------
 id     | integer |           | plain   |
Child tables: x2,
              x3
```



Pg 9.0: Admin

- **'samehost' and 'samenet' in pg_hba.conf** (Stef Walter)
  - samehost вместо CIDR сервера
  - samenet вместо текущего подсети
  - Удобно для массовой установки

```
# TYPE DATABASE USER CIDR-ADDRESS METHOD
local  all      all                trust
host   all all  127.0.0.1/32       trust
host   all all  0.0.0.0/0          md5

# TYPE DATABASE USER CIDR-ADDRESS METHOD
local  all      all                trust
host   all      all samehost       trust
host   all      all samenet        md5
```

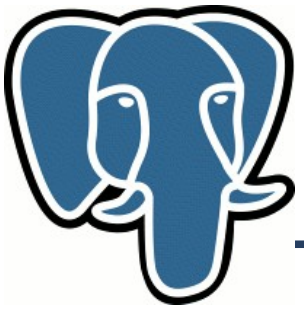


Pg 9.0: Admin

- **Multi-threaded pgbench (-j N)** (ITAGAKI Takahiro)

Here are results of multi-threaded pgbench runs on Fedora 11 with intel core i7 (8 logical cores = 4 physical cores * HT). -j8 (8 threads) was the best and the tps is 4.5 times of -j1, that is a traditional result.

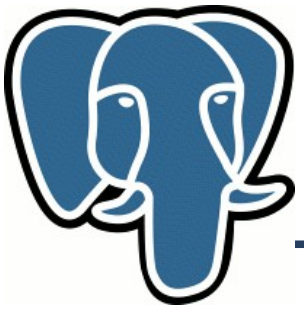
```
$ pgbench -i -s10
$ pgbench -n -S -c64 -j1 => tps = 11600.158593
$ pgbench -n -S -c64 -j2 => tps = 17947.100954
$ pgbench -n -S -c64 -j4 => tps = 26571.124001
$ pgbench -n -S -c64 -j8 => tps = 52725.470403
$ pgbench -n -S -c64 -j16 => tps = 38976.675319
$ pgbench -n -S -c64 -j32 => tps = 28998.499601
$ pgbench -n -S -c64 -j64 => tps = 26701.877815
```



Pg 9.0: Dev

- **Deferrable UNIQUE constraints** (Dean Rasheed)
  - Отложенная проверка UNIQUE, проверяется при COMMIT

```
CREATE TABLE distributors (
did integer,
name varchar(40) UNIQUE DEFERRABLE
);
```



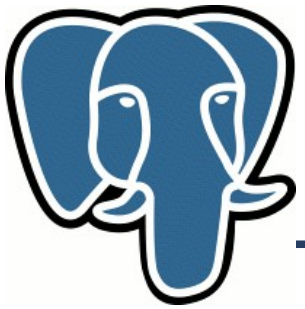
Pg 9.0: Dev

- **Deferrable UNIQUE constraints** (Dean Rasheed)

```
# CREATE TABLE test (i INT4 PRIMARY KEY);

# INSERT INTO test (i) values (1), (2), (3);

# select * from test;
i
---
1
2
3
(3 rows)
# update test set i = i + 2;
ERROR: duplicate key value violates unique constraint "test_pkey"
```

Так как проверяется после каждого обновления

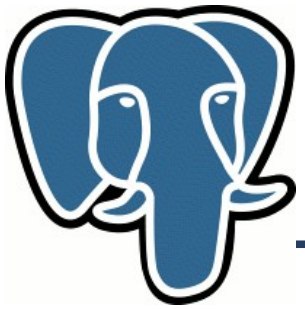
```
# CREATE TABLE test (i INT4 PRIMARY KEY
DEFERRABLE INITIALLY DEFERRED
);
```



Pg 9.0: Dev

- **DROP IF EXISTS for columns and constraints** (Andres Freund)

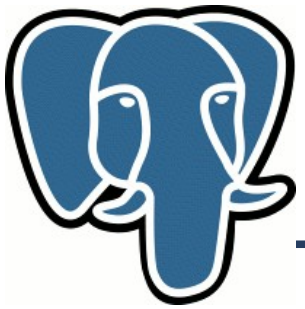
```
# alter table test drop column if exists c;
NOTICE: column "c" of relation "test" does not exist, skipping
ALTER TABLE

