

Управление доступом Привилегии



Авторские права

© Postgres Professional, 2017 год.

Авторы: Егор Рогов, Павел Лузанов

Использование материалов курса

Некоммерческое использование материалов курса (презентации, демонстрации) разрешается без ограничений. Коммерческое использование возможно только с письменного разрешения компании Postgres Professional. Запрещается внесение изменений в материалы курса.

Обратная связь

Отзывы, замечания и предложения направляйте по адресу:

edu@postgrespro.ru

Отказ от ответственности

Компания Postgres Professional не несет никакой ответственности за любые повреждения и убытки, включая потерю дохода, нанесенные прямым или непрямым, специальным или случайным использованием материалов курса. Компания Postgres Professional не предоставляет каких-либо гарантий на материалы курса. Материалы курса предоставляются на основе принципа «как есть» и компания Postgres Professional не обязана предоставлять сопровождение, поддержку, обновления, расширения и изменения.

Виды привилегий для разных объектов

Категории ролей с точки зрения управления доступом

Выдача и отзыв привилегий и право перевыдачи

Групповые привилегии

Привилегии по умолчанию

Примеры управления доступом

Таблицы

SELECT	чтение данных	} можно задавать 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	

Привилегии (права) определяют права доступа ролей к объектам.

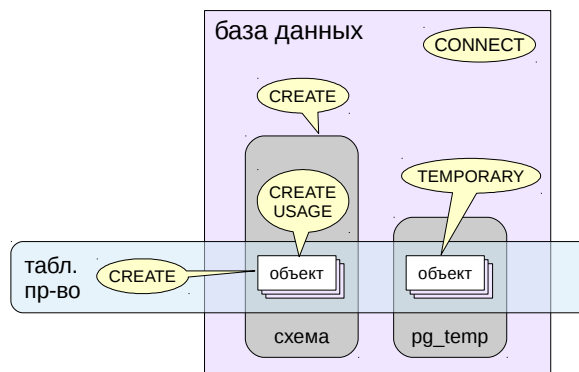
<https://postgrespro.ru/docs/postgresql/10/ddl-priv>

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.

Виды привилегий

Табличные пространства,
базы данных, схемы



Функции

EXECUTE

выполнение с правами:

SECURITY INVOKER — вызвавшего (по умолчанию),

SECURITY DEFINER — создавшего

4

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 (точнее, с правами ее текущего владельца).

Суперпользователи

полный доступ ко всем объектам — проверки не выполняются

Владельцы

доступ в рамках выданных привилегий
(изначально получает полный набор)

а также действия, не регламентируемые привилегиями,
например: удаление, выдача и отзыв привилегий и т. п.

Остальные роли

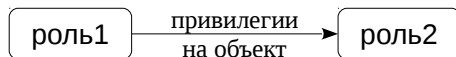
доступ исключительно в рамках выданных привилегий

В целом можно сказать, что доступ роли к объекту определяется привилегиями. Но можно выделить три категории ролей.

1. Проще всего с ролями с атрибутом суперпользователя. Такие роли могут делать все, что угодно — для них проверки разграничения доступа не выполняются.
2. 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.
3. Все остальные роли имеют доступ к объекту только в рамках выданных им привилегий.

Выдача привилегии

роль1: GRANT привилегии ON объект TO *роль2*;



одна привилегия может быть независимо выдана разными ролями

Отзыв привилегии

роль1: REVOKE привилегии ON объект FROM *роль2*;

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

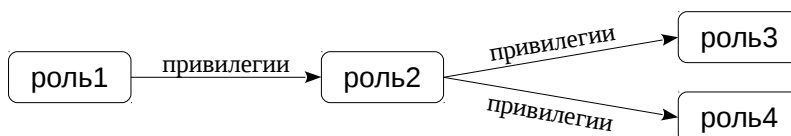
Синтаксис команд GRANT и REVOKE достаточно сложен и позволяет указывать как отдельные, так и все возможные привилегии; как отдельные объекты, так и группы объектов, входящие в определенные схемы и т. п.

