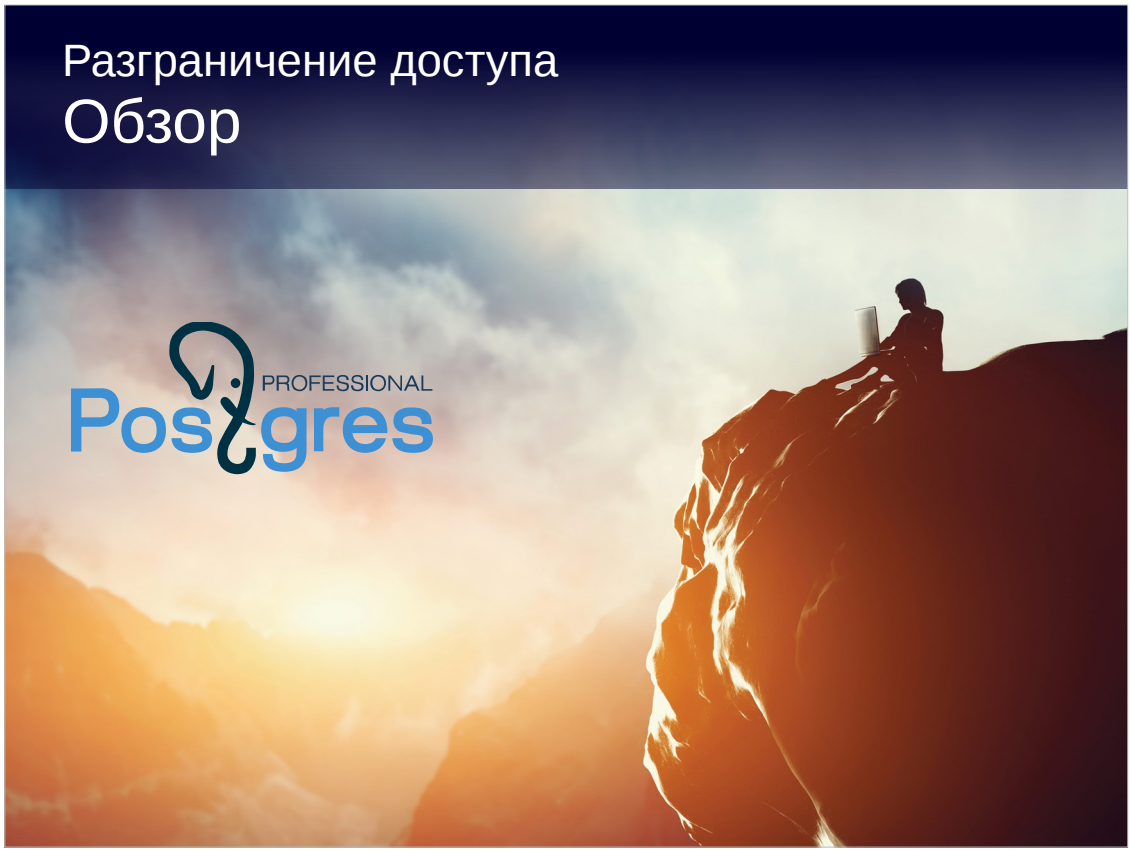


Авторы: Егор Рогов, Павел Лузанов

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

Темы

Роли

Привилегии

Политики защиты строк

Подключение к серверу

Роли

Два в одном:
пользователь СУБД
и группа пользователей

Роль совмещает в себе два понятия: пользователь СУБД и группа пользователей.

В свое время в PostgreSQL для этих понятий были отдельные сущности, и это до сих пор находит отражение, например, в названии некоторых команд (CREATE USER и CREATE ROLE). Но потом их объединили.

Роли

Роль

- пользователь СУБД (не связана с пользователем ОС)
- может включать в себя другие роли — быть «групповой ролью»

Псевдороль public

- неявно включает в себя все остальные роли

Любая роль может рассматриваться и как пользователь СУБД, и в то же время может включать в себя другие роли.

