**Министерство науки и высшего образования Российской Федерации**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

# ОТЧЕТ

## по Лабораторной работе № 5

«процедуры, функции, триггеры в PostgreSQL»

**по дисциплине «Проектирование и реализация баз данных»**

**Обучающийся** Соболев Артём

**Факультет** прикладной информатики

**Группа** K3240

**Направление подготовки** 09.03.03 Прикладная информатика

**Образовательная программа** Мобильные и сетевые технологии 2023

**Преподаватель** Говорова Марина Михайловна

Санкт-Петербург

2025

# ВВЕДЕНИЕ

**Цель работы:** овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

**Практическое задание:**

1. Создать 3 процедуры для индивидуальной БД согласно варианту  (часть 4 ЛР 2). Допустимо использование IN/OUT параметров. Допустимо создать авторские процедуры. (3 балла)
2. Создать триггеры для индивидуальной БД согласно варианту

## ВЫПОЛНЕНИЕ

### Процедуры

Создать хранимые процедуры:

- Для повышения стипендии отличникам на 10%.
- Для перевода студентов на следующий курс.
- Для изменения оценки при успешной пересдаче экзамена.

Процедура для повышения стипендии отличникам на 10%. Код процедуры представлен в листинге 1.

Листинг 1 – Процедура для повышения стипендии отличникам на 10%

```
CREATE OR REPLACE PROCEDURE increase_scholarship()
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE award_scholarship
    SET count = count * 1.10
    WHERE EXISTS (
        SELECT 1
        FROM student
        WHERE student.id = award_scholarship.student_id
          AND EXISTS (SELECT 1 FROM education
            JOIN curriculum ON education.group_id =
curriculum.education_group_id
            JOIN composition_curriculum ON
composition_curriculum.curriculum_id = curriculum.id
            WHERE education.student_id = student.id
          )
          AND NOT EXISTS (SELECT 1 FROM education
              JOIN curriculum ON education.group_id =
curriculum.education_group_id
              JOIN composition_curriculum ON
composition_curriculum.curriculum_id = curriculum.id
              LEFT JOIN attestation ON attestation.student_id =
student.id AND attestation.discipline_id =
composition_curriculum.discipline_id
              WHERE education.student_id = student.id AND
(attestation.id IS NULL OR attestation.grade <> 5)
```

```
            )
    );
END;
$$;
```

Создание процедуры и проверка его работоспособности были выполнены в консоли SQL Shell (psql), выполнение представлено на рисунках 1–3.

```
session_db=# CREATE OR REPLACE PROCEDURE increase_scholarship()
session_db-# LANGUAGE plpgsql
session_db-# AS $$
session_db$# BEGIN
session_db$#     UPDATE award_scholarship
session_db$#     SET count = count * 1.10
session_db$#     WHERE EXISTS (
session_db$#         SELECT 1
session_db$#         FROM student
session_db$#         WHERE student.id = award_scholarship.student_id
session_db$#           AND EXISTS (SELECT 1 FROM education
session_db$#             JOIN curriculum ON education.group_id = curriculum.education_group_id
session_db$#             JOIN composition_curriculum ON composition_curriculum.curriculum_id = curriculum.id
session_db$#             WHERE education.student_id = student.id
session_db$#           )
session_db$#           AND NOT EXISTS (SELECT 1 FROM education
session_db$#               JOIN curriculum ON education.group_id = curriculum.education_group_id
session_db$#               JOIN composition_curriculum ON composition_curriculum.curriculum_id = curriculum.id
session_db$#               LEFT JOIN attestation ON attestation.student_id = student.id AND attestation.discipline_id = composition_curriculum.discipline_id
session_db$#               WHERE education.student_id = student.id AND (attestation.id IS NULL OR attestation.grade <> 5)
session_db$#           )
session_db$#     );
session_db$# END;
session_db$# $$;
CREATE PROCEDURE
```

