

ЗАЩИЩЕННАЯ СИСТЕМА УПРАВЛЕНИЯ
БАЗАМИ ДАННЫХ «ЈАТОВА»

Руководство по настройке. Часть 17.
Выявление и предотвращение исполнения нетипичных
SQL-запросов.
Компонент «SQL_Firewall»

643.72410666.00067-07 98 01-17

Листов 24

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

АННОТАЦИЯ

В документе приведены сведения, необходимые для установки и эксплуатации компонента «SQL_Firewall» (далее – компонент или SQL_Firewall). Настоящее руководство предназначено для администратора защищенной системы управления базами данных (СУБД) «Jatoba».

Администратор СУБД «Jatoba» должен иметь навыки по работе с СУБД PostgreSQL или защищенной СУБД Jatoba (ООО «Газинформсервис»).



Все примеры в данном документе приведены для СУБД «Jatoba» версии ядра 4.x, для других версий все шаги выполняются аналогично, разница состоит в именах директорий.

Например, СУБД «Jatoba» версии 6.x по умолчанию устанавливается в директорию:

- ОС Windows – «C:\Program Files\GIS\Jatoba\6\bin»;
- ОС Linux – «/usr/jatoba-6/bin».

Версия компонента – 0.8.1



Важная информация

Для сертифицированной версии СУБД «Jatoba» поддерживается работа только на ОС, указанных в формуляре на поставку!

Степени важности примечаний, применяемые в документе:



Важная информация – указания, требующие особого внимания



Дополнительная информация – указания, позволяющие упростить работу с изделием

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

СОДЕРЖАНИЕ

1. Назначение компонента.....	4
1.1. Условия применения.....	4
2. Установка и настройка.....	5
2.1. Установка компонента «SQL_Firewall» в ОС GNU/Linux.....	5
2.2. Установка/удаление в ОС Microsoft Windows.....	6
2.3. Настройка конфигурационного файла postgresql.conf.....	8
2.4. Установка расширения компонента «SQL_Firewall».....	8
3. Функциональные возможности	10
3.1. Режимы функционирования компонента	12
3.1.1. Режим обучения «learning»	12
3.1.2. Режим применения «enforcing».....	13
3.1.3. Режим разрешающий «permissive»	13
3.1.4. Режим отключения компонента «disabled»	14
4. Описание операций.....	15
4.1. Функции просмотра	15
4.1.1. Просмотр правил брандмауэра (sql_firewall.sql_firewall_statements)	15
4.1.2. Просмотр статистики (sql_firewall.sql_firewall_stat)	16
4.2. Управление функциями мониторинга	17
4.2.1. Экспорт правил компонента (sql_firewall_export_rule)	17
4.2.2. Импорт правил компонента (sql_firewall_import_rule).....	18
4.2.3. Очистка правил (sql_firewall_reset).....	19
4.2.4. Очистка предупреждений и ошибок (sql_firewall_stat_reset).....	20
5. Удаление	22
Перечень сокращений.....	23

1. НАЗНАЧЕНИЕ КОМПОНЕНТА

Компонент не является межсетевым экраном (firewall) в классическом понимании, как программный или программно-аппаратный элемент компьютерной сети, применяемого для фильтрации трафика.

Компонент SQL_Firewall предназначен для защиты базы данных от SQL-инъекций или неожиданных запросов.

Компонент просматривает запросы, которые могут быть выполнены, и предотвращает либо предупреждает о выполнении запросов, которые не найдены в установленных правилах по аналогии с брандмауэром, т.е. фильтрует трафик SQL-запросов.

1.1. Условия применения

Компонент «SQL_Firewall» может использоваться совместно с СУБД «Jatoba» версии 1.x и выше.

В текущей реализации не поддерживается управление с помощью пользовательского веб-интерфейса для администраторов «Jatoba data safe».

2. УСТАНОВКА И НАСТРОЙКА

Установка компонента должна производиться от имени пользователя, обладающего административными привилегиями в системе. Компонент штатным образом может быть установлен только с СУБД «Jatoba» (см. документ «Защищенная система управления базами данных «Jatoba». Руководство по установке).

2.1. Установка компонента «SQL_Firewall» в ОС GNU/Linux

Установка компонента осуществляется в процессе установки СУБД «Jatoba», также компонент можно установить опционально после основной инсталляции СУБД.

Установку компонента возможно провести двумя способами:

- 1) установка из локального репозитория (CDROM) – производится из файлов, записанных на компакт-диск или скопированных с него;
- 2) установка непосредственно из deb/rpm-файлов – производится опционально, по усмотрению пользователя.

Компонент выполнен в виде отдельного deb или rpm-пакета. Установка компонента осуществляется средствами пакетного менеджера ОС. Для разных типов пакетных менеджеров команда установки немного отличается. Ниже приведены основные типы:

- для систем на основе пакетного менеджера APT (к таким системам относятся все ОС семейства Debian, использующие deb-пакеты) команда установки следующая:

```
apt-get install jatoba4-sql-firewall
```

- для систем на основе пакетных менеджеров YUM/DNF (к таким системам относятся все ОС семейства RedHat и вышедшие из нее, использующие rpm-пакеты) команда установки следующая:

```
yum install jatoba4-sql-firewall
```

Отдельного уточнения требуют операционные системы ALT Linux и openSUSE.

- ALT Linux использует пакетный менеджер APT, но распространяется в виде rpm-пакетов и для нее команда установки выглядит аналогично Debian:

```
apt-get install jatoba4-sql-firewall
```

– openSUSE также распространяется в виде rpm-пакетов, но использует собственный пакетный менеджер zypper, для нее команда установки выглядит следующим образом:

```
zypper install jatoba4-sql-firewall
```

Установка компонента в составе других версий СУБД «Jatoba» осуществляется аналогично. Отличие будет только в номере версии СУБД, в составе которой он распространяется. Например, jatoba5-sql-firewall и т.п.

