Министерство науки и образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени   
Н.Э. Баумана

(национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)



**УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ К ЛАБОРАТОРНОЙ РАБОТЕ**

**ПО КУРСУ «БАЗЫ ДАННЫХ»**

**Лабораторная работа №1** **«Разграничение прав в PostgreSQL»**

Авторы:

Кудрявцев А.П., [kudryavtsevap@bmstu.ru](mailto:kudryavtsevap@bmstu.ru)

Фомин М.М.

Москва, 2025

## Управление доступом к данным

Одним из основных способов обеспечения конфиденциальности и целостности информации в БД выступает механизм аутентификации. Аутентификация– это процедура проверки подлинности пользователя (точнее, его идентификатора). Обычно пользователь подтверждает то, что он является именно тем, за кого он себя выдает, путем ввода в систему уникальной (неизвестной другим) информации о себе. В простейшем случае это символьный пароль, но возможен и более сложный подход подтверждения подлинности, в том числе биометрическая аутентификация или электронные способы аутентификации. Если пользователь успешно прошел процедуру аутентификации, СУБД осуществляет его авторизацию. Авторизация– это процедура предоставления пользователю определенных ресурсов и прав на их использование. Все дальнейшее взаимодействие пользователя с объектами БД строго регламентируется в соответствии с назначенными правами. Пользователи могут обладать разными правами на один и тот же объект, одни могут лишь просматривать данные, другие – только добавлять, третьи – осуществлять чтение, вставку и редактирование.

Привилегиями доступаназывают разрешение на использование определенной услуги управления данными для доступа к объекту данных (например, чтение, запись или исполнение), предоставляемое идентифицированному пользователю.

PostgreSQL управляет правами доступа к базе данных, используя концепцию ролей. Роль может рассматриваться как пользователь базы данных или как группа пользователей базы данных, в зависимости от настройки роли. Роли могут владеть объектами базы данных (например, таблицами и функциями) и назначать привилегии на эти объекты другим ролям, чтобы контролировать доступ к тем или иным объектам. Кроме того, можно предоставить членство в роли другой роли, что позволит роли-участнику использовать привилегии, назначенные другой роли.

Концепция ролей объединяет понятия «пользователи» и «группы». То есть нет пользователей и групп, есть только роли. Любая роль может выполнять функции пользователя, группы или и того, и другого.

Так как привилегии не назначаются пользователям непосредственно и приобретаются ими только через свою роль (или роли), управление индивидуальными правами пользователя, по сути, сводится к назначению ему ролей. Это упрощает такие операции, как добавление пользователя или смена подразделения пользователем.

Так, например, можно определить роли «Администратор», «Аналитик», «Пользователь» и в дальнейшем управлять доступом, используя роли, а не адресные привилегии доступа к объектам.

**При создании объекта БД ему назначается владелец. Обычно владельцем является роль, выполнившая оператор создания. Для большинства типов объектов изначально только владелец (или суперпользователь) может выполнять какие-либо действия с объектом. Чтобы разрешить другим ролям использовать объект, необходимо предоставить соответствующие привилегии*.***

Существуют различные виды привилегий:  
SELECT, INSERT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER, CREATE, CONNECT, TEMPORARY, EXECUTE, USAGE, SET, ALTER SYSTEM, и MAINTAIN. Привилегии, применяемые к конкретному объекту БД, различаются в зависимости от его типа (таблица, функция и т. д.).

* 1. Создание и модификация пользователей

Роли базы данных являются глобальными для всей установки кластера базы данных (не для отдельной базы данных). Для создания роли используется команда SQL CREATE ROLE:

CREATE ROLE *имя* [ [ WITH ] *параметр* [ ... ] ]

**Параметры**

SUPERUSER | NOSUPERUSER

| CREATEDB | NOCREATEDB

| CREATEROLE | NOCREATEROLE

| INHERIT | NOINHERIT

| LOGIN | NOLOGIN

| REPLICATION | NOREPLICATION

| CONNECTION LIMIT предел\_подключений

| PASSWORD 'пароль' | PASSWORD NULL

| VALID UNTIL 'дата\_время'

| IN ROLE имя\_роли [, ...]

| ROLE имя\_роли [, ...]

| ADMIN имя\_роли [, ...]

Прежде всего отметим, что бывают обычные пользователи и суперпользователи. Если задан атрибут SUPERUSER, то ограничения, налагаемые на обычного пользователя, не действуют. Суперпользователь может удалять любые объекты (базы данных и т. д.), какие захочет.

Важный момент: для создания базы данных у пользователя должны быть права на уровне экземпляра.

В команде после имени можно указать параметры создаваемой роли. В частности, можно указать параметр LOGIN, определяющий, разрешается ли новой роли вход на сервер, то есть, может ли эта роль стать начальным авторизованным именем при подключении клиента.

Существует также команда CREATE USER, которая является синонимом команды CREATE ROLE:

CREATE USER *имя* [ [ WITH ] *параметр* [ ... ] ]

…

| LOGIN | NOLOGIN

Единственное отличие между командами состоит в том, что для команды CREATE USER по умолчанию подразумевается использование параметра LOGIN, а в виде CREATE ROLE по умолчанию используется NOLOGIN.

**Параметры**

*имя -* имя создаваемой роли.

SUPERUSER, NOSUPERUSER - эти предложения определяют, будет ли эта роль «суперпользователем», который может переопределить все ограничения доступа в базе данных. Статус суперпользователя несёт опасность и назначать его следует только в случае необходимости. Создать нового суперпользователя может только суперпользователь. В отсутствие этих предложений по умолчанию подразумевается NOSUPERUSER.

CREATEDB, NOCREATEDB – эти предложения определяют, сможет ли роль создавать базы данных. Указание CREATEDB даёт новой роли это право, а NOCREATEDB запрещает роли создавать базы данных. По умолчанию подразумевается NOCREATEDB. Только суперпользователи или роли с правом CREATEDB могут указывать CREATEDB.

CREATEROLE, NOCREATEROLE - эти предложения определяют, сможет ли роль создавать, изменять и удалять другие роли, добавлять для них комментарии и изменять метку безопасности в любых ролях. Подробнее возможности, предоставляемые этим правом, описаны в Создание роли. Если предложение не указано, по умолчанию используется NOCREATEROLE.

INHERIT, NOINHERIT – эти предложения влияют, будет ли роль «наследовать» привилегии ролей, членом которых она является. Роль с атрибутом INHERIT может автоматически использовать любые права, назначенные всем ролям, в которые она включена, непосредственно или опосредованно. Иначе права другой роли можно использовать только после переключения на нее командой SET ROLE.

LOGIN, NOLOGIN – эти предложения определяют, разрешается ли новой роли вход на сервер; то есть, может ли эта роль стать начальным авторизованным именем при подключении клиента. Можно считать, что роль с атрибутом LOGIN соответствует пользователю. Роли без этого атрибута бывают полезны для управления доступом в базе данных, но это не пользователи в обычном понимании. По умолчанию подразумевается вариант NOLOGIN, за исключением вызова CREATE ROLE в виде CREATE USER.

REPLICATION, NOREPLICATION – эти предложения определяют, будет ли роль ролью репликации. Чтобы роль могла подключаться к серверу в режиме репликации (в режиме физической или логической репликации) и создавать/удалять слоты репликации, у неё должен быть этот атрибут (либо это должна быть роль суперпользователя). Роль, имеющая атрибут REPLICATION, обладает очень большими привилегиями, и, поэтому, этот атрибут должны иметь только роли, фактически используемые для репликации. По умолчанию подразумевается вариант NOREPLICATION. Выдавать атрибут REPLICATION могут только суперпользователи или роли с REPLICATION.

BYPASSRLS, NOBYPASSRLS – эти предложения определяют, будут ли для роли игнорироваться все политики защиты на уровне строк (RLS). По умолчанию подразумевается вариант NOBYPASSRLS. Выдавать атрибут BYPASSRLS могут только суперпользователи или роли с BYPASSRLS.

CONNECTION LIMIT *предел\_подключений* – если роли разрешён вход, этот параметр определяет, сколько параллельных подключений может установить роль. Значение -1 (по умолчанию) снимает ограничение. Заметьте, что под это ограничение подпадают только обычные подключения. Ни подготовленные транзакции, ни соединения фоновых рабочих процессов в расчёт не берутся.

PASSWORD *'пароль'*, PASSWORD NULL – задаёт пароль роли. (Пароль полезен только для ролей с атрибутом LOGIN, но задать его можно и для ролей без такого атрибута.) Если проверка подлинности по паролю не будет использоваться, этот параметр можно опустить. При указании пустого значения будет задан пароль NULL, что не позволит данному пользователю пройти проверку подлинности по паролю. При желании пароль NULL можно установить явно, указав PASSWORD NULL.

VALID UNTIL *'дата\_время'* – предложение VALID UNTIL устанавливает дату и время, после которого пароль роли перестаёт действовать. Если это предложение отсутствует, срок действия пароля будет неограниченным.

IN ROLE *имя\_роли***–** с предложением IN ROLE новая роль автоматически становится членом указанных существующих ролей.

ROLE *имя\_роли*–с предложением ROLE одна или нескольких указанных существующих ролей автоматически добавляются как члены новой роли.

ADMIN *имя\_роли*–предложение ADMIN работает подобно ROLE, но перечисленные в нём роли добавляются как члены новой роли с включённым атрибутом ADMIN, что даёт им право включать в новою роль другие роли.

Создадим несколько ролей. Начнем с создания групповой роли admin, в которую будем включать пользователей с привилегией SUPERUSER. Подсоединимся к любой БД сервера с правами SUPERUSER (по умолчанию это postgres), с помощью утилиты psql (рис. 1.1.1):

Пользователь postgres

Server [localhost]:

Database [postgres]:

Port [5432]:

Username [postgres]:

Пароль пользователя postgres:

psql (16.8)

ПРЕДУПРЕЖДЕНИЕ: Кодовая страница консоли (866) отличается от основной

страницы Windows (1251).

8-битовые (русские) символы могут отображаться некорректно.

