PostgresPro

Построение защищенных БД с использованием мандатного разграничения доступа в PostgreSQL

Валерий Попов
Николай Чадаев

Защита данных

- Предметная область
- Решение, его составные части
- Конфигурации и Настройки
- Тестирование и модификации
- Выводы

Предметная область:

«POSTGRES — Первое знакомство» v.5

П.Лузанов, Е.Рогов, И.Левшин.

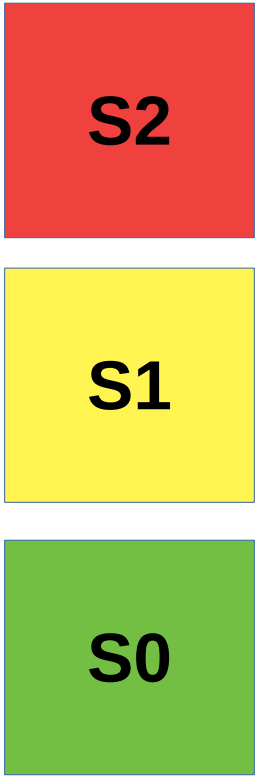
ООО ППГ 2019

Демонстрационная база «Авиаперевозки».

Будут защищаются данные о бронировании авиабилетов.

Уровни безопасности данных:

PostgresPro



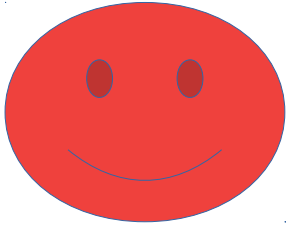
- Секретные

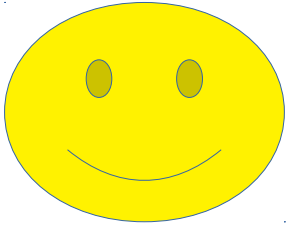
- Для служебного пользования

- Открытые

Безопасный доступ к данным:

PostgresPro





?