Роли никак не связаны с именами пользователей ОС, хотя некоторые программы это предполагают, выбирая значения по умолчанию.

При создании кластера определяется одна начальная роль, имеющая суперпользовательский доступ. В дальнейшем роли можно создавать, изменять и удалять.

Также существует псевдороль public, в которую всегда неявно включены все остальные роли.

https://postgrespro.ru/docs/postgresql/9.6/user-manag.html

https://postgrespro.ru/docs/postgresql/9.6/database-roles.html

Атрибуты

Атрибуты определяют свойства роли

`LOGIN`	возможность подключения
`SUPERUSER`	суперпользователь
`CREATEDB`	возможность создавать базы данных
`CREATEROLE`	возможность создавать роли
`REPLICATION`	использование протокола репликации

и другие

Роль обладает некоторыми атрибутами, определяющими ее общие особенности и права (не связанные с правами доступа к объектам).

Обычно атрибуты имеют два варианта, например, CREATEDB (дает право на создание БД) и NOCREATEDB (не дает такого права). Как правило, по умолчанию выбирается ограничивающий вариант.

Если у роли нет атрибута LOGIN, она не сможет подключиться к серверу. Такие роли тоже имеют смысл в качестве групповых.

В таблице перечислены лишь некоторые из атрибутов. Атрибуты INHERIT и BYPASSRLS рассматривается чуть дальше в этой теме.

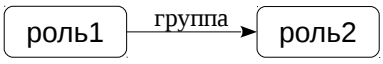
https://postgrespro.ru/docs/postgresql/9.6/role-attributes.html

https://postgrespro.ru/docs/postgresql/9.6/sql-createrole.html

Участие в групповой роли

Включение роли в группу

```
роль1: GRANT группа TO роль2;
```



Исключение роли из группы

```
роль1: REVOKE группа FROM роль2;
```


Роль может быть включена в другую роль подобно тому, как пользователь Unix может быть включен в группу.

Однако PostgreSQL не делает различий между ролями-пользователями и ролями-группами. Поэтому любая роль может быть включена в любую другую. При этом возможно появление цепочек включений (но циклы не допускаются).

Смысл такого включения состоит в том, что для роли становятся доступны атрибуты (и привилегии, о которых пойдет речь дальше), которыми обладает групповая роль. Чтобы воспользоваться правами, которые дают атрибуты групповой роли, необходимо переключиться в нее командой SET ROLE.

Правом на включение и исключение других ролей в данную роль обладают:

- сама эта роль,

- роль с атрибутом SUPERUSER (суперпользователь)

- роль с атрибутом CREATEROLE.

https://postgrespro.ru/docs/postgresql/9.6/role-membership.html

Привилегии

Определяют права доступа ролей к объектам

Привилегии определяются на пересечении объектов кластера и ролей. Они ограничивают действия, доступные для ролей в отношении этих объектов.

Виды привилегий

Таблицы

`SELECT`	чтение данных	можно на уровне столбцов
`INSERT`	вставка строк	
`UPDATE`	изменение строк	
`REFERENCES`	внешний ключ	
`DELETE`	удаление строк	
`TRUNCATE`	очистка таблицы	
`TRIGGER`	создание триггеров	

Представления — SELECT и TRIGGER

Последовательности

`SELECT`	currval		
`UPDATE`		nextval	setval
`USAGE`	currval	nextval	

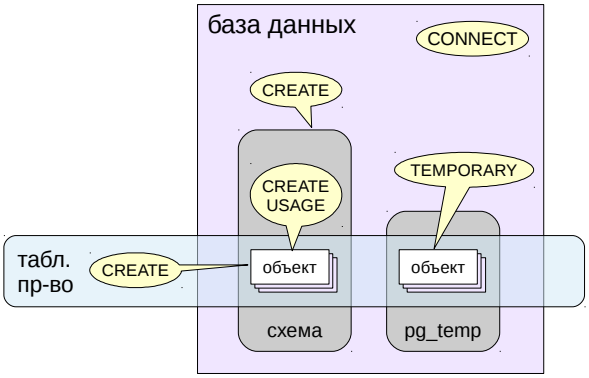
Список возможных привилегий отличается для объектов различных типов. Привилегии для основных объектов приведены на этом и следующем слайдах.

