Авторы: Алексей Береснев, Илья Баштанов, Павел Толмачев

Обратная связь
Отзывы, замечания и предложения направляйте по адресу:
edu@postgrespro.ru

Темы

Расширение pg_proaudit

Возможности pg_proaudit

Аудит ролей и объектов

Сохранение настроек

Запись событий аудита

Расширение pg_proaudit

Аудит — средство обеспечения безопасности

- не отладка (поиск ошибок и неисправностей)
- не мониторинг (выявление аномалий и проблем)

Не зависит от обычного журнала сообщений

- отдельный фоновый процесс
- отдельный протокол событий безопасности

Доступ к расширению и протоколам событий безопасности должен быть предоставлен только уполномоченным ролям

Аудит регистрирует действия, связанные с объектами кластера баз данных. При этом поток регистрируемых событий может быть очень велик, что затрудняет анализ инцидентов. Поэтому требуется точный выбор объектов наблюдения.

Расширение pg_proaudit разработано в Postgres Professional для регистрации событий, связанных с безопасностью. Оно реализует концепцию независимости подсистемы аудита. Для этого запускается отдельный фоновый процесс pg_proaudit logging worker, который протоколирует события независимо от записи в обычный журнал сообщений сервера.

Место размещения протокола о событиях в файловой системе или настройки для передачи протокола в syslog определяются с помощью параметров конфигурации расширения pg_proaudit.

Пользователь Postgres Pro с атрибутом SUPERUSER должен давать доступ к расширению pg_proaudit и протоколам событий безопасности только пользователям, исполняющим роль администратора информационной безопасности.

Установка расширения pg_proaudit

Создадим базу данных.

```
=> CREATE DATABASE audit;
```

```
CREATE DATABASE
```

Расширение pg_proaudit требует загрузки одноименной разделяемой библиотеки:

```
=> ALTER SYSTEM SET shared_preload_libraries = 'pg_proaudit';
```

```
ALTER SYSTEM
```

Необходимо рестартовать СУБД.

```
student$ sudo systemctl restart postgrespro-ent-13.service
```

```
student$ psql -d audit
```

Создадим расширение pg_proaudit:

```
=> CREATE EXTENSION pg_proaudit;
```

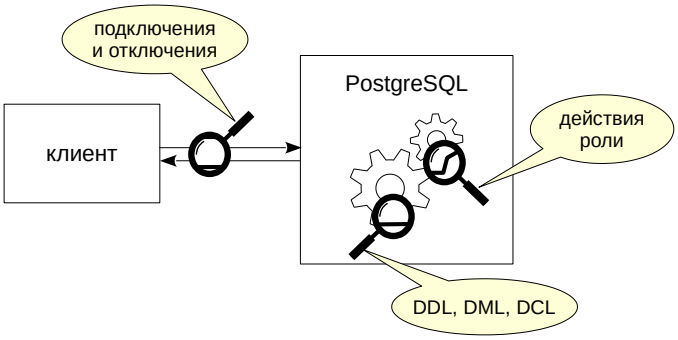
```
CREATE EXTENSION
```

В результате в списке процессов экземпляра должен быть виден процесс pg_proaudit.

```
student$ ps f -C postgres
```

```
    PID TTY      STAT   TIME COMMAND
 269732 ?        Ss     0:00 /opt/pgpro/ent-13/bin/postgres -D /var/lib/pgpro/ent-13
 269733 ?        Ss     0:00  \_ postgres: logger
 269735 ?        Ss     0:00  \_ postgres: checkpointer
 269736 ?        Ss     0:00  \_ postgres: background writer
 269737 ?        Ss     0:00  \_ postgres: walwriter
 269738 ?        Ss     0:00  \_ postgres: autovacuum launcher
 269739 ?        Ss     0:00  \_ postgres: stats collector
 269740 ?        Ss     0:00  \_ postgres: pg_proaudit logging worker
 269741 ?        Ss     0:00  \_ postgres: logical replication launcher
 269747 ?        Ss     0:00  \_ postgres: cfs-worker-0
 269768 ?        Ss     0:00  \_ postgres: student audit [local] idle
```

