Савицкий Вадим Сергеевич
бакалавр, студент
Гурский Станислав Михайлович
бакалавр, студент
Жаркова Оксана Михайловна
канд. физ.-мат. наук, доцент
ФГБОУ ВО «Кубанский государственный университет»
г. Краснодар, Краснодарский край

ИСПОЛЬЗОВАНИЕ МЕХАНИЗМА ROW LEVEL SECURITY В POSTGRESQL ПРИ ПОСТРОЕНИИ ОБРАЗОВАТЕЛЬНЫХ ПЛАТФОРМ

Аннотация: *в статье рассматриваются особенности применения механизма Row Level Security (RLS) – политики безопасности на уровне строк в системе управления базами данных PostgreSQL при разработке многопользовательских образовательных платформ. Проведён анализ ограничений классического подхода к разграничению доступа на уровне приложения и обоснована необходимость переноса логики безопасности на уровень СУБД. Описаны базовые принципы настройки RLS: от включения политик на таблице до создания дифференцированных правил для ролей студента, педагога и администратора. Приведены практические примеры SQL-запросов и схема интеграции RLS в архитектуру образовательной платформы. Продемонстрировано, что применение данного механизма позволяет обеспечить надёжную защиту персональных данных обучающихся и соответствие требованиям Федерального закона №152-ФЗ.*

Ключевые слова: *Row Level Security, RLS, PostgreSQL, образовательная платформа, разграничение доступа, персональные данные, информационная безопасность, цифровая образовательная среда, политики безопасности, многопользовательская система, цифровизация образования, защита данных, LMS, СУБД, роли пользователей.*

Введение.

Цифровая трансформация образования сопровождается стремительным ростом числа платформ дистанционного обучения, электронных журналов и систем управления учебным контентом. Такие системы аккумулируют значительные объёмы персональных данных: успеваемость обучающихся, персональные сведения педагогов, учебные материалы с ограниченным доступом. Перед разработчиком неизбежно встаёт вопрос по разграничению доступа для различных ролей.

Традиционный подход предполагает реализацию проверок прав доступа в коде приложения. Однако такой подход обладает принципиальным недостатком: любой прямой запрос к базе данных в обход приложения – через инструмент администрирования, скрипт или при эксплуатации уязвимости – полностью снимает все ограничения. Помимо этого, логика безопасности оказывается рассеяна по всему коду, что увеличивает вероятность ошибки разработчика [1].

Альтернативой в данном случае выступает механизм Row Level Security (RLS) – встроенная возможность СУБД PostgreSQL, позволяющая определять политики доступа непосредственно на уровне таблиц базы данных. При включённом RLS СУБД автоматически фильтрует строки при каждом запросе, руководствуясь заданными правилами, – независимо от того, каким способом выполняется этот запрос [1].

Цель данной статьи – объяснить принципы работы механизма RLS, показать базовые сценарии его настройки применительно к образовательной платформе, а также разобрать практические примеры SQL-кода.

Материал ориентирован на разработчиков образовательных платформ и специалистов по информационной безопасности образовательных организаций, желающих освоить надёжный способ защиты данных на уровне СУБД.

Архитектура доступа в образовательной платформе.

Для понимания принципов работы RLS необходимо разобраться в том, как организован доступ к данным в типовой образовательной платформе. В отличие от монолитных систем с единственной учётной записью базы данных, платформа

обслуживает несколько категорий пользователей с принципиально различными правами.

Ключевая особенность RLS состоит в том, что он работает прозрачно для приложения. Каждый SQL-запрос автоматически дополняется условием из политики, и пользователь физически не может получить строки, к которым у него нет доступа, даже выполняя запрос SELECT * FROM grades [1].

Типовая структура ролей образовательной платформы включает три уровня:

– студент – видит только собственные оценки и назначенные ему материалы;

– педагог – видит данные курсов, которые он ведёт;

– администратор – имеет неограниченный доступ ко всем данным.

Ключевым элементом интеграции является передача идентификатора и роли текущего пользователя в контекст сессии PostgreSQL. Приложение выполняет SET LOCAL перед каждым запросом, после чего все политики RLS применяются автоматически.

Схема взаимодействия компонентов платформы с применением RLS представлена на рисунке 1.

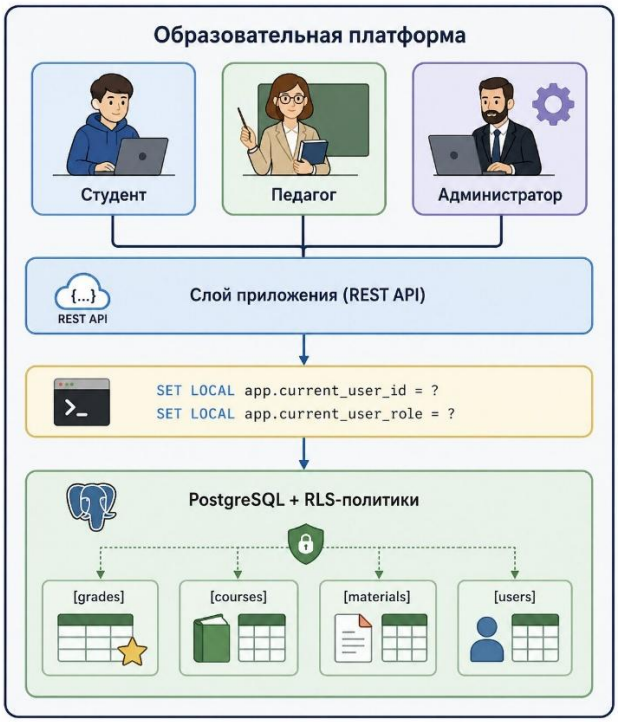

Рис. 1. Архитектура образовательной платформы с применением RLS

Настройка Row Level Security в PostgreSQL.