Удаление модуля также осуществляется средствами пакетного менеджера ОС. Вместо команды install нужно использовать соответствующую данному пакетному менеджеру команду удаления (remove, purge, erase и т.п.).

Для получения детальной информации по пакетному менеджеру рекомендуется обратиться к документации по ОС.

2.2. Установка/удаление в ОС Microsoft Windows

Компонент распространяется в составе инсталляционного msi-файла СУБД «Jatoba». Для установки компонента «SQL_Firewall» при выборе режима «Полный», модуль установится автоматически.

При режиме «Выборочный» потребуется в списке компонент дополнительно выбрать «Контроль запросов пользователей (sql_firewall)» для последующей его установки.

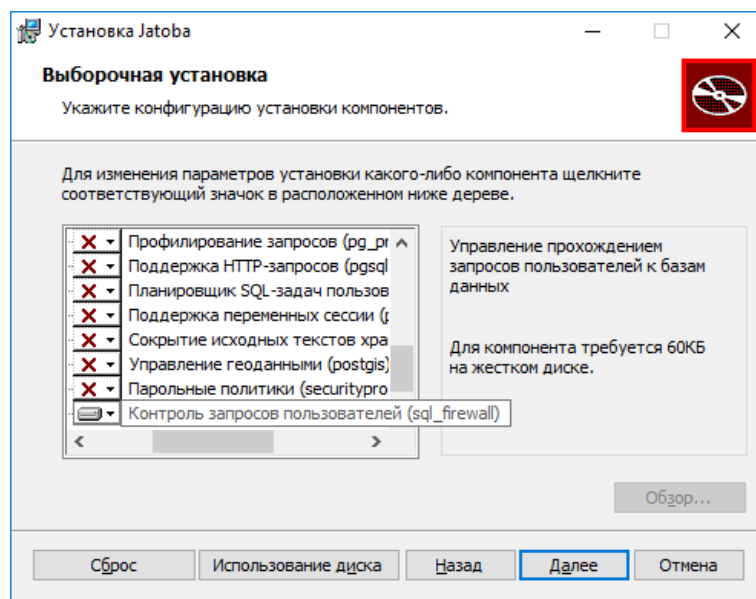


Рисунок 2.1 – Выбор компонента при установке

Удаление «SQL_Firewall» происходит также через инсталлятор СУБД «Jatoba». Удаление компонента осуществляется либо вместе с удалением СУБД «Jatoba», либо отдельно его можно выключить из состава установленных компонент СУБД (через управление приложениями в панели управления Microsoft Windows).

Для получения детальной информации по установке/удалению программ в Microsoft Windows рекомендуется обратиться к документации по ОС.

2.3. Настройка конфигурационного файла postgresql.conf

В разделе «Shared Library Preloading», для последующей загрузки расширения, установить параметр:

```
shared_preload_libraries = 'sql_firewall'
```

```
# - Shared Library Preloading -  
  
shared_preload_libraries = 'sql_firewall'      # (change requires restart)  
#local_preload_libraries = ''  
#session_preload_libraries = ''  
#jit_provider = 'llvmjit'                      # JIT library to use
```

Рисунок 2.2 – Параметр загрузки расширения

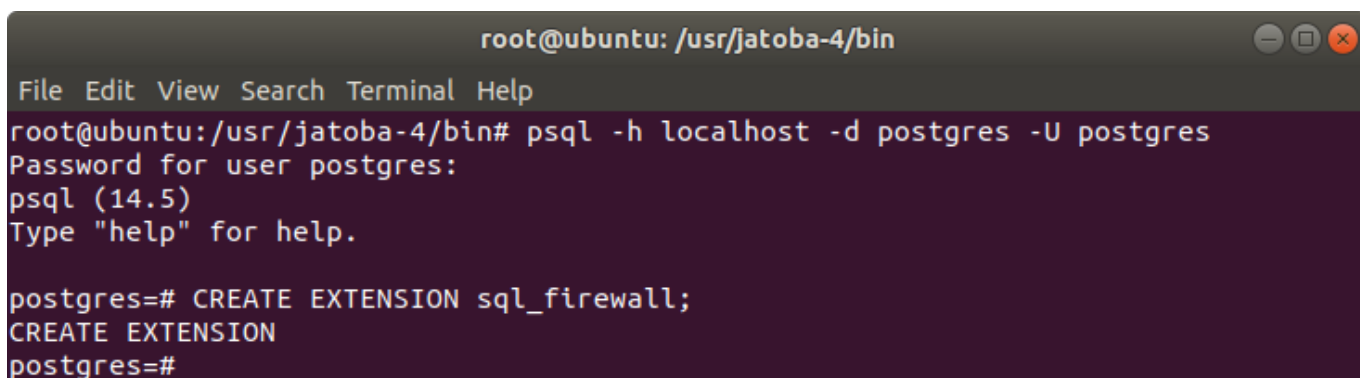
Для применения параметров потребуется перезапустить СУБД.

2.4. Установка расширения компонента «SQL_Firewall»

После перезагрузки СУБД и загрузки расширения станет доступной установка расширения «SQL_Firewall».