Рисунок 1 – Создание процедуры

```
session_db=# SELECT student.id, student.gradebook_number, student.last_name, student.first_name, student.patronymic, award_scholarship.count, scholarship.scholarship_name FROM
student
session_db-# JOIN award_scholarship ON award_scholarship.student_id = student.id
session_db-# JOIN scholarship ON scholarship.id = award_scholarship.scholarship_id
session_db-# WHERE EXISTS (SELECT 1 FROM education
session_db(#     JOIN curriculum ON education.group_id = curriculum.education_group_id
session_db(#     JOIN composition_curriculum ON composition_curriculum.curriculum_id = curriculum.id
session_db(#     WHERE education.student_id = student.id
session_db(# )
session_db-# AND NOT EXISTS (SELECT 1 FROM education
session_db(#     JOIN curriculum ON education.group_id = curriculum.education_group_id
session_db(#     JOIN composition_curriculum ON composition_curriculum.curriculum_id = curriculum.id
session_db(#     LEFT JOIN attestation ON attestation.student_id = student.id
session_db(#                          AND attestation.discipline_id = composition_curriculum.discipline_id
session_db(#     WHERE education.student_id = student.id
session_db(#       AND (attestation.id IS NULL OR attestation.grade <> 5)
session_db(# );
 id | gradebook_number | last_name | first_name |  patronymic   | count |          scholarship_name
----+------------------+-----------+------------+---------------+-------+------------------------------------
 11 |              253 | Родина    | Алексей    | Михайлович    |  5000 | Стипендия за отличную успеваемость
 12 |              988 | Дроздов   | Артем      | Григорьевич   |  3000 | Научная стипендия
 19 |              908 | Кузнецова | Мария      | Григорьевна   |  7500 | Стипендия за активное участие
 25 |              386 | Тихонова  | Вера       | Александровна | 10000 | Креативная стипендия
(4 строки)
```

Рисунок 2 – Просмотр данных

```
session_db=# CALL increase_scholarship();
CALL
session_db=# SELECT student.id, student.gradebook_number, student.last_name, student.first_name, student.patronymic, award_scholarship.count, scholarship.scholarship_name, awar
d_scholarship.description FROM student
session_db-# JOIN award_scholarship ON award_scholarship.student_id = student.id
session_db-# JOIN scholarship ON scholarship.id = award_scholarship.scholarship_id
session_db-# WHERE EXISTS (SELECT 1 FROM education
session_db(#     JOIN curriculum ON education.group_id = curriculum.education_group_id
session_db(#     JOIN composition_curriculum ON composition_curriculum.curriculum_id = curriculum.id
session_db(#     WHERE education.student_id = student.id
session_db(# )
session_db-# AND NOT EXISTS (SELECT 1 FROM education
session_db(#     JOIN curriculum ON education.group_id = curriculum.education_group_id
session_db(#     JOIN composition_curriculum ON composition_curriculum.curriculum_id = curriculum.id
session_db(#     LEFT JOIN attestation ON attestation.student_id = student.id
session_db(#                          AND attestation.discipline_id = composition_curriculum.discipline_id
session_db(#     WHERE education.student_id = student.id
session_db(#       AND (attestation.id IS NULL OR attestation.grade <> 5)
session_db(# );
 id | gradebook_number | last_name | first_name |  patronymic   | count |          scholarship_name          |            description
----+------------------+-----------+------------+---------------+-------+------------------------------------+------------------------------------
 11 |              253 | Родина    | Алексей    | Михайлович    |  5500 | Стипендия за отличную успеваемость | Стипендия за отличную успеваемость
 12 |              988 | Дроздов   | Артем      | Григорьевич   |  3300 | Научная стипендия                  | Научная стипендия
 19 |              908 | Кузнецова | Мария      | Григорьевна   |  8250 | Стипендия за активное участие      | Активное участие в университете
 25 |              386 | Тихонова  | Вера       | Александровна | 11000 | Креативная стипендия               | Креативное мышление
(4 строки)
```

Рисунок 3 – Вызов процедуры и просмотр обновленных данных

Процедура для перевода студентов на следующий курс. Код процедуры представлен в листинге 2.

