1. Подготовка окружения

1.1. Подключение к базе данных в pgAdmin 4

Откройте pgAdmin 4.

Подключитесь к нужному серверу.

Разверните дерево до нужной базы данных (Database) и откройте в ней Query Tool для выполнения SQL-команд.

1.2. Создание ролей (если они не существуют)

Если у вас пользователи (роли) alice и bob уже созданы, пропустите этот шаг. Если же их нет – создайте их (в примере пароли произвольные; вы можете задать любые другие):

sql

Copy

Edit

-- Создание ролей alice и bob, если нужно

CREATE ROLE alice LOGIN PASSWORD 'alicepass';

CREATE ROLE bob LOGIN PASSWORD 'bobpass';

Убедитесь, что роли действительно созданы и отображаются в списке ролей (Schemas -> Login/Group Roles).

2. Создание и заполнение таблиц

2.1. Таблица со связыванием логинов и отделов

Создадим таблицу, которая будет хранить сведения о том, в каком отделе (department) работает каждый пользователь (login):

sql

Copy

Edit

CREATE TABLE users\_depts (

login text,

department text

);

Заполним её демо-данными (для alice – отдел PR, для bob – отдел Sales):

sql

Copy

Edit

INSERT INTO users\_depts VALUES ('alice', 'PR'), ('bob', 'Sales');

2.2. Таблица с данными о доходах и расходах

Создадим таблицу revenue, которая хранит информацию о суммах (amount) по разным отделам (department). Для наглядности добавим демо-данные.

sql

Copy

Edit

CREATE TABLE revenue (

department text,

amount numeric(10,2)

);

Вставим несколько записей (в примере – случайные данные, но можно добавить и статические):

sql

Copy

Edit

-- Для отдела PR - расход (отрицательные суммы)

INSERT INTO revenue

SELECT 'PR', -random() \* 100.00

FROM generate\_series(1, 100);

-- Для отдела Sales - доход (положительные суммы)

INSERT INTO revenue

SELECT 'Sales', random() \* 1000.00

FROM generate\_series(1, 100);

(Функция random() в PostgreSQL возвращает значение от 0 до 1, поэтому умножаем на какую-то базовую величину для распределения.)

Проверим, что данные вставились (как суперпользователь/владелец таблиц):

sql

Copy

Edit

SELECT department, amount

FROM revenue

LIMIT 10;

3. Настройка механизма Row Level Security (RLS)

3.1. Включение RLS для таблицы

Чтобы политики (политики доступа к строкам) начали работать, нужно включить RLS для таблицы:

sql

Copy

Edit

ALTER TABLE revenue ENABLE ROW LEVEL SECURITY;

Важно: если RLS включён, но не создано никаких политик, доступ к строкам будет запрещён (по умолчанию все операции блокируются, пока политики не заданы).

3.2. Создание политики для SELECT

Создадим политику, которая позволит выбирать строки только из «своего» отдела.

Рассмотрим основную логику предиката:

sql

Copy

Edit

department = (SELECT department

FROM users\_depts

WHERE login = current\_user)

Таким образом, у пользователя будет видна только та строка в revenue, у которой значение department совпадает с его отделом из таблицы users\_depts.

sql

Copy

Edit

CREATE POLICY rls\_revenue\_select

ON revenue

FOR SELECT

TO PUBLIC

USING (

department = (

SELECT department

FROM users\_depts

WHERE login = current\_user

)

);

FOR SELECT – политика применяется к операциям чтения (SELECT).

TO PUBLIC – политика распространяется сразу на все роли (включая alice, bob, и др.). Можно вместо PUBLIC явно перечислить роли alice, bob, если нужно более точечно.

3.3. Создание политики для UPDATE и DELETE

Чтобы пользователи могли модифицировать только свои строки (обновлять/удалять данные для своего отдела), создадим политику с FOR UPDATE и FOR DELETE.

Для UPDATE:

USING (...) описывает условие, при котором строка «допущена» к обновлению (т.е. пользователь её «видит» для изменения).

WITH CHECK (...) описывает условие, при котором система разрешает совершить UPDATE (т.е. «новые» данные после изменения тоже должны оставаться в рамках дозволенного).