Возможности pg_proaudit



Классы фиксируемых событий

подключения и отключения
команды DDL — создание, изменение и удаление объектов баз данных
команды DML — обращение к объектам баз данных
команды DCL — управление доступом к объектам баз данных
действия, выполняемые заданной ролью

5

Расширение pg_proaudit позволяет отслеживать действия заданных ролей или следить за операциями с заданными объектами, например, таблицами, или же с определенными классами объектов. Классы фиксируемых событий:

- команды создания, изменения и удаления объектов
- команды управления доступом к объектам баз данных
- обращения к объектам баз данных
- подключение к базе данных
- отключение от базы данных
- любые команды, выполняемые заданным пользователем

Фиксируемые события можно задать с помощью функций расширения или параметров конфигурации сервера.

Полный список регистрируемых событий приведен в документации: https://postgrespro.ru/docs/enterprise/13/pg-proaudit#PG-PROAUDIT-EVENT-TYPES

Аудит ролей и объектов

Управление аудитом ролей

pg_proaudit_set_role
pg_proaudit_reset_role

Управление аудитом объектов или типов объектов

pg_proaudit_set_object
pg_proaudit_reset_object

Отключение аудита всех событий

pg_proaudit_reset

Текущие настройки

представление pg_proaudit_settings

Расширение pg_proaudit по-разному управляет аудитом для ролей и объектов СУБД.

Для отслеживания действий, выполняемых конкретными ролями, используются функции:

- pg_proaudit_set_role — регистрация событий для роли;
- pg_proaudit_reset_role — отключение регистрации.

Если нужно отслеживать события определенного объекта или типа объектов базы данных для всех ролей, используют функцию:

- pg_proaudit_set_object — регистрация событий для объекта или типа объектов;
- pg_proaudit_reset_object — отключение регистрации.

Функция pg_proaudit_reset отключает регистрацию всех событий.

Регистрация событий аудита запускается и останавливается немедленно.

Текущие настройки аудита можно получить с помощью представления pg_proaudit_settings.

Аудит ролей и объектов

Включим аудит на все события, связанные с управлением ролями:

```
=> SELECT pg_proaudit_set_object(
  event_type => 'ALL',
  object_type => 'ROLE'
);

 pg_proaudit_set_object
------------------------
 t
(1 row)
```

Аудит включается немедленно.

Зарегистрируем нового пользователя и отдельной командой дадим право начинать сеанс.

```
=> CREATE ROLE alice;

CREATE ROLE

=> ALTER ROLE alice LOGIN;

ALTER ROLE
```

А теперь назначим пользователю пароль:

```
=> ALTER ROLE alice PASSWORD 'alice';

ALTER ROLE
```

Представление pg_proaudit_settings используется для проверки настроек аудита:

```
=> SELECT * FROM pg_proaudit_settings;

 db_name | event_type | object_type | object_name | role_name
---------+------------+-------------+-------------+-----------
 audit   | all        | role        | 0           |
(1 row)
```

Ожидаем, что в протокол аудита попали три записи. Проверим с помощью команды, выводящей последние три строки из файла протокола, имеющего самую свежую дату модификации.

```
student$ sudo tail -3 /var/lib/pgpro/ent-13/pg_proaudit/postgresql-2024-01-16_213509.log

2024-01-16 21:35:09.870 MSK,"student","audit",269768,INFO,1,1,"CREATE ROLE","ROLE","alice",SUCCESS,"","CREATE ROLE alice;",
2024-01-16 21:35:09.881 MSK,"student","audit",269768,INFO,2,1,"ALTER ROLE","ROLE","alice",SUCCESS,"","ALTER ROLE alice LOGIN;",
2024-01-16 21:35:09.894 MSK,"student","audit",269768,INFO,3,1,"ALTER ROLE","ROLE","alice",SUCCESS,"","ALTER ROLE alice password <REDACTED>",
```