Подробнее об этом смотрите документацию psql, раздел

"Notes for Windows users".

Введите "help", чтобы получить справку.

postgres=# \! chcp 1251

Текущая кодовая страница: 1251

postgres=#

Рисунок 1.1.1

Выполним команду CREATE ROLE (рис. 1.1.2):

Пользователь postgres

postgres=# CREATE ROLE admin WITH SUPERUSER;

CREATE ROLE

postgres=#

Рисунок 1.1.2

Этой командой мы создали роль суперпользователя, но без права подключения к серверу. Это обычная практика при создании групповой роли, групповая роль не должна иметь логин и пароль.

Следующим шагом, добавим пользователя adm1 в группу admin  
 (рис. 1.1.3):

Пользователь postgres

postgres=# CREATE ROLE adm1 WITH IN ROLE admin LOGIN PASSWORD '12345';

CREATE ROLE

postgres=#

Рисунок 1.1.3

Пользователь adm1 имеет право подключиться к серверу с заданным паролем.

Запустим еще одну утилиту psql от имени adm1 и создадим тестовую БД (рис. 1.1.4):

Пользователь adm1

Server [localhost]:

Database [postgres]:

Port [5432]:

Username [postgres]: adm1

Пароль пользователя adm1:

psql (16.8)

ПРЕДУПРЕЖДЕНИЕ: Кодовая страница консоли (866) отличается от основной

страницы Windows (1251).

8-битовые (русские) символы могут отображаться некорректно.

Подробнее об этом смотрите документацию psql, раздел

"Notes for Windows users".

Введите "help", чтобы получить справку.

postgres=> \! chcp 1251

Текущая кодовая страница: 1251

postgres=> CREATE DATABASE db\_test;

ОШИБКА: нет прав для создания базы данных

postgres=>

Рисунок 1.1.4

У суперпользователя нет прав! Может нам удастся создать таблицу в БД postgres (рис. 1.1.5):

Пользователь adm1

postgres=> CREATE TABLE test (id int);

ОШИБКА: нет доступа к схеме public

СТРОКА 1: CREATE TABLE test (id int);

^

postgres=>

Рисунок 1.1.5

И здесь неудача. Все дело в том, что привилегии роли LOGIN, SUPERUSER, CREATEDB и CREATEROLE никогда не наследуются как обычные права на объекты базы данных. Чтобы ими воспользоваться, необходимо переключиться на роль, имеющую этот атрибут, с помощью команды SET ROLE. Таким образом мы, подключившись, как adm1, должны переключится на роль admin и внести изменения, что и сделаем   
(рис. 1.1.6):

Пользователь adm1

postgres=> SET ROLE admin;

SET

postgres=# CREATE DATABASE db\_test;

CREATE DATABASE

postgres=# \l

Список баз данных

Имя | Владелец | Кодировка | Провайдер локали | LC\_COLLATE |

-----------+----------+-----------+------------------+------------+

db\_test | admin | UTF8 | libc | ru-RU |

demo | postgres | UTF8 | libc | ru-RU |

postgres | postgres | UTF8 | libc | ru-RU |

template0 | postgres | UTF8 | libc | ru-RU |

| | | | |

template1 | postgres | UTF8 | libc | ru-RU |

| | | | |

(5 строк)

postgres=#

Рисунок 1.1.6

Работает, но вводить лишнюю команду, имея хоть и опосредованно привилегию суперпользователя, значит себя не уважать. Воспользуемся командой:

ALTER ROLE *указание\_роли* [ WITH ] *параметр* [ ... ]

**Параметры**

SUPERUSER | NOSUPERUSER

| CREATEDB | NOCREATEDB

| CREATEROLE | NOCREATEROLE

| INHERIT | NOINHERIT

| LOGIN | NOLOGIN

| REPLICATION | NOREPLICATION

| BYPASSRLS | NOBYPASSRLS

| CONNECTION LIMIT *предел\_подключений*

| PASSWORD '*пароль*' | PASSWORD NULL

| VALID UNTIL '*дата\_время*'

Перечисленные параметры имеют те же значение что и при создании роли.

Мы подсоединились к серверу (рис. 1.1.6) и переключились на роль admin, которая, является суперпользователем и тем самым имеет возможность назначить привилегию SUPERUSER роли adm1 (рис. 1.1.7):

postgres=# ALTER ROLE adm1 WITH SUPERUSER;

ALTER ROLE

Рисунок 1.1.7

Переключимся обратно на роль adm1 и создадим БД (рис. 1.1.8):

postgres=# SET ROLE adm1;

SET

postgres=# CREATE DATABASE db\_test3;

CREATE DATABASE

Рисунок 1.1.8

Утилита psql имеет команду \du для просмотра ролей сервера, введем ее (рис. 1.1.9):

postgres=# \du

Список ролей

Имя роли | Атрибуты

----------+------------------------------------------------------------------------

adm1 | Суперпользователь

admin | Суперпользователь, Вход запрещён

postgres | Суперпользователь, Создаёт роли, Создаёт БД, Репликация, Пропускать RLS

postgres=#

Рисунок 1.1.9

После получения прав суперпользователя, желательно изменить пароль, если он был вам выдан. Пароль меняется так же с помощью ALTER ROLE:

ALTER ROLE adm1 WITH PASSWORD 'новый пароль';

Создадим себе помощника adm2, включим его в группу admin и сделаем его суперпользователем. Для добавления и удаления роли из группы используется команда ALTER GROUP:

ALTER GROUP *указание\_роли* ADD USER *имя\_пользователя* [, ... ]

ALTER GROUP *указание\_роли* DROP USER *имя\_пользователя* [, ... ]

Здесь *указание\_роли*:

имя\_роли

| CURRENT\_ROLE

| CURRENT\_USER

| SESSION\_USER

Выполним задуманное (рис. 1.1.10):

Пользователь adm1

postgres=# CREATE ROLE adm2 WITH SUPERUSER LOGIN PASSWORD '23456';

CREATE ROLE

postgres=# ALTER GROUP admin ADD USER adm2;

ALTER ROLE

postgres=# \du

Список ролей

Имя роли | Атрибуты

----------+------------------------------------------------------------------------

adm1 | Суперпользователь

adm2 | Суперпользователь

admin | Суперпользователь, Вход запрещён

postgres | Суперпользователь, Создаёт роли, Создаёт БД, Репликация, Пропускать RLS

postgres=#

Рисунок 1.1.10

Заведем еще одного пользователя adm3 с привилегией SUPERUSER и добавим в роль admin, но с помощью pgAdmin 4[[1]](#footnote-1). Запустим pgAdmin от имени суперпользователя. Развернем дерево объектов в левой панели (рис. 1.1.11):

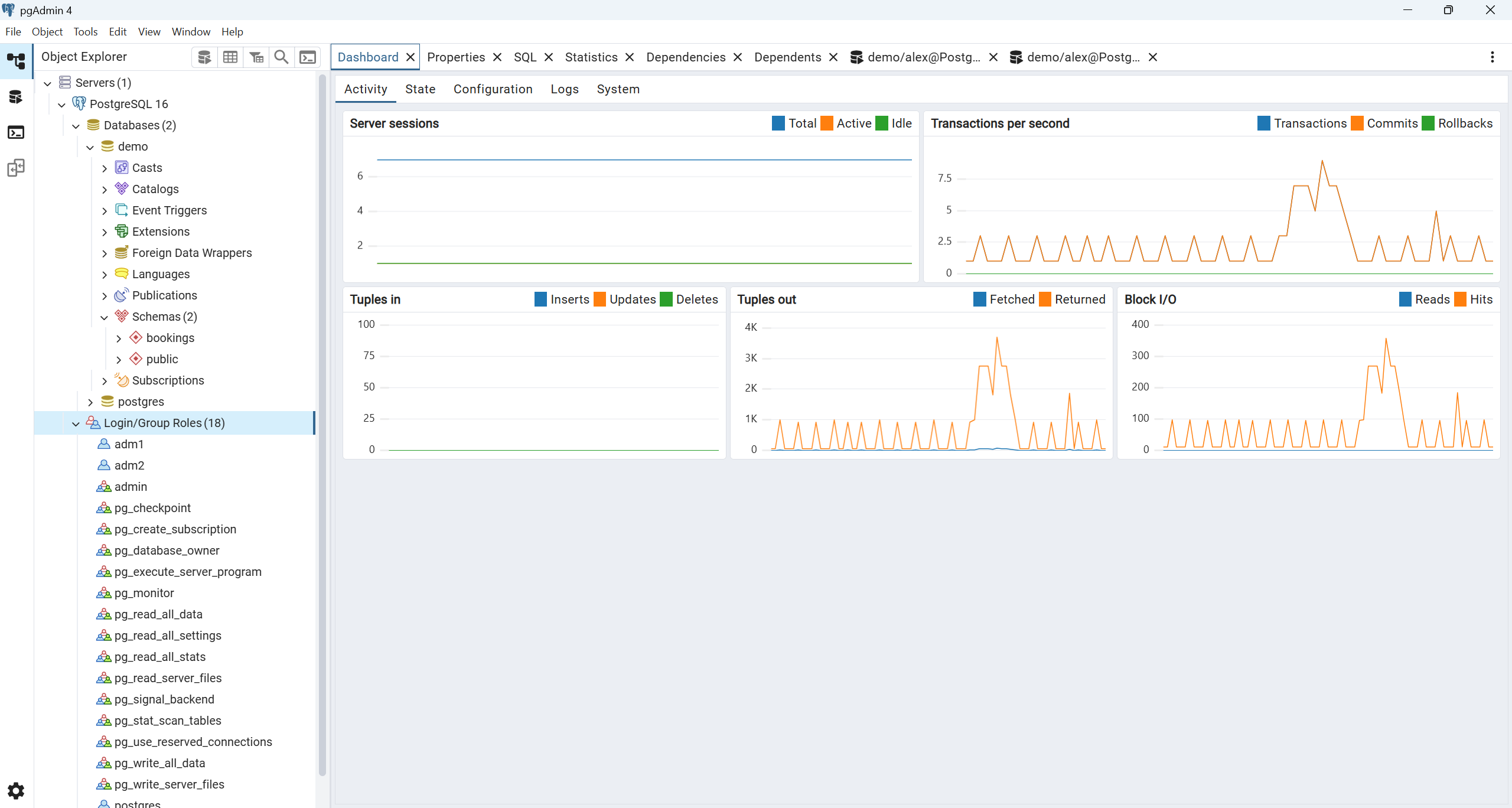


Рисунок 1. 1.11

Щелкнем ПКМ[[2]](#footnote-2) по узлу *Login/Group Roles* и выполним команду   
*Create-> Login/Group Role…* (рис. 1.1.12):