<https://postgrespro.ru/docs/postgresql/10/sql-grant>

<https://postgrespro.ru/docs/postgresql/10/sql-revoke>

Выдача привилегии с правом передачи

роль1: GRANT привилегии ON объект TO роль2 WITH GRANT OPTION;



Отзыв привилегии

роль1: REVOKE привилегии ON объект FROM роль2 CASCADE;

Отзыв права передачи

роль1: REVOKE GRANT OPTION FOR привилегии ON объект FROM роль2 CASCADE;

обязательно,
если привилегия
была передана
другим ролям

При выдаче роли полномочия можно передать ей право дальнейшей передачи (и отзыва) этого полномочия. Это выполняется командой GRANT ... WITH GRANT OPTION (похожая конструкция WITH ADMIN OPTION для атрибутов рассматривалась в предыдущей теме «Роли»). Если роль воспользуется этим правом, образуется иерархия ролей.

Отозвать привилегию можно с помощью команды REVOKE. Роль может отозвать привилегию только у той роли, которой она его выдала. Например, *роль1* не может отозвать привилегию непосредственно у *роль3* или *роль4*.

Однако при отзыве привилегии у *роль2* привилегия будет автоматически отозвана у всех ролей в иерархии. Для этого надо указать ключевое слово CASCADE (если иерархия непуста, то без CASCADE будет ошибка).

Право передачи можно отозвать, не отзывая у роли саму привилегию. Это выполняется с помощью команды REVOKE GRANT OPTION FOR. Слово CASCADE имеет здесь такое же значение, как и при отзыве привилегии.

Роль получает привилегии своих групповых ролей

поведение зависит от атрибута роли:

INHERIT (по умолчанию) *NOINHERIT*

автоматически наследует привилегии группы требуется явное переключение с помощью SET ROLE

Преднастроенные роли

<code>pg_signal_backend</code>	— сигналы обслуживающим процессам	} <code>pg_monitor</code>
<code>pg_read_all_settings</code>	— чтение конфигурационных параметров	
<code>pg_read_all_stats</code>	— доступ к статистике	
<code>pg_stat_scan_tables</code>	— статистика, вызывающая блокировки	

8

Роль может получать привилегии для доступа к объекту не только непосредственно, но и от групповых ролей, в которые она входит.

Роль с атрибутом *INHERIT* (по умолчанию) автоматически обладает привилегиями всех групп, в которые она входит. Это касается и псевдороль `public`, в которую неявно входят все роли.

Если же роль создана как *NOINHERIT*, то она может воспользоваться привилегиями группы, только выполнив команду `SET ROLE` и таким образом переключившись на эту групповую роль. Все действия, совершаемые ролью, будут совершаться от имени групповой роли (например, групповая роль будет владельцем созданных объектов).

<https://postgrespro.ru/docs/postgresql/10/role-membership>

В PostgreSQL уже есть ряд преднастроенных групп, обладающих рядом специальных привилегий для выполнения задач, которые обычно может совершать только суперпользователь. Большая часть из них появилась в версии PostgreSQL 10.

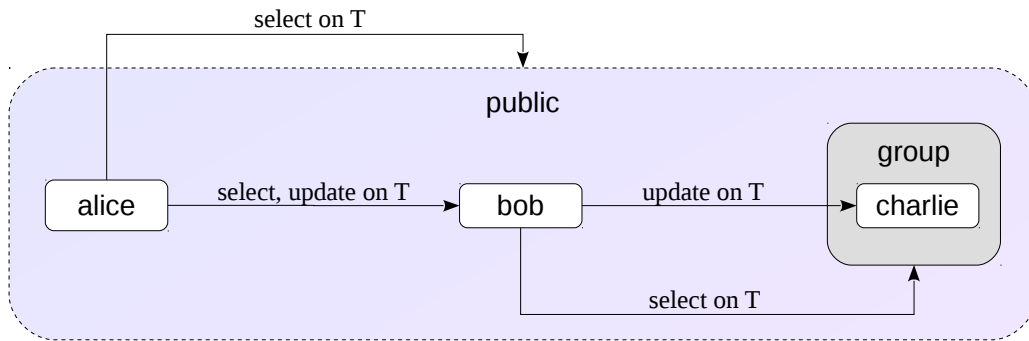
<https://postgrespro.ru/docs/postgresql/10/default-roles>