Включим аудит событий подключения. Здесь функции pg_proaudit_set_object вместо имени объекта передаем NULL.

```
=> SELECT pg_proaudit_set_object(
  event_type => 'AUTHENTICATE',
  object_type => NULL
);

 pg_proaudit_set_object
------------------------
 t
(1 row)
```

Что теперь покажет представление pg_proaudit_settings?

```
=> SELECT * FROM pg_proaudit_settings;

 db_name |  event_type  | object_type | object_name | role_name
---------+--------------+-------------+-------------+-----------
 audit   | all          | role        | 0           |
 audit   | authenticate |             | 0           |
(2 rows)
```

Начнем другой сеанс от имени alice:

```
student$ psql "host=localhost dbname=audit user=alice password=alice"
```

И снова проверим протокол аудита:

```
student$ sudo tail -1 /var/lib/pgpro/ent-13/pg_proaudit/postgresql-2024-01-16_213509.log

2024-01-16 21:35:11.089 MSK,"alice","audit",270123,INFO,1,1,"AUTHENTICATE","","audit",SUCCESS,"conntype: host, method: scram, host: 127.0.0.1",,
```

Следующая задача — включить регистрацию всех действий пользователя alice. Для этого воспользуемся функцией pg_proaudit_set_role:

```
=> SELECT pg_proaudit_set_role (
  event_type => 'ALL',
  role_oid => 'alice'::regrole
);

 pg_proaudit_set_role
----------------------
 t
(1 row)
```

Как это отобразилось в представлении pg_proaudit_settings?

```
=> SELECT * FROM pg_proaudit_settings;

 db_name |  event_type  | object_type | object_name | role_name
---------+--------------+-------------+-------------+-----------
 audit   | all          | role        | 0           |
 audit   | authenticate |             | 0           |
 audit   | all          |             | 0           | alice
(3 rows)
```

Пользователь alice создает в своем сеансе таблицу tab1, и сведения об этом сразу появляются в протоколе аудита.

```
  => CREATE TABLE tab1(
    id int,
    txt text
  );

  CREATE TABLE
```

```
student$ sudo tail -4 /var/lib/pgpro/ent-13/pg_proaudit/postgresql-2024-01-16_213509.log
```

```
2024-01-16 21:35:12.243 MSK,"alice","audit",270123,INFO,2,1,"CREATE TABLE","TABLE","public.tab1",SUCCESS,"","CREATE TABLE tab1(
  id int,
  txt text
);",
```

Отменим аудит всех действий пользователя alice:

```
=> SELECT pg_proaudit_reset_role(
  event_type => 'ALL',
  role_oid => 'alice'::regrole
);
```

```
 pg_proaudit_reset_role
------------------------
 t
(1 row)
```

```
=> SELECT * FROM pg_proaudit_settings;
```

```
 db_name |  event_type  | object_type | object_name | role_name
---------+--------------+-------------+-------------+-----------
 audit   | all          | role        | 0           |
 audit   | authenticate |             | 0           |
(2 rows)
```

Сфокусируем аудит на таблице tab1. Для объектов типа TABLE тип события ALL включает регистрацию команд SELECT, INSERT, UPDATE, DELETE, TRUNCATE, COPY, а также CREATE, ALTER, DROP.

```
=> SELECT pg_proaudit_set_object(
  event_type => 'ALL',
  object_oid => 'tab1'::regclass
);
```

```
 pg_proaudit_set_object
------------------------
 t
(1 row)
```

```
=> SELECT * FROM pg_proaudit_settings;
```

```
 db_name |  event_type  | object_type | object_name | role_name
---------+--------------+-------------+-------------+-----------
 audit   | all          | role        | 0           |
 audit   | authenticate |             | 0           |
 audit   | all          |             | public.tab1 |
(3 rows)
```