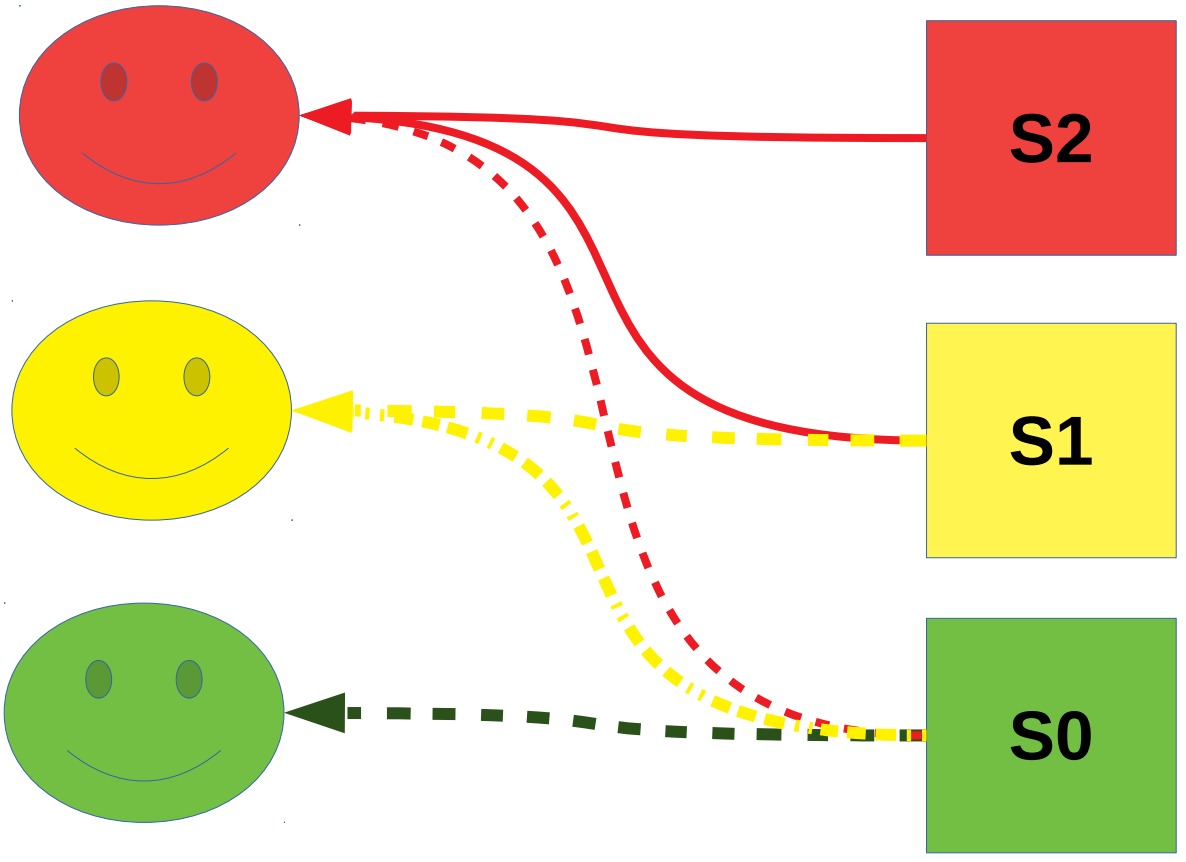
S2

S1

S0

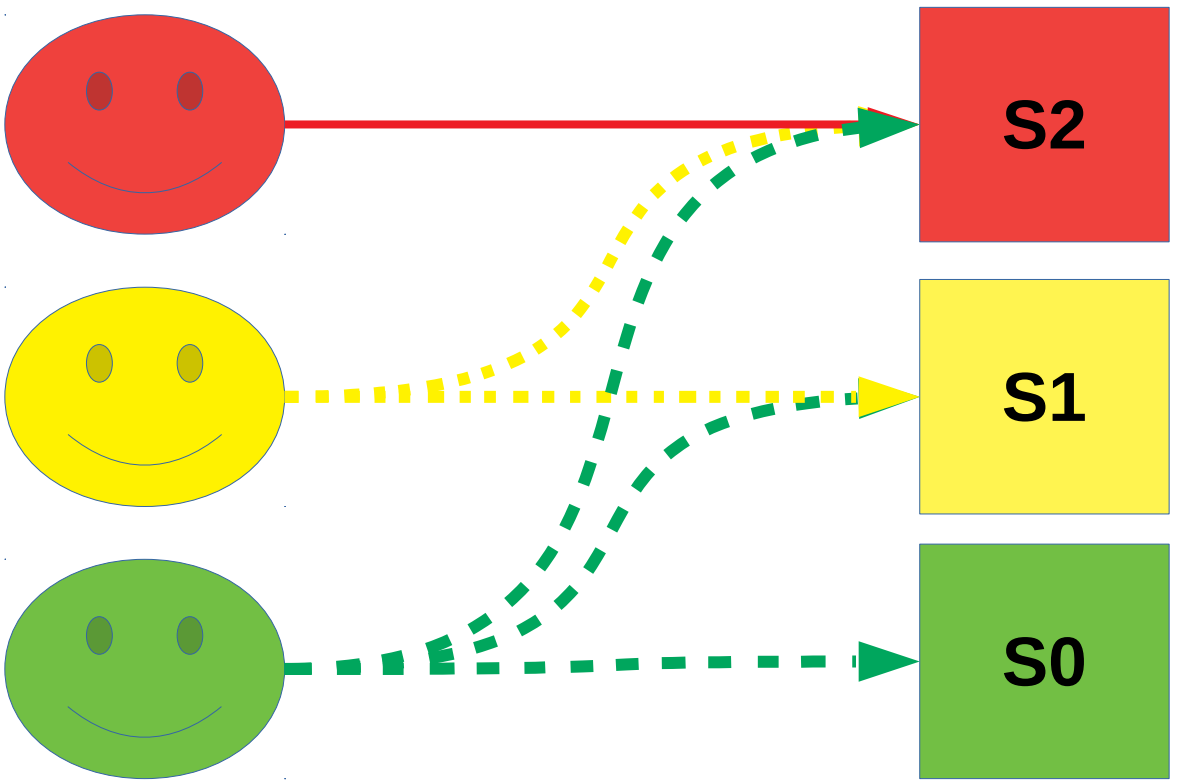
Безопасное чтение:

PostgresPro



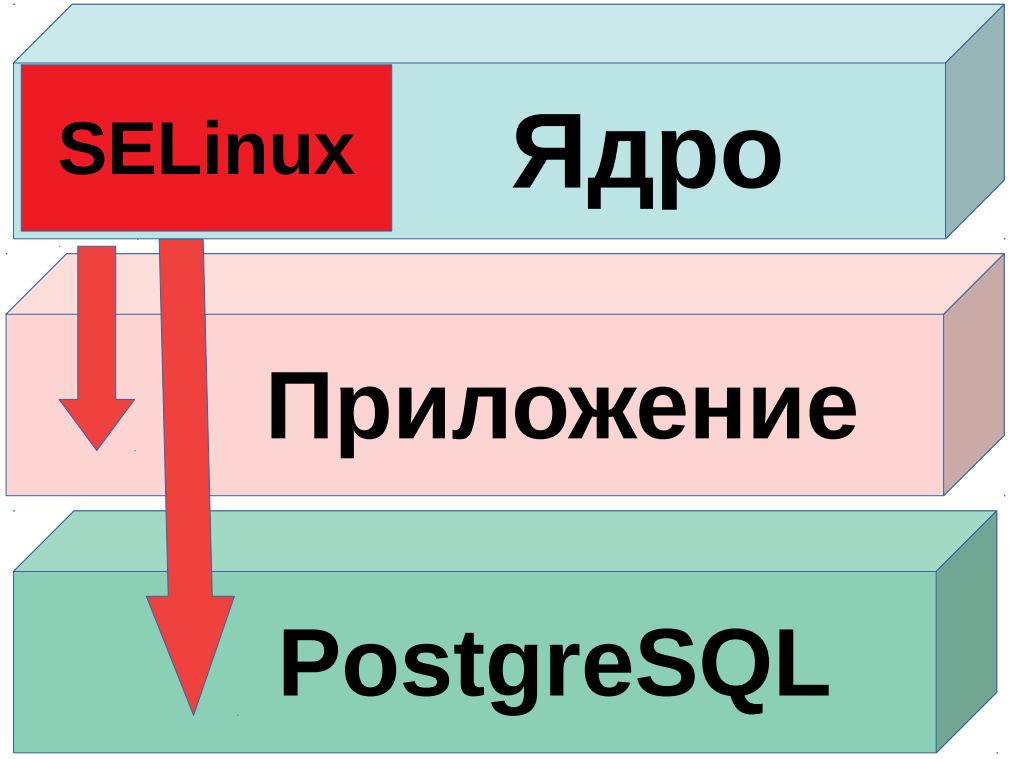
Безопасная запись:

PostgresPro



PostgresPro

SELinux, функциональность:



PostgresPro

SELinux, метка безопасности:

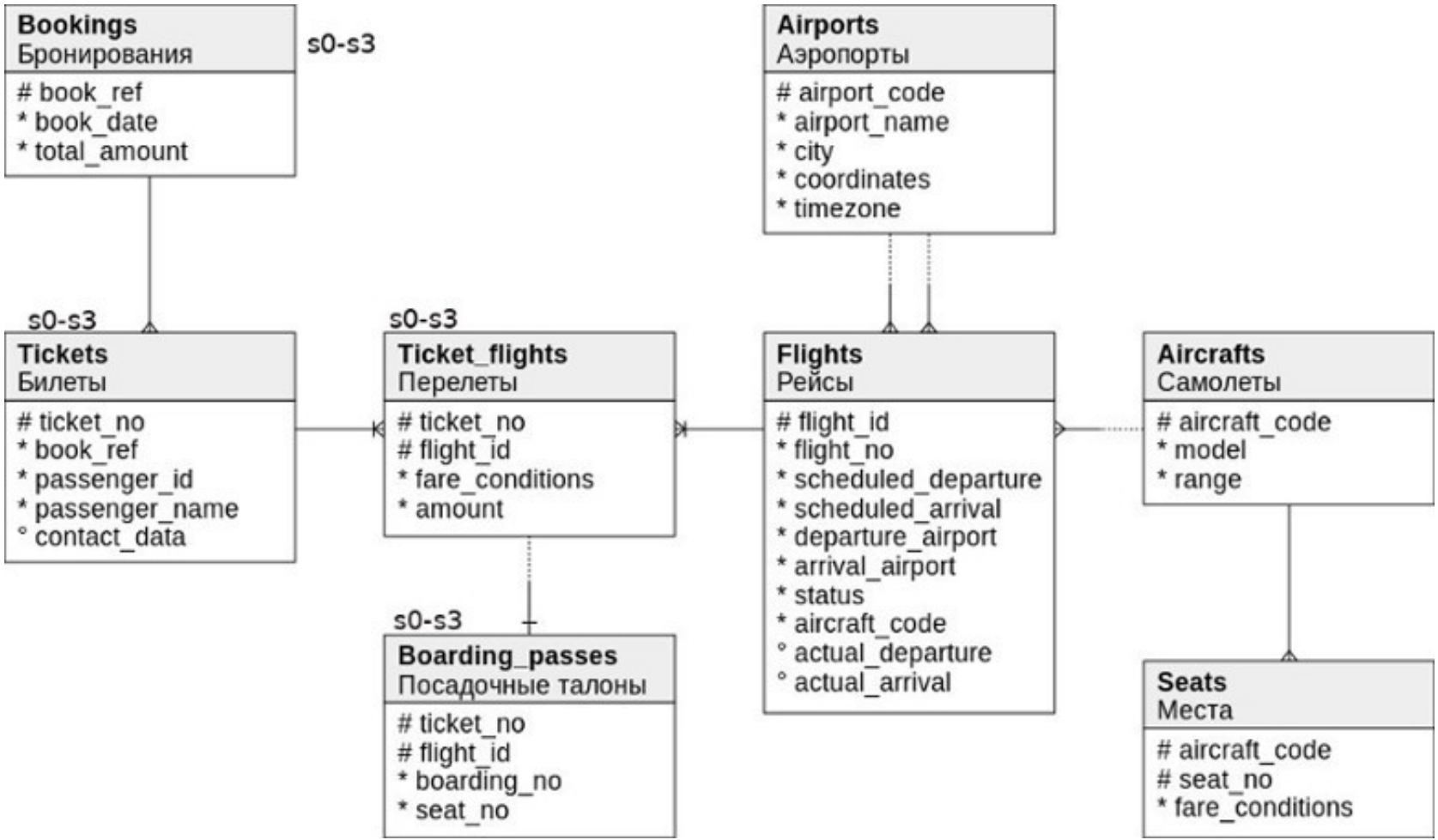
<SE-user>:<SE-role>:
<SE-type/domain>:
<Sensitivity>: <Category>

Пример:

```
demo_x5=# SELECT sepgsql_getcon();
-----------------------------------------
system_u:object_r:sysadm_t:s0:c0
```

Модель базы:

PostgresPro



Защищённое решение:

- CentOS 7.6
- SELInux MLS-policy 1.20
- SEPGSQL
- PostgresPro Enterprise 11.5
  + RLS-policy
  + Хранилище меток строк.

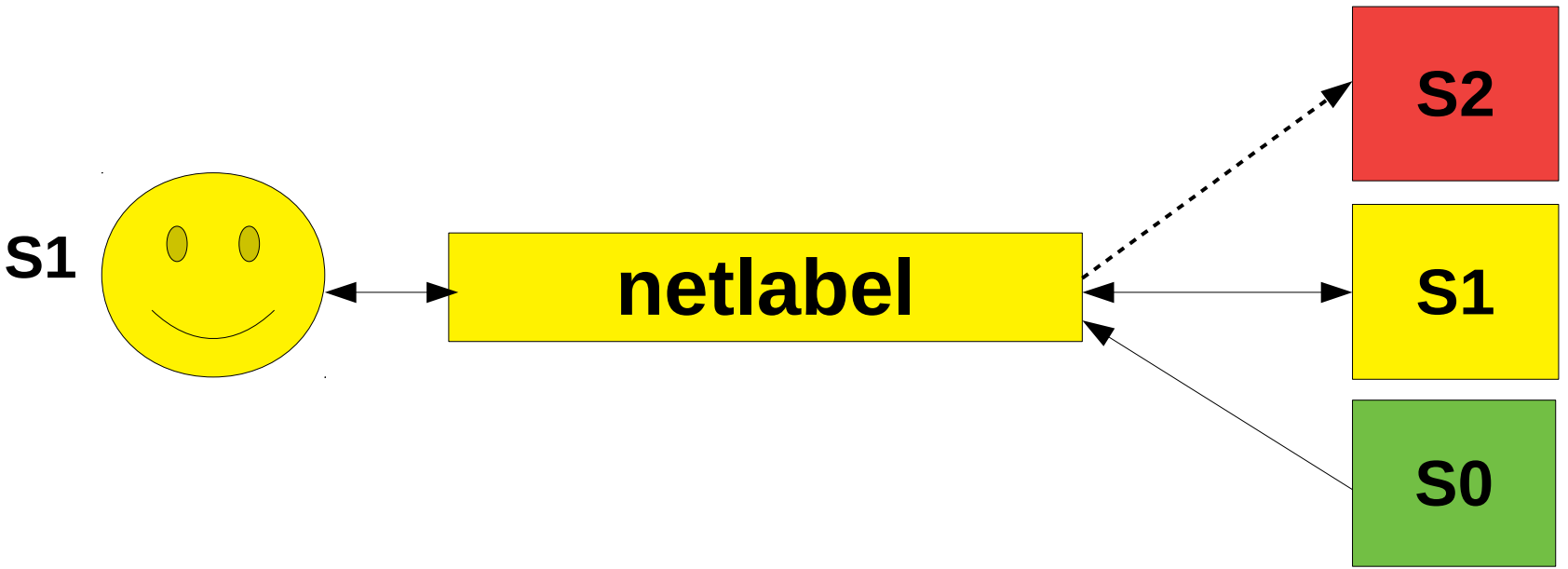
SELinux. Клиент, Сеть, БД:

PostgresPro

Клиент
SE-user, Linux-user
s1:c0.c5

Сеть
IP xxx.xxx.xxx.xxx
s1:c0.c5

PostgreSQL
SE-user, DB-user
s1:c0.c5



Защищаемые объекты, метки:

PostgresPro

system_u:object_r:

```
sepgsql_db_t:s0-s3:c0.c15                    -  База
sepgsql_schema_t:s0-s3:c0.c15                -  Схема
sepgsql_seq_t:s0                  - Последовательность
sepgsql_table_t:s0-s3:c0.c15                 - Таблица
sepgsql_table_t:s0                           - Столбец
sepgsql_tuple_t:s0:c0.c15                    - Строка
```

PostgresPro

Защищаемые объекты:

Строка таблицы, как хранить метку ??