# alter table test drop constraint if exists typo_pkey;
NOTICE: constraint "typo_pkey" of relation "test" does not exist,
skipping
ALTER TABLE
```



Pg 9.0: Dev

- **Hstore improvements** (Andrew Gierth)
  - Remove the 64K limit on the lengths of keys and values within an hstore. (This changes the on-disk format, but the old format can still be read.)
  - Add support for btree/hash opclasses for hstore for actual indexing purposes as to allow use of GROUP BY, DISTINCT, etc.
  - Add various other new functions and operators.

```
# select 'a=>1, b=>2, c=>3, d=>4'::hstore -> ARRAY['b', 'd',
'c'];
# select 'a=>1, b=>2, c=>3, d=>4'::hstore ? 'a';
# select array['a','1','b','2','c','3']::hstore;
.................................................
```

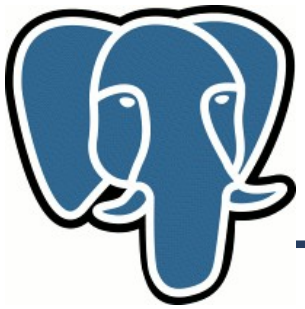


Pg 9.0: Dev

- **Hstore improvements** (Andrew Gierth)

```
# CREATE TABLE test (h hstore);
# INSERT INTO test (h) VALUES ('a=>123');
# INSERT INTO test (h) VALUES ('a=>123');
# INSERT INTO test (h) VALUES ('a=>256');
# INSERT INTO test (h) VALUES ('a=>256,b=>512');
# SELECT h, count(*) FROM test group by h;
```

h	count
"a"=>"123"	2
"a"=>"256", "b"=>"512"	1
"a"=>"256"	1



Pg 9.0: Dev

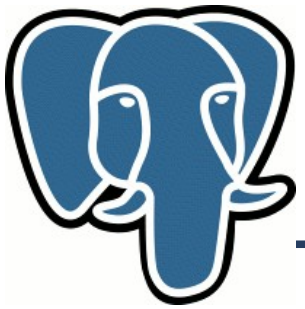
- **ORDERED AGGREGATES** (Andrew Gierth)
  - ORDER BY в агрегатах – контроль в каком порядке данные поступают в функцию-агрегат

Пример: Сортированные даты заказа (ordered)
Раньше:

```
# select buyer, sum(total), array_agg(ordered)
from ( select * from orders order by buyer,
ordered ) x group by buyer order by buyer;
```

Сейчас:

```
# select buyer, sum(total), array_agg(ordered
order by ordered ) from orders group by buyer;
```



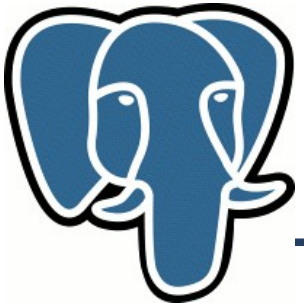
Pg 9.0: Dev

- **string_agg()** агрегатная функция
  - На два порядка быстрее array_to_string

```
# select string_agg( relname, ', ' ) from
(select relname from pg_class where relkind = 'r' limit 5 )x;
string_agg
-----------------------------------------------------------------
pg_statistic, pg_type, pg_attribute, x, active_threads_by_forum_id
```

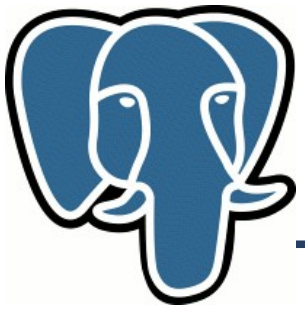
- **pg_table_size() , pg_indexes_size()** - более дружеские, чем pg_relation_size()