Аналогично можно создавать и собственные групповые роли, например, для управления резервным копированием и т. п.

Вопрос

Привилегии на таблицу T получены Бобом от Алисы.

Если Алиса выполнит
REVOKE ALL ON T FROM bob CASCADE,
какие привилегии останутся у Чарли?



9

У Чарли останется только привилегия на чтение T, полученная от public. Все привилегии, выданные Бобом другим ролям, будут отозваны.

По умолчанию роль public получает ряд привилегий

для баз данных	CONNECT (подключение) TEMPORARY (создание временных таблиц)
для схемы public	CREATE (создание объектов) USAGE (доступ к объектам)
для схем pg_catalog и information_schema	USAGE (доступ к объектам)
для функций	EXECUTE (выполнение)

Удобно, но не безопасно

По умолчанию псевдороль public получает ряд привилегий (то есть, фактически, их получают все роли):

- подключение и создание временных таблиц для **всех** баз данных
- использование схемы **public** и создание в ней объектов
- использование схем **pg_catalog** и **information_schema**
- выполнение **всех** функций

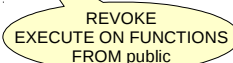
Такое поведение может оказаться нежелательным. В этом случае надо явно отзывать у public привилегии.

<https://postgrespro.ru/docs/postgresql/10/sql-grant>

Дополнительные привилегии по умолчанию

```
ALTER DEFAULT PRIVILEGES  
[ IN SCHEMA схема ]  
GRANT привилегии ON класс_объектов TO роль;
```

```
ALTER DEFAULT PRIVILEGES  
[ IN SCHEMA схема ]  
REVOKE привилегии ON класс_объектов FROM роль;
```



```
REVOKE  
EXECUTE ON FUNCTIONS  
FROM public
```

Можно настроить дополнительные привилегии, выставляемые по умолчанию, с помощью команды ALTER DEFAULT PRIVILEGES.

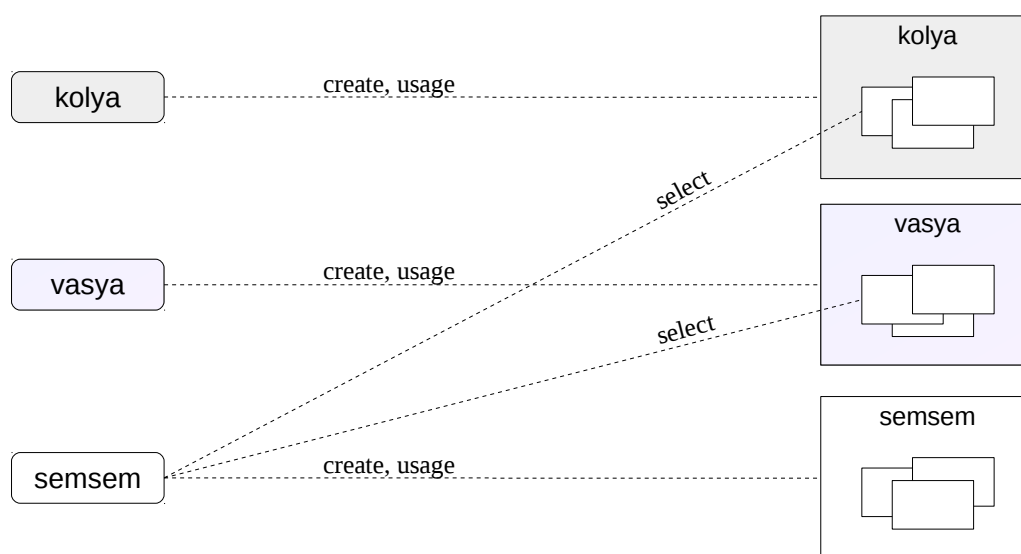
Привилегии, которые получает псевдороль public, здесь не отражаются и не могут быть отключены. Поэтому, даже если отозвать у public все привилегии на все существующие объекты, эта роль будет получать привилегии на все новые объекты, создающиеся в системе.

