**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

A logo of a triangle

Description automatically generated

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

|  |  |
| --- | --- |
| Институт | компьютерных наук |
| Кафедра | автоматизированных систем управления |

ЛАБОРАТОРНАЯ РАБОТА №6

По дисциплине: "Базы данных"

На тему: "Программирование серверной логики базы данных"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ПИ-22-1 | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | Насонов Н.С. |
| Руководитель | |  |  |  |
| к.т.н., доцент  ученая степень, ученое звание | |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | Алексеев В.А. |

Липецк, 2024 г.

**Цель работы**

Изучить возможности программирования серверной логики базы данных с использованием триггеров и хранимых процедур. Получить практические навыки программирования триггеров и хранимых процедур для выбранной СУБД.

**Задание**

в БД триггеры для поддержки бизнес-логики информационной системы (не менее 2-х). Спроектировать транзакцию и реализовать её в виде хранимой процедуры. Проверить правильность работы созданных объектов БД.

По желанию могут быть реализованы также пользовательские (хранимые) функции БД.

**1. Триггер №1**

**1.1. Описание логики работы на естественном языке**

Триггер проверяет корректность добавляемой оценки в таблицу mark. Если значение score\_mark выходит за пределы допустимого диапазона (0–100), выполнение операции прерывается, а добавление записи отменяется.

**1.2. Фрагмент физической схемы данных, отображающий таблицу, с которой взаимодействует триггер**

Фрагмент физической схемы данных, отображающий таблицу, с которой взаимодействует триггер представлен на рисунке 1.

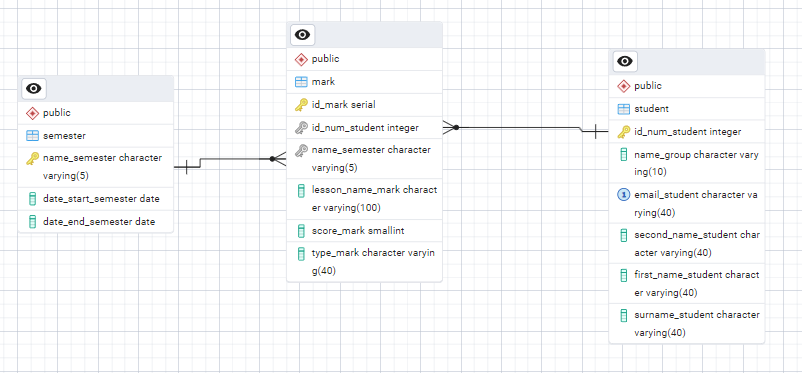


Рисунок 1 – Фрагмент физической схемы данных

**1.3. Текст процедуры триггера на языке SQL**

CREATE OR REPLACE FUNCTION check\_mark\_score()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.score\_mark < 0 OR NEW.score\_mark > 100 THEN

RAISE EXCEPTION 'Оценка должна быть в диапазоне от 0 до 100. Некорректное значение: %', NEW.score\_mark;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_check\_mark\_score

BEFORE INSERT OR UPDATE ON mark

FOR EACH ROW

EXECUTE FUNCTION check\_mark\_score();

**1.4. Пример выполнения**

INSERT INTO mark (id\_num\_student, name\_semester, lesson\_name\_mark, score\_mark, type\_mark)

VALUES (1220060405, '2023В', 'Физика', 150, 'Экзамен');

На рисунке 2 представлена таблица после выполнения запроса.

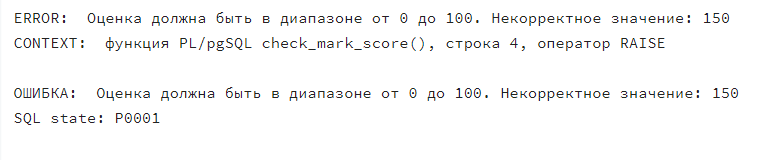


Рисунок 2 – Пример выполнения триггера

**2. Триггер №2**

**2.1. Описание логики работы на естественном языке**

Триггер срабатывает перед вставкой записи в таблицу mark. Он автоматически присваивает значение в поле type\_mark в зависимости от поля score\_mark.