```
=# select  pg_table_size('_accrg7175'),
pg_indexes_size('_accrg7175'), pg_total_relation_size('_accrg7175');
 pg_table_size | pg_indexes_size | pg_total_relation_size
---------------+-----------------+------------------------
     308690944 |       669655040 |              978345984
```



Pg 9.0: Dev

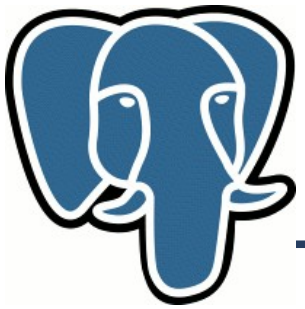
- Listen/Notify (Payloads)
  - Раньше использовалась системная таблица pg_listener как хранилище всех слушателей
  - В 9.0 используется очередь в памяти (mmap)
    - Сильно производительней
    - Вместимость 2,147,483,647 извещений
    - Совместимость с Host Standby (HS slaves пока не могут получать извещения от мастера, planned)
  - NOTIFY channel [, payload]
    - Payload – строка с length(payload) < 8K



Pg 9.0: Dev

- **PL/pgSQL by default**
- **Named function arguments** (Pavel Stehule)
  funcname(value AS arg1, anothervalue AS arg2)

```
CREATE OR REPLACE FUNCTION test(
IN x TEXT DEFAULT 'DefaultX',
IN y TEXT DEFAULT 'DefaultY',
IN z TEXT DEFAULT 'DefaultZ',
OUT o TEXT
) as $$
BEGIN
o := printf( 'x=[%] , y=[%] , z=[%]', x, y, z );
RETURN;
END;
$$ language plpgsql;

# select test('c' as z);
```



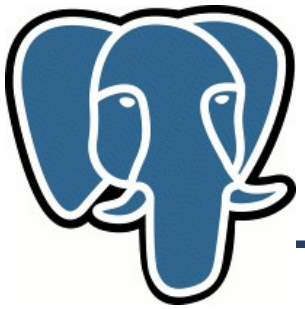
Pg 9.0: Dev

- Анонимные процедуры (Petr Jelinek)

  - Пример: Дать все привилегии пользователю webuser

```
DO $$DECLARE r record;
BEGIN
FOR r IN SELECT table_schema, table_name FROM information_schema.tables
WHERE table_type = 'VIEW' AND table_schema = 'public'
LOOP
EXECUTE 'GRANT ALL ON ' || quote_ident(r.table_schema) || '.' ||
quote_ident(r.table_name) || ' TO webuser';
END LOOP;
END$$;
```

- pl/python – Python 3.1, анонимные процедуры
- pl/perl – анонимные процедуры, переписан plperl.c, use strict, ошибки в лог - elog(WARNING) , функции quote_literal, encode_bytea,...

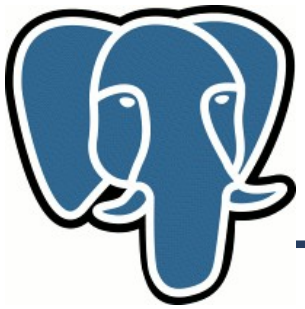


Pg 9.0: Triggers

- **Triggers on columns** (Itagaki Takahiro)

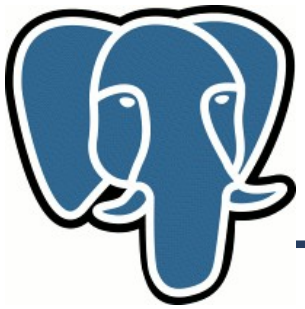
  Можно отслеживать изменение (UPDATE) только колонок, не всей строки ! Раньше надо было проверять 'ручками' :)

  CREATE TRIGGER <name>
  BEFORE UPDATE OF <col1>, <col12>, …
  ON <table>
  FOR EACH ROW
  EXECUTE PROCEDURE <procedure>;



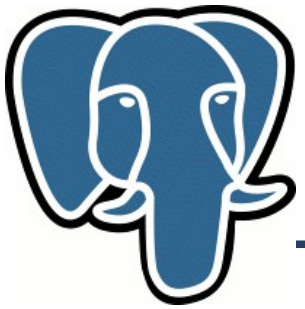
Pg 9.0: Triggers

```
CREATE OR REPLACE FUNCTION trg_test_u() RETURNS TRIGGER AS $$

BEGIN

IF NEW.title IS DISTINCT FROM OLD.title OR NEW.body IS DISTINCT
FROM OLD.BODY THEN