Больше всего привилегий определено для таблиц. Некоторые из них можно определить не только для всей таблицы, но и для отдельных столбцов.

https://postgrespro.ru/docs/postgresql/9.6/sql-grant.html

Виды привилегий

Табличные пространства,
базы данных, схемы



Функции

EXECUTE	выполнение с правами: SECURITY INVOKER — вызвавшего (по умолчанию), SECURITY DEFINER — создавшего

Для табличных пространств есть привилегия CREATE, разрешающая создание объектов в этом пространстве.

Для баз данных привилегия CREATE разрешает создавать схемы в этой БД, а для схемы привилегия CREATE разрешает создавать объекты в этой схеме.

Поскольку точное имя схемы для временных объектов заранее неизвестно, привилегия на создание временных таблиц вынесено на уровень БД (TEMPORARY).

Привилегия USAGE схемы разрешает обращаться к объектам в этой схеме.

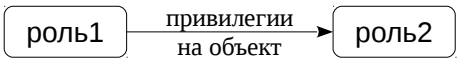
Привилегия CONNECT базы данных разрешает подключение к этой БД.

Для функций есть единственная привилегия EXECUTE, разрешающая выполнение этой функции. Тонкий момент связан с тем, от имени какого пользователя будет выполняться функция. Если функция создана как SECURITY INVOKER (по умолчанию), она выполняется с правами вызывающего пользователя. Если же указать фразу SECURITY DEFINER, функция работает с правами создавшего ее пользователя.

Управление привилегиями

Выдача привилегий

```
роль1: GRANT привилегии ON объект TO роль2;
```



Отзыв привилегий

```
роль1: REVOKE привилегии ON объект FROM роль2;
```


Право выдачи и отзыва привилегий на объект имеет владелец этого объекта (и суперпользователь).

Синтаксис команд GRANT и REVOKE достаточно сложен и позволяет указывать как отдельные, так и все возможные привилегии; как отдельные объекты, так и группы объектов, входящие в определенные схемы и т. п.

https://postgrespro.ru/docs/postgresql/9.6/sql-grant.html

https://postgrespro.ru/docs/postgresql/9.6/sql-revoke.html

Суперпользователи

Кто входит в категорию

роли с атрибутом SUPERUSER

Права

полный доступ ко всем объектам — проверки не выполняются

В целом можно сказать, что доступ роли к объекту определяется привилегиями. Но имеет смысл выделить три категории ролей и рассмотреть их по отдельности.

Проще всего с ролями с атрибутом суперпользователя. Такие роли могут делать все, что угодно — для них проверки разграничения доступа не выполняются.

Владелец объекта

Кто входит в категорию

- изначально — роль, создавшая объект (можно сменить)
- а также роли, включенные в роль владельца

Права

- изначально все привилегии для объекта (можно отозвать)
- действия со своими объектами, не регламентируемые привилегиями, например: удаление, выдача и отзыв привилегий и т. п.

У каждого объекта есть роль, владеющая этим объектом («владелец»). Изначально это роль, создавшая объект, хотя потом владельца можно сменить. Неочевидный момент: владельцем считается не только сама роль-владелец, но и любая другая роль, включенная в нее.

Владелец объекта сразу получает полный набор привилегий для этого объекта.

В принципе, эти привилегии можно отозвать, но владелец объекта обладает также неотъемлемым правом совершать действия, не регламентируемые привилегиями. В частности, он может выдавать и отзывать привилегии (в том числе и себе самому), удалять объект и т. п.