PostgresPro

Хранение метки строки таблицы:

Вариант №1 Централизованное хранение, логическая модель:

Сущность **«Метка строки»**, её атрибуты:

«Имя таблицы» - *Родительский объект*
«Глобальный ID строки» - *Защищаемый объект*
«Метка строки» - *Контекст безопасности*

PostgresPro

Хранение меток строк:

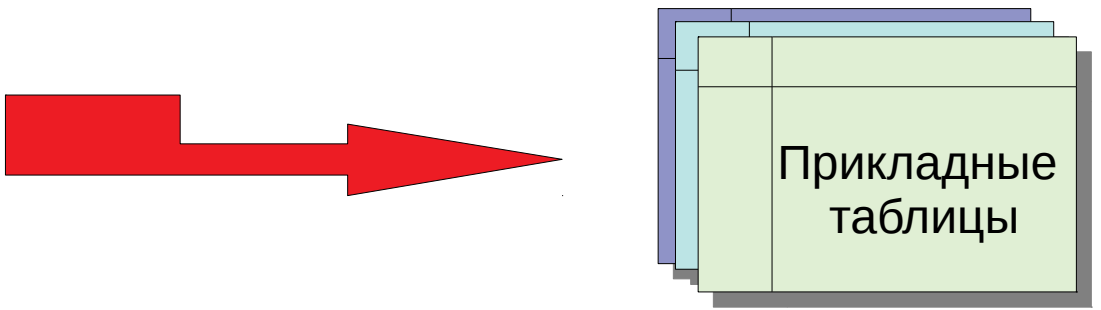
Вариант №1 Централизованное хранение:

```
CREATE SEQUENCE sepgsql.all_rec_id_seq
                                    INCREMENT 1 START 1;
--
CREATE TABLE sepgsql.sepgsql_tuple_z (
              table_name      text  NOT NULL
             ,rec_id          int   NOT NULL
            ,security_label   text  NOT NULL
) PARTITION BY LIST (table_name);
_
ALTER TABLE sepgsql.sepgsql_tuple_z
           ADD CONSTRAINT pk_sepgsql_tuple_z
                        PRIMARY KEY (table_name, rec_id);
```

Хранение меток строк:

```
ALTER TABLE <имя таблицы>
    ADD COLUMN  rec_id  int  NOT NULL
          DEFAULT nextval  ('sepgsql.all_rec_id_seq'::regclass);

ALTER TABLE <имя таблицы>
          ADD CONSTRAINT <имя таблицы>_ak1 UNIQUE (rec_id);
```



Модификация прикладных таблиц

PostgresPro

SEPGSQL – Дополнительные функции:

«GET» - получить метку защищаемого объекта

«SET» - установить метку объекта

«CREATE» - создать метку строки, при этом учитываются контексты безопасности таблицы и клиента.

«CHECK» - сравнение меток клиента и строки.

«AUTH» - отображение дискреционных прав

Триггерные функции обрабатывают события **«Insert»**, **«Delete»**, **«Update»** в прикладных таблицах.

Защищаемые объекты:

PostgresPro

Как управлять доступом к строкам таблицы ?

Политики RLS на основе меток SELinux.

Политики RLS:

PostgresPro

Начиная с версии Postgres 9.5 могут быть использованы для управления доступом к строкам таблиц.

SELECT
INSERT
UPDATE
DELETE

USING <выражение> (для существующей строки)
WITH CHECK (для новой строки)

Должны учитывать права выданные в MLS-Policy для защищаемого объекта класса "db_table" и пользовательского домена.

Политики RLS, определения:

PostgresPro

```
ALTER TABLE <имя таблицы>  ENABLE ROW LEVEL SECURITY;



CREATE POLICY mls_select ON <имя таблицы> FOR
SELECT
  USING
  (sepgsql_check_tuple_label_t ('<имя таблицы>',
      rec_id)
  );
```

Политики RLS, определения:

PostgresPro

```
CREATE POLICY mls_insert ON <имя таблицы> FOR
INSERT
  WITH CHECK (
    sepgsql_create_tuple_label ('<имя таблицы>')
                          IS NOT NULL
  );
```

Политики RLS, определения:

PostgresPro

```
CREATE POLICY mls_update ON <имя таблицы> FOR
UPDATE
 USING (
   sepgsql_check_tuple_label_t ('<имя таблицы>', rec_id)
  )
  WITH CHECK (
   sepgsql_check_tuple_label_t (
                  '<имя таблицы>', rec_id, 'update'
        )
  );
```

Политики RLS, определения:

PostgresPro

```
CREATE POLICY mls_delete ON <имя таблицы> FOR
DELETE
 USING (
   sepgsql_check_tuple_label_t ('<имя таблицы>', rec_id)
  )
  WITH CHECK (
   sepgsql_check_tuple_label_t (
                        '<имя таблицы>', rec_id, 'delete'
        )
  );
```

Субъект и объект:

Пользователь:

```
demo_x6=> SELECT session_user, sepgsql_getcon();
-------------+-------------------------------------
session_user |        sepgsql_getcon
-------------+-------------------------------------
user1        | staff_u:staff_r:staff_t:s1:c0
```

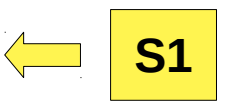


Таблица:

```
demo_x6=> SELECT sepgsql_get_table_label ('bookings.bookings');
-------------------------------------------------------------
       sepgsql_get_table_label
-------------------------------------------------------------
system_u:object_r:sepgsql_table_t:s0-s3:c0.c15
```

Объект, уровни безопасности строк:

Хранилище меток:

```
demo_x6=> SELECT * FROM sepgsql.sepgsql_tuple_z
   WHERE  (table_name = 'bookings.bookings') AND ( rec_id >= 10037033);
    table_name      |  rec_id  |                security_label
--------------------+----------+--------------------------------------
 bookings.bookings | 10037033 | system_u:object_r:sepgsql_tuple_t:s0:c0
 bookings.bookings | 10037034 | system_u:object_r:sepgsql_tuple_t:s1:c0
 bookings.bookings | 10037035 | system_u:object_r:sepgsql_tuple_t:s0:c0
```

Запрос:

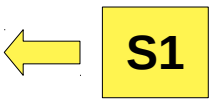
```
demo_x6=> SELECT book_ref, book_date, total_amount, rec_id FROM
bookings.bookings WHERE ( rec_id >= 10037033);
 book_ref |       book_date         | total_amount |  rec_id
----------+-------------------------+--------------+----------
 FFFFF1   | 2016-05-31 21:44:00+03 |      7200.00 | 10037033
 FFFFF7   | 2016-08-29 21:12:00+03 |     73600.00 | 10037034
 FFFFF9   | 2015-12-09 09:19:00+03 |     62600.00 | 10037035
```

Политики RLS в действии:

PostgresPro

Пользователь:

```
demo_x6=> SELECT session_user, sepgsql_getcon();
--------------+-------------------------------------
session_user |        sepgsql_getcon
--------------+-------------------------------------
user1        | staff_u:staff_r:staff_t:s1:c0
```



Запрос:

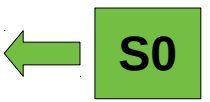
```
demo_x6=> SELECT book_ref, book_date, total_amount, rec_id
             FROM bookings.bookings WHERE ( rec_id >= 10037033);
 book_ref |       book_date        | total_amount |  rec_id
----------+------------------------+--------------+----------
 FFFFF1   | 2016-05-31 21:44:00+03 |      7200.00 | 10037033
 FFFFF7   | 2016-08-29 21:12:00+03 |     73600.00 | 10037034
 FFFFF9   | 2015-12-09 09:19:00+03 |     62600.00 | 10037035
```

Политики RLS в действии:

PostgresPro

Пользователь:

```
demo_x6=> SELECT session_user, sepgsql_getcon();
--------------+-------------------------------------
session_user |        sepgsql_getcon
--------------+-------------------------------------
user0        | staff_u:staff_r:staff_t:s0:c0
```



Запрос:

```
demo_x6=> SELECT book_ref, book_date, total_amount, rec_id
             FROM bookings.bookings WHERE ( rec_id >= 10037033);
 book_ref |        book_date         | total_amount |  rec_id
----------+--------------------------+--------------+----------
 FFFFF1   | 2016-05-31 21:44:00+03 |      7200.00 | 10037033
 FFFFF9   | 2015-12-09 09:19:00+03 |     62600.00 | 10037035
```

Данные уровня «S1», («book_ref» = FFFFF7) недоступны для пользователя «user0»

PostgresPro

Конфигурации и Настройки:

- CentOS 7.6
- SELinux
- PostgresPro Enterprise 11.5
- База данных

CentOS 7.6, установка пакетов:

```
#>yum install netlabel_tools
#>yum install selinux-policy-mls
#>yum install libsemanage-python
#>yum install policycoreutils-python
#>yum install setools-libs
#>yum install setools-console
#>yum install selinux-policy-devel
#>yum install xinetd

#>yum install postgrespro-ent-11
```

Настройка «/etc/selinux/config»:

PostgresPro

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#   enforcing  - SELinux security policy is enforced.
#   permissive - SELinux prints warnings instead of enforcing.
#   disabled   - No SELinux policy is loaded.
```

SELINUX=permissive

```
# SELINUXTYPE= can take one of three two values:
#  targeted  - Targeted processes are protected,
#  minimum   - Modification of targeted policy.
               Only selected processes are protected.
#  mls - Multi Level Security protection.
```

SELINUXTYPE=mls

Настройка SELinux, relabeling:

PostgresPro

```
#>pwd
/
#>touch .autorelabel
#>l .autorelabel
-rw-r--r--. 1 root root 0 дек 20 12:34 /.autorelabel
#>
#>init 6
```

При перезагрузке на консоль будет выведено сообщение.

** Предупреждение. Требуется повторная расстановка меток политики SELinux MLS

PostgresPro

SELinux, смотрим что получилось:

```
#>sestatus

SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             mls
Current mode:                   permissive
Mode from config file:          permissive
Policy MLS status:              enabled
Policy deny_unknown status:     denied
Max kernel policy version:      31
```

PostgresPro Enterprise 11.5, настройки:

```
#>l
/var/lib/pgpro/ent-11/data/postgresql.conf

-rw-------. 1 postgres postgres 21919 ноя 27 19:26
                                /var/lib/pgpro/ent-11/data/postgresql.conf

# - Kernel Resource Usage -

shared_preload_libraries = 'sepgsql'

#>service postgrespro-ent-11 restart
```

PostgresPro Enterprise 11.5, контексты процессов:

system_u:system_r:

```
#>ps efZ -C postgres
LABEL                          PID                                  COMMAND
init_t:s0-s15:c0.c1023 3288 ? Ss    0:00 /opt/pgpro/ent-11/bin/postgres
                                           -D  /var/lib/pgpro/ent-11/data
init_t:s0-s15:c0.c1023 3674 ? Ss    0:00  \_ postgres: logger
init_t:s0-s15:c0.c1023 3731 ? Ss    0:00  \_ postgres: checkpointer
init_t:s0-s15:c0.c1023 3732 ? Ss    0:00  \_ postgres: background writer
init_t:s0-s15:c0.c1023 3733 ? Ss    0:00  \_ postgres: walwriter
init_t:s0-s15:c0.c1023 3734 ? Ss    0:00  \_ postgres: autovacuum launcher
init_t:s0-s15:c0.c1023 3735 ? Ss    0:00  \_ postgres: stats collector
init_t:s0-s15:c0.c1023 3736 ? Ss    0:00  \_ postgres:
                                            logical replication launcher
init_t:s0-s15:c0.c1023 3747 ? Ss    0:00  \_ postgres: cfs-worker-0
#>
```

PostgresPro

PostgresPro Enterprise 11.5, установка расширений:

```
template1=# CREATE EXTENSION pg_pro_selinux;
CREATE EXTENSION

demo_x6=# CREATE EXTENSION pg_pro_selinux;
CREATE EXTENSION

demo_x6=# CREATE EXTENSION pg_promls_selinux;
CREATE EXTENSION
```

Оформление модуля «sepgsql» в виде расширения «pg_pro_selinux», гарантирует сохранение контекста безопасности базы при её восстановлении.

PostgresPro

База данных, постинсталляционные скрипты:

Эти скрипты, должны быть выполнены после установки расширения «pg_promls_selinux» в прикладной базе.

pg_promls_selinux_security_labels–0.3.sql

0_pg_promls_selinux_security_add_one–0.3.sql

1_pg_promls_selinux_security_rvk_grt–0.3.sql

Наполнение скриптов

«0_pg_promls_selinux_security_add_one–0.3.sql»
«1_pg_promls_selinux_security_rvk_grt–0.3.sql»

специфично для каждой прикладной базы.

PostgresPro

Скрипт «pg_promls_selinux_security_labels–0.3.sql»

Установка меток для функций из расширения выполняющихся в доверенном контексте «sepgsql_trusted_proc_exec_t».

0.3 – номер текущей версии расширения «pg_promls_selinux»

PostgresPro

Скрипт «0_pg_promls_selinux_security_add_one–0.3.sql»

Модификация прикладной базы для работы в режиме MLS

1) Создать дополнительные секции.
2) Модифицировать прикладные таблицы.
3) Установить метки на таблицы и секции.
4) Создать метки строк.
5) Связать триггеры и прикладные таблицы.
6) Создать политики RLS.

0.3 – номер текущей версии расширения «pg_promls_selinux»

PostgresPro

Скрипт «1_pg_promls_selinux_security_rvk_grt–0.3.sql»

Установка необходимых дискреционных прав в прикладной базе

1) Определение основных ролей.

2) Отзыв прав у псевдороли PUBLIC.

3) Назначения дискреционных прав на объекты БД для основных ролей.

0.3 – номер текущей версии расширения «pg_promls_selinux»

DB role = Linux User:

PostgresPro

```
CREATE ROLE user0 LOGIN
 NOSUPERUSER INHERIT NOCREATEDB
    NOCREATEROLE NOREPLICATION
            VALID UNTIL '9999-12-31 00:00:00';

CREATE ROLE user1 LOGIN
 NOSUPERUSER INHERIT
    NOCREATEDB NOCREATEROLE NOREPLICATION
          VALID UNTIL '9999-12-31 00:00:00';
```

Пользователи Linux:

```
#> adduser user0
#> adduser user1

#> passwd user0 -> «user000»
#> passwd user1 -> «user111»
```

SE-users + Linux-users = SE-logins:

PostgresPro

```
#>semanage user -l
      staff_u  s0-s15:c0.c1023  staff_r
```

```
#>semanage login -a -s staff_u \
                        -r s0:c0.c1023 user0
```

```
$>runcon -l s0:c0 psql demo_x6 -U user0
demo_x6=> SELECT session_user, sepgsql_getcon();
 session_user |          sepgsql_getcon
--------------+-----------------------------
 user0        | staff_u:staff_r:staff_t:s0:c0
```

PostgresPro

Тестирование:

- Оценка производительности
- Модификация решения

PostgresPro

Оценка производительности:

Влияние:

- SEPGSQL
- RLS
- Централизованного хранилища меток

Что осталось за «бортом» ?

Нагрузочное тестирование в условиях мандатного разделения доступа

User0, уровень s0:c0, выполняет запрос:

PostgresPro

```
SELECT tf.ticket_no
      ,f.departure_airport
      ,f.arrival_airport
      ,f.scheduled_arrival
      ,lead (f.scheduled_departure) OVER w AS next_departure
      ,(lead (f.scheduled_departure) OVER w -
f.scheduled_arrival) AS gap

FROM bookings b
   JOIN tickets t          ON (t.book_ref  = b.book_ref)
   JOIN ticket_flights tf ON (t.ticket_no = tf.ticket_no)
   JOIN flights f          ON (f.flight_id = tf.flight_id)
WHERE (b.book_date = bookings.now()::date - INTERVAL '7 day')

WINDOW w AS
   (PARTITION BY tf.ticket_no ORDER BY f.scheduled_departure);
```

Результат №0:

PostgresPro

Выполнил «user0» (Работает RLS)

Время: **467061,077 мс (07:47,061)** **!!!**

Выполнил «postgres» (Владелец базы, без RLS)

Время: **9,679 мс**