**2.2. Фрагмент физической схемы данных, отображающий таблицу, с которой взаимодействует триггер**

Фрагмент физической схемы данных, отображающий таблицу, с которой взаимодействует триггер представлен на рисунке 3.

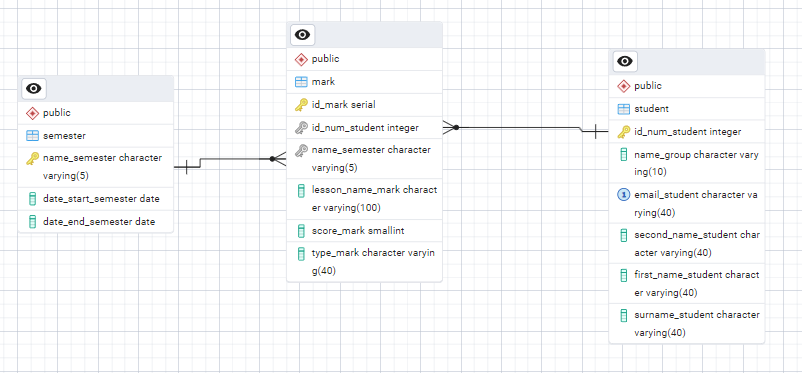


Рисунок 3 – Фрагмент физической схемы данных

**2.3. Текст процедуры триггера на языке SQL**

CREATE OR REPLACE FUNCTION set\_type\_mark()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.score\_mark >= 93 THEN

NEW.type\_mark := 'Отлично';

ELSIF NEW.score\_mark >= 80 THEN

NEW.type\_mark := 'Хорошо';

ELSIF NEW.score\_mark >= 53 THEN

NEW.type\_mark := 'Удовлетворительно';

ELSE

NEW.type\_mark := 'Неудовлетворительно';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger\_set\_type\_mark

BEFORE INSERT ON mark

FOR EACH ROW

EXECUTE FUNCTION set\_type\_mark();

**2.4. Пример выполнения**

Запрос для добавления оценки представлен ниже.

INSERT INTO mark (id\_num\_student, name\_semester, lesson\_name\_mark, score\_mark, type\_mark)

VALUES (1220060405, '2023В', 'Физика', 82, 'Экзамен');

На рисунке 4 представлена таблица после выполнения запроса.

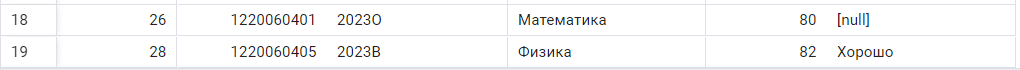


Рисунок 4 – Пример выполнения триггера

**3. Хранимая процедура**

**3.1. Описание логики транзакции на естественном языке**

1. Процедура принимает на вход:
   * Порог среднего балла (threshold\_score).
   * Название семестра (current\_semester).
   * Тип бюджета (budget\_type).
2. Находит id\_budget из таблицы budget, соответствующий текущему семестру и указанному типу бюджета.
3. Для студентов, чей средний балл за текущий семестр превышает порог:
   * Добавляет записи о стипендии в таблицу scholarship.

**3.2. Блок-схема алгоритма транзакции**

На рисунке 5 представлена блок-схема алгоритма процедуры.



Рисунок 5 – Блок-схема транзакции

**3.3. Фрагмент физической схемы данных, отображающий таблицы, с которыми работает хранимая процедура, и их взаимосвязи**