В сеансе пользователя alice выполним вставку в таблицу tab1 и проверим протокол аудита.

```
=> INSERT INTO tab1 VALUES (1984, 'Big brother is watching you...');
```

```
INSERT 0 1
```

```
=> SELECT * FROM tab1;
```

```
  id  |              txt
------+-------------------------------
 1984 | Big brother is watching you...
(1 row)
```

```
student$ sudo tail -2 /var/lib/pgpro/ent-13/pg_proaudit/postgresql-2024-01-16_213509.log
```

```
2024-01-16 21:35:13.434 MSK,"alice","audit",270123,INFO,3,1,"INSERT","TABLE","public.tab1",SUCCESS,"","INSERT INTO tab1 VALUES (1984, 'Big brother is watching you...');",
2024-01-16 21:35:13.445 MSK,"alice","audit",270123,INFO,4,1,"SELECT","TABLE","public.tab1",SUCCESS,"","SELECT * FROM tab1;",
```

Сохранение настроек

Настройки хранятся в файле PGDATA/pg_proaudit.conf

- pg_proaudit_save — сохраняет настройки регистрации событий
- pg_proaudit_reload — загружает настройки регистрации

Файл pg_proaudit.conf можно редактировать напрямую, но после этого надо перечитать настройки

Настройки аудита, произведенные SQL-командами расширения pg_proaudit, будут потеряны после перезагрузки сервера, если не сохранить их в файле настроек pg_proaudit.conf.

Файл pg_proaudit.conf размещается в каталоге кластера данных (PGDATA), изменить его расположение нельзя.

Для сохранения текущих настроек в этот файл предназначена функция pg_proaudit_save. Аудита можно настраивать и непосредственным редактированием файла pg_proaudit.conf, но это не так удобно, как вызов функций расширения.

Если изменить настройки непосредственно в файле pg_proaudit.conf, их нужно считать и применить функцией pg_proaudit_reload.

Настройки из файла автоматически считываются при запуске сервера.

Файл pg_proaudit.conf

Функция pg_proaudit_save записывает настройки аудита в файл pg_proaudit.conf. Если не сохранить настройки, они будут потеряны при остановке сервера.

```
=> SELECT pg_proaudit_save();

 pg_proaudit_save
------------------

(1 row)
```

Заглянем в файл pg_proaudit.conf:

```
student$ sudo cat /var/lib/pgpro/ent-13/pg_proaudit.conf

audit,all,role,0,
audit,authenticate,,0,
audit,all,,16571,
```

А теперь сбросим все настройки аудита:

```
=> SELECT pg_proaudit_reset();

 pg_proaudit_reset
-------------------

(1 row)
```

Никакие настройки не действуют:

```
=> SELECT * FROM pg_proaudit_settings;

 db_name | event_type | object_type | object_name | role_name
---------+------------+-------------+-------------+-----------
(0 rows)
```

Можно (но не очень удобно) менять файл pg_proaudit.conf любыми средствами редактирования текста. Например, удалим строку конфигурации аудита событий подключения:

```
student$ sudo sed -i '/authenticate/d' /var/lib/pgpro/ent-13/pg_proaudit.conf
```

Вот что осталось в файле конфигурации:

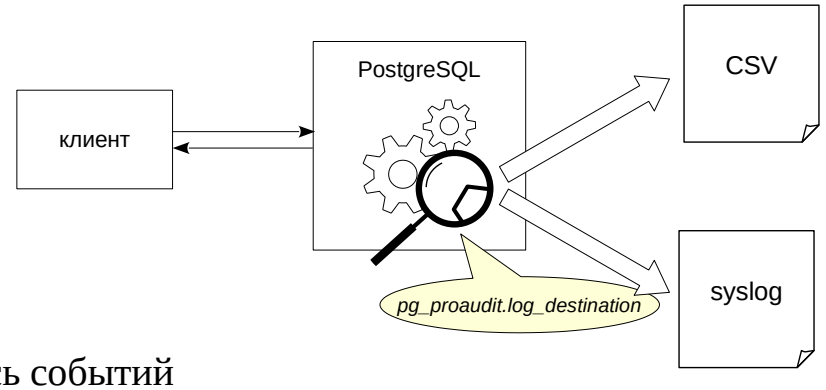
```
student$ sudo cat /var/lib/pgpro/ent-13/pg_proaudit.conf

audit,all,role,0,
audit,all,,16571,
```