Листинг 2 – Для перевода студентов на следующий курс.

```
CREATE OR REPLACE PROCEDURE increment_group_numbers()
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE education_group
    SET group_number = group_number + 100;
END;
$$;
```

Создание процедуры и проверка его работоспособности представлено на рисунках 4–6.

```
session_db=# CREATE OR REPLACE PROCEDURE increment_group_numbers()
session_db-# LANGUAGE plpgsql
session_db-# AS $$
session_db$# BEGIN
session_db$#     UPDATE education_group
session_db$#     SET group_number = group_number + 100;
session_db$# END;
session_db$# $$;
CREATE PROCEDURE
```

Рисунок 4 – Создание процедуры

```
session_db=# SELECT group_number FROM education_group;
 group_number
--------------
          101
          102
          103
          104
          105
          106
          107
          108
          109
          110
         1001
         3001
         3002
         3003
         3004
         3005
         9123
(17 строк)
```

Рисунок 5 – Просмотр данных до вызова процедуры

```
session_db=# CALL increment_group_numbers();
CALL
session_db=# SELECT group_number FROM education_group;
 group_number
--------------
          201
          202
          203
          204
          205
          206
          207
          208
          209
          210
         1101
         3101
         3102
         3103
         3104
         3105
         9223
(17 строк)
```

Рисунок 6 – Вызов процедуры и просмотр обновленных данных

Процедура для изменения оценки. Код процедуры представлен в листинге 3.

Листинг 3 – Процедура для изменения оценки.

```
CREATE OR REPLACE PROCEDURE update_grade(
    attestation_id INT,
    new_grade INT
)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE attestation
    SET grade = new_grade, attempt_number = attempt_number + 1
    WHERE id = attestation_id
      AND new_grade > grade
      AND new_grade BETWEEN 2 AND 5;
END;
$$;
```

Создание процедуры и проверка его работоспособности представлено на рисунках 7–9.

```
session_db=# CREATE OR REPLACE PROCEDURE update_grade(
session_db(#     attestation_id INT,
session_db(#     new_grade INT
session_db(# )
session_db-# LANGUAGE plpgsql
session_db-# AS $$
session_db$# BEGIN
session_db$#     UPDATE attestation
session_db$#     SET grade = new_grade, attempt_number = attempt_number + 1
session_db$#     WHERE id = attestation_id
session_db$#       AND new_grade > grade
session_db$#       AND new_grade BETWEEN 2 AND 5;
session_db$# END;
session_db$# $$;
CREATE PROCEDURE
```

Рисунок 7 – Создание процедуры

```
session_db=# SELECT grade, attempt_number FROM attestation WHERE id = 32;
 grade | attempt_number
-------+----------------
     2 |              1
(1 строка)
```

Рисунок 8 – Просмотр данных до вызова процедуры

```
session_db=# CALL update_grade(32, 3);
CALL
session_db=# SELECT grade, attempt_number FROM attestation WHERE id = 32;
 grade | attempt_number
-------+----------------
     3 |              2
(1 строка)
```

Рисунок 9 – Вызов процедуры и просмотр обновленных данных

**Триггеры**

Для своей БД я придумал следующие 7 триггеров:

1. Триггер, который не допускает добавление одного и того же преподавателя дважды в одну комиссию.
2. Триггер, который проверяет, что год старта в учебном плане не раньше начала программы.
3. Триггер, который автоматически завершает предыдущую запись в истории должностей при добавлении новой.
4. Триггер, который удаляет назначенную стипендию, если вставлена оценка 2.
5. Триггер, который триггер на попытку вставки экзамена с устаревшей датой (например, меньше сегодняшнего дня).
6. Триггер, который не даёт студенту быть одновременно зачисленным в несколько групп.
7. Триггер, который запрещает изменение оценок по завершённым сессиям (где дата экзамена в прошлом).

Триггер, который не допускает добавление одного и того же преподавателя дважды в одну комиссию. Код функции и триггера представлен в листинге 4.