NEW.fts_data := to_tsvector( coalesce( NEW.title, '' ) || ' '
|| coalesce( NEW.body || '' ) );

END IF;

RETURN NEW;

END;

$$ language plpgsql;


CREATE TRIGGER trg_test_u BEFORE UPDATE ON test FOR EACH ROW
EXECUTE PROCEDURE trg_test_u();
```



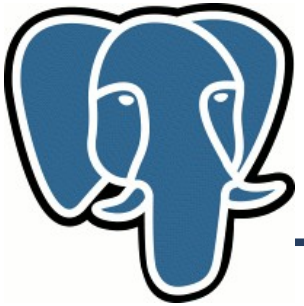
Pg 9.0: Triggers

```
CREATE OR REPLACE FUNCTION trg_test_u() RETURNS TRIGGER AS $$

BEGIN

NEW.fts_data := to_tsvector( coalesce( NEW.title, '' ) || ' '
|| coalesce( NEW.body || '' ) );

RETURN NEW;

END;

$$ language plpgsql;

CREATE TRIGGER trg_test_u BEFORE UPDATE OF title, body ON test
FOR EACH ROW EXECUTE PROCEDURE trg_test_u();
```

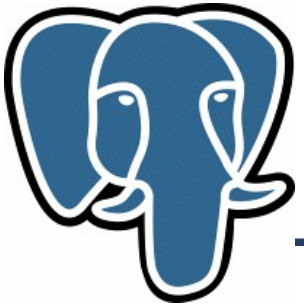


Pg 9.0: Triggers

- **Triggers with WHEN clauses** (Itagaki Takahiro)
  CREATE TRIGGER <trigger>
  AFTER UPDATE ON <table>
  FOR EACH ROW
  WHEN (OLD.* IS DISTINCT FROM NEW.*)
  EXECUTE PROCEDURE <function>;
  - Произвольное логическое выражение
  - Показывается в \d
  - Значительное ускорение для AFTER триггеров



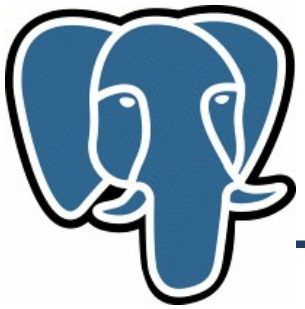
9.0 ?



Pg 9.0

- **EXCLUSION CONSTRAINTS** (Jeff Davis)
  - Общая инфраструктура для ограничений
  - UNIQUE – это оператор = (Btree)
    - любой коммутативный (x*y=y*x) оператор с индексной поддержкой (не только BTree !)

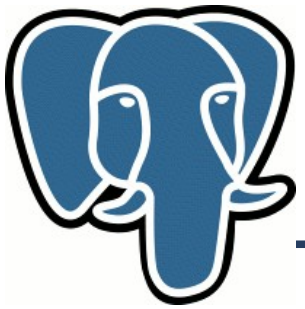
- Как реализовать уникальность более сложных типов ? Например, многоугольники ?
  - Триггер – безумно медленно !



Pg 9.0

- **EXCLUSION CONSTRAINTS** (Jeff Davis)
  Add exclusion constraints, which generalize the concept of uniqueness to support any indexable commutative operator, not just equality. Two rows violate the exclusion constraint if "row1.col OP row2.col" is TRUE for each of the columns in the constraint.

```
Не допустить: одинаковая комната в одно время
CREATE TABLE reservation (
room TEXT,
professor TEXT,
during PERIOD,
EXCLUDE USING gist (room   WITH =,
                    during WITH &&)
);
```

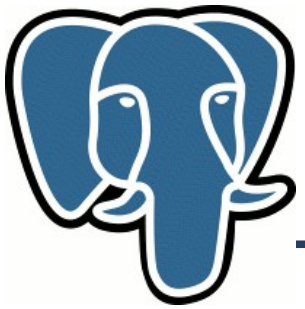


Pg 9.0

- **EXCLUSION CONSTRAINTS** (Jeff Davis)
  - Для оператора =, обычный UNIQUE
    CREATE TABLE test (i INT4,
    EXCLUDE (i WITH =));

Пример: Все комнаты должны быть одинаковые (unUnique) !

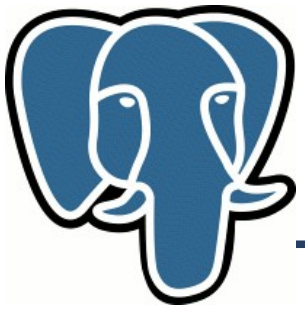
```
CREATE TABLE reservation (
room TEXT,
professor TEXT,
during PERIOD,
EXCLUDE USING gist (room WITH <>)
);
```



Pg 9.0

- **EXCLUSION CONSTRAINTS** (Jeff Davis)
  - Можно использовать разные типы индексов