А теперь считаем настройки из файла:

```
=> SELECT pg_proaudit_reload();

 pg_proaudit_reload
--------------------

(1 row)


=> SELECT * FROM pg_proaudit_settings;

 db_name | event_type | object_type | object_name | role_name
---------+------------+-------------+-------------+-----------
 audit   | all        | role        | 0           |
 audit   | all        |             | public.tab1 |
(2 rows)
```

Настройки аудита восстановлены, за исключением протоколирования подключений.

Запись событий аудита



Запись событий

- в CSV-файлы на сервере
- в канал журналирования syslog
- одновременно в оба приемника

10

Параметр расширения *pg_proaudit.log_destination* определяет способ фиксации событий безопасности:

- csvlog — записывать события в файл CSV;
- syslog — передавать события в syslog.

Формат строк CSV предопределен в расширении. Параметр *pg_proaudit.log_filename* задает шаблон имен файлов CSV. Параметр *pg_proaudit.log_rotation_size* задает максимальный размер файла протокола в килобайтах. Другой способ настройки ротации файлов протоколов — по времени, устанавливается параметром *pg_proaudit.log_rotation_age*.

Чтобы просматривать сообщения аудита в формате CSV из СУБД, можно подключить расширение file_fdw и использовать внешнюю таблицу (FOREIGN TABLE).

Передача сообщений в syslog удобнее для централизованной сборки протоколов. Популярные реализации службы syslog предоставляют возможности сложной фильтрации, копирования и передачи по сети сообщений протоколов, но формат строк сообщений в таких службах может быть разным.

Полный список параметров конфигурации записи событий аудита в протокол содержится в документации: https://postgrespro.ru/docs/enterprise/13/pg-proaudit#PG-PROAUDIT-SECURITY-EVENT-LOG-CONFIGURATION-PARAMETERS

Настройка записи событий аудита

Посмотрим значения параметров конфигурации по умолчанию:

```
=> SELECT name, setting
FROM pg_settings
WHERE name LIKE 'pg_proaudit%';
```

```
                 name                 |            setting
--------------------------------------+-------------------------------
 pg_proaudit.log_catalog_access       | off
 pg_proaudit.log_command_text         | on
 pg_proaudit.log_destination          | csvlog
 pg_proaudit.log_directory            | pg_proaudit
 pg_proaudit.log_filename             | postgresql-%Y-%m-%d_%H%M%S.log
 pg_proaudit.log_rotation_age         | 1440
 pg_proaudit.log_rotation_size        | 10240
 pg_proaudit.log_truncate_on_rotation | off
(8 rows)
```

Отменим запись в протокол текста команд при регистрации событий аудита:

```
=> ALTER SYSTEM SET pg_proaudit.log_command_text = off;
```

```
ALTER SYSTEM
```

Перечитаем конфигурацию.

```
=> SELECT pg_reload_conf();
```

```
 pg_reload_conf
----------------
 t
(1 row)
```

Удалим строки в таблице tab1 в сеансе пользователя alice:

```
=> DELETE FROM tab1;
```

```
DELETE 1
```

Как теперь записываются события аудита? Сравните с предыдущей записью:

```
student$ sudo tail -2 /var/lib/pgpro/ent-13/pg_proaudit/postgresql-2024-01-16_213509.log
```

```
2024-01-16 21:35:13.445 MSK,"alice","audit",270123,INFO,4,1,"SELECT","TABLE","public.tab1",SUCCESS,"","SELECT * FROM tab1;",
2024-01-16 21:35:14.816 MSK,"alice","audit",270123,INFO,5,1,"DELETE","TABLE","public.tab1",SUCCESS,"",,
```