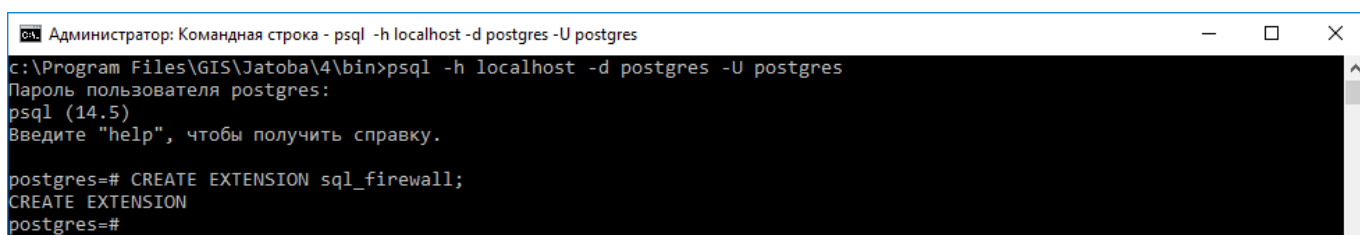
Расширение устанавливается на всех узлах кластера при помощи SQL-команды:

```
CREATE EXTENSION sql_firewall;
```



The screenshot shows a terminal window titled 'root@ubuntu: /usr/jatoba-4/bin'. The user enters the command 'psql -h localhost -d postgres -U postgres'. The prompt changes to 'postgres=#'. The user then enters 'CREATE EXTENSION sql_firewall;'. The prompt returns to 'postgres=#'.

Рисунок 2.3 – Команда установки расширения в ОС GNU/Linux



The screenshot shows a Windows command prompt window titled 'Администратор: Командная строка - psql -h localhost -d postgres -U postgres'. The user enters the command 'psql -h localhost -d postgres -U postgres'. The prompt changes to 'postgres=#'. The user then enters 'CREATE EXTENSION sql_firewall;'. The prompt returns to 'postgres=#'.

Рисунок 2.4 – Команда установки расширения в ОС Windows

В результате выполнения SQL-команды будет создано расширение (extension) «sql_firewall» в схеме данных.

3. ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ

Компонент «SQL_Firewall» выполнен в форме расширения СУБД и предназначен для нормализации записи запросов.

Нормализация – процесс, при котором похожие запросы, как правило, отличающиеся только константами (хотя точные правила несколько сложнее) признаются эквивалентными и отслеживаются как одна запись. Это особенно полезно для неподготовленных запросов.

Нормализация реализуется с помощью запросов-"отпечатков" (fingerprinting queries), которые выборочно сериализуют те поля узлов каждого дерева запроса, которые оцениваются как значимые.

Это называется слепок запроса «query jumble». Он отличается от обычной сериализации тем, что информация, такая как сопоставления переменных или значения констант, игнорируется как несущественная для запроса.

Слепок «jumble» получается в результате парсинг-анализа каждого запроса, и 32(64)-битный хеш слепка сохраняется в поле запроса «Query.queryId».

Затем сервер распространяет это значение, делая его доступным в плановом дереве, сгенерированном из запроса.

Исполнитель («executor» – главная процедура, непосредственно выполняющая запрос) может затем использовать это значение, чтобы приписать временные затраты на исполнение запроса ориентируясь по подходящему «queryId».

Для упрощения представления записей, создается "демонстрационная" строка SQL-запроса, в которой константы заменены на символ «?», чтобы было понятнее, что может из себя представлять нормализованная запись.

Чтобы сэкономить на общей памяти «shared memory», и чтобы избежать усечения длинных строк SQL-запроса, эти строки хранятся во временном внешнем файле с текстом SQL-запросов.

Смещения в этот файл хранятся в общей памяти «shared memory» в хэш-таблице.

Каждая запись в хэш-таблице основанная на id (хэше) запроса, рассматривается как "правило" файервола.

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

"Выученные" правила выводятся для каждого отдельного исходного запроса.

Отслеживается только то количество разных запросов, которое будет соответствовать указанному в конфигурации объему общей памяти «shared memory» – 100 до 5000.

Проверка каждого нового запроса на соответствие правилам производится при попытке записи в хэш-таблицу. Это может произойти сразу после парсинга запроса, или в процессе выполнения команд.

В зависимости от режима расширения записывается сообщение об ошибке и, если это ошибка, вызывается еще и `pg_unreachable()` за счет механики «ereport».

Также расширение предоставляет возможность импортировать или экспортировать правила при помощи вызова соответствующих функций расширения. При импорте и экспорте правила сохраняются в файл на диске.

Примечание о блокировках: чтобы создать или удалить запись в общей хэш-таблице, нужно захватывать монопольную блокировку `pgss->lock`. Изменение любого поля в записи, кроме счетчиков, требует того же. Чтобы найти запись, нужно удерживать общую блокировку.

Чтобы прочитать или обновить счетчики в записи, необходимо удерживать общую или монопольную блокировку (чтобы запись не исчезла), а также использовать мьютекс `spinlock` записи хэш-таблицы.

Переменная общего состояния `pgss->extent` (следующее свободное для записи место во внешнем файле текста SQL-запросов) должна быть доступна только при удержании либо спин-блокировки `pgss->mutex`, либо монопольной блокировки на `pgss->lock`.

Мьютекс используется, чтобы разрешить резервирование файлового пространства, удерживая только общую блокировку на `pgss->lock`.

Перезапись всего внешнего файла текста SQL запросов требует удержания монопольной блокировки `pgss->lock`. Это позволяет читать или записывать отдельные записи в файле, удерживая только общую блокировку.

3.1. Режимы функционирования компонента

Компонент «SQL_Firewall» функционирует в следующих режимах, указанных в параметре `sql_firewall.firewall`:

- "learning" – режим обучения;
- "enforcing" – режим применения;
- "permissive" – режим разрешающий любые SQL запросы;
- "disabled" – режим отключенного модуля.

3.1.1. Режим обучения «learning»

В режиме обучения модуль собирает информацию о пользователе и его запросах. Собирает, записывает и связывает информацию об идентификаторе пользователя «userid» и идентификаторе SQL-запроса «query id».

«query id» устанавливается по дереву синтаксического анализа, аналогично `pg_stat_statements`.

Режим обучения инициализируется установкой параметра:

```
sql_firewall.firewall = 'learning'
```

в конфигурационном файле СУБД `postgresql.conf` и последующей перезагрузкой СУБД.

```
# - Shared Library Preloading -  
  
shared_preload_libraries = 'sql_firewall'          # (change requires restart)  
sql_firewall.firewall = 'learning'  
#local_preload_libraries = ''  
#session_preload_libraries = ''  
#jit_provider = 'llvmjit'                          # JIT library to use
```

Рисунок 3.1 – Инициализация режима обучения

По умолчанию в режиме обучения может обработаться 5000 запросов. Запросы превышающие данное значение не будут изучаться.