Листинг 4 – Создание функции и триггера, который не допускает добавление одного и того же преподавателя дважды в одну комиссию.

```
CREATE OR REPLACE FUNCTION check_commission_member()
RETURNS TRIGGER AS $$
BEGIN
    IF EXISTS (SELECT 1 FROM attestation_commission WHERE staff_id =
NEW.staff_id AND attestation_id = NEW.attestation_id) THEN
        RAISE EXCEPTION 'Преподаватель уже добавлен в комиссию для
аттестации';
    END IF;
```

```
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER t_check_commission_member BEFORE INSERT ON
attestation_commission FOR EACH ROW EXECUTE FUNCTION
check_commission_member();
```

Создание функции и триггера, а также его работоспособности представлено на рисунках 10–11.

```
session_db=# CREATE OR REPLACE FUNCTION check_commission_member()
session_db-# RETURNS TRIGGER AS $$
session_db$# BEGIN
session_db$#     IF EXISTS (SELECT 1 FROM attestation_commission WHERE staff_id = NEW.staff_id AND attestation_id = NEW.attestation_id) THEN
session_db$#         RAISE EXCEPTION 'Преподаватель уже добавлен в комиссию для аттестации';
session_db$#     END IF;
session_db$#     RETURN NEW;
session_db$# END;
session_db$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
session_db=#
session_db=#
session_db=# CREATE TRIGGER t_check_commission_member BEFORE INSERT ON attestation_commission FOR EACH ROW EXECUTE FUNCTION check_commission_member();
CREATE TRIGGER
```

Рисунок 10 – Создание функции и триггера

```
session_db=# SELECT * FROM attestation_commission
session_db-# WHERE staff_id = 1 AND attestation_id = 2;
 id | staff_id | attestation_id
----+----------+----------------
(0 строк)


session_db=# INSERT INTO attestation_commission (staff_id, attestation_id)
session_db-# VALUES (1, 2);
INSERT 0 1
session_db=# SELECT * FROM attestation_commission
session_db-# WHERE staff_id = 1 AND attestation_id = 2;
 id | staff_id | attestation_id
----+----------+----------------
 68 |        1 |              2
(1 строка)


session_db=# INSERT INTO attestation_commission (staff_id, attestation_id)
session_db-# VALUES (1, 2);
ОШИБКА:  Преподаватель уже добавлен в состав аттестационной комиссии
КОНТЕКСТ:  функция PL/pgSQL prevent_duplicate_commission_member(), строка 8, оператор RAISE
session_db=# SELECT * FROM attestation_commission
session_db-# WHERE attestation_id = 2;
 id | staff_id | attestation_id
----+----------+----------------
  2 |        2 |              2
 68 |        1 |              2
(2 строки)
```

Рисунок 11 – Проверка триггера

Триггер, который проверяет, что год старта в учебном плане не раньше начала программы. Код функции и триггера представлен в листинге 5.

Листинг 5 – Создание функции и триггера, который проверяет, что год старта в учебном плане не раньше начала программы

```
CREATE OR REPLACE FUNCTION validate_curriculum_year()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.start_year < EXTRACT(YEAR FROM CURRENT_DATE)::INT THEN
        RAISE EXCEPTION 'Год начала учебы не может быть меньше текущего
года';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER t_validate_curriculum_year BEFORE INSERT ON
composition_curriculum FOR EACH ROW EXECUTE FUNCTION
validate_curriculum_year();
```

Создание функции и триггера, а также его работоспособности представлено на рисунках 12–13.

```
session_db=# CREATE OR REPLACE FUNCTION validate_curriculum_year()
session_db-# RETURNS TRIGGER AS $$
session_db$# BEGIN
session_db$#     IF NEW.start_year < EXTRACT(YEAR FROM CURRENT_DATE)::INT THEN
session_db$#         RAISE EXCEPTION 'Год начала учебы не может быть меньше текущего года';
session_db$#     END IF;
session_db$#     RETURN NEW;
session_db$# END;
session_db$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
session_db=#
session_db=#
session_db=# CREATE TRIGGER t_validate_curriculum_year BEFORE INSERT ON composition_curriculum FOR EACH ROW EXECUTE FUNCTION validate_curriculum_year();
CREATE TRIGGER
```