Остальные роли

Postgres PROFESSIONAL

Кто входит в категорию

все остальные (не суперпользователи и не владельцы)

Права

доступ в рамках выданных привилегий

обычно наследуют привилегии групповых ролей
(но атрибут NOINHERIT требует явного переключения роли)

Наконец, все остальные роли имеют доступ к объекту только в рамках выданных им привилегий. В том числе учитываются и привилегии групповых ролей, в которые эти роли входят (включая псевдороль public, в которую неявно включены все остальные роли).

Обычно роль сразу обладает всеми «групповыми» полномочиями. Это поведение можно изменить, указав роли атрибут NOINHERIT — тогда, чтобы воспользоваться привилегиями групповых ролей, надо будет явно переключиться с помощью SET ROLE.

https://postgrespro.ru/docs/postgresql/9.6/ddl-priv.html

Чтобы проверить, есть ли у роли необходимая привилегия в отношении некоторого объекта, можно воспользоваться функциями has_*_privilege:

https://postgrespro.ru/docs/postgresql/9.6/functions-info.html

Выданные привилегии удобно показывают команды psql, описывающие объект (см. раздаточный материал по системному каталогу).

Привилегии по умолчанию

Привилегии псевдороли public

- подключение к любой базе данных
- доступ к схеме public и создание в ней объектов
- доступ к системному каталогу
- выполнение любых функций
- привилегии выдаются автоматически для каждого нового объекта

Настраиваемые привилегии по умолчанию

- возможность дополнительно выдать или отозвать привилегии для только что созданного объекта

Псевдороль public имеет довольно широкий спектр привилегий по умолчанию. В частности, право подключения к любой базе данных, доступ к системному каталогу и схеме public, право выполнения любых функций. Причем эти привилегии появляются автоматически при создании новых объектов. Например, как только создается новая функция, public немедленно получает привилегию на ее выполнение.

Это, с одной стороны, позволяет комфортно работать, не задумываясь о привилегиях, но с другой, создает определенные сложности, если разграничение доступа действительно необходимо.

Существует также механизм «привилегий по умолчанию», который позволяет автоматически выдавать необходимые привилегии при создании нового объекта. Этот механизм может использоваться и для отзыва права выполнения функций у псевдороли public.

https://postgrespro.ru/docs/postgresql/9.6/sql-alterdefaultprivileges

Политики защиты строк

Дополнение к системе привилегий
для разграничения доступа к таблицам на уровне строк

Привилегии позволяют разграничить доступ на уровне таблиц и столбцов, а политики защиты строк (row-level security) — на уровне строк. Этот механизм появился в PostgreSQL 9.5.

Политика защиты строк

Защита включается явно для каждой таблицы

- не действует на владельца (если не включить принудительно)
- не действует на роли с атрибутом BYPASSRLS
- не действует на ограничения целостности

Политика определяет доступность строки

- отдельные предикаты для существующих и для новых строк
- предикаты вычисляются с правами вызывающего
- политика определяется для таблицы в разрезе ролей и команд SELECT, INSERT, UPDATE, DELETE
- доступ возможен, если он разрешен хотя бы одной политикой

Защита строк по умолчанию выключена. При необходимости ее надо включать явно для каждой таблицы.

Политики защиты строк не действуют на владельца таблицы (как правило), на роли со специальным атрибутом BYPASSRLS и на ограничения целостности (уникальность, внешние ключи).

Политики определяется для таблицы для набора команд (SELECT, INSERT, UPDATE, DELETE) и для определенных ролей. По сути, каждая из политик — это предикат, вычисляемый для каждой строки. Если предикат истинен — доступ к строке разрешается (достаточно, чтобы доступ разрешила хотя бы одна политика).

Можно указывать разные предикаты для доступа к существующим строкам и для добавления новых строк (тогда, например, операция UPDATE сработает, только если оба предикаты будут истинны).

https://postgrespro.ru/docs/postgresql/9.6/ddl-rowsecurity

Подключение

При появлении нового клиента
сервер решает, разрешить ли подключение

Перед тем, как разграничивать доступ к объектам внутри базы данных, для каждого нового клиента сервер определяет, надо ли вообще разрешить подключение к БД.