Параметр изучаемых запросов может быть установлен от 100 до 5000, который устанавливается в конфигурационном файле СУБД `postgresql.conf`

```
# - Shared Library Preloading -  
#local_preload_libraries = ''  
#session_preload_libraries = ''  
shared_preload_libraries = 'sql_firewall'      # (change requires restart)  
sql_firewall.firewall = 'disabled'  
sql_firewall.firewall = 5000
```

Рисунок 3.2 – Установка параметра количества обрабатываемых SQL - запросов

3.1.2. Режим применения «enforcing»

В режиме применения «enforcing», модуль проверяет находятся ли идентификатор пользователя «userid» и его SQL-запрос «query id» в паре, которая была ранее записана в режиме обучения «learning».

Если обнаруживается несоответствие, то пользователю выдается предупреждение о предотвращении выполнения.

Режим применения инициализируется установкой параметра:

```
sql_firewall.firewall = 'enforcing'
```

в конфигурационном файле СУБД postgresql.conf и последующей перезагрузкой СУБД.

```
# - Shared Library Preloading -  
  
shared_preload_libraries = 'sql_firewall'      # (change requires restart)  
sql_firewall.firewall = 'enforcing'  
#local_preload_libraries = ''  
#session_preload_libraries = ''  
#jit_provider = 'llvmjit'                    # JIT library to use
```

Рисунок 3.3 – Инициализация режима применения

3.1.3. Режим разрешающий «permissive»

В разрешающем режиме «permissive» модуль проверяет все запросы, но позволяет их выполнение, даже в случае, если таких запросов нет в правилах модуля. При несовпадениях выполняет запрос и сигнализирует пользователю о выявленном несоответствии.

Режим разрешения инициализируется установкой параметра:

```
sql_firewall.firewall = 'permissive'
```

в конфигурационном файле СУБД postgresql.conf и последующей перезагрузкой СУБД.

```
# - Shared Library Preloading -  
  
shared_preload_libraries = 'sql_firewall'          # (change requires restart)  
sql_firewall.firewall = 'permissive'  
#local_preload_libraries = ''
```

Рисунок 3.4 – Инициализация разрешающего режима

3.1.4. Режим отключения компонента «disabled»

В данном режиме модуль отключен и не проводит ни каких действий и включен по умолчанию.

Режим отключения компонента инициализируется установкой параметра:

```
sql_firewall.firewall = 'disabled'
```

в конфигурационном файле СУБД postgresql.conf и последующей перезагрузкой СУБД.

```
# - Shared Library Preloading -  
  
shared_preload_libraries = 'sql_firewall'          # (change requires restart)  
sql_firewall.firewall = 'disabled'  
#local_preload_libraries = ''  
#session_preload_libraries = ''  
#jit_provider = 'llvmjit'                          # JIT library to use
```

Рисунок 3.5 – Инициализация отключения модуля

4. ОПИСАНИЕ ОПЕРАЦИЙ

4.1. Функции просмотра

Компонент обладает функциональными возможностями просмотра:

- правил брандмауэра;
- статистики брандмауэра.

4.1.1. Просмотр правил брандмауэра (sql_firewall.sql_firewall_statements)

Представление sql_firewall_statements показывает правила брандмауэра и счетчик выполнения для каждого запроса.

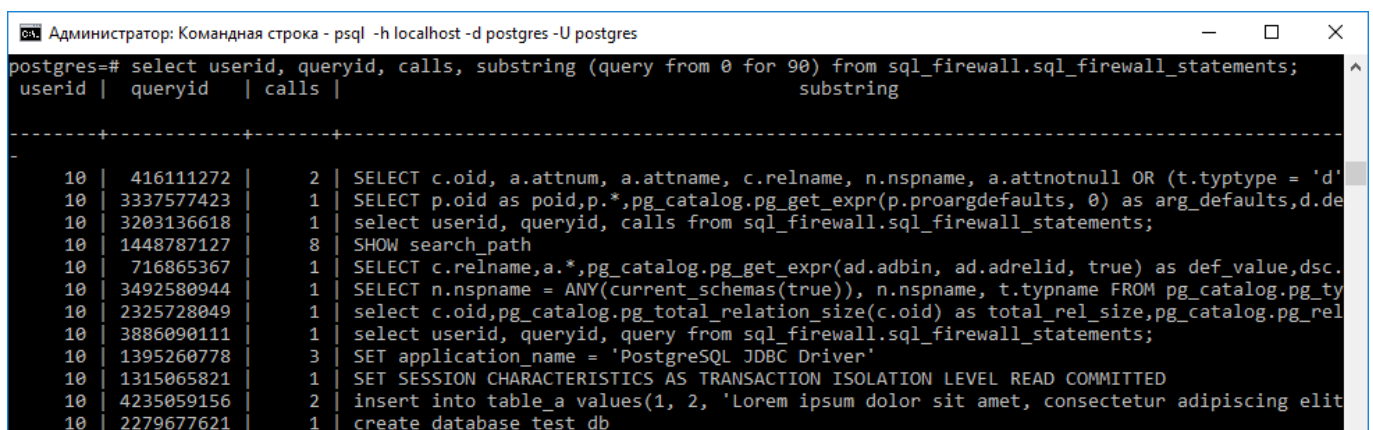
Просмотр правил брандмауэра выполняется SQL-командой:

```
SELECT * from sql_firewall.sql_firewall_statements;
```

Данную SQL-команду целесообразно использовать для специального программного обеспечения по управлению СУБД.

