МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

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

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №6

по дисциплине

“Базы данных”

Студент АС-21-1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Станиславчук С. М.

(подпись, дата)

Руководитель

Доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Алексеев В. А.

(подпись, дата)

Липецк 2023

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

Реализовать в БД триггеры для поддержки бизнес-логики

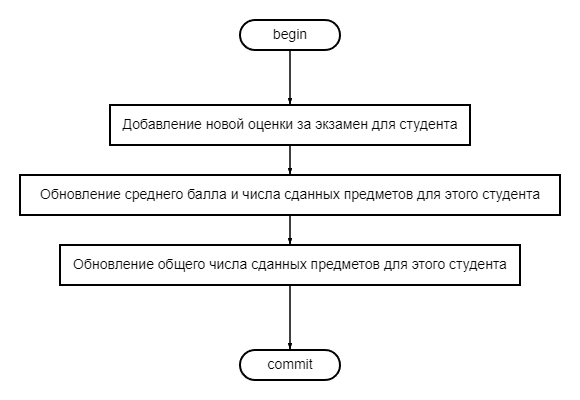
информационной системы (не менее 2-х). Реализовать хранимую процедуру для транзакции, разработанной в лабораторной работе #5. Проверить правильность работы созданных объектов БД.

По желанию могут быть реализованы также пользовательские (хранимые)

функции БД

Ход работы

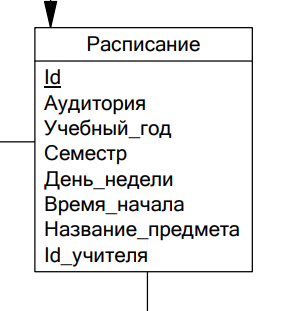
Блок-схема транзакции для лр#5:



3.1 Занятость преподавателя

В одно и то же преподаватель не может вести более одной пары.

Триггер: при вставке записи в таблицу расписания (schedule), проверять, не накладываются ли новые данные на уже существующие по времени занятия для преподавателя.



Триггер:

CREATE OR REPLACE FUNCTION check\_schedule\_conflict()

RETURNS TRIGGER AS $$

BEGIN

IF EXISTS (

SELECT 1

FROM schedule s

WHERE s.id\_teacher = NEW.id\_teacher

AND s.weekday\_schedule = NEW.weekday\_schedule

AND s.start\_time\_schedule = NEW.start\_time\_schedule

)

THEN

RAISE EXCEPTION 'Этот преподаватель занят в это время!';

END IF;

RETURN NEW;

END;