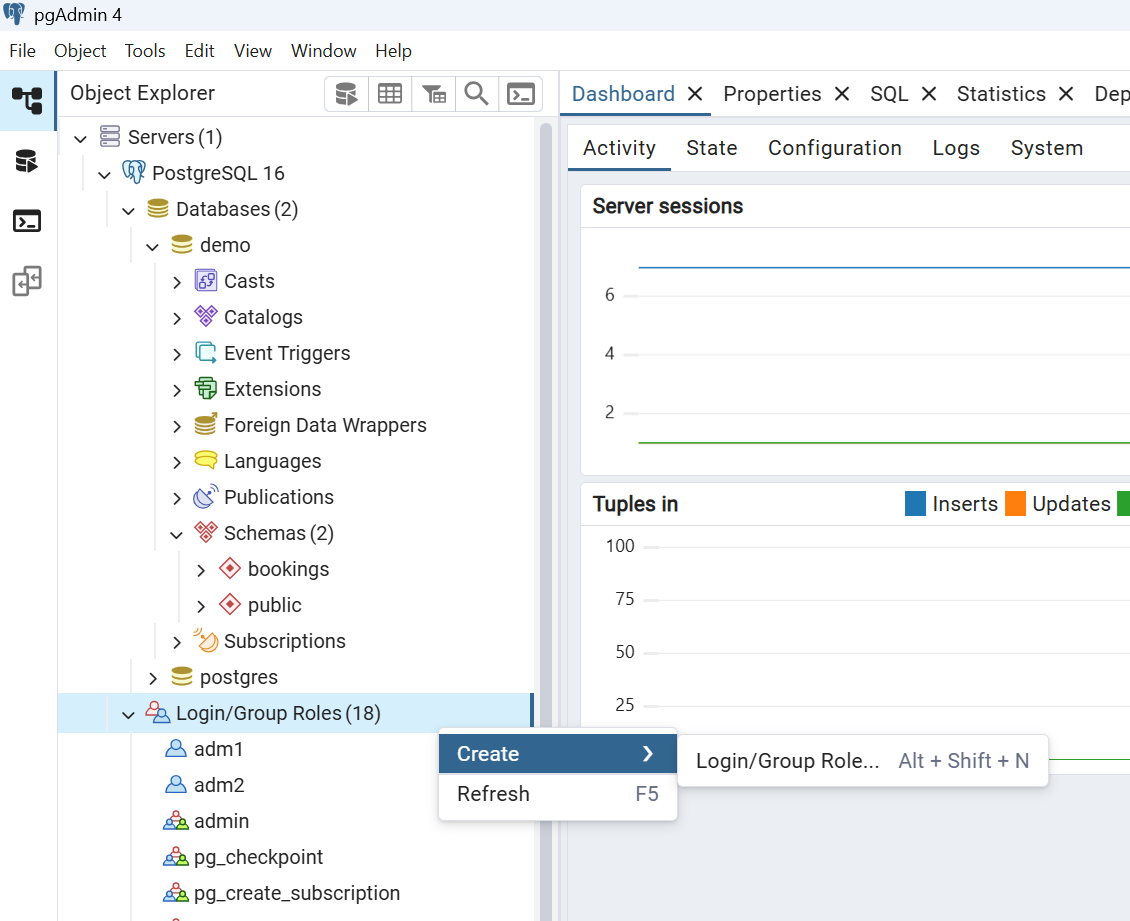


Рисунок 1.1.12

Откроется диалог *Group Role – Login/Group Roles* (рис. 1.1.13):

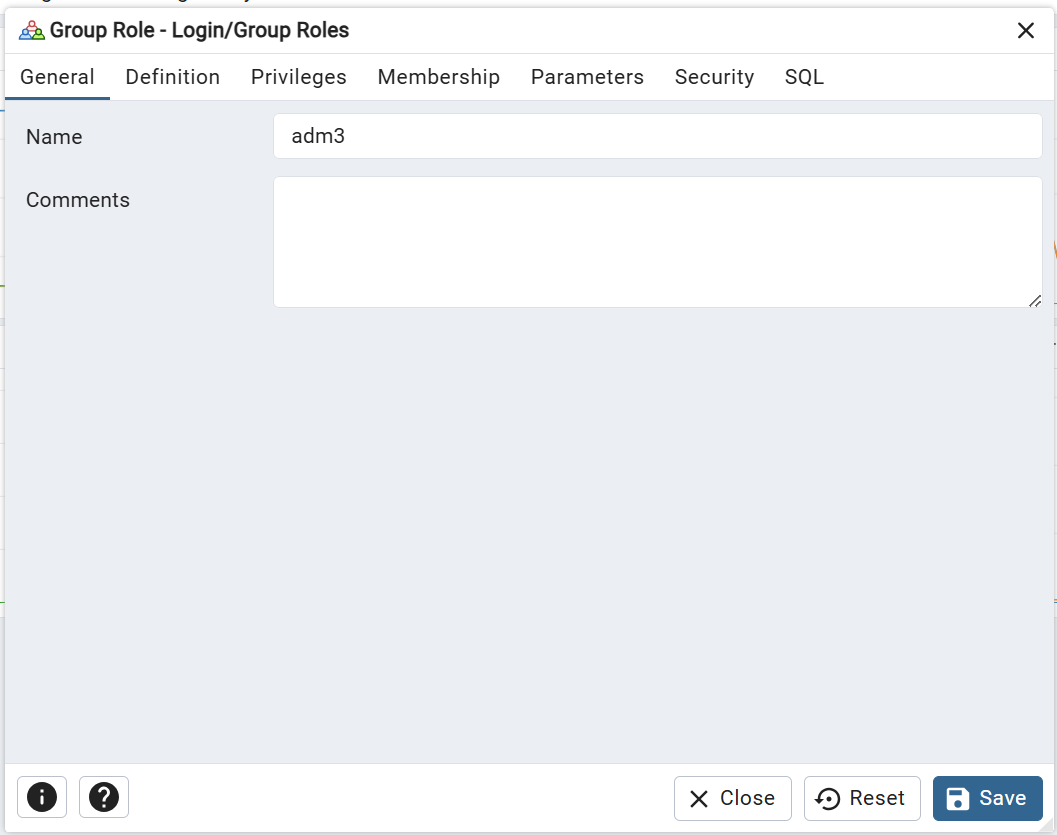


Рисунок 1.1.13

В поле Name имя пользователя и перейдем на вкладку *Definition*   
(рис. 1.1.14):

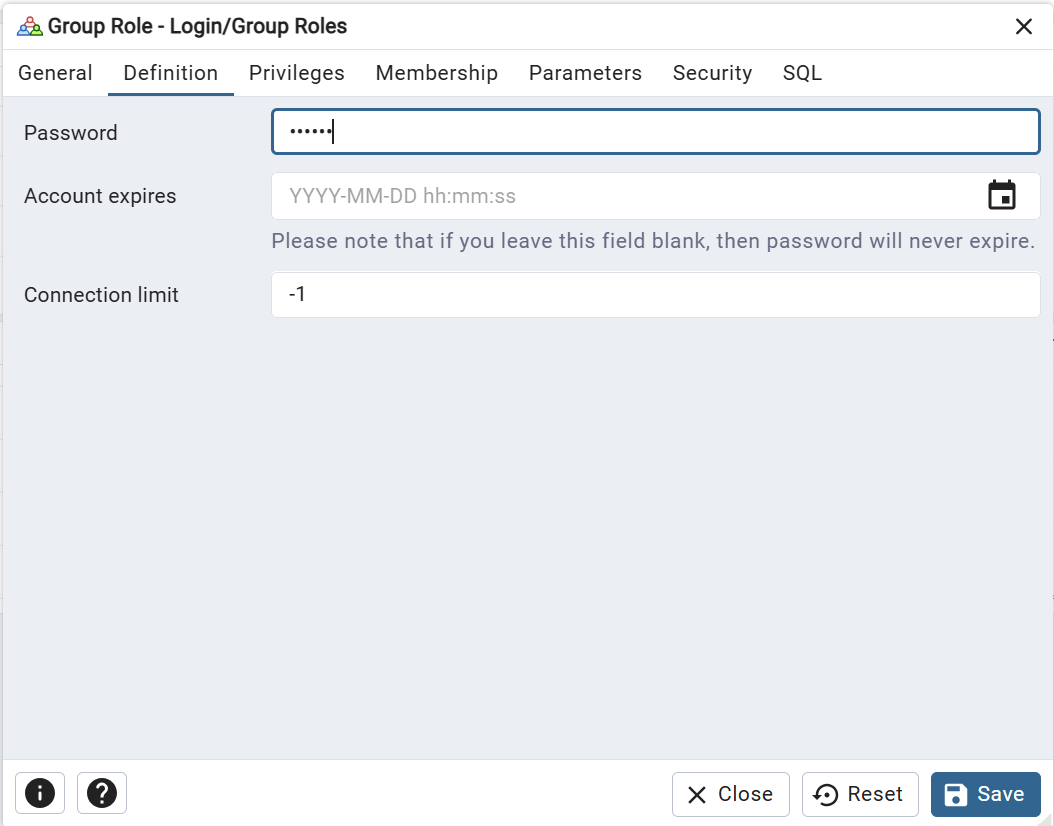


Рисунок 1.1.14

Зададим пароль и далее вкладка *Privileges* (рис. 1.1.15):