На рисунке 6 представлена схема данных, с которой взаимодействует процедура

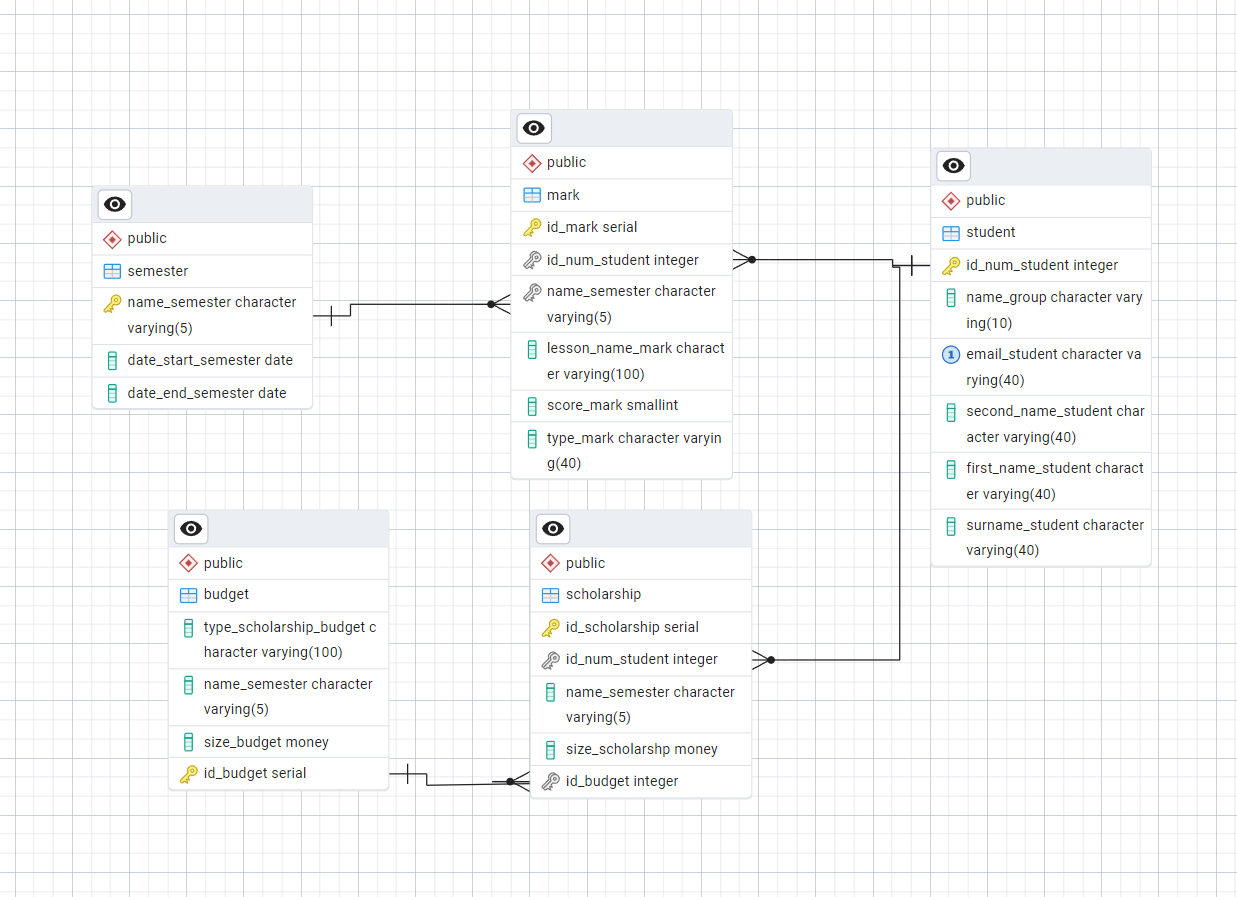


Рисунок 6 – Фрагмент физической схемы данных

**3.4. Текст хранимой процедуры на языке SQL**

CREATE OR REPLACE PROCEDURE assign\_scholarships(current\_semester VARCHAR, budget\_type VARCHAR)

LANGUAGE plpgsql

AS $$

DECLARE

target\_budget\_id INTEGER;

BEGIN

SELECT id\_budget

INTO target\_budget\_id

FROM budget

WHERE name\_semester = current\_semester AND type\_scholarship\_budget = budget\_type

LIMIT 1;

IF target\_budget\_id IS NULL THEN

RAISE EXCEPTION 'Бюджет не найден для семестра % и типа %', current\_semester, budget\_type;

END IF;

DELETE FROM scholarship

WHERE name\_semester = current\_semester;

INSERT INTO scholarship (id\_num\_student, name\_semester, size\_scholarship, id\_budget)

SELECT

m.id\_num\_student,

current\_semester,

4500+(AVG(m.score\_mark)-80)\*100,

target\_budget\_id

FROM

mark m

WHERE

m.name\_semester = current\_semester

GROUP BY

m.id\_num\_student, m.name\_semester

HAVING

AVG(m.score\_mark) >= 80;

RAISE NOTICE 'Стипендии начислены студентам за семестр: % и тип бюджета: %', current\_semester, budget\_type;

END;

$$;

**3.5. Скриншот с содержимым таблицы до выполнения хранимой процедуры**

На рисунке 7 представлено содержимое таблицы scholarship до выполнения хранимой процедуры.