Рисунок 12 – Создание функции и триггера

```
session_db=# INSERT INTO composition_curriculum (hours_count, start_year, discipline_id, curriculum_id)
session_db-# VALUES (72, 2014, 1, 1);
ОШИБКА:  Год начала учебы не может быть меньше текущего года
КОНТЕКСТ:  функция PL/pgSQL validate_curriculum_year(), строка 4, оператор RAISE
session_db=# SELECT * FROM composition_curriculum
session_db-# WHERE start_year = 2014;
 id | hours_count | start_year | discipline_id | curriculum_id
----+-------------+------------+---------------+---------------
(0 строк)


session_db=# INSERT INTO composition_curriculum (hours_count, start_year, discipline_id, curriculum_id)
session_db-# VALUES (72, 2028, 1, 1);
INSERT 0 1
session_db=# SELECT * FROM composition_curriculum
session_db-# WHERE start_year = 2028;
 id | hours_count | start_year | discipline_id | curriculum_id
----+-------------+------------+---------------+---------------
 24 |          72 |       2028 |             1 |             1
(1 строка)
```

Рисунок 13 – Проверка триггера

Триггер, который автоматически завершает предыдущую запись в истории должностей при добавлении новой. Код функции и триггера представлен в листинге 6.

Листинг 6 – Создание функции и триггера, который автоматически завершает предыдущую запись в истории должностей при добавлении новой

```
CREATE OR REPLACE FUNCTION end_previous_position()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE staff_history SET end_date = NEW.start_date - INTERVAL '1
day' WHERE staff_id = NEW.staff_id AND end_date IS NULL;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER t_end_previous_position
BEFORE INSERT ON staff_history
FOR EACH ROW
EXECUTE FUNCTION end_previous_position();
```

Создание функции и триггера, а также его работоспособности представлено на рисунках 14–15.

```
session_db=# CREATE OR REPLACE FUNCTION end_previous_position()
session_db-# RETURNS TRIGGER AS $$
session_db$# BEGIN
session_db$#     UPDATE staff_history SET end_date = NEW.start_date - INTERVAL '1 day' WHERE staff_id = NEW.staff_id AND end_date IS NULL;
session_db$#     RETURN NEW;
session_db$# END;
session_db$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
session_db=#
session_db=#
session_db=# CREATE TRIGGER t_end_previous_position
session_db-# BEFORE INSERT ON staff_history
session_db-# FOR EACH ROW
session_db-# EXECUTE FUNCTION end_previous_position();
CREATE TRIGGER
```

Рисунок 14 – Создание функции и триггера

```
session_db=# SELECT * FROM staff_history WHERE staff_id = 1;
 id | start_date |  end_date  | position_id | staff_id
----+------------+------------+-------------+----------
  1 | 2019-01-01 | 2020-12-31 |           1 |        1
(1 строка)


session_db=# INSERT INTO staff_history (start_date, end_date, position_id, staff_id)
session_db-# VALUES ('2024-01-01', NULL, 2, 1);
INSERT 0 1
session_db=# SELECT * FROM staff_history WHERE staff_id = 1;
 id | start_date |  end_date  | position_id | staff_id
----+------------+------------+-------------+----------
  1 | 2019-01-01 | 2020-12-31 |           1 |        1
 30 | 2024-01-01 |            |           2 |        1
(2 строки)


session_db=# INSERT INTO staff_history (start_date, end_date, position_id, staff_id)
session_db-# VALUES ('2025-01-01', NULL, 3, 1);
INSERT 0 1
session_db=# SELECT * FROM staff_history WHERE staff_id = 1;
 id | start_date |  end_date  | position_id | staff_id
----+------------+------------+-------------+----------
  1 | 2019-01-01 | 2020-12-31 |           1 |        1
 30 | 2024-01-01 | 2024-12-31 |           2 |        1
 31 | 2025-01-01 |            |           3 |        1
(3 строки)
```

Рисунок 15 – Проверка триггера

Триггер, который удаляет назначенную стипендию, если вставлена оценка 2. Код функции и триггера представлен в листинге 7.