```
=# create table boxes_unique (
       id integer,
       box box,
      exclude using btree (id  with =),
      exclude using gist (box with &&)
);
=# \d boxes_unique
 Table "public.boxes_unique"
 Column |  Type   | Modifiers
--------+---------+-----------
 id     | integer |
 box    | box     |
Indexes:
    "boxes_unique_box_excl" EXCLUDE USING gist (box WITH &&)
    "boxes_unique_id_excl" EXCLUDE USING btree (id WITH =)
```

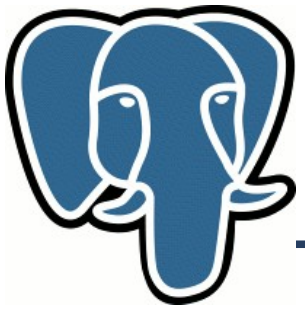


Pg 9.0

- **EXCLUSION CONSTRAINTS** (Jeff Davis)
  - Можно комбинировать разные типы индексов

Попытаемся вставить два вложенных квадрата

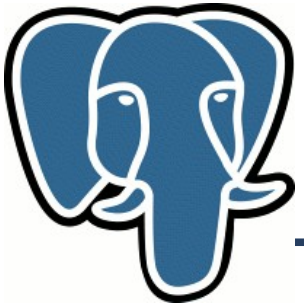
```
=# insert into boxes_unique values(1, '((0,0),(2,2))');
INSERT 0 1
Time: 1.191 ms
postgres=# insert into boxes_unique values(1, '((0,0),(1,1))');
ERROR:  conflicting key value violates exclusion constraint
"boxes_unique_id_excl"
DETAIL:  Key (id)=(1) conflicts with existing key (id)=(1).