Итоги

Расширение pg_proaudit протоколирует события аудита независимо от обычного журнала сообщений

Аудиту подлежат подключения и отключения, операции с объектами баз данных и действия ролей

Журнал аудита может сохраняться как в формате CSV, так и передаваться службе syslog

Практика

1. Настройте аудит для записи в CSV и syslog одновременно.
2. Включите аудит регистрации ролей и создания таблиц.
3. Зарегистрируйте новую роль observed1 и настройте аудит подключений и отключений для этой роли.
4. Создайте таблицу tab1 так, чтобы роль observed1 могла выполнять команды DML с этой таблицей, владельцем таблицы должен быть другой пользователь.
5. Настройте регистрацию команд DML, выполненных любым пользователем, для таблицы tab1.

1. После подключения аудита используйте команду ALTER SYSTEM SET pg_proaudit.log_destination = 'csvlog,syslog'; для записи сообщений аудита и в файл и в syslog. Сообщения syslog будут записываться в файл /var/log/syslog, для чтения сообщений необходимо в командной строке Bash использовать sudo в сеансе пользователя student.

2. Для настройки аудита используйте функцию pg_proaudit_set_object.

В заданиях с 3 по 5 просматривать сообщения аудита в файле CSV нужно в каталоге /var/lib/pgpro/ent-13/pg_proaudit. Это удобно делать из сеанса пользователя ОС postgres, который можно запустить командой sudo -i -u postgres. Просматривать файл журнала сообщений /var/log/syslog проще всего в сеансе пользователя ОС student с помощью команды sudo grep pg_proaudit /var/log/syslog | tail.

1. Запись сообщений аудита и в CSV и в syslog

Создадим базу данных.

```
=> CREATE DATABASE audit;
```

```
CREATE DATABASE
```

Загрузка библиотеки:

```
=> ALTER SYSTEM SET shared_preload_libraries = 'pg_proaudit';
```

```
ALTER SYSTEM
```

Подключение библиотеки требует перезагрузки СУБД.

```
student$ sudo systemctl restart postgrespro-ent-13.service
```

```
student$ psql -d audit
```

Создадим расширение pg_proaudit:

```
=> CREATE EXTENSION pg_proaudit;
```

```
CREATE EXTENSION
```

Настройка записи сообщений аудита одновременно в CSV файл и в syslog:

```
=> ALTER SYSTEM SET pg_proaudit.log_destination = 'csvlog,syslog';
```

```
ALTER SYSTEM
```

Установка имени программы — идентификатора в записях syslog:

```
=> ALTER SYSTEM SET syslog_ident = 'MY_audit';
```

```
ALTER SYSTEM
```

```
=> SELECT pg_reload_conf();
```

```
 pg_reload_conf
----------------
 t
(1 row)
```

Наличие идентификатора облегчит поиск записей аудита в журнале сообщений.

---

2. Аудит регистрации пользователей и создания новых таблиц

Аудит на все события, связанные с управлением ролями:

```
=> SELECT pg_proaudit_set_object('ALL', 'ROLE');
```

```
 pg_proaudit_set_object
------------------------
 t
(1 row)
```

Аудит событий создания таблиц:

```
=> SELECT pg_proaudit_set_object('CREATE TABLE', 'TABLE');
```

```
 pg_proaudit_set_object
------------------------
 t
(1 row)
```

Получившиеся настройки аудита:

```
=> SELECT * FROM pg_proaudit_settings;
```

```
 db_name |  event_type  | object_type | object_name | role_name
---------+--------------+-------------+-------------+-----------
 audit   | all          | role        | 0           |
 audit   | create table | table       | 0           |
(2 rows)
```

---

3. Регистрация событий создания и управления пользователями

Регистрация нового пользователя:

```
=> CREATE ROLE observed1 LOGIN PASSWORD 'obs123';
```

```
CREATE ROLE
```

Второй сеанс от имени observed1:

```
student$ psql "dbname=audit user=observed1 password=obs123"
```