При работе в CLI целесообразно изменить ее и использовать ограничение количества символов в поле «query» оператором «substring»:

```
SELECT userid, queryid, calls, substring (query from 0 for 90)  
from sql_firewall.sql_firewall_statements;
```



```
Администратор: Командная строка - psql -h localhost -d postgres -U postgres
postgres=# select userid, queryid, calls, substring (query from 0 for 90) from sql_firewall.sql_firewall_statements;
userid | queryid | calls | substring
-----+-----+-----+-----
10 | 416111272 | 2 | SELECT c.oid, a.attnum, a.attname, c.relname, n.nspname, a.attnotnull OR (t.typtype = 'd'
10 | 3337577423 | 1 | SELECT p.oid as poid,p.*,pg_catalog.pg_get_expr(p.proargdefaults, 0) as arg_defaults,d.de
10 | 3203136618 | 1 | select userid, queryid, calls from sql_firewall.sql_firewall_statements;
10 | 1448787127 | 8 | SHOW search_path
10 | 716865367 | 1 | SELECT c.relname,a.*,pg_catalog.pg_get_expr(ad.adbin, ad.adrelid, true) as def_value,dsc.
10 | 3492580944 | 1 | SELECT n.nspname = ANY(current_schemas(true)), n.nspname, t.typname FROM pg_catalog.pg_ty
10 | 2325728049 | 1 | select c.oid,pg_catalog.pg_total_relation_size(c.oid) as total_rel_size,pg_catalog.pg_rel
10 | 3886090111 | 1 | select userid, queryid, query from sql_firewall.sql_firewall_statements;
10 | 1395260778 | 3 | SET application_name = 'PostgreSQL JDBC Driver'
10 | 1315065821 | 1 | SET SESSION CHARACTERISTICS AS TRANSACTION ISOLATION LEVEL READ COMMITTED
10 | 4235059156 | 2 | insert into table_a values(1, 2, 'Lorem ipsum dolor sit amet, consectetur adipiscing elit
10 | 2279677621 | 1 | create database test_db
```

Рисунок 4.1 – Просмотр правил брандмауэра в ОС Windows

```

root@ubuntu: /usr/jatoba-4/bin
File Edit View Search Terminal Help
postgres=# select * from sql_firewall.sql_firewall_statements;
userid | queryid | query | calls
-----+-----+-----+-----
      10 | 3226304309 | select * from table_b; | 1
      10 | 2886906670 | select * from table_c; | 1
(2 rows)

postgres=#

```

Рисунок 4.2 – Просмотр правил брандмауэра в ОС GNU/Linux

В полученном списке отражены поля:

- userid – идентификационный номер пользователя (идентификационный номер 10 присваивается роли postgres);
- Queryid – идентификационный номер запроса;
- Query – тело запроса;
- Calls – вызовы.

4.1.2. Просмотр статистики (sql_firewall.sql_firewall_stat)

В представлении sql_firewall_stat есть два счетчика: "sql_warning" и "sql_error".

«sql_warning» показывает количество выполненных запросов с предупреждениями в «разрешающем» режиме ([permissive](#)).

«sql_error» показывает количество предотвращенных запросов в «применительном» режиме ([enforcing](#)).

Просмотр статистика выполняется SQL-командой:

```
SELECT * from sql_firewall.sql_firewall_stat;
```

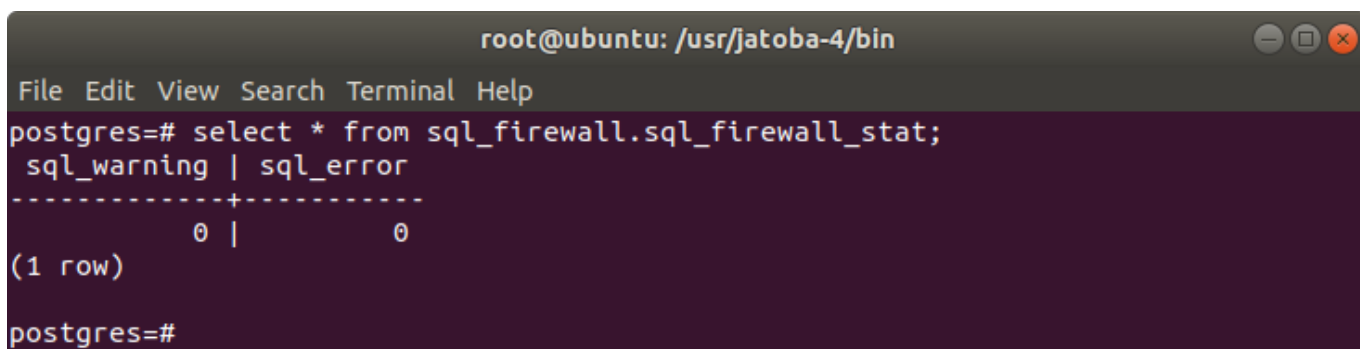
```

Администратор: Командная строка - psql -h localhost -d postgres -U postgres
postgres=# select * from sql_firewall.sql_firewall_stat;
sql_warning | sql_error
-----+-----
          0 |          0
(1 строка)

postgres=#

```

Рисунок 4.3 – Просмотр статистики в ОС Windows



```
root@ubuntu: /usr/jatoba-4/bin
File Edit View Search Terminal Help
postgres=# select * from sql_firewall.sql_firewall_stat;
 sql_warning | sql_error
-----+-----
          0 |          0
(1 row)
postgres=#
```

Рисунок 4.4 – Просмотр статистики в ОС GNU/Linux

4.2. Управление функциями мониторинга

Компонент обладает функциональными возможностями управления правилами, такими как:

- экспорт правил (п. 4.2.1);
- импорт правил (п. 4.2.2);
- очистка правил (п. 4.2.3);
- очистка предупреждений и ошибок (п. 4.2.4).

4.2.1. Экспорт правил компонента (sql_firewall_export_rule)

Функциональная возможность экспорта правил компонента «sql_firewall» доступна в режиме «[disabled](#)» от имени и с правами привилегированного пользователя с атрибутом «Superuser».