При подключении

1. Строки pg_hba.conf просматриваются сверху вниз
2. Выбирается первая запись, которой соответствуют параметры подключения (тип, база, пользователь и адрес)

```
# TYPE          DATABASE        USER            ADDRESS                 METHOD
local           all             postgres                                peer
local           all             all                                     peer
host            all             all             127.0.0.1/32            md5
host            all             all             ::1/128                 md5
```

local — сокет
host — TCP/IP

all — любая БД
имя БД

all — любая роль
имя роли

параметр listen_addresses

all — любой IP
IP/маска
доменное имя

Настройки подключения определяются в конфигурационном файле pg_hba.conf (от «host-based authentication»). Как и с основным конфигурационным файлом postgresql.conf, изменения вступают в силу только после перечитывания файла сервером (select pg_reload_conf() в SQL, либо pg_ctl reload из операционной системы).

При появлении клиента конфигурационный файл просматривается сверху вниз. Для каждой строки определяется, подходит ли она к запрашиваемому клиентом подключению (по соответствию типа подключения, имени БД, имени пользователя и IP-адресу).

Ниже перечислены только самые основные возможности. Подробнее см. https://postgrespro.ru/docs/postgresql/9.6/client-authentication.html

Подключение — local (unix-сокеты, не доступно в Windows) или host (подключение по протоколу TCP/IP).

База данных – ключевое слово all (соответствует любой БД) или имя конкретной роли.

Пользователь — all или имя конкретной роли.

Адрес — all, конкретный IP-адрес с маской или доменное имя. Не указывается для типа local.
По умолчанию PostgreSQL слушает входящие соединения только с localhost; обычно параметр listen_addresses ставят в значение «*» (слушать все интерфейсы) и дальше регулируют доступ средствами pg_hba.conf.

При подключении

3. Выполняется аутентификация указанным методом и проверка привилегии CONNECT
4. Удачно — доступ разрешается, иначе — запрещается (если не подошла ни одна запись — доступ запрещается)

```
# TYPE          DATABASE      USER              ADDRESS           METHOD
local           all           postgres                            peer
local           all           all                                 peer
host            all           all               127.0.0.1/32      md5
host            all           all               ::1/128           md5
```

trust — верить
reject — отказать
md5 — запросить пароль
peer — спросить ОС

Когда в файле найдена подходящая строка, выполняется аутентификация указанным в этой строке методом, а также проверяется наличие привилегии CONNECT. Если результат проверки успешен, то подключение разрешается, иначе — запрещается (другие строки при этом уже не рассматриваются).

Если ни одна из строк не подошла, то доступ также запрещается.

Таким образом, записи в файле должны идти сверху вниз от частного к общему.

Существует множество методов аутентификации: https://postgrespro.ru/docs/postgresql/9.6/auth-methods.html. Ниже перечислены только самые основные.

Метод trust безусловно разрешает подключение. Если вопросы безопасности не важны, можно указать «все all» и метод trust — тогда будут разрешены все подключения.

Метод reject наоборот, безусловно запрещает подключение.

Наиболее распространен метод md5, который запрашивает у пользователя пароль и проверяет его соответствие паролю, который хранится в системном каталоге кластера.

Метод peer запрашивает имя пользователя у операционной системы и разрешает подключение, если имя пользователя ОС и пользователя БД совпадают. (При желании можно установить и другие соответствия имен.)

Парольная аутентификация

На сервере

пароль устанавливается при создании или изменении роли:
`ALTER|CREATE ROLE PASSWORD 'пароль' VALID UNTIL 'время'`

пользователю без пароля будет отказано в доступе

пароль хранится в системном каталоге pg_authid

Ввод пароля на клиенте

вручную

из переменной окружения PGPASSWORD

из файла ~/.pgpass (строки в формате *узел:порт:база:роль:пароль*)

Если используется аутентификация по паролю, при создании роли должен быть указан эталонный пароль, иначе в доступе будет отказано.