```
     ticket_no   | departure_airport | arrival_airport |   scheduled_arrival    |     next_departure     |       gap
-- --------------+-------------------+-----------------+------------------------+------------------------+-----------------
    0005432985626 | KHV               | DME             | 2016-10-16 19:55:00+03 | 2016-10-25 19:40:00+03 | 8 days 23:45:00
    0005432985626 | DME               | KHV             | 2016-10-26 03:40:00+03 |                        |
    0005435653879 | DME               | URJ             | 2016-10-18 12:00:00+03 | 2016-10-19 10:55:00+03 | 22:55:00
    0005435653879 | URJ               | TJM             | 2016-10-19 11:20:00+03 | 2016-10-28 06:30:00+03 | 8 days 19:10:00
    0005435653879 | TJM               | URJ             | 2016-10-28 06:55:00+03 | 2016-10-28 15:35:00+03 | 08:40:00
    0005435653879 | URJ               | DME             | 2016-10-28 17:50:00+03 |                        |
    0005435886245 | LED               | VKO             | 2016-10-20 10:10:00+03 | 2016-10-25 10:25:00+03 | 5 days 00:15:00
    0005435886245 | VKO               | LED             | 2016-10-25 11:15:00+03 |                        |
    0005435886246 | LED               | VKO             | 2016-10-20 10:10:00+03 | 2016-10-25 10:25:00+03 | 5 days 00:15:00
    0005435886246 | VKO               | LED             | 2016-10-25 11:15:00+03 |                        |
   (10 строк)
```

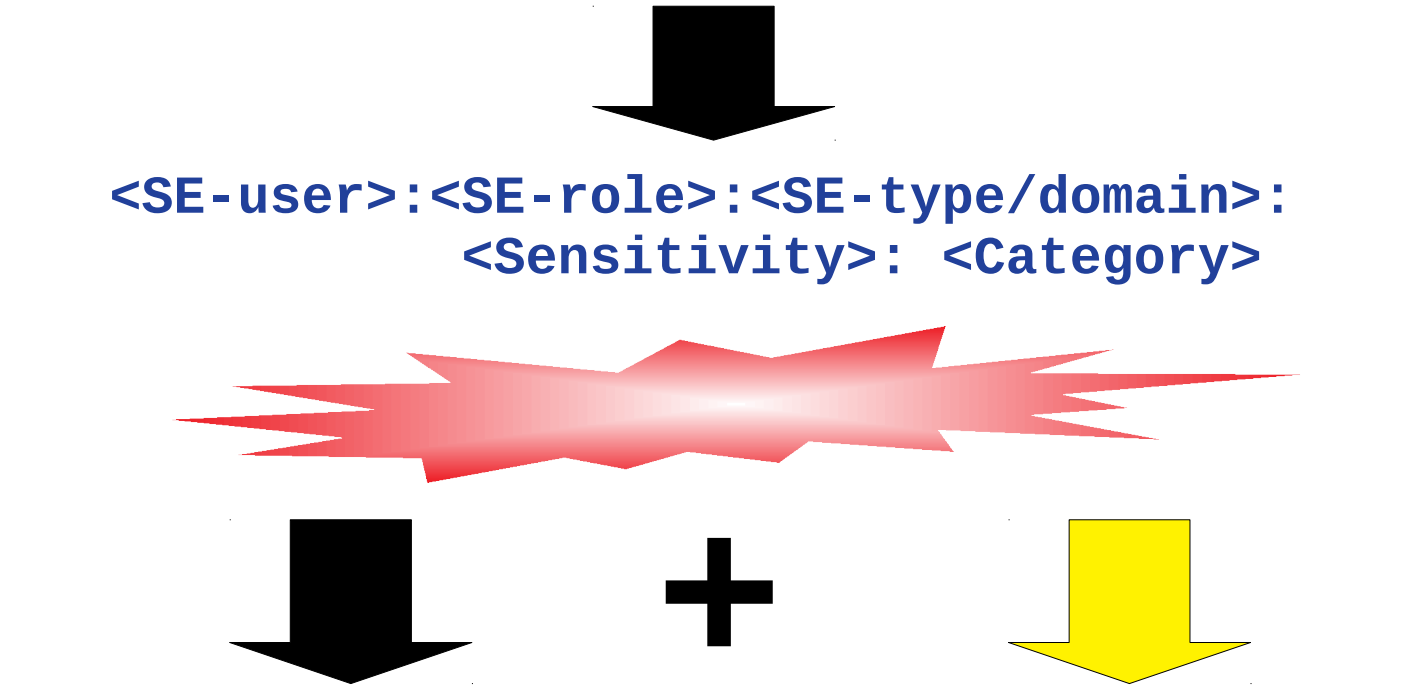
Анализ плана запроса:

PostgresPro

```
->  Nested Loop  (cost=0.43..580385.87 rows=1 width=14)
          (actual time=259074.599..463827.555 rows=4 loops=1)
  ->  Seq Scan on bookings b
                                (cost=0.00..580359.93 rows=2 width=7)
        (actual time=259074.276..463826.472 rows=3 loops=1)
                        Filter:
(sepgsql_check_tuple_label_t ('bookings.bookings'::text,
                                  rec_id, true
                                )
      AND
(book_date =
  (('2016-10-13 17:00:00+03'::timestamp with time zone)::date -
    '7 days'::interval)
             )
            Rows Removed by Filter: 2111107
```

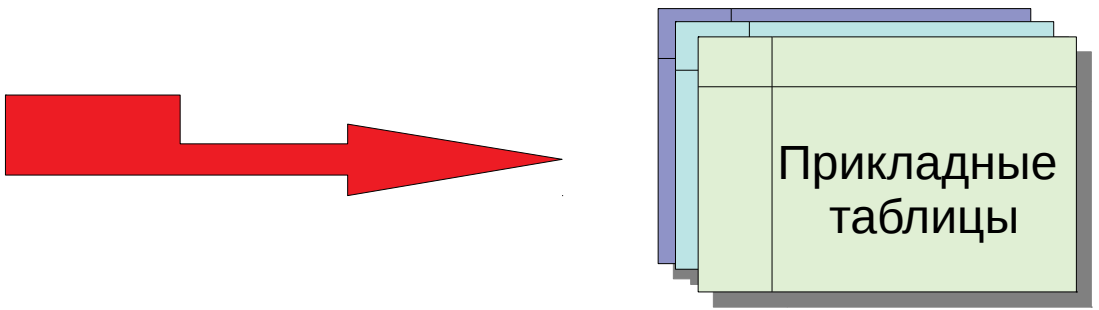
Модификация архитектуры:

PostgresPro



Часть метки в строку таблицы:

```
ALTER TABLE <имя таблицы>
    ADD COLUMN  s_lvs varchar (10) NOT NULL
        DEFAULT
            sepgsql_create_lvl_label (sepgsql_getcon());
```



Модификация прикладных таблиц

Новые определения политик:

PostgresPro

```
CREATE POLICY mls_select ON <имя таблицы> FOR
SELECT
  USING
  (sepgsql_check_tuple_label
                    (sepgsql_getcon(), s_lvs)
  );

CREATE POLICY mls_select ON <имя таблицы> FOR
SELECT
  USING
    (sepgsql_check_tuple_label_t
                          ('<имя таблицы>', rec_id)
    );
```

Новые определения политик:

PostgresPro

```
CREATE POLICY mls_update ON <имя таблицы> FOR
UPDATE
    USING (
      sepgsql_check_tuple_label (sepgsql_getcon(), s_lvs)
    )
    WITH CHECK (sepgsql_check_tuple_label_t (
                                '<имя таблицы>', rec_id, 'update'
                  )
    );