Листинг 7 – Создание функции и триггера, который удаляет назначенную стипендию, если вставлена оценка 2

```
CREATE OR REPLACE FUNCTION remove_scholarship()
RETURNS trigger
LANGUAGE plpgsql
AS $$
BEGIN
    IF NEW.grade = 2 THEN DELETE FROM award_scholarship WHERE student_id
= NEW.student_id;
    END IF;
    RETURN NEW;
END;
$$;

CREATE TRIGGER t_remove_scholarship AFTER INSERT ON attestation FOR
EACH ROW EXECUTE FUNCTION remove_scholarship_on_grade_two();
```

Создание функции и триггера, а также его работоспособности представлено на рисунках 16–17.

```
session_db=# CREATE OR REPLACE FUNCTION remove_scholarship()
session_db-# RETURNS trigger
session_db-# LANGUAGE plpgsql
session_db-# AS $$
session_db$# BEGIN
session_db$#     IF NEW.grade = 2 THEN DELETE FROM award_scholarship WHERE student_id = NEW.student_id;
session_db$#     END IF;
session_db$#     RETURN NEW;
session_db$# END;
session_db$# $$;
CREATE FUNCTION
session_db=#
session_db=#
session_db=# CREATE TRIGGER t_remove_scholarship AFTER INSERT ON attestation FOR EACH ROW EXECUTE FUNCTION remove_scholarship_on_grade_two();
CREATE TRIGGER
```

Рисунок 16 – Создание функции и триггера

```
session_db=# INSERT INTO award_scholarship (start_date, end_date, description, student_id, scholarship_id, count)
session_db-# VALUES ('2025-01-01', '2025-06-30', 'Базовая стипендия', 123, 1, 5000);
INSERT 0 1
session_db=#
session_db=# SELECT * FROM award_scholarship WHERE student_id = 123;
 id  | start_date |  end_date  |    description    | student_id | scholarship_id | count
-----+------------+------------+-------------------+------------+----------------+-------
 173 | 2025-01-01 | 2025-06-30 | Базовая стипендия |        123 |              1 |  5000
(1 строка)


session_db=# INSERT INTO attestation (student_id, discipline_id, grade, attempt_number, attestation_type)
session_db-# VALUES (123, 3, 2, 1, 'экзамен');
INSERT 0 1
session_db=# SELECT * FROM award_scholarship WHERE student_id = 123;
 id | start_date | end_date | description | student_id | scholarship_id | count
----+------------+----------+-------------+------------+----------------+-------
(0 строк)
```

Рисунок 17 – Проверка триггера

Триггер, который триггер на попытку вставки экзамена с устаревшей датой. Код функции и триггера представлен в листинге 8.

Листинг 8 – Создание функции и триггера, который триггер на попытку вставки экзамена с устаревшей датой

```
CREATE OR REPLACE FUNCTION check_exam_date()
RETURNS trigger
LANGUAGE plpgsql
AS $$
BEGIN
    IF NEW.exam_date < CURRENT_DATE THEN
        RAISE EXCEPTION 'Нельзя назначить экзамен в прошлом';
    END IF;
    RETURN NEW;
END;
$$;

CREATE TRIGGER t_check_exam_date BEFORE INSERT ON session_schedule FOR
EACH ROW EXECUTE FUNCTION check_exam_date_not_past();
```

Создание функции и триггера, а также его работоспособности представлено на рисунках 18–19.

```
session_db=# CREATE OR REPLACE FUNCTION check_exam_date()
session_db-# RETURNS trigger
session_db-# LANGUAGE plpgsql
session_db-# AS $$
session_db$# BEGIN
session_db$#     IF NEW.exam_date < CURRENT_DATE THEN
session_db$#         RAISE EXCEPTION 'Нельзя назначить экзамен в прошлом';
session_db$#     END IF;
session_db$#     RETURN NEW;
session_db$# END;
session_db$# $$;
CREATE FUNCTION
session_db=#
session_db=#
session_db=# CREATE TRIGGER t_check_exam_date BEFORE INSERT ON session_schedule FOR EACH ROW EXECUTE FUNCTION check_exam_date_not_past();
CREATE TRIGGER
```