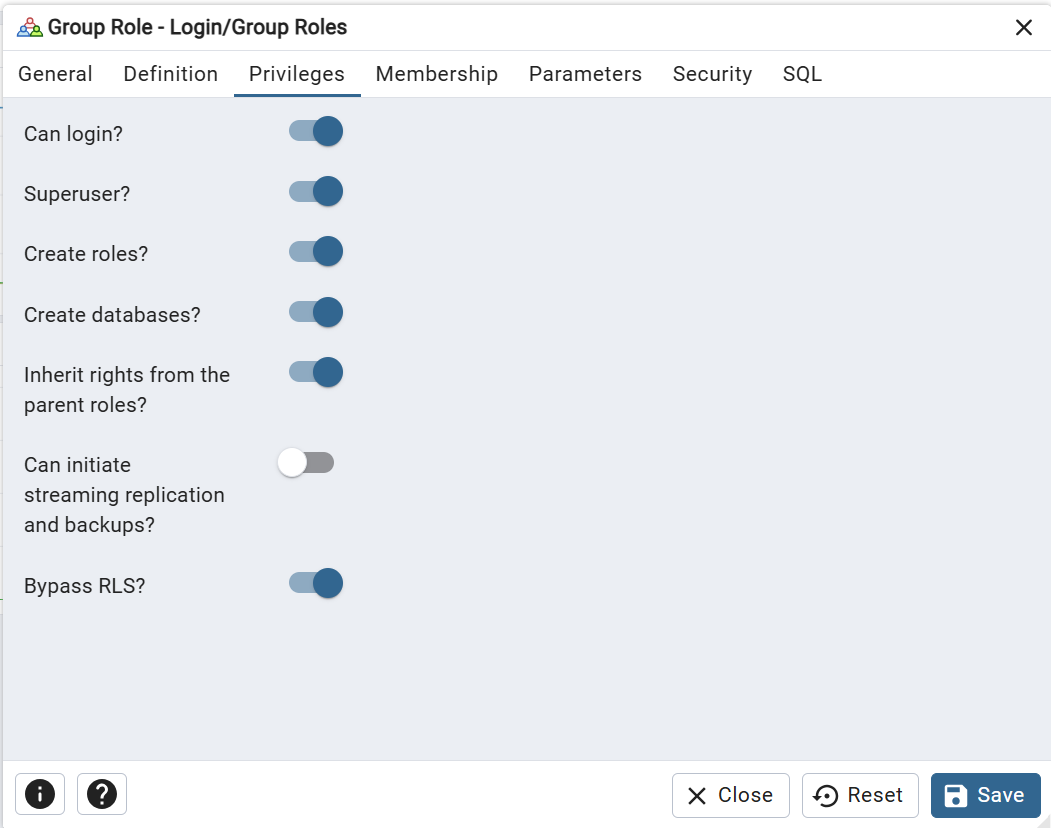


Рисунок 1.1.15

Разрешим adm3 все, кроме репликации и backups. На следующей вкладке *Membership* сделаем его членом группы admin. Роль, членом которой хотим сделать adm3 выбирается из выпадающего списка, в котором наряду с предопределёнными групповыми ролями, перечислены и роли, созданные нами (рис. 1.1.16):

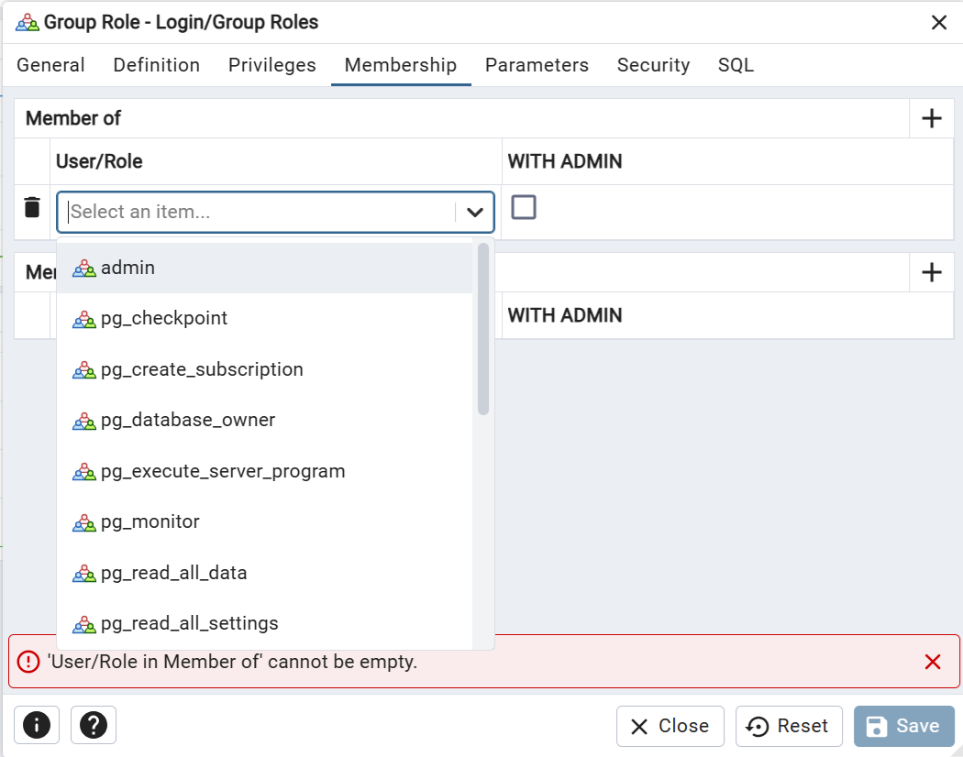


Рисунок 1.1.16

Следующие две вкладки мы пропустим и перейдем к вкладке *SQL* (рис. 1.1.17):

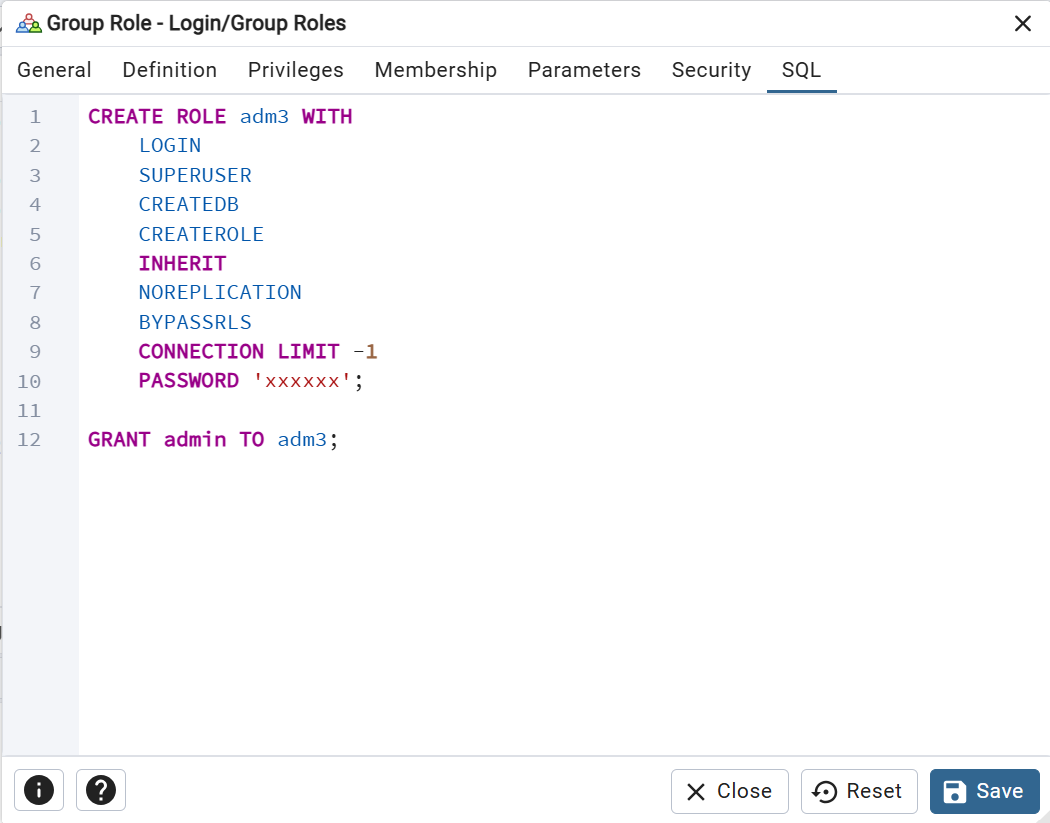


Рисунок 1.1.17

В скрипте перечислены все привилегии, кроме репликации и предложение, включающее adm3 в группу admin.