CREATE POLICY mls_update ON <имя таблицы> FOR
UPDATE
  USING (sepgsql_check_tuple_label_t ('<имя таблицы>', rec_id))
  WITH CHECK (sepgsql_check_tuple_label_t (
                                        '<имя таблицы>', rec_id, 'update'
        )
  );
```

Результат №1:

PostgresPro

Выполнил «user0» (Работает RLS)

Время: **5783.286 мс** **быстрее в 81 раз**

Выполнил «postgres» (Владелец базы, без RLS)

Время: **9,679 мс**

```
--    ticket_no   | departure_airport | arrival_airport |   scheduled_arrival    |     next_departure     |       gap
-- --------------+-------------------+-----------------+------------------------+------------------------+-----------------
    0005432985626 | KHV               | DME             | 2016-10-16 19:55:00+03 | 2016-10-25 19:40:00+03 | 8 days 23:45:00
    0005432985626 | DME               | KHV             | 2016-10-26 03:40:00+03 |                        |
    0005435653879 | DME               | URJ             | 2016-10-18 12:00:00+03 | 2016-10-19 10:55:00+03 | 22:55:00
    0005435653879 | URJ               | TJM             | 2016-10-19 11:20:00+03 | 2016-10-28 06:30:00+03 | 8 days 19:10:00
    0005435653879 | TJM               | URJ             | 2016-10-28 06:55:00+03 | 2016-10-28 15:35:00+03 | 08:40:00
    0005435653879 | URJ               | DME             | 2016-10-28 17:50:00+03 |                        |
    0005435886245 | LED               | VKO             | 2016-10-20 10:10:00+03 | 2016-10-25 10:25:00+03 | 5 days 00:15:00
    0005435886245 | VKO               | LED             | 2016-10-25 11:15:00+03 |                        |
    0005435886246 | LED               | VKO             | 2016-10-20 10:10:00+03 | 2016-10-25 10:25:00+03 | 5 days 00:15:00
    0005435886246 | VKO               | LED             | 2016-10-25 11:15:00+03 |                        |
   (10 строк)
```

Анализ изменённого плана:

PostgresPro

```
->  Nested Loop  (cost=0.43..585663.66 rows=1 width=14)
                                (2147.138..5762.258 rows=4 loops=1)
  ->  Seq Scan on bookings b
                                (cost=0.00..585637.70 rows=2 width=7)
        (actual time=2146.868..5761.935 rows=3 loops=1)

                        Filter:

sepgsql_check_tuple_label (sepgsql_getcon(), s_lvs)  AND
(book_date =
  (('2016-10-13 17:00:00+03'::timestamp with time  zone)::date -
    '7 days'::interval)
               )
             Rows Removed by Filter: 2111107
```

Победа над SEQ SCAN:

PostgresPro

```
CREATE OR REPLACE FUNCTION bookings.bookings_check_dt_lp
        ( p_arg_1 timestamp with time zone -- Аргумент 1
        , p_arg_2 timestamp with time zone -- Аргумент 2
        )
 RETURNS boolean
  AS
    -- ------------------------------------------------------
    --      2019-11-29 Nick Тест.
    -- ------------------------------------------------------
    $$
     SELECT (p_arg_1 = p_arg_2);
    $$
      LANGUAGE sql
```

LEAKPROOF;

User0, уровень s0:c0, новый запрос:

PostgresPro

```
SELECT tf.ticket_no
  ,f.departure_airport
  ,f.arrival_airport
  ,f.scheduled_arrival
  ,lead (f.scheduled_departure)  OVER w AS next_departure
  ,(lead (f.scheduled_departure) OVER w -
                                 f.scheduled_arrival) AS gap
FROM bookings b
   JOIN tickets t          ON (t.book_ref  = b.book_ref)
   JOIN ticket_flights tf ON (t.ticket_no = tf.ticket_no)
   JOIN flights f          ON (f.flight_id = tf.flight_id)
WHERE (bookings.bookings_check_dt_lp (
     b.book_date, (bookings.now()::date - INTERVAL '7 day'))
)
WINDOW w AS (PARTITION BY tf.ticket_no ORDER BY
                                       f.scheduled_departure);
```

Результат №2:

Выполнил «user0» (Работает RLS)

Время: **31.257 мс** **быстрее в 14 943 раз**

Выполнил «postgres» (Владелец базы, без RLS)

Время: **9,679 мс**

```
--    ticket_no   | departure_airport | arrival_airport |   scheduled_arrival    |     next_departure     |       gap
-- ---------------+-------------------+-----------------+------------------------+------------------------+-----------------
    0005432985626 | KHV               | DME             | 2016-10-16 19:55:00+03 | 2016-10-25 19:40:00+03 | 8 days 23:45:00
    0005432985626 | DME               | KHV             | 2016-10-26 03:40:00+03 |                        |
    0005435653879 | DME               | URJ             | 2016-10-18 12:00:00+03 | 2016-10-19 10:55:00+03 | 22:55:00
    0005435653879 | URJ               | TJM             | 2016-10-19 11:20:00+03 | 2016-10-28 06:30:00+03 | 8 days 19:10:00
    0005435653879 | TJM               | URJ             | 2016-10-28 06:55:00+03 | 2016-10-28 15:35:00+03 | 08:40:00
    0005435653879 | URJ               | DME             | 2016-10-28 17:50:00+03 |                        |
    0005435886245 | LED               | VKO             | 2016-10-20 10:10:00+03 | 2016-10-25 10:25:00+03 | 5 days 00:15:00
    0005435886245 | VKO               | LED             | 2016-10-25 11:15:00+03 |                        |
    0005435886246 | LED               | VKO             | 2016-10-20 10:10:00+03 | 2016-10-25 10:25:00+03 | 5 days 00:15:00
    0005435886246 | VKO               | LED             | 2016-10-25 11:15:00+03 |                        |
   (10 строк)
```

Выводы:

- Решение по созданию защищённой БД на основе SELinux + RLS имеет шансы на дальнейшее развитие и успех.
- Необходим компромисс между строго централизованной и распределённой схемами хранения меток.
- Функциональность модуля «SEPGSQL» расширяется.

Источники:

PostgresPro

- **Joe Conway (Crunchy Data)**, «MLS-Postgres» PgConf US-2016, pgDay Paris-2017.
- **Paul Moore (RedHat)**, «SELinux Network Access Controls». December 2012.

PostgresPro

ВОПРОСЫ ??