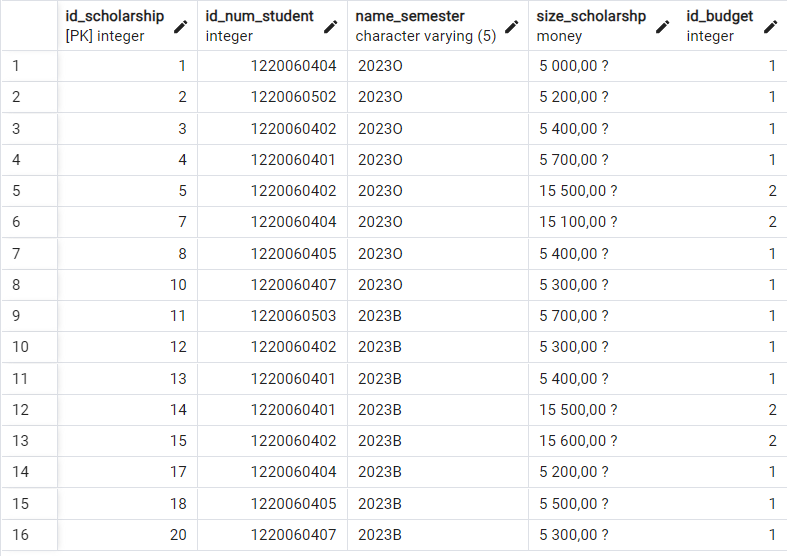


Рисунок 7 – Таблица scholarship до вызова процедуры

**3.6. Формулировка запроса или описание действия, вызывающего исполнения хранимой процедуры**

На рисунке 8 представлен вызов хранимой процедуры.

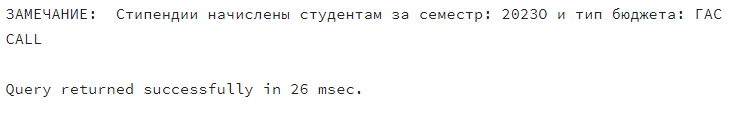


Рисунок 8 – Вызов процедуры

Ниже представлен корректный запрос на выполнение хранимой процедуры.

CALL assign\_scholarships('2023О','ГАС')

**3.7. Скриншот с содержимым таблицы после выполнения хранимой процедуры**

На рисунке 9 представлено содержимое таблицы scholarship после выполнения хранимой процедуры.

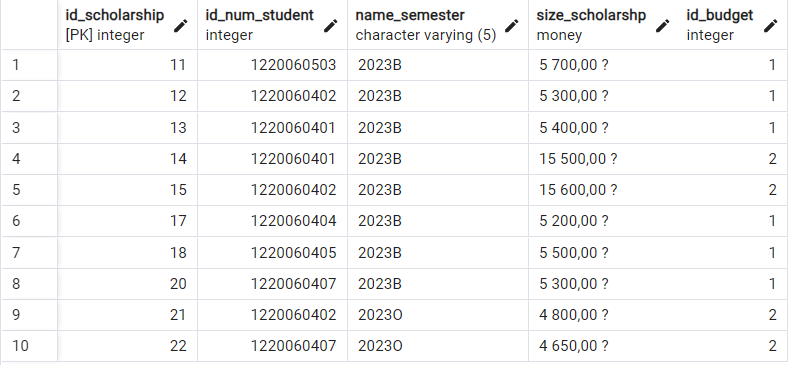


Рисунок 9 – Таблица scholarship после вызова процедуры

**Вывод**

В ходе выполнения лабораторной работы были изучены возможности программирования серверной логики базы данных с использованием триггеров и хранимых процедур. Приобретены практические навыки программирования триггеров и хранимых процедур.