postgres=# insert into boxes_unique values(2, '((0,0),(1,1))');
ERROR:  conflicting key value violates exclusion constraint
"boxes_unique_box_excl"
DETAIL:  Key (box)=((1,1),(0,0)) conflicts with existing key
(box)=((2,2),(0,0)).
```



Pg 9.0

- **EXCLUSION CONSTRAINTS** (Jeff Davis)
  - Можно использовать с WHERE (partial index)

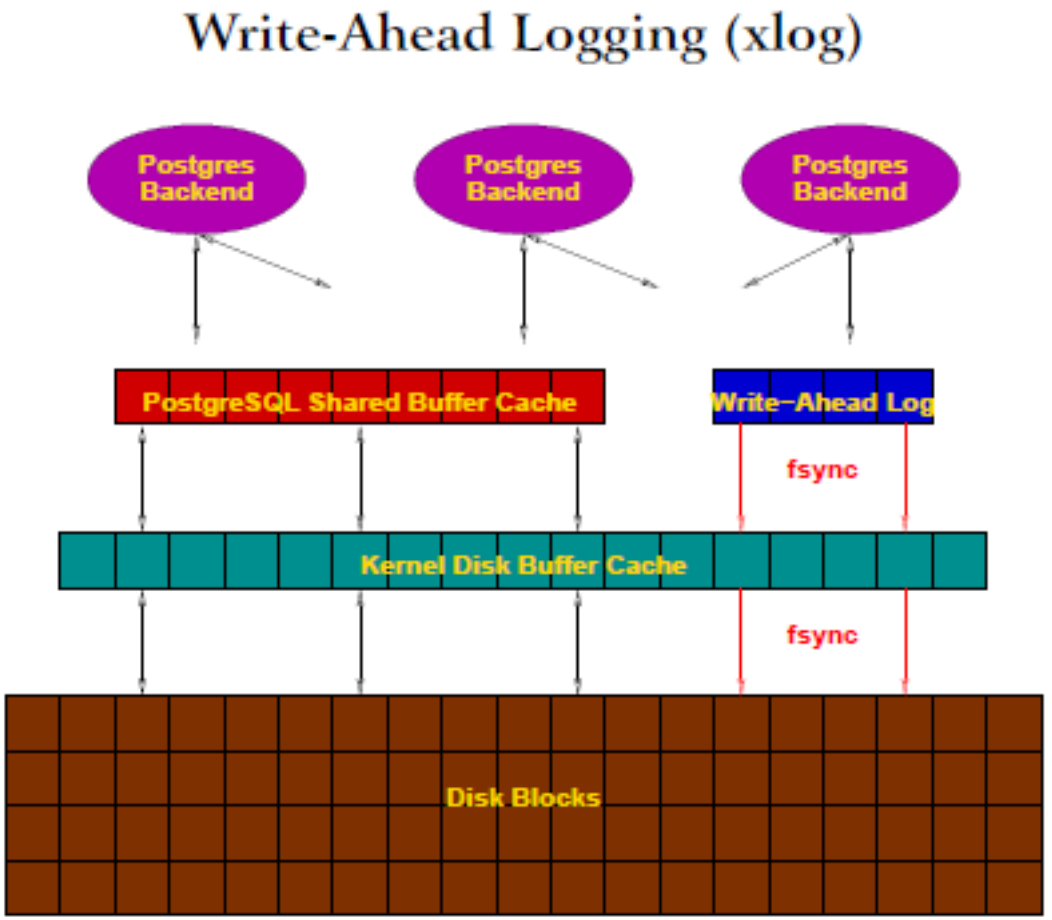
```
Одинаковые квадратики нельзя вставлять в избранную область.
=# create table boxes_unique (
       id integer,
       box box,
      exclude using btree (id  with =),
      exclude using gist (box with &&)
              WHERE      (box <@ '((0,0),(5,5))')
);
postgres=# insert into boxes_unique values(1, '((0,0),(1,1))');
INSERT 0 1
postgres=# insert into boxes_unique values(2, '((0,0),(10,10))');
INSERT 0 1
postgres=# insert into boxes_unique values(3, '((0,0),(4,4))');
ERROR:  conflicting key value violates exclusion constraint
"boxes_unique_box_excl"