Зайдем в свойства (Properties) групповой роли admin (щелкнув по ней ПКМ) и посмотрим ее состав (рис. 1.1.18):

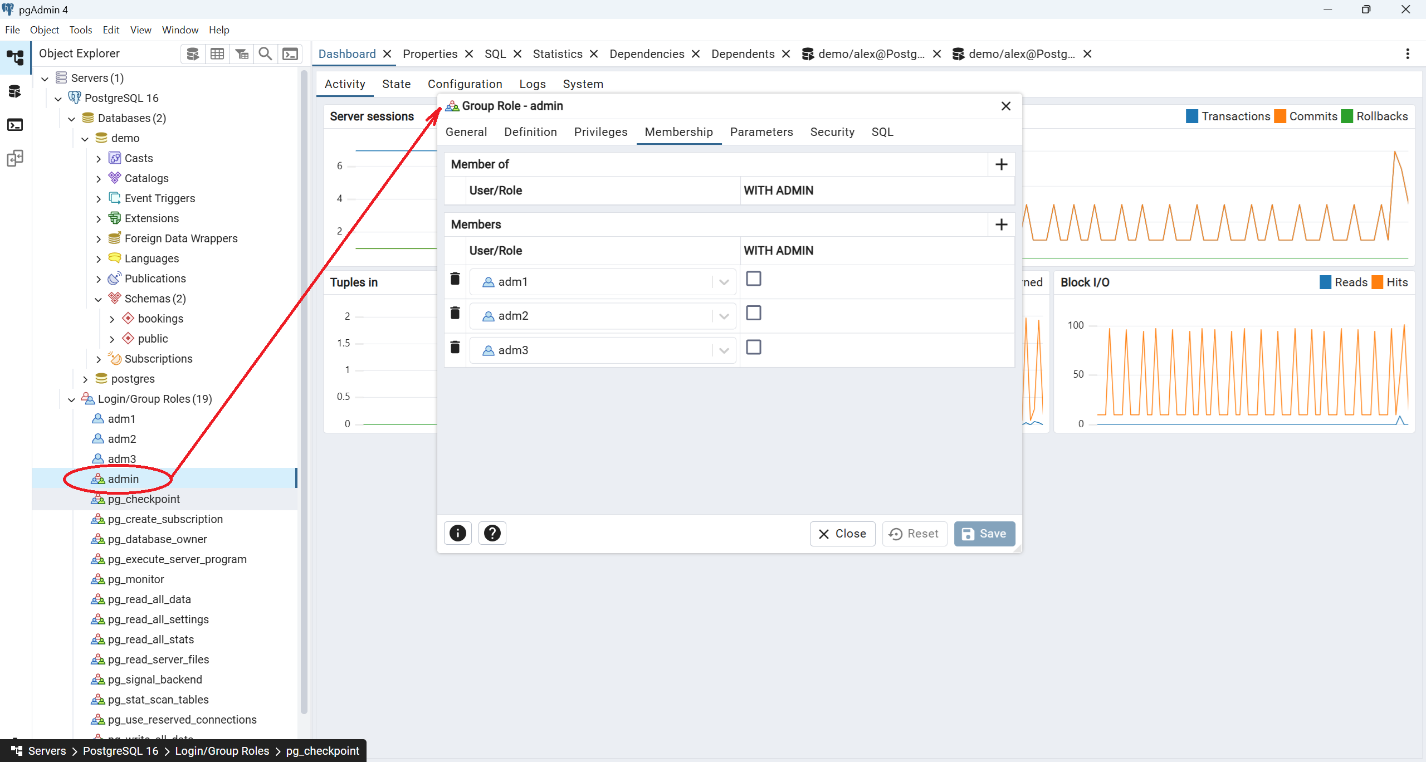


Рисунок 1.1.18

Результат ожидаемый.

* 1. Безопасность на уровне базы данных

Заведем пользователей, которые могут подсоединиться к БД demo и читать данные таблиц, т. е. выполнять SELECT. Начнем с подключения к БД. Для баз данных команда GRANT позволяет задать следующие права доступа:

GRANT {{ CREATE | CONNECT | TEMPORARY | TEMP } [, ...] | ALL [PRIVILEGES]}

ON DATABASE *имя\_бд* [, ...]

TO *указание\_роли* [, ...] [ WITH GRANT OPTION ]

**Параметры**

CREATE – это право позволяет создавать схемы в конкретной базе. Отметим, чтоCREATE не разрешает создавать таблицы.

CONNECT [–](https://postgrespro.ru/docs/postgresql/current/ddl-priv#DDL-PRIV-TEMPORARY) позволяет подключаться к базе данных. Это право проверяется при   
установлении соединения.

TEMPORARY или TEMP [–](https://postgrespro.ru/docs/postgresql/current/ddl-priv#DDL-PRIV-TEMPORARY) позволяет создавать временные таблицы в определённой базе данных.

Обратная команда:

REVOKE [ GRANT OPTION FOR ]

{{ CREATE | CONNECT | TEMPORARY | TEMP } [, ...] | ALL [ PRIVILEGES ]}

ON DATABASE *имя\_бд* [, ...]

FROM *указание\_роли* [, ...]

[ CASCADE ]

**Параметры**

GRANT OPTION FOR – если указана эта опция, то отзывается только право передачи права, но не само право. Без этого указания отзывается и право, и право распоряжаться им.

CASCADE – если пользователь обладает правом с правом передачи и он дал его другим пользователям, последнее право считается зависимым. Когда первый пользователь лишается самого права или права передачи и существуют зависимые права, эти зависимые права также отзываются, если дополнительно указано CASCADE; в противном случае операция завершается ошибкой.

Приступим (рис. 1.2.1):

Пользователь postgres

demo=# CREATE ROLE readers;

CREATE ROLE

demo=# CREATE ROLE fio1 WITH IN ROLE readers LOGIN PASSWORD '123';

CREATE ROLE

demo=# \du

Список ролей

Имя роли | Атрибуты

----------+------------------------------------------------------------------------

adm1 | Суперпользователь

adm2 | Суперпользователь

adm3 | Суперпользователь, Создаёт роли, Создаёт БД, Пропускать RLS

admin | Суперпользователь, Вход запрещён

fio1 |

postgres | Суперпользователь, Создаёт роли, Создаёт БД, Репликация, Пропускать RLS  
 readers | Вход запрещён

Рисунок 1.2.1

Пользователь fio1 пока не имеет прав для подключения к БД demo, проверим. Зайдем с другого терминала и подключимся от имени пользователя fio1   
(рис. 1.2.2):

Пользователь fio1

Server [localhost]:

Database [postgres]: demo

Port [5432]:

Username [postgres]: fio1

Пароль пользователя fio1:

psql (16.8)

ПРЕДУПРЕЖДЕНИЕ: Кодовая страница консоли (866) отличается от основной

страницы Windows (1251).

8-битовые (русские) символы могут отображаться некорректно.

Подробнее об этом смотрите документацию psql, раздел

"Notes for Windows users".

Введите "help", чтобы получить справку.

demo=>

Рисунок 1.2.2

Подсоединились! В чем дело? А дело в том, что каждая БД имеет специальную групповую роль public, в которую автоматом включаются все роли и которая имеет привилегию CONNECT. Чтобы этого не происходило необходимо отозвать эту привилегию у public (рис. 1.2.3):

Пользователь postgres

demo=# REVOKE ALL ON DATABASE demo FROM public;

REVOKE

demo=#

Рисунок 1.2.3

Повторим попытку подключения (рис. 1.2.4):

Пользователь fio1

Server [localhost]:

Database [postgres]: demo

Port [5432]:

Username [postgres]: fio1

Пароль пользователя fio1:

psql: ошибка: подключиться к серверу "localhost" (::1), порту 5432 не удалось: ВАЖНО: доступ к базе "demo" запрещён

ПОДРОБНОСТИ: Пользователь не имеет привилегии CONNECT.

Для продолжения нажмите любую клавишу . . .

Рисунок 1.2.4

Предоставим право подключения роли readers к БД demo (рис. 1.2.5):

Пользователь postgres

demo=# GRANT CONNECT ON DATABASE demo TO readers;

GRANT

demo=#

Рисунок 1.2.5

Войдем с правами fio1 и проведем пару тестов (рис. 1.2.6):

Пользователь fio1

Server [localhost]:

Database [postgres]: demo

Port [5432]:

Username [postgres]: fio1

Пароль пользователя fio1:

psql (16.8)

ПРЕДУПРЕЖДЕНИЕ: Кодовая страница консоли (866) отличается от основной

страницы Windows (1251).

8-битовые (русские) символы могут отображаться некорректно.

Подробнее об этом смотрите документацию psql, раздел

"Notes for Windows users".

Введите "help", чтобы получить справку.

demo=> \! chcp 1251

Текущая кодовая страница: 1251

demo=> SELECT \* FROM bookings.aircrafts\_data;

ОШИБКА: нет доступа к схеме bookings

СТРОКА 1: SELECT \* FROM bookings.aircrafts\_data;

^

demo=> CREATE SCHEMA rrr;

ОШИБКА: нет доступа к базе данных demo

demo=> CREATE table test (id int);

ОШИБКА: нет доступа к схеме public

СТРОКА 1: CREATE table test (id int);

^

demo=>

Рисунок 1.2.6

Пользователь fio1 еще не имеет возможности получить данные – нет доступа к схеме bookings.

Перед тем, как приступить к предоставлению привилегий доступа к схеме, рассмотрим, как выполнить предыдущие операции с помощью pgAdmin.

Откроем pgAdmin с правами суперпользователя и создадим групповую роль writer\_aircrafts (без привилегии LOGIN), содержащая пользователя foi2 (рис. 1.2.7).

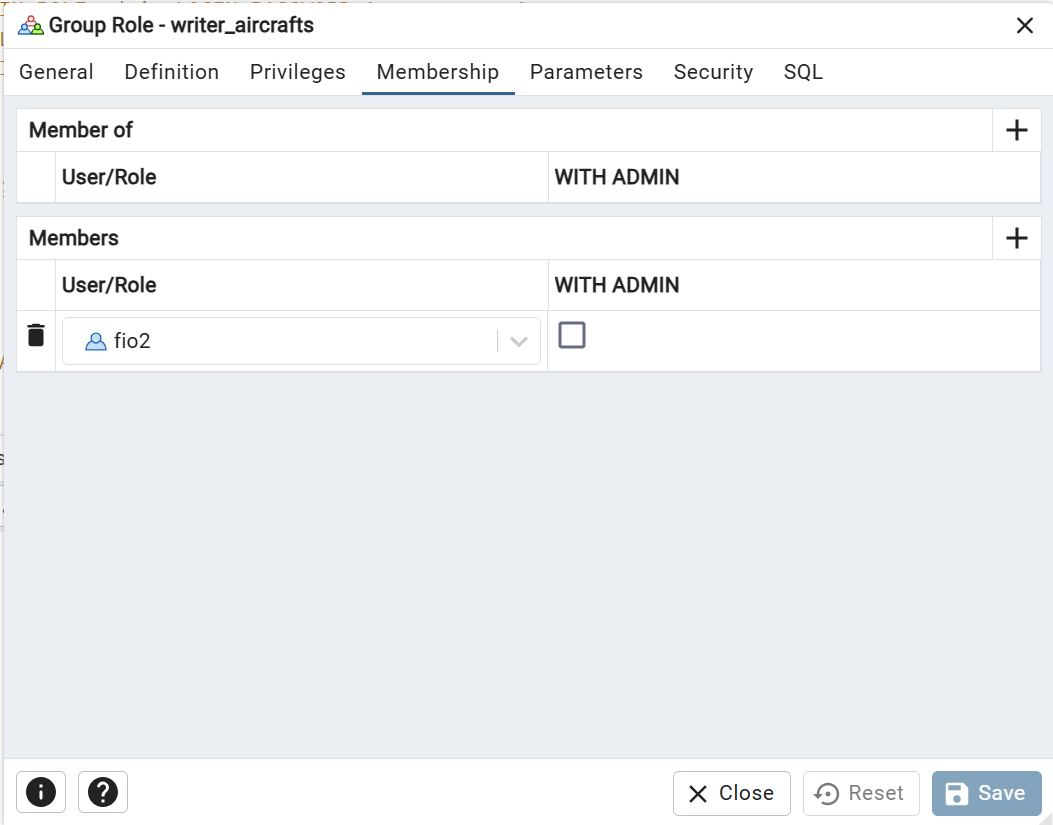


Рисунок 1.2.7

Групповую роль writer\_aircrafts наделим привилегиями, позволяющими редактировать записи таблицы aircrafts\_data. Но сначала необходимо предоставить возможность подключиться к БД demo.