Командой ls -ltr $PGDATA/pg_proaudit можно получить сортированный по времени модификации список файлов аудита.

```
student$ sudo ls -ltr /var/lib/pgpro/ent-13/pg_proaudit
```

```
total 4
-rw-r----- 1 postgres postgres 157 янв 16 21:46 postgresql-2024-01-16_214622.log
```

Проверяем CSV-файл аудита, который был изменен последним:

```
student$ sudo tail -1 /var/lib/pgpro/ent-13/pg_proaudit/postgresql-2024-01-16_214622.log
```

```
2024-01-16 21:46:22.692 MSK,"student","audit",303879,INFO,1,1,"CREATE ROLE","ROLE","observed1",SUCCESS,"","CREATE ROLE observed1 LOGIN password <REDACTED>",
```

Получите последние две записи в журнале /var/log/syslog по установленному идентификатору MY_audit:

```
student$ sudo grep MY_audit /var/log/syslog | tail -2
```

```
Jan 16 21:46:22 student MY_audit[303879]: AUDIT [1-1] INFO: USER: student, DB: audit, PID: 303879, EVENT TYPE: CREATE ROLE, OBJECT TYPE: ROLE, OBJECT NAME: observed1, SUCCESS
Jan 16 21:46:22 student MY_audit[303879]: AUDIT [1-1] STATEMENT: CREATE ROLE observed1 LOGIN password <REDACTED>
```

---

4. Создание таблицы и предоставление прав на нее

Суперпользователь student создает таблицу и выдает на нее права observed1.

```
=> CREATE TABLE tab1(n integer, txt text);
```

```
CREATE TABLE
```

```
=> GRANT SELECT,INSERT,UPDATE,DELETE ON tab1 TO observed1;
```

```
GRANT
```

Проверяем CSV-файл аудита: должно быть видно CREATE TABLE, но не GRANT.

```
student$ sudo tail -2 /var/lib/pgpro/ent-13/pg_proaudit/postgresql-2024-01-16_214622.log

2024-01-16 21:46:22.692 MSK,"student","audit",303879,INFO,1,1,"CREATE ROLE","ROLE","observed1",SUCCESS,"","CREATE ROLE observed1 LOGIN password <REDACTED>",
2024-01-16 21:46:23.876 MSK,"student","audit",303879,INFO,2,1,"CREATE TABLE","TABLE","public.tab1",SUCCESS,"","CREATE TABLE tab1(n integer, txt text);",
```

Поиск записей в журнале /var/log/syslog по установленному идентификатору MY_audit:

```
student$ sudo grep MY_audit /var/log/syslog | tail -4

Jan 16 21:46:22 student MY_audit[303879]: AUDIT [1-1] INFO: USER: student, DB: audit, PID: 303879, EVENT TYPE: CREATE ROLE, OBJECT TYPE: ROLE, OBJECT NAME: observed1, SUCCESS
Jan 16 21:46:22 student MY_audit[303879]: AUDIT [1-1] STATEMENT: CREATE ROLE observed1 LOGIN password <REDACTED>
Jan 16 21:46:23 student MY_audit[303879]: AUDIT [2-1] INFO: USER: student, DB: audit, PID: 303879, EVENT TYPE: CREATE TABLE, OBJECT TYPE: TABLE, OBJECT NAME: public.tab1, SUCCESS
Jan 16 21:46:23 student MY_audit[303879]: AUDIT [2-1] STATEMENT: CREATE TABLE tab1(n integer, txt text);
```

---

5. Регистрация команд DML с таблицей tab1

Установка регистрации команд DML для tab1.

```
=> SELECT pg_proaudit_set_object('SELECT', 'tab1'::regclass);

 pg_proaudit_set_object
------------------------
 t
(1 row)


=> SELECT pg_proaudit_set_object('INSERT', 'tab1'::regclass);

 pg_proaudit_set_object
------------------------
 t
(1 row)


=> SELECT pg_proaudit_set_object('UPDATE', 'tab1'::regclass);

 pg_proaudit_set_object
------------------------
 t
(1 row)


=> SELECT pg_proaudit_set_object('DELETE', 'tab1'::regclass);

 pg_proaudit_set_object
------------------------
 t
(1 row)
```