SQL-команда экспорта правил SQL-брандмауэра имеет синтаксис:

```
sql_firewall_export_rule('/path/to/rule.txt')
```

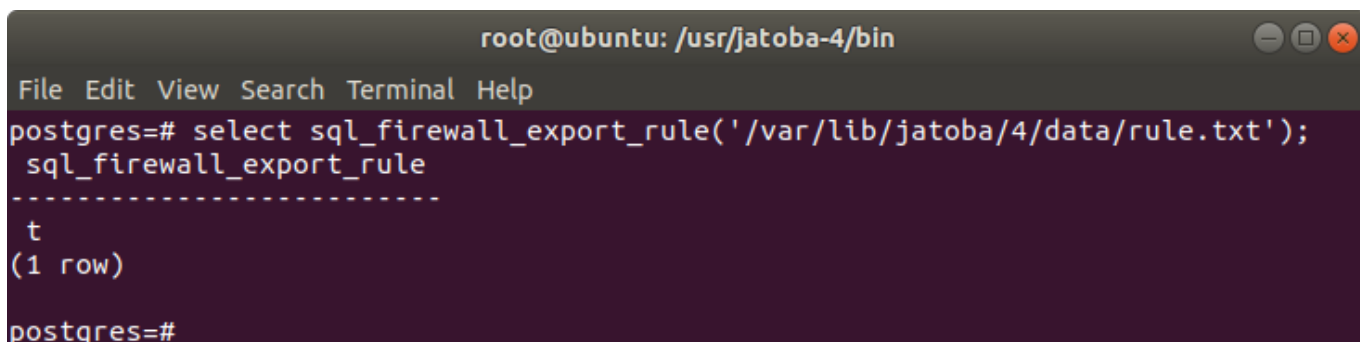
Для предотвращения ошибки доступа к файлу рекомендуется размещать его в директории DATA, где у системной учетной записи postgres есть полные права на чтение и запись.

В рассматриваемом примере для ОС GNU/Linux файл rule.txt расположен по пути:

```
/var/lib/jatoba/4/data/rule.txt
```

SQL-команда будет иметь вид:

```
SELECT  
sql_firewall_export_rule('/var/lib/jatoba/4/data/rule.txt');
```



```
root@ubuntu: /usr/jatoba-4/bin  
File Edit View Search Terminal Help  
postgres=# select sql_firewall_export_rule('/var/lib/jatoba/4/data/rule.txt');  
sql_firewall_export_rule  
-----  
t  
(1 row)  
postgres=#
```

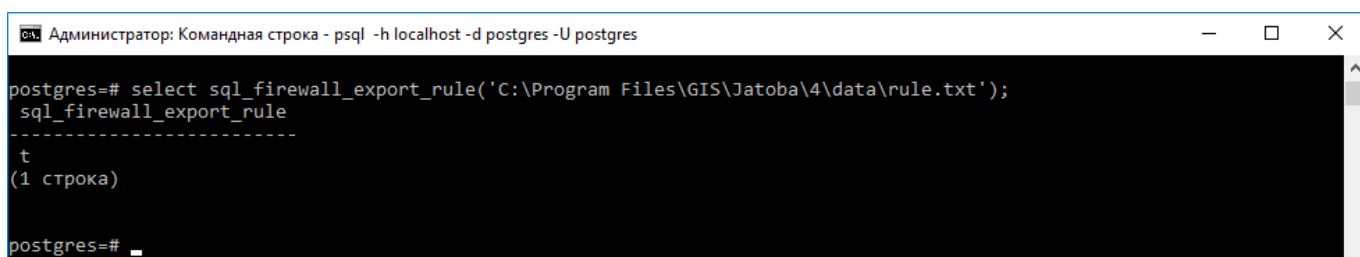
Рисунок 4.5 – Выполнение команды экспорта правил в ОС GNU/Linux

В рассматриваемом примере для ОС Windows, файл rule.txt расположен по пути:

```
C:\Program Files\GIS\Jatoba\4\data\rule.txt
```

SQL-команда будет иметь вид:

```
select sql_firewall_export_rule('C:\Program  
Files\GIS\Jatoba\4\data\rule.txt');
```



```
Администратор: Командная строка - psql -h localhost -d postgres -U postgres  
postgres=# select sql_firewall_export_rule('C:\Program Files\GIS\Jatoba\4\data\rule.txt');  
sql_firewall_export_rule  
-----  
t  
(1 строка)  
postgres=#
```

Рисунок 4.6 – Выполнение команды экспорта правил в ОС Windows

4.2.2. Импорт правил компонента (sql_firewall_import_rule)

Функциональная возможность импорта правил компонента «sql_firewall» доступна в режиме «[disabled](#)» от имени и с правами привилегированного пользователя с атрибутом «Superuser».

SQL-команда экспорта правил SQL-брандмауэра имеет синтаксис:

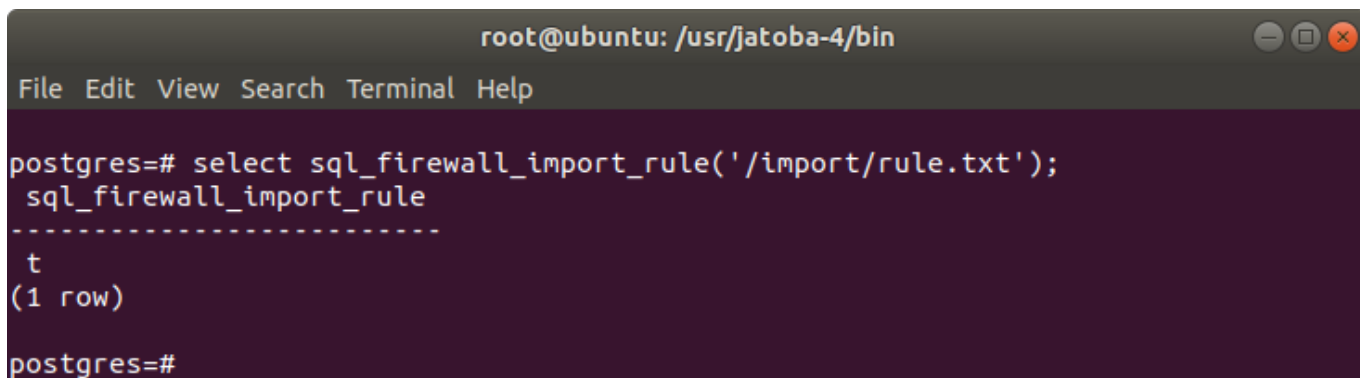
```
sql_firewall_import_rule('/path/to/rule.txt')
```

В рассматриваемом примере файл rule.txt расположен в директории «import».