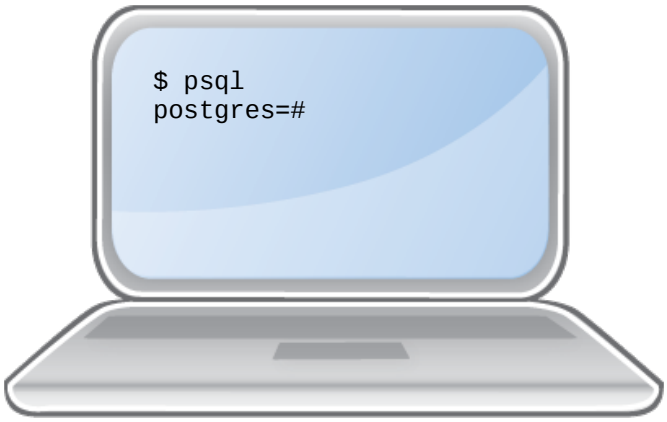
Пароль хранится в системном каталоге в таблице pg_authid.

Пользователь может вводить пароль вручную, а может автоматизировать ввод. Для этого есть две возможности.

Во-первых, можно задать пароль в переменной окружения $PGPASSWORD на клиенте. Однако это неудобно, если приходится подключаться к нескольким базам, и не рекомендуется из соображений безопасности.

Во-вторых, можно задать пароли в файле ~/.pgpass на клиенте. К файлу должен иметь доступ только владелец, иначе PostgreSQL проигнорирует его.

Демонстрация



Итоги

Роли, привилегии и политики — гибкий механизм, позволяющий по-разному организовать работу

- можно легко разрешить все всем
- можно строго разграничить доступ, если это необходимо

При создании новых ролей надо позаботиться о возможности их подключения к серверу

Практика

1. Создайте две роли (пароль должен совпадать с именем):
   – employee — сотрудник магазина,
   – buyer — покупатель.
2. Убедитесь, что созданные роли могут подключиться к БД.
3. Отзовите у роли public права выполнения всех функций и подключения к БД.
4. Разграничьте доступ таким образом, чтобы:
   – сотрудник мог только заказывать книги, а также добавлять авторов и книги,
   – покупатель мог только приобретать книги.
5. Проверьте выполненные настройки в приложении.

1. Сотрудник — внутренний пользователь приложения, аутентификация выполняется на уровне СУБД.

Покупатель — внешний пользователь. В реальном интернет-магазине управление такими пользователями ложится на приложение, а все запросы поступают в СУБД от одной «обобщенной» роли (buyer). Идентификатор конкретного покупателя может передаваться как параметр (но в нашем приложении мы этого не делаем).

4. Вообще говоря, разграничение доступа должно быть заложено, в том числе, в приложение. В нашем учебном приложении разграничение не сделано специально: вместо этого на веб-странице можно явно выбрать роль, от имени которой пойдет запрос в СУБД. Это позволяет посмотреть, как поведет себя серверная часть при некорректной работе приложения.

Итак, пользователям нужно выдать:

- Право подключения к БД bookstore и доступ к схеме bookstore.

- Доступ к представлениям, к которым происходит непосредственное обращение.

- Доступ к функциям, которые вызываются как часть API. Если оставить функции SECURITY INVOKER, придется выдавать доступ и ко всем «нижележащим» объектам (таблицам, другим функциям). Однако удобнее просто объявить API-функции как SECIRUTY DEFINER.

Разумеется, ролям нужно выдать привилегии только на те объекты, доступ к которым у них должен быть.

Практика

1. Создайте базу данных, схему и в ней таблицу с двумя столбцами: ключ и значение.
2. Создайте роль.
3. Выясните IP-адрес виртуальной машины и настройте систему так, чтобы подключение по этому адресу разрешалось только созданной роли к созданной базе данных при вводе верного пароля.
4. Настройте разграничение доступа таким образом, чтобы созданная роль могла читать таблицу и изменять в ней значения, но не ключи.

3. IP-адрес можно узнать командой ifconfig.