Рисунок 18 – Создание функции и триггера

```
session_db=# INSERT INTO session_schedule (classroom, area, exam_date, education_group_id, attestation_id)
session_db-# VALUES (101, 'Корпус А', '2025-06-01 10:00:00', 1, 2);
INSERT 0 1
session_db=# SELECT * FROM session_schedule
session_db-# WHERE education_group_id = 1 AND attestation_id = 2;
 id | classroom |   area   |      exam_date      | education_group_id | attestation_id
----+-----------+----------+---------------------+--------------------+----------------
 18 |       101 | Корпус А | 2025-06-01 10:00:00 |                  1 |              2
(1 строка)


session_db=# INSERT INTO session_schedule (classroom, area, exam_date, education_group_id, attestation_id)
session_db-# VALUES (101, 'Корпус А', '2024-04-01 10:00:00', 1, 2);
ОШИБКА:  Нельзя назначить экзамен в прошлом
КОНТЕКСТ:  функция PL/pgSQL check_exam_date_not_past(), строка 4, оператор RAISE
session_db=# SELECT * FROM session_schedule
session_db-# WHERE education_group_id = 1 AND attestation_id = 2;
 id | classroom |   area   |      exam_date      | education_group_id | attestation_id
----+-----------+----------+---------------------+--------------------+----------------
 18 |       101 | Корпус А | 2025-06-01 10:00:00 |                  1 |              2
(1 строка)
```

Рисунок 19 – Проверка триггера

Триггер, который не даёт студенту быть одновременно зачисленным в несколько групп. Код функции и триггера представлен в листинге 9.

Листинг 9 – Создание функции и триггера, который не даёт студенту быть одновременно зачисленным в несколько групп

```
CREATE OR REPLACE FUNCTION validate_group_for_student()
RETURNS TRIGGER AS $$
BEGIN
   IF EXISTS (SELECT 1 FROM education WHERE student_id = NEW.student_id
AND (NEW.start_date <= end_date AND NEW.end_date >= start_date)) THEN
        RAISE EXCEPTION 'Студент уже зачислен в другую группу в этот
период';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER t_validate_group_for_student BEFORE INSERT ON education
FOR EACH ROW EXECUTE FUNCTION validate_group_for_student();
```

Создание функции и триггера, а также его работоспособности представлено на рисунках 20–21.

```
session_db=# CREATE OR REPLACE FUNCTION validate_group_for_student()
session_db-# RETURNS TRIGGER AS $$
session_db$# BEGIN
session_db$#    IF EXISTS (SELECT 1 FROM education WHERE student_id = NEW.student_id AND (NEW.start_date <= end_date AND NEW.end_date >= start_date)) THEN
session_db$#        RAISE EXCEPTION 'Студент уже зачислен в другую группу в этот период';
session_db$#     END IF;
session_db$#     RETURN NEW;
session_db$# END;
session_db$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
session_db=#
session_db=#
session_db=# CREATE TRIGGER t_validate_group_for_student BEFORE INSERT ON education FOR EACH ROW EXECUTE FUNCTION validate_group_for_student();
CREATE TRIGGER
```

Рисунок 20 – Создание функции и триггера

```
session_db=# SELECT * FROM education
session_db-# WHERE student_id = 5;
 id | start_date |  end_date  | group_id | student_id
----+------------+------------+----------+------------
  5 | 2020-09-01 | 2024-06-30 |        5 |          5
(1 строка)


session_db=# INSERT INTO education (start_date, end_date, group_id, student_id)
session_db-# VALUES ('2025-01-01', '2025-12-31', 2, 5);
INSERT 0 1
session_db=# SELECT * FROM education
session_db-# WHERE student_id = 5;
 id | start_date |  end_date  | group_id | student_id
----+------------+------------+----------+------------
  5 | 2020-09-01 | 2024-06-30 |        5 |          5
 83 | 2025-01-01 | 2025-12-31 |        2 |          5
(2 строки)


session_db=# INSERT INTO education (start_date, end_date, group_id, student_id)
session_db-# VALUES ('2025-09-01', '2026-09-01', 2, 5);
ОШИБКА:  Студент уже зачислен в другую группу в этот период
КОНТЕКСТ:  функция PL/pgSQL validate_group_for_student(), строка 4, оператор RAISE
session_db=# SELECT * FROM education
session_db-# WHERE student_id = 5;
 id | start_date |  end_date  | group_id | student_id
----+------------+------------+----------+------------
  5 | 2020-09-01 | 2024-06-30 |        5 |          5
 83 | 2025-01-01 | 2025-12-31 |        2 |          5
(2 строки)
```

Рисунок 21 – Проверка триггера

Триггер, который запрещает вставку, если уже есть 3 попытки у студента по дисциплине. Код функции и триггера представлен в листинге 10.