В ОС GNU/Linux SQL-команда будет иметь вид:

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

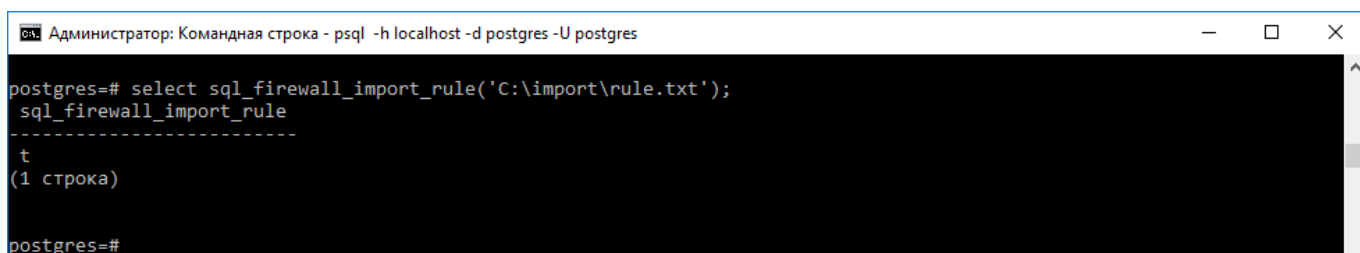
```
SELECT sql_firewall_import_rule('/import/rule.txt');
```



```
root@ubuntu: /usr/jatoba-4/bin
File Edit View Search Terminal Help
postgres=# select sql_firewall_import_rule('/import/rule.txt');
 sql_firewall_import_rule
-----
 t
(1 row)
postgres=#
```

Рисунок 4.7 – Выполнение команды импорта правил в ОС GNU/Linux
В ОС Windows SQL-команда будет иметь вид:

```
SELECT sql_firewall_import_rule('C:\import\rule.txt');
```



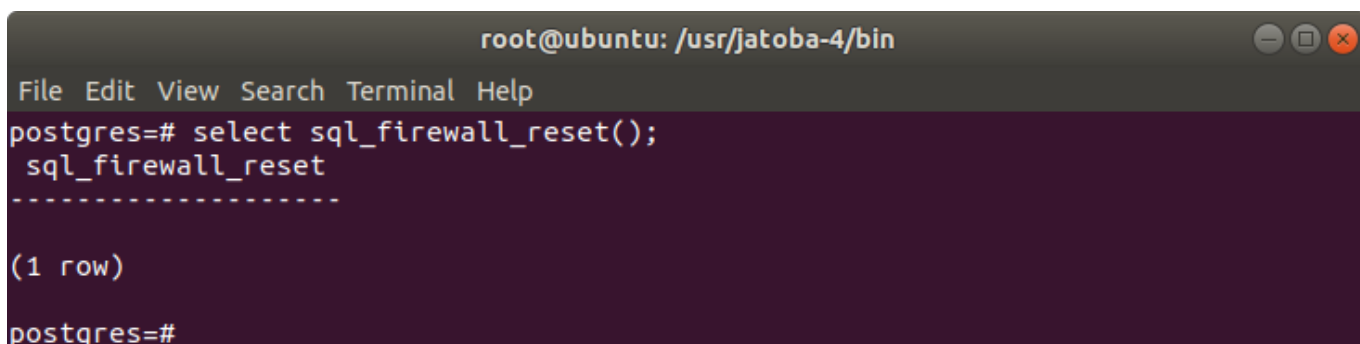
```
Администратор: Командная строка - psql -h localhost -d postgres -U postgres
postgres=# select sql_firewall_import_rule('C:\import\rule.txt');
 sql_firewall_import_rule
-----
 t
(1 строка)
postgres=#
```

Рисунок 4.8 – Выполнение команды импорта правил в ОС Windows

4.2.3. Очистка правил (sql_firewall_reset)

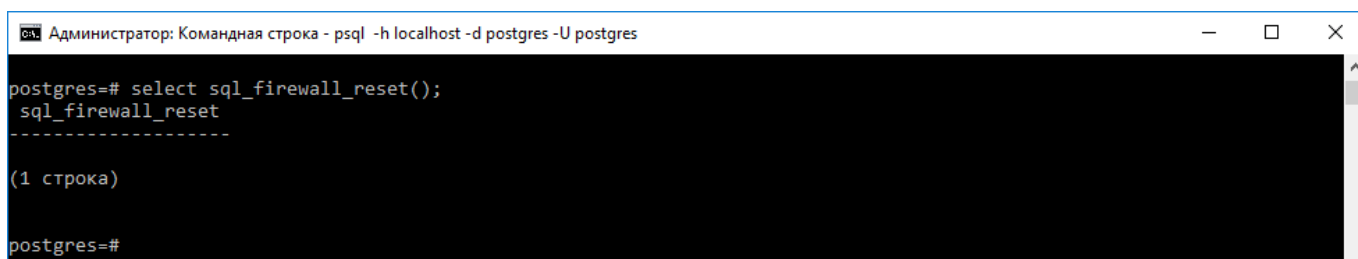
Функциональная возможность очистки правил компонента «sql_firewall» доступна в режиме «[disabled](#)» от имени и с правами привилегированного пользователя с атрибутом «Superuser» и выполняется SQL-командой:

```
SELECT sql_firewall_reset();
```



```
root@ubuntu: /usr/jatoba-4/bin
File Edit View Search Terminal Help
postgres=# select sql_firewall_reset();
 sql_firewall_reset
-----
(1 row)
postgres=#
```