DETAIL:  Key (box)=((4,4),(0,0)) conflicts with existing key (
```

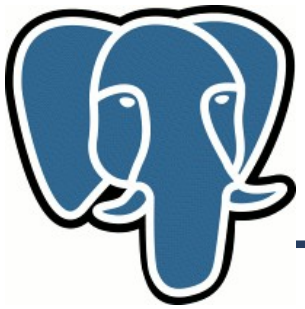


Pg 9.0: SR/HS

Streaming Replication
&
HOT Standby

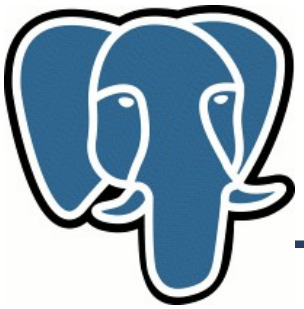
Pg 9.0: WAL (xlog)



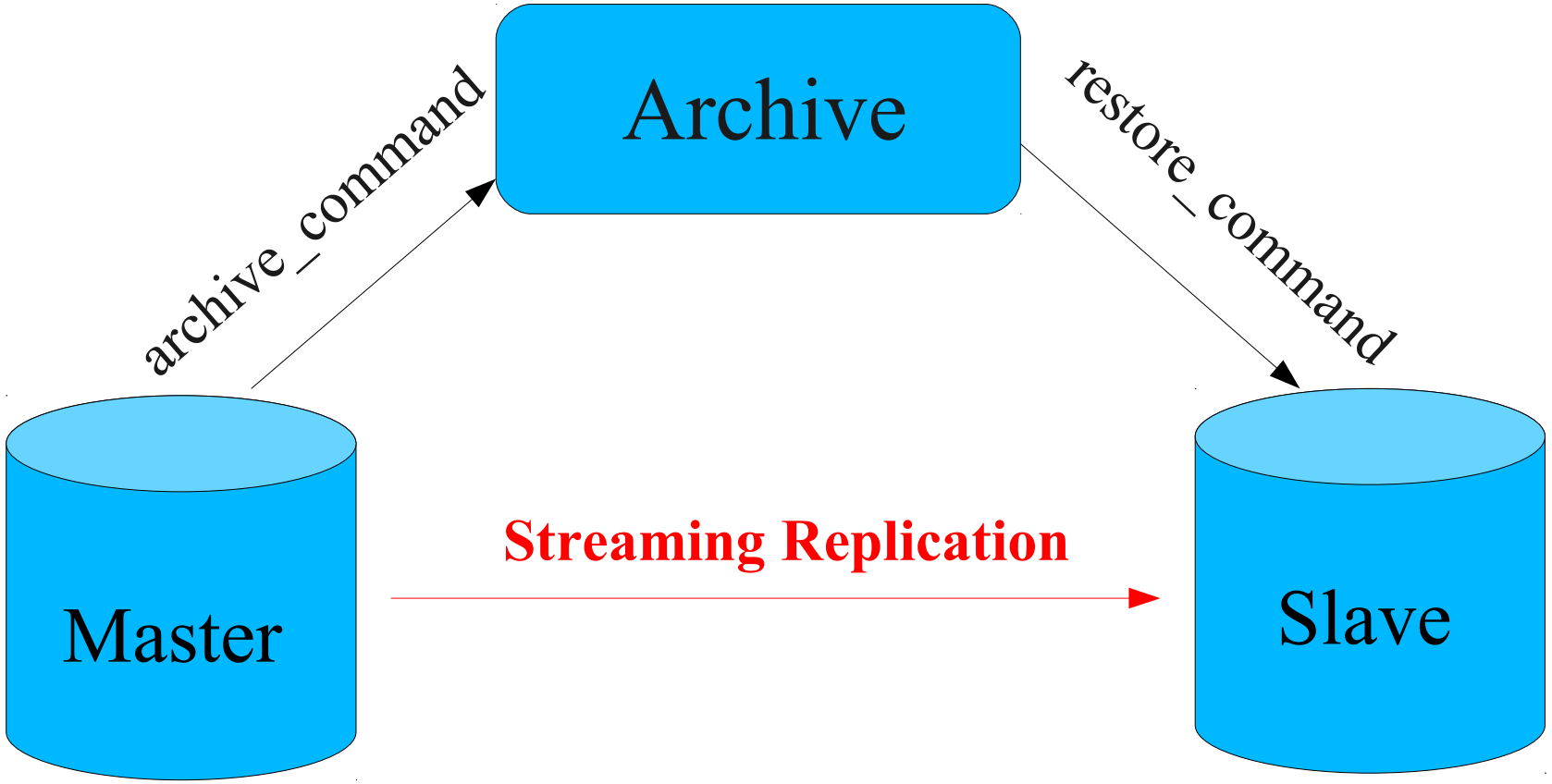


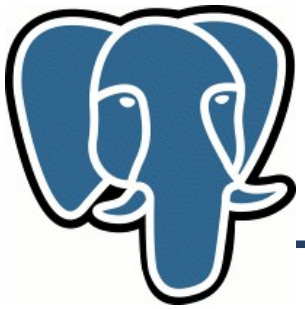
Pg 9.0: SR/HS

- Streaming Replication (Потоковая репликация)
  - WAL-записи пересылаются на standby по мере появления
  - Асинхронная репликация
  - Раньше WAL пересылался 16 MB кусками (проблема с низким WAL-трафиком, archive_timeout для более частого архивирования, много места на диске)
- Hot Standby
  - Можно выполнять r/o запросы
  - Раньше WARM Standby простаивал.



Pg 9.0: SR/HS


Archive
archive_command
restore_command
Master
Streaming Replication
Slave



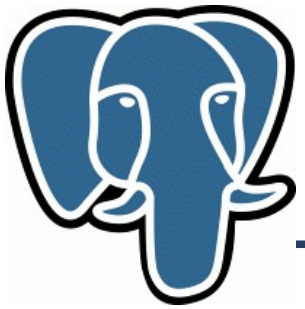
Pg 9.0: SR/HS

- Настройка мастера (postgresql.conf)
  - Настройка постоянного архивирования WAL