Щелкнем ПКМ по узлу demo на панели объектов и выберем меню *Properties…*. Откроется диалог, в котором выберем вкладку Security   
(рис. 1.2.8):

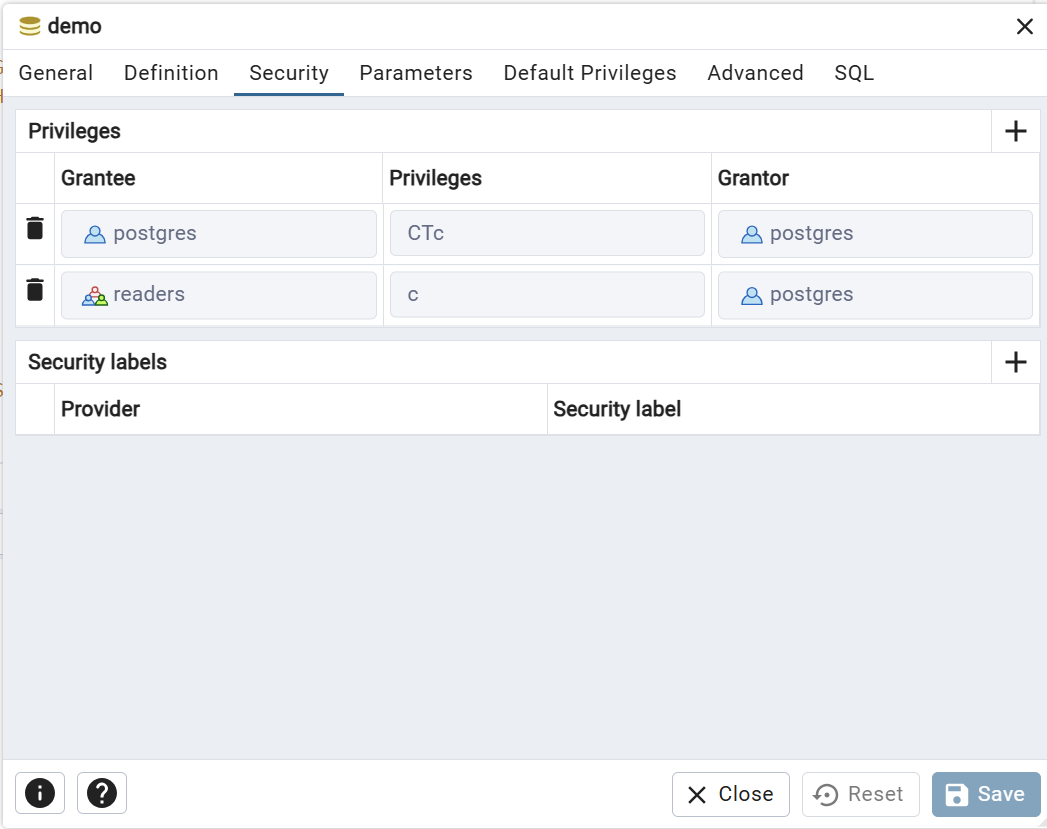


Рисунок 1.2.8

Нажмем на «+», чтобы добавить роль writer\_aircrafts и наделим ее привилегией CONNECT (рис. 1.2.9):

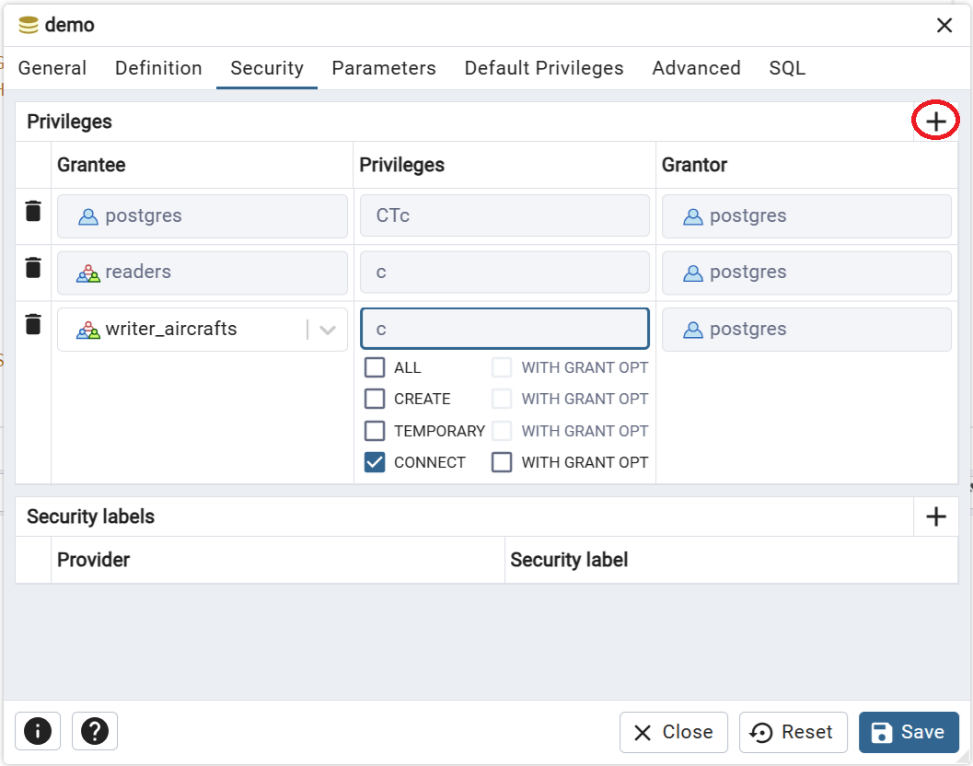


Рисунок 1.2.9

Убедимся, что мы сделали все правильно, открыв вкладку SQL  
 (рис. 1.2.10):

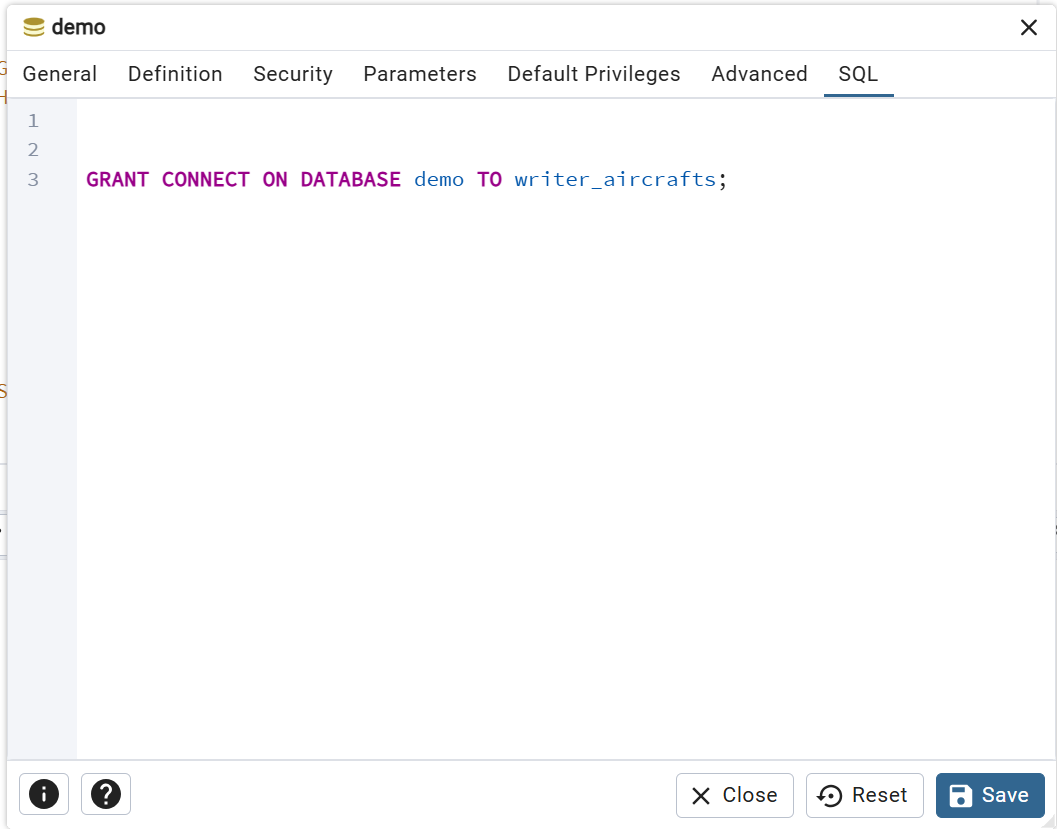


Рисунок 1.2.10

Сохраним. Проверим, что пользователь может подключиться к demo, проще всего это сделать, запустив утилиту psql (рис. 1.2.11):

Server [localhost]:

Database [postgres]: demo

Port [5432]:

Username [postgres]: fio2

Пароль пользователя fio2:

psql (16.8)

ПРЕДУПРЕЖДЕНИЕ: Кодовая страница консоли (866) отличается от основной

страницы Windows (1251).