Рисунок 4.9 – SQL-команда очистки правил в ОС GNU/Linux



```
Администратор: Командная строка - psql -h localhost -d postgres -U postgres

postgres=# select sql_firewall_reset();
 sql_firewall_reset
-----
(1 строка)
postgres=#
```

Рисунок 4.10 – SQL-команда очистки правил в ОС Windows

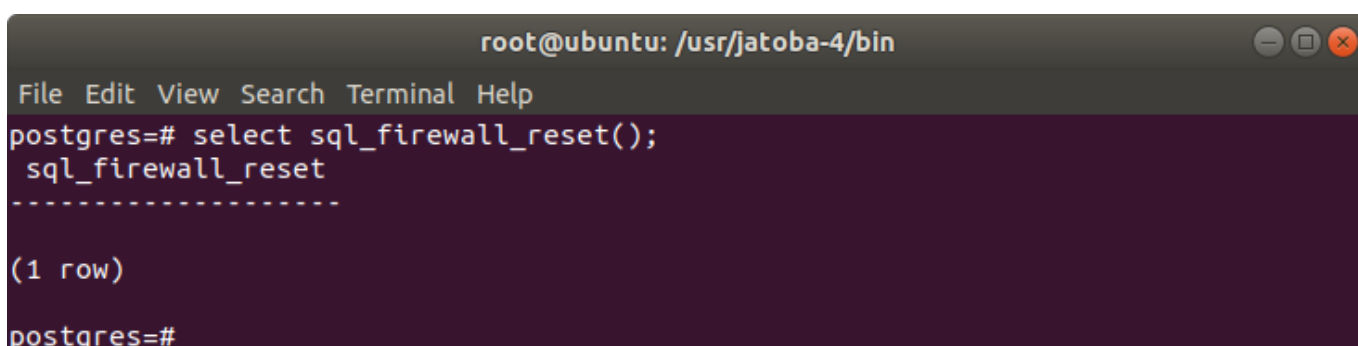
В итоге очистятся сформированные правила и перезапишется файл «rule.txt»

Проверить выполнение очистки правил компонента возможно вызовом функции «[sql_firewall.sql_firewall_statements](#)».

4.2.4. Очистка предупреждений и ошибок (sql_firewall_stat_reset)

Функциональная возможность очистки правил компонента «sql_firewall» доступна в режиме «[disabled](#)» от имени и с правами привилегированного пользователя с атрибутом «Superuser» и выполняется SQL-командой:

```
SELECT sql_firewall_reset();
```

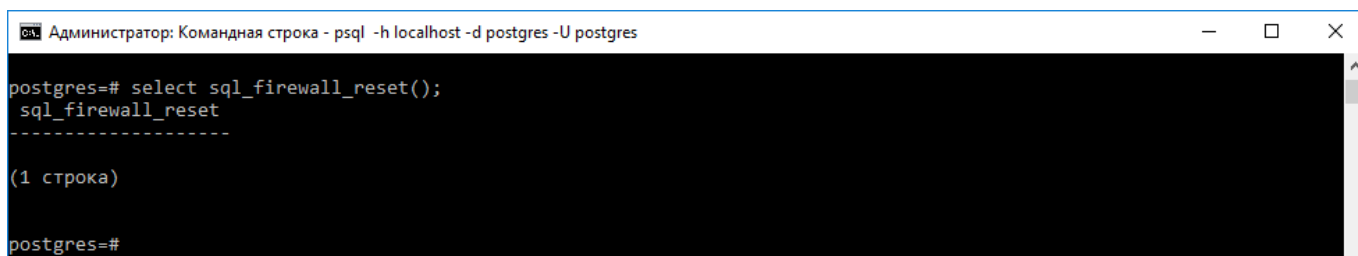


```
root@ubuntu: /usr/jatoba-4/bin

File Edit View Search Terminal Help

postgres=# select sql_firewall_reset();
 sql_firewall_reset
-----
(1 row)
postgres=#
```

Рисунок 4.11 – SQL-команда очистки предупреждений и ошибок в ОС GNU/Linux



```
Администратор: Командная строка - psql -h localhost -d postgres -U postgres

postgres=# select sql_firewall_reset();
 sql_firewall_reset
-----
(1 строка)
postgres=#
```

Рисунок 4.12 – SQL-команда очистки правил в ОС Windows

По результатам выполнения команды выполнится очистка счетчиков предупреждений и ошибок.

Проверить выполнение очистки правил компонента возможно вызовом функции «[sql_firewall.sql_firewall_stat](#)», через SQL-команду:

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------

```
SELECT * from sql_firewall.sql_firewall_stat;
```

5. УДАЛЕНИЕ

Для полного удаления компонента необходимо выполнить следующие действия:

- 1) удалить расширение SQL-командой:

```
DROP EXTENSION sql_firewall
```

- 2) удалить или закомментировать в конфигурационном файле postgresql.conf загрузку компонента:

```
#shared_preload_libraries = 'sql_firewall'  
#sql_firewall.firewall = 'disabled'  
#sql_firewall.max = 5000
```

- 3) перезапустить СУБД.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

SQL	–	Structured Query Language – язык структурированных запросов
БД	–	База данных
ОС	–	Операционная система
СУБД	–	Система управления базами данных

[illegible]

№ изменения: _____	Подпись отв. лица: _____	Дата внесения изм: _____
--------------------	--------------------------	--------------------------