Листинг 10 – Создание функции и триггера, который запрещает вставку, если уже есть 3 попытки у студента по дисциплине

```
CREATE OR REPLACE FUNCTION check_number_attempt()
RETURNS trigger
LANGUAGE plpgsql
```

```
AS $$
BEGIN
    IF (SELECT COUNT(*) FROM attestation WHERE student_id =
NEW.student_id AND discipline_id = NEW.discipline_id) >= 3 THEN
        RAISE EXCEPTION 'Нельзя добавить больше 3 попыток для одного
студента по одной дисциплине';
    END IF;
    RETURN NEW;
END;
$$;

CREATE TRIGGER t_check_number_attempt BEFORE INSERT ON attestation FOR
EACH ROW EXECUTE FUNCTION check_number_attempt();
```

Создание функции и триггера, а также его работоспособности представлено на рисунках 24–25.

```
session_db=# CREATE OR REPLACE FUNCTION check_number_attempt()
session_db-# RETURNS trigger
session_db-# LANGUAGE plpgsql
session_db-# AS $$
session_db$# BEGIN
session_db$#     IF (SELECT COUNT(*) FROM attestation WHERE student_id = NEW.student_id AND discipline_id = NEW.discipline_id) >= 3 THEN
session_db$#         RAISE EXCEPTION 'Нельзя добавить больше 3 попыток для одного студента по одной дисциплине';
session_db$#     END IF;
session_db$#     RETURN NEW;
session_db$# END;
session_db$# $$;
CREATE FUNCTION
session_db=#
session_db=#
session_db=# CREATE TRIGGER t_check_number_attempt BEFORE INSERT ON attestation FOR EACH ROW EXECUTE FUNCTION check_number_attempt();
CREATE TRIGGER
```

Рисунок 24 – Создание функции и триггера

```
session_db=# INSERT INTO attestation (student_id, discipline_id, grade, attempt_number, attestation_type)
session_db-# VALUES (7, 4, 2, 1, 'экзамен');
INSERT 0 1
session_db=# INSERT INTO attestation (student_id, discipline_id, grade, attempt_number, attestation_type)
session_db-# VALUES (7, 4, 2, 2, 'экзамен');
INSERT 0 1
session_db=# INSERT INTO attestation (student_id, discipline_id, grade, attempt_number, attestation_type)
session_db-# VALUES (7, 4, 2, 3, 'экзамен');
INSERT 0 1
session_db=# SELECT * FROM attestation
session_db-# WHERE student_id = 7 AND discipline_id = 4;
 id  | student_id | discipline_id | grade | attempt_number | attestation_type
-----+------------+---------------+-------+----------------+------------------
 372 |          7 |             4 |     2 |              1 | экзамен
 373 |          7 |             4 |     2 |              2 | экзамен
 374 |          7 |             4 |     2 |              3 | экзамен
(3 строки)


session_db=# INSERT INTO attestation (student_id, discipline_id, grade, attempt_number, attestation_type)
session_db-# VALUES (7, 4, 5, 3, 'экзамен');
ОШИБКА:  Нельзя добавить больше 3 попыток для одного студента по одной дисциплине
КОНТЕКСТ:  функция PL/pgSQL check_number_attempt(), строка 4, оператор RAISE
session_db=# SELECT * FROM attestation
session_db-# WHERE student_id = 7 AND discipline_id = 4;
 id  | student_id | discipline_id | grade | attempt_number | attestation_type
-----+------------+---------------+-------+----------------+------------------
 372 |          7 |             4 |     2 |              1 | экзамен
 373 |          7 |             4 |     2 |              2 | экзамен
 374 |          7 |             4 |     2 |              3 | экзамен
(3 строки)
```

Рисунок 25 – Проверка триггера

# ВЫВОД

В ходе выполнения лабораторной работы получены навыки создания и использования процедур, функций и триггеров в СУБД PostgreSQL. Были созданы хранимые процедуры, а также триггеры.