8-битовые (русские) символы могут отображаться некорректно.

Подробнее об этом смотрите документацию psql, раздел

"Notes for Windows users".

Введите "help", чтобы получить справку.

demo=>

Рисунок 1.2.11

* 1. Задание прав на уровне схемы

Рассмотрим команду предоставление прав на схему:

GRANT { { CREATE | USAGE } [, ...] | ALL [ PRIVILEGES ] }

ON SCHEMA *имя\_схемы* [, ...]

TO *указание\_роли* [, ...] [ WITH GRANT OPTION ]

**Параметры**

CREATE – означает, что пользователь может создавать в схеме объекты.

USAGE –означает, что пользователю разрешено войти в схему.

От имени суперпользователя дадим привилегию USAGE (рис. 1.3.1):

Пользователь postgres

demo=# GRANT USAGE ON SCHEMA bookings TO readers;

GRANT

Рисунок 1.3.1

Но это еще не значит, что пользователь сможет что-то из находящего в этой схеме использовать; соответствующие права еще не предоставлены. Пока что пользователю разрешено только просматривать эту схему в системном каталоге (рис. 1.3.2):

Пользователь fio1

demo=> SELECT \* FROM bookings.aircrafts\_data;

ОШИБКА: нет доступа к таблице aircrafts\_data

demo=> \dt

Список отношений

Схема | Имя | Тип | Владелец

----------+-----------------+---------+----------

bookings | aircrafts\_data | таблица | postgres

bookings | airports\_data | таблица | postgres

bookings | boarding\_passes | таблица | postgres

bookings | bookings | таблица | postgres

bookings | flights | таблица | postgres

bookings | seats | таблица | postgres

bookings | ticket\_flights | таблица | postgres

bookings | tickets | таблица | postgres

(8 строк)

Рисунок 1.3.2

Поскольку владелец таблиц postgres, нам необходимо выдать привилегии SELECT на таблицы для роли readers.

Проделаем тоже самое и для роли writer\_aircrafts, но используя pgAdmin. Последовательность действий аналогична, описанной выше для БД demo. Нажимаем ПКМ на узле bookings и выбираем из меню *Properties…*. Выбираем во вкладку Security и добавляем роль с привилегией USAGE   
(рис. 1.3.3):

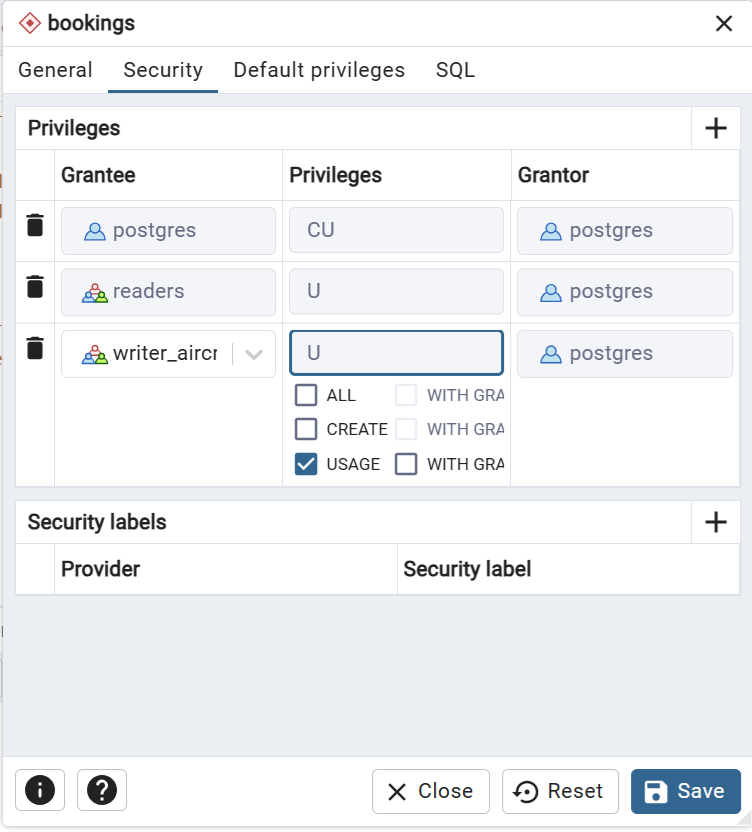


Рисунок 1.3.3

Сохраним. Теперь необходимо выдать привилегии для работы с таблицами.

* 1. Работа с таблицами

Для предоставления прав на таблицы надо использовать команду все той же командой GRANT:

GRANT {{SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIGGER}

[, ...] | ALL [ PRIVILEGES ] }

ON { [ TABLE ] *имя\_таблицы* [, ...]

| ALL TABLES IN SCHEMA *имя\_схемы* [, ...] }

TO *указание\_роли* [, ...] [ WITH GRANT OPTION ]

Параметры

SELECT – позволяет выполнять SELECT для любого столбца или перечисленных столбцов в заданной таблице, представлении, или другом объекте табличного вида. Также позволяет выполнять COPY TO. Помимо этого, данное право требуется для обращения к существующим значениям столбцов в UPDATE, DELETE или MERGE.

INSERT – позволяет вставлять с помощью INSERT строки в заданную таблицу, представление и т. п. Также позволяет выполнять COPY FROM.

UPDATE – позволяет изменять с помощью UPDATE данные во всех, либо только перечисленных, столбцах в заданной таблице, представлении и т. п. (На практике для любой нетривиальной команды UPDATE потребуется и право SELECT, так как она должна обратиться к столбцам таблицы, чтобы определить, какие строки подлежат изменению, и/или вычислить новые значения столбцов.)

DELETE – позволяет удалять с помощью команды DELETE строки из таблицы, представления и т. п. (На практике для любой нетривиальной команды DELETE потребуется также право SELECT, так как она должна обратиться к колонкам таблицы, чтобы определить, какие строки подлежат удалению.)

TRUNCATE – позволяет опустошать таблицу с помощью TRUNCATE.

REFERENCES – позволяет создавать ограничение внешнего ключа, обращающееся к таблице или определённым столбцам таблицы.

TRIGGER – позволяет создавать триггер для таблицы, представления и т. п.

Для группы readers мы выдадим пава на просмотр всех таблиц в схеме bookings (рис. 1.4.1):

Пользователь postgres

demo=# GRANT SELECT ON ALL TABLES IN SCHEMA bookings TO readers;

GRANT

demo=#

Рисунок 1.4.1

Выполним SELECT с правами пользователя fio1 (рис. 1.4.2):

Пользователь fio1

demo=> SELECT \* FROM bookings.aircrafts\_data;

aircraft\_code | model | range

---------------+------------------------------------------------------------+-------

773 | {"en": "Boeing 777-300", "ru": "Боинг 777-300"} | 11100

763 | {"en": "Boeing 767-300", "ru": "Боинг 767-300"} | 7900

SU9 | {"en": "Sukhoi Superjet-100", "ru": "Сухой Суперджет-100"} | 3000

320 | {"en": "Airbus A320-200", "ru": "Аэробус A320-200"} | 5700

321 | {"en": "Airbus A321-200", "ru": "Аэробус A321-200"} | 5600

319 | {"en": "Airbus A319-100", "ru": "Аэробус A319-100"} | 6700

733 | {"en": "Boeing 737-300", "ru": "Боинг 737-300"} | 4200

CN1 | {"en": "Cessna 208 Caravan", "ru": "Сессна 208 Караван"} | 1200

CR2 | {"en": "Bombardier CRJ-200", "ru": "Бомбардье CRJ-200"} | 2700

(9 строк)

demo=>

Рисунок 1.4.2

Аналогичным образом, можно создать групповые роли с правами, позволяющие модифицировать данные в таблицах. Предоставим привилегии на изменения таблицы aircrafts\_data для роли writer\_aircrafts. Вызовем команду контекстного меню *Properties…* для таблицы aircrafts\_data и во вкладке *Security* добавим роль, предоставив привилегии только на добавление и изменение записей (рис. 1.4.3):

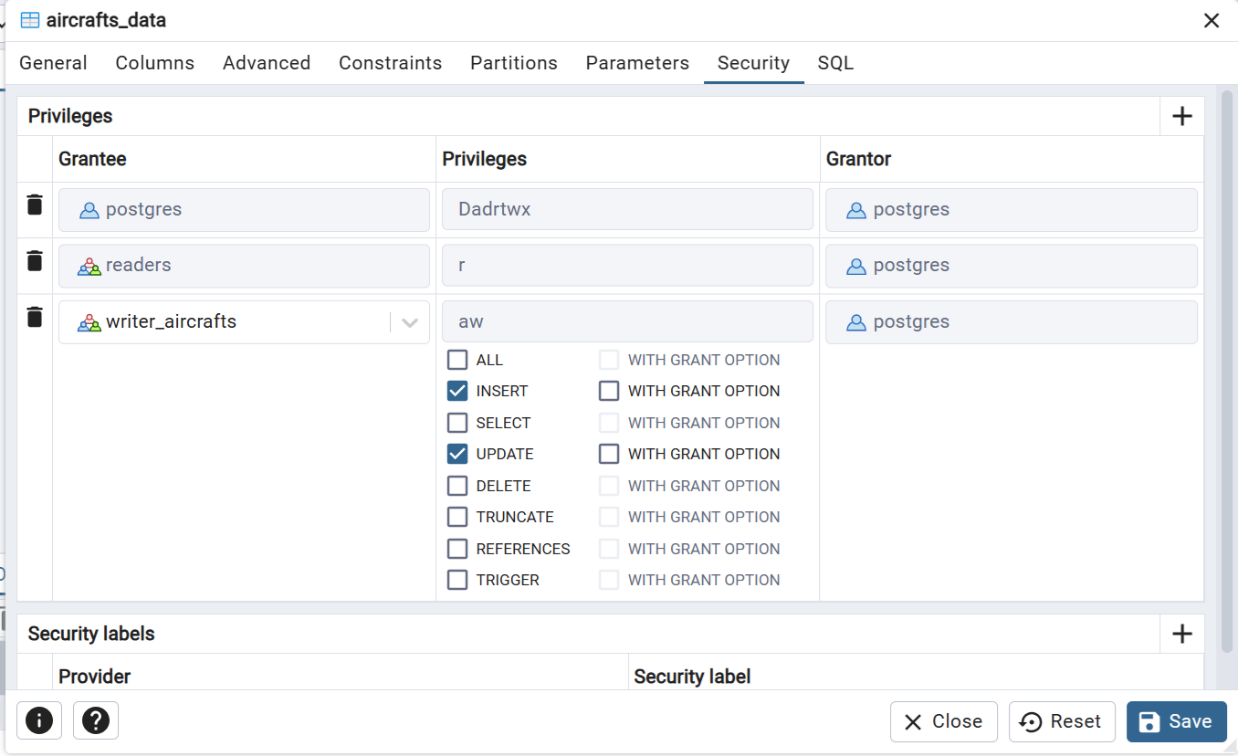


Рисунок 1.4.3

Войдем во вкладку и полюбопытствуем, какой скрипт нам приготовили (рис. 1.4.4):