Но с помощью привилегий по умолчанию можно автоматически отзывать у public привилегию выполнения функций, как только она появляется.

<https://postgrespro.ru/docs/postgresql/10/sql-alterdefaultprivileges>



Пример 1



Механизмы управления доступом (роли, атрибуты и привилегии, а также схемы) весьма гибки и позволяют в разных случаях организовать работу удобным образом. Приведем два примера.

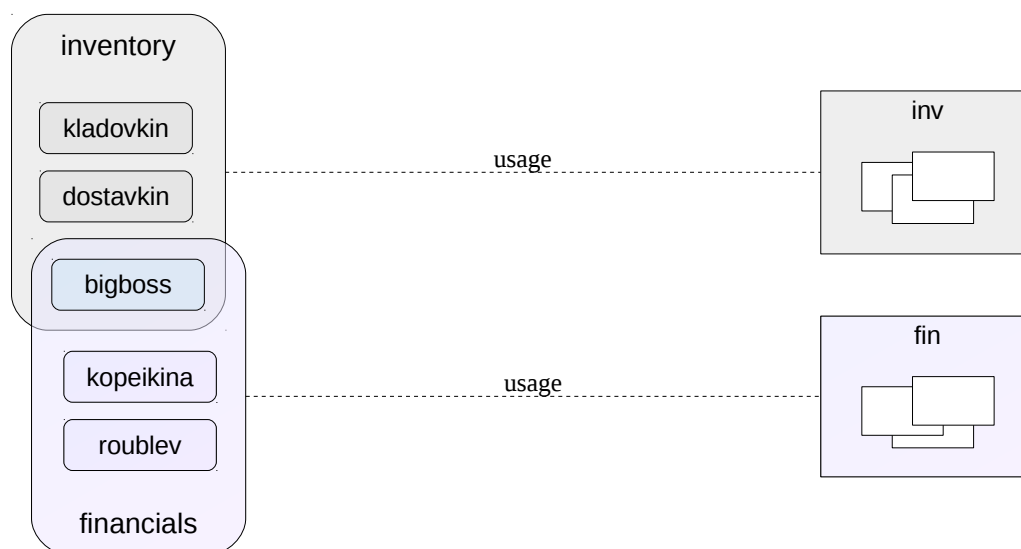
Пример простого использования.

Семен Семенович и его студенты Коля и Вася занимаются научными исследованиями и хранят результаты измерений в базе данных.

Системный администратор создал на факультетском сервере БД каждому из них своего пользователя и одноименную схему. Путь поиска он оставил по умолчанию. Групповые роли не используются.

Каждый пользователь является владельцем объектов, которые он создает в своей схеме. Кроме того, Коля и Вася дают доступ к некоторым своим таблицам Семену Семеновичу, чтобы он мог посмотреть их результаты.

Пример 2



Более сложный пример.

На выделенный сервер БД установлена система автоматизации предприятия. Она состоит из двух модулей — складского и финансового.