Процесс настройки RLS начинается с включения механизма на конкретной таблице с помощью команды ALTER TABLE. По умолчанию после включения RLS все строки становятся недоступны для пользователей, не являющихся владельцами таблицы, – пока не будет создана хотя бы одна политика [3].

Пример включения RLS на таблице оценок и создания базовой политики для студента представлен на рисунке 2.

```
-- Включение RLS на таблице с оценками
ALTER TABLE grades ENABLE ROW LEVEL SECURITY;

-- Политика: студент видит только свои оценки
CREATE POLICY student_see_own_grades
ON grades
FOR SELECT
USING (
  student_id = current_setting('app.current_user_id')::INTEGER
);
```

Рис. 2. Включение RLS и создание политики для роли студента

Параметр USING задаёт условие, которое PostgreSQL автоматически подставляет в каждый SELECT-запрос к таблице. Функция current_setting читает значение из контекста текущей сессии – того самого, которое приложение устанавливает через SET LOCAL [4].

Для роли педагога логика усложняется: необходимо выбирать оценки только по тем курсам, которые ведёт данный педагог. Пример соответствующей политики представлен на рисунке 3.

```
-- Политика: педагог видит оценки только своих курсов
CREATE POLICY teacher_see_course_grades
ON grades
FOR SELECT
USING (
  course_id IN (
    SELECT course_id
    FROM courses
    WHERE teacher_id =
      current_setting('app.current_user_id')::INTEGER
  )
);
```

Рис. 3. Политика RLS для роли педагога

Администратор, как правило, освобождается от действия политик через специальную роль PostgreSQL с атрибутом BYPASSRLS, либо для него создаётся отдельная политика, возвращающая true для всех строк [5].

Типовой сценарий применения RLS в образовательной платформе выглядит следующим образом:

– аутентификация пользователя – приложение проверяет логин и пароль, определяет идентификатор и роль;

– установка контекста сессии – перед выполнением запросов приложение передаёт данные пользователя в PostgreSQL через SET LOCAL;

– автоматическая фильтрация – СУБД применяет политики RLS ко всем запросам в рамках сессии;

– возврат результата – пользователь получает только те данные, к которым у него есть доступ согласно политикам [4].

Защита журнала успеваемости.

Рассмотрим полный практический пример для таблицы grades, содержащей записи об успеваемости. Пример создания таблицы, включения RLS и определения всех трёх политик представлен на рисунке 4.

```sql
-- Создание таблицы оценок
CREATE TABLE grades (
  id          SERIAL PRIMARY KEY,
  student_id  INTEGER NOT NULL,
  course_id   INTEGER NOT NULL,
  grade       NUMERIC(4,2),
  graded_at   TIMESTAMP DEFAULT NOW()
);

-- Включение RLS
ALTER TABLE grades ENABLE ROW LEVEL SECURITY;

-- Студент видит только свои записи
CREATE POLICY policy_student
ON grades FOR SELECT
USING (
  current_setting('app.role') = 'student'
  AND student_id =
    current_setting('app.user_id')::INTEGER
);

-- Педагог видит записи по своим курсам
CREATE POLICY policy_teacher
ON grades FOR SELECT
USING (
  current_setting('app.role') = 'teacher'
  AND course_id IN (
    SELECT id FROM courses
    WHERE teacher_id =
      current_setting('app.user_id')::INTEGER
  )
);

-- Администратор видит все записи
CREATE POLICY policy_admin
ON grades FOR SELECT
USING (
  current_setting('app.role') = 'admin'
);

```

Рис. 4. Полный пример настройки RLS для таблицы оценок

На стороне приложения перед выполнением любого запроса к защищённым таблицам необходимо установить контекст сессии. Пример передачи контекста из серверного кода представлен на рисунке 5.

```sql
-- Выполняется приложением перед каждым запросом
BEGIN;
SET LOCAL app.user_id = '42';
SET LOCAL app.role    = 'student';

-- Этот запрос вернёт только оценки студента с id = 42
SELECT * FROM grades;

COMMIT;
```

Рис. 5. Установка контекста сессии и выполнение защищённого запроса

Заключение.

Был проведён анализ механизма Row Level Security как инструмента обеспечения информационной безопасности образовательных платформ. Понимание принципов работы RLS – автоматическая подстановка условий фильтрации на уровне СУБД – является основой для построения надёжной многоуровневой защиты данных. На практических примерах продемонстрирована настройка политик для типовых ролей образовательного процесса: от простой фильтрации по идентификатору студента до связанных подзапросов для педагога. Показана схема интеграции RLS в архитектуру приложения через механизм контекста сессии. Применение данного подхода позволяет обеспечить соответствие требованиям законодательства о персональных данных и повысить устойчивость системы к угрозам несанкционированного доступа.

References

1. PostgreSQL Documentation. Row Security Policies / The PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/current/ddl-rowsecurity.html (date of request: 03.05.2026).

2. PostgreSQL Documentation. CREATE POLICY / The PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/current/sql-createpolicy.html (date of request: 03.05.2026).

3. PostgreSQL Documentation. ALTER TABLE / The PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/current/sql-altertable.html (date of request: 03.05.2026).

4. PostgreSQL Documentation. System Administration Functions: current_setting / The PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/current/functions-admin.html (date of request: 03.05.2026).

5. PostgreSQL Documentation. Database Roles / The PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/current/user-manag.html (date of request: 03.05.2026).