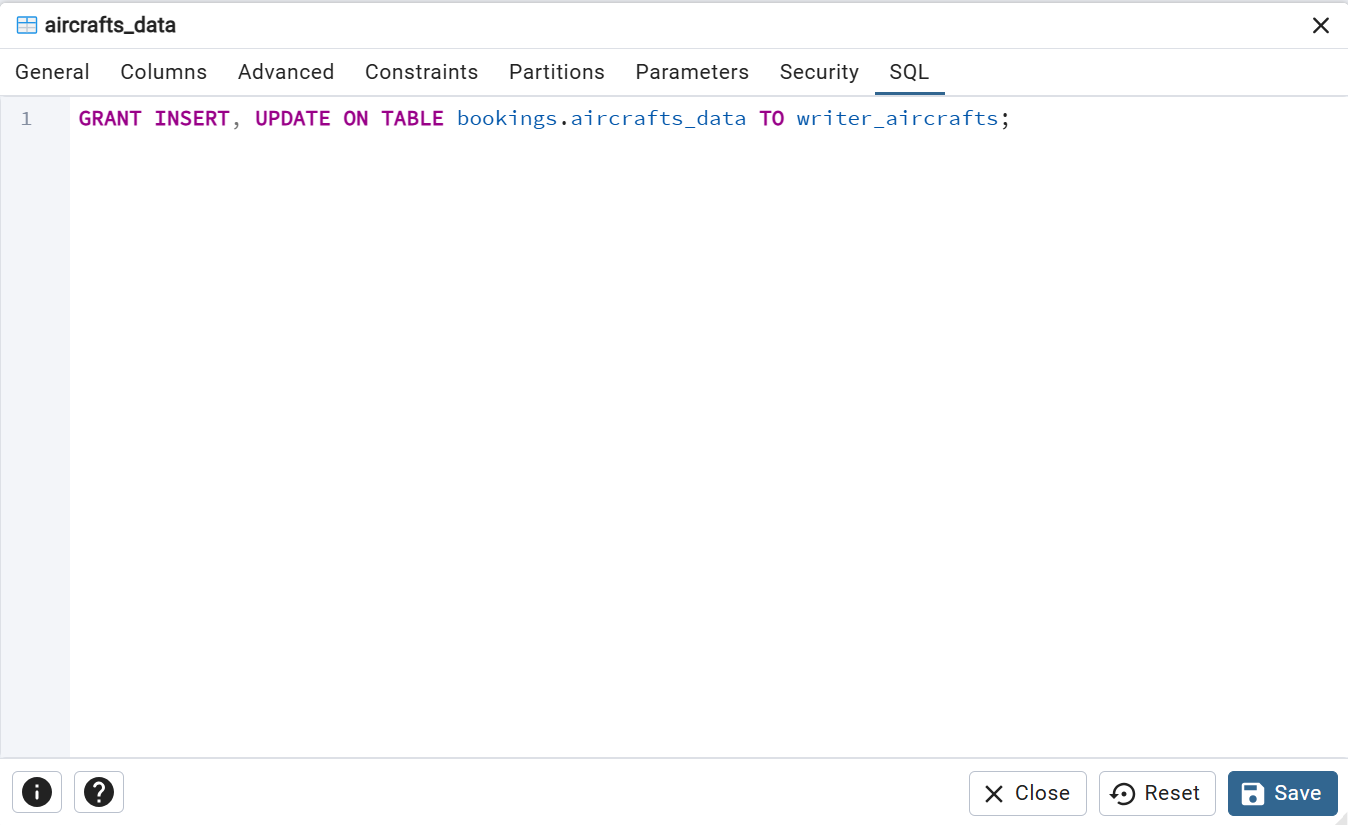


Рисунок 1.4.4

Теперь можем сохранить. Откроем psql или pgAdmin, как пользователь fio2 и добавим строку (рис. 14.5):

Server [localhost]:

Database [postgres]: demo

Port [5432]:

Username [postgres]: fio2

Пароль пользователя fio2:

psql (16.8)

ПРЕДУПРЕЖДЕНИЕ: Кодовая страница консоли (866) отличается от основной

страницы Windows (1251).

8-битовые (русские) символы могут отображаться некорректно.

Подробнее об этом смотрите документацию psql, раздел

"Notes for Windows users".

Введите "help", чтобы получить справку.

demo=> \! chcp 1251

Текущая кодовая страница: 1251

^

demo=> INSERT INTO bookings.aircrafts\_data ( aircraft\_code, model, range)   
VALUES('WWW', '{"en": "Sukhoi Superjet-200", "ru": "Сухой Суперджет-200" }', 7000);

INSERT 0 1

demo=>

Рисунок 1.4.5

Строка добавлена. При попытки удалить любую строку из таблицы aircrafts\_data мы получим сообщение об отсутствия доступа.

Теперь посмотрим на содержание таблицы и следующей строке наберем SELECT (рис. 1.4.6):

Пользователь fio2

demo=> SELECT \* FROM bookings.aircrafts\_data;

ОШИБКА: нет доступа к таблице aircrafts\_data

demo=>

Рисунок 1.4.6

Результат ожидаемый, нет прав. Вместо того чтобы предоставлять привилегии на SELECT для роли writer\_aircrafts, включим ее в роль readers, тем самым разрешим выполнять выборку из всех таблиц, т.к. привилегия SELECT наследуется (рис. 1.4.7):

Пользователь postgres

demo=# ALTER GROUP readers add USER writer\_aircrafts;

ALTER ROLE

demo=#

Рисунок 1.4.7

Повторим SELECT (рис. 1.4.8):

Пользователь fio2

demo=> SELECT \* FROM bookings.aircrafts\_data;

aircraft\_code | model | range

---------------+------------------------------------------------------------+------

773 | {"en": "Boeing 777-300", "ru": "Боинг 777-300"} | 11100

763 | {"en": "Boeing 767-300", "ru": "Боинг 767-300"} | 7900

SU9 | {"en": "Sukhoi Superjet-100", "ru": "Сухой Суперджет-100"} | 3000

320 | {"en": "Airbus A320-200", "ru": "Аэробус A320-200"} | 5700

321 | {"en": "Airbus A321-200", "ru": "Аэробус A321-200"} | 5600

319 | {"en": "Airbus A319-100", "ru": "Аэробус A319-100"} | 6700

733 | {"en": "Boeing 737-300", "ru": "Боинг 737-300"} | 4200

CN1 | {"en": "Cessna 208 Caravan", "ru": "Сессна 208 Караван"} | 1200

CR2 | {"en": "Bombardier CRJ-200", "ru": "Бомбардье CRJ-200"} | 2700

WWW | {"en": "Sukhoi Superjet-200", "ru": "Сухой Суперджет-200"} | 7000

(10 строк)

demo=>

Рисунок 1.4.8

Последний пример является демонстрацией удобства использования групповых ролей.

* 1. Примечание

Мы не рассмотрели возможность в pgAdmin выдавать привилегии ролям сразу для нескольких объектов БД. Это можно сделать, выбрав из контекстного меню для БД или схемы команду *Grant Wizard…*. Разобраться, как это работает, в силу изложенного выше материала, не составит труда и оставляется для самостоятельной проработки.

## Практическое задание

Создать групповые роли: admin, reader и две роли writer. Каждая роль должна содержать по пользователю.

Роль члены роли admin должны обладать привилегиями суперпользователя. От имени суперпользователя из роли admin создать БД, схему и две таблицы.

Первая таблица (таблица 1) должна содержать id с типом serial. Вторая таблица (таблица 2) должна содержать id с типом identity.

Групповая роль reader должна обладать привилегиями чтения данных из таблиц БД.

Первая групповая роль writer1 должна иметь привилегии на чтение и изменения таблицы 1.

Вторая групповая роль writer2 должна иметь привилегии на чтение и изменения таблицы 2.

От имени пользователей из ролей writer добавьте, измените и удалите данные в соответствующих таблицах.

Привести скрипты для каждой операции по созданию ролей и данных. Объяснить принципиальную разницу в типах данных serial и identity.

## Список использованных источников

1. Фомин М.М. Реляционные базы данных. Учебное пособие для бакалавров <https://e-learning.bmstu.ru/iu6/mod/resource/view.php?id=6634>
2. PostgreSQL 11. Мастерство разработки. Ганс-Юрген Шениг / пер. с анг. А. А. Слинкина. – М.: ДМК Пресс, 2019.
3. Документация PostgreSQL https://postgrespro.ru/docs

1. Версия 9.6 [↑](#footnote-ref-1)
2. ПКМ – сокращение «правая клавиша мыши». [↑](#footnote-ref-2)