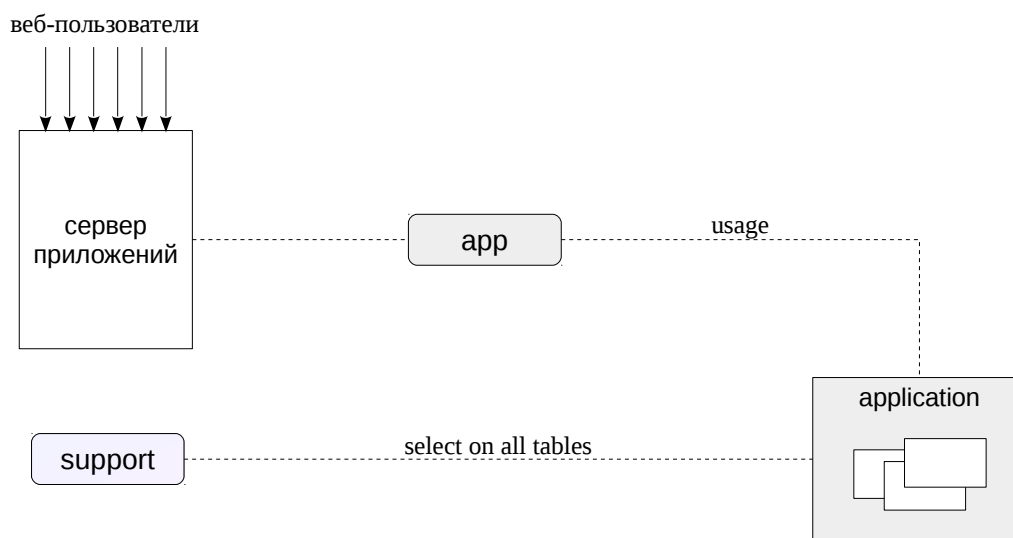
Для каждого из модулей создана своя схема (`inv` для склада и `fin` для финансов) и своя групповая роль (`inventory` для склада и `financials` для финансов). Групповая роль является владельцем всех объектов в своей схеме.

В складском отделе работают Кладовкин и Доставкин. Для каждого создан свой пользователь и включен в группу `inventory`.

В финансовом отделе работают Копейкина и Рублев. Для каждого создан свой пользователь и включен в группу `financials`.

Для генерального директора предприятия на всякий случай создан пользователь, включенный в обе группы, хотя вряд ли он когда-либо воспользуется системой.

Пример 3



15

Очень часто бывает так, что аутентификация пользователей информационной системы происходит не в СУБД, а на сервере приложений. Действительно, если у системы тысячи пользователей и есть возможность онлайн-регистрации, нет никакого смысла управлять ими в СУБД.

В этом случае сервер приложений подключается к базе данных под одной заранее созданной ролью, а информацию о пользователе передает, при необходимости, в виде какого-то контекста.

Но даже и в этом случае в СУБД наверняка понадобятся несколько ролей. Например, специальная роль для технической поддержки, сотрудники которой смогут читать таблицы продуктивного сервера для быстрого решения проблем.

Привилегии определяют права доступа ролей к объектам
Роли, атрибуты и привилегии, схемы — гибкий механизм,
позволяющий по-разному организовать работу

можно легко разрешить все всем

можно строго разграничить доступ, если это необходимо

Организация работы таким образом, чтобы одни пользователи имели полный доступ к таблицам, а другие могли только запрашивать, но не изменять информацию.

1. Создайте новую базу данных и две роли: `writer` и `reader`.
2. Отзовите у роли `public` все привилегии на схему `public`, выдайте роли `writer` обе привилегии, а роли `reader` — только `usage`.
3. Настройте привилегии по умолчанию так, чтобы роль `reader` получала доступ на чтение к таблицам, принадлежащим `writer` в схеме `public`.
4. Создайте пользователей `w1` в группе `writer` и `r1` в группе `reader`.
5. Под пользователем `writer` создайте таблицу.
6. Убедитесь, что `r1` имеет доступ к таблице только на чтение, а `w1` имеет к ней полный доступ, включая удаление.