```
archive_mode = on
archive_command = 'cp i %p  ARCHIVE_DIR/%f <
/dev/null'
max_wal_senders = 5
```

  - Запуск мастера и создание опорного бэкапа

```
$ bin/pg_ctl start -D data-master
$ bin/psql postgres -c
"SELECT pg_start_backup('mybackup', true)"
0/1000020
$ cp -a data-master/ data-standby
$ bin/psql postgres -c "SELECT pg_stop_backup()"
0/10000D0
```



Pg 9.0: SR/HS

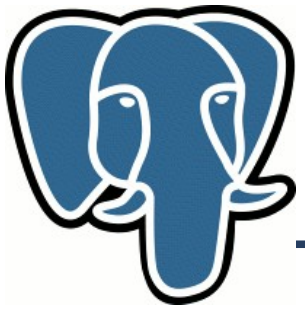
- Настройка standby (data-standby/recovery.conf)
  - Восстановиться из опорного бэкапа

```
restore_command = 'cp ARCHIVE_DIR/%f %p'
  standby_mode = 'true'
  primary_conninfo = 'host=localhost port=5432'
  trigger_file='/tmp/standbytrigger'
```

  - Запуск сервера

    port=5433 (postgresql.conf)

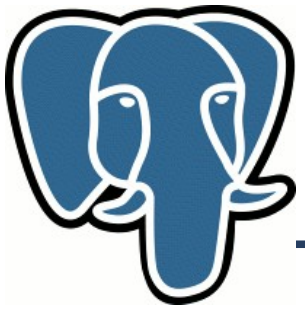
    $ bin/postmaster -D data-standby



Pg 9.0: SR/HS

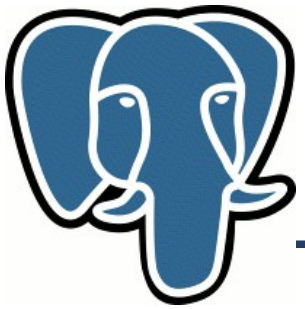
```
$ ps ax| grep postgres
17451 pts/4 S 0:00 postgres -D data-master
17455 ? Ss 0:00 postgres: writer process
17456 ? Ss 0:00 postgres: wal writer process
17457 ? Ss 0:00 postgres: autovacuum launcher process
17458 ? Ss 0:00 postgres: archiver process last was 000000010000000000000005
17459 ? Ss 0:00 postgres: stats collector process
17573 ? Ss 0:00 postgres: startup process recovering 000000010000000000000006
17576 ? Ss 0:00 postgres: writer process
17578 ? Ss 0:00 postgres: stats collector process
17584 ? Ss 0:00 postgres: wal receiver process streaming 0/6014708
17585 ? Ss 0:49 postgres: wal sender process postgres 127.0.0.1(56056)
Streaming 0/6014708
```

walsender на мастере
Walreceiver на standby



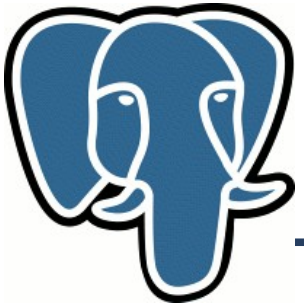
Pg 9.0: SR/HS

- Failover (standby → master)
  - Сервер обнаруживает специальный файл (trigger_file в recovery.conf)
  - Создается новая ветка WAL
  - Существующие соединения (только на чтение) становятся read-write
- Failover можно использовать с популярными решениями типа Hearbeat
- Мастер может стать standby после восстановления опорного бэкапа



Pg 9.0: SR/HS

- SR/HS и долгоживущие запросы на slave. Записи на мастере могут быть удалены за время работы запросов на slave. Регулировки:
  - Задержка удаления старых записей на мастере
    vacuum_defer_cleanup_age (postgresql.conf)
  - Задержка использования WAL на slave
    max_standby_delay (postgresql.conf)
    - 30 секунд по-умолчанию
    - -1 – ждать до бесконечности (чтобы не отменять долгий запрос)



Спасибо за внимание !

ВОПРОСЫ ?