Проще всего (наглядно) написать так:

sql

Copy

Edit

CREATE POLICY rls\_revenue\_update

ON revenue

FOR UPDATE

TO PUBLIC

USING (

department = (

SELECT department

FROM users\_depts

WHERE login = current\_user

)

)

WITH CHECK (

department = (

SELECT department

FROM users\_depts

WHERE login = current\_user

)

);

Для DELETE:

sql

Copy

Edit

CREATE POLICY rls\_revenue\_delete

ON revenue

FOR DELETE

TO PUBLIC

USING (

department = (

SELECT department

FROM users\_depts

WHERE login = current\_user

)

);

Если нужно разграничить и INSERT, то добавляем аналогичную политику с FOR INSERT и WITH CHECK, чтобы при вставке строк пользователь мог вставлять только свой отдел.

4. Выдача привилегий на таблицы

Помимо RLS, нужно не забыть выдать обычные привилегии на операции, которые пользователь может делать (SELECT, INSERT, UPDATE, DELETE). Если не выдать эти права, даже при наличии RLS-политик пользователь не сможет выполнять соответствующие запросы.

sql

Copy

Edit

-- Даем Alice и Bob права на SELECT, UPDATE, DELETE над таблицей revenue

GRANT SELECT, UPDATE, DELETE ON revenue TO alice, bob;

-- Аналогично, если нужно, выдаем права на вставку

-- GRANT INSERT ON revenue TO alice, bob;

-- Чтобы видеть данные из вспомогательной таблицы users\_depts,

-- также выдать как минимум право SELECT (если нужно):

GRANT SELECT ON users\_depts TO alice, bob;

5. Проверка результата

5.1. Проверка от лица суперпользователя / владельца

Суперпользователь (или владелец схемы) по умолчанию видит все строки (на суперпользователя RLS не действует):

sql

Copy

Edit

SELECT department, amount

FROM revenue;

-- Будут видны и department = 'PR', и department = 'Sales', все 200 строк (по 100 на отдел)

5.2. Проверка под alice

В pgAdmin 4 можно открыть новое подключение к базе от имени alice (либо сменить роль командой SET ROLE alice; в Query Tool, если права позволяют), и выполнить:

sql

Copy

Edit

SELECT current\_user;

-- Должно показать "alice"

SELECT department, amount

FROM revenue

LIMIT 10;

Ожидаем, что alice увидит только строки, где department = 'PR'.

Если попробовать UPDATE или DELETE для строки, где department не совпадает с alice-подразделением, то должна прийти ошибка или команда просто «ничего не затронет» (если условие USING не выполняется — строка для alice «невидима»).

5.3. Проверка под bob

Аналогично для bob:

sql

Copy

Edit

SELECT current\_user;

-- Должно показать "bob"

SELECT department, amount

FROM revenue

LIMIT 10;

bob увидит только строки, где department = 'Sales'.

6. Дополнительные замечания

Учет NULL: Если таблица users\_depts не содержит нужного login, то предикат department = (SELECT department ... ) вернёт department = NULL, что эквивалентно false при сравнении. Тогда пользователь не увидит ни одной строки.

Суперпользователь или роль с атрибутом BYPASSRLS не подчиняются этим политикам по умолчанию и всегда видят все данные.

Если требуется «жёстко» ограничить и владельца таблицы, можно использовать опцию ALTER TABLE ... FORCE ROW LEVEL SECURITY;, но обычно это не нужно.

Сочетание нескольких политик: если политик несколько, разрешительные (PERMISSIVE) складываются по логическому ИЛИ, а ограничительные (RESTRICTIVE) – по логическому И (“и при этом”), что позволяет гибко комбинировать условия.

Не забудьте, что политики действуют только после включения ENABLE ROW LEVEL SECURITY для соответствующей таблицы (или FORCE ROW LEVEL SECURITY, если нужно принудительно применять и к владельцу).

Таким образом, с помощью вышеописанных шагов, в pgAdmin 4 вы получите раздельный доступ на уровне строк (RLS) для разных пользователей: каждый будет видеть (и при наличии соответствующих политик – изменять) только те строки, которые соответствуют его отделу в таблице users\_depts.