Получившиеся настройки аудита:

```
=> SELECT * FROM pg_proaudit_settings;

 db_name |  event_type  | object_type | object_name | role_name
---------+--------------+-------------+-------------+-----------
 audit   | all          | role        | 0           |
 audit   | create table | table       | 0           |
 audit   | select       |             | public.tab1 |
 audit   | insert       |             | public.tab1 |
 audit   | update       |             | public.tab1 |
 audit   | delete       |             | public.tab1 |
(6 rows)
```

Команды в сеансе observed1:

```
=> INSERT INTO tab1 VALUES (1, 'Один');

INSERT 0 1

=> SELECT * FROM tab1;

 n | txt
---+------
 1 | Один
(1 row)


=> UPDATE tab1 SET n=2;

UPDATE 1

=> DELETE FROM tab1;

DELETE 1
```

Проверяем CSV-файл аудита:

```
student$ sudo tail -4 /var/lib/pgpro/ent-13/pg_proaudit/postgresql-2024-01-16_214622.log

2024-01-16 21:46:25.128 MSK,"observed1","audit",304154,INFO,1,1,"INSERT","TABLE","public.tab1",SUCCESS,"","INSERT INTO tab1 VALUES (1, 'Один');",
2024-01-16 21:46:25.147 MSK,"observed1","audit",304154,INFO,2,1,"SELECT","TABLE","public.tab1",SUCCESS,"","SELECT * FROM tab1;",
2024-01-16 21:46:25.167 MSK,"observed1","audit",304154,INFO,3,1,"UPDATE","TABLE","public.tab1",SUCCESS,"","UPDATE tab1 SET n=2;",
2024-01-16 21:46:25.179 MSK,"observed1","audit",304154,INFO,4,1,"DELETE","TABLE","public.tab1",SUCCESS,"","DELETE FROM tab1;",
```

Поиск записей в журнале /var/log/syslog по установленному идентификатору MY_audit:

```
student$ sudo grep MY_audit /var/log/syslog | tail -8

Jan 16 21:46:25 student MY_audit[304154]: AUDIT [1-1] INFO: USER: observed1, DB: audit, PID: 304154, EVENT TYPE: INSERT, OBJECT TYPE: TABLE, OBJECT NAME: public.tab1, SUCCESS
Jan 16 21:46:25 student MY_audit[304154]: AUDIT [1-1] STATEMENT: INSERT INTO tab1 VALUES (1, 'Один');
Jan 16 21:46:25 student MY_audit[304154]: AUDIT [2-1] INFO: USER: observed1, DB: audit, PID: 304154, EVENT TYPE: SELECT, OBJECT TYPE: TABLE, OBJECT NAME: public.tab1, SUCCESS
Jan 16 21:46:25 student MY_audit[304154]: AUDIT [2-1] STATEMENT: SELECT * FROM tab1;
Jan 16 21:46:25 student MY_audit[304154]: AUDIT [3-1] INFO: USER: observed1, DB: audit, PID: 304154, EVENT TYPE: UPDATE, OBJECT TYPE: TABLE, OBJECT NAME: public.tab1, SUCCESS
Jan 16 21:46:25 student MY_audit[304154]: AUDIT [3-1] STATEMENT: UPDATE tab1 SET n=2;
Jan 16 21:46:25 student MY_audit[304154]: AUDIT [4-1] INFO: USER: observed1, DB: audit, PID: 304154, EVENT TYPE: DELETE, OBJECT TYPE: TABLE, OBJECT NAME: public.tab1, SUCCESS
Jan 16 21:46:25 student MY_audit[304154]: AUDIT [4-1] STATEMENT: DELETE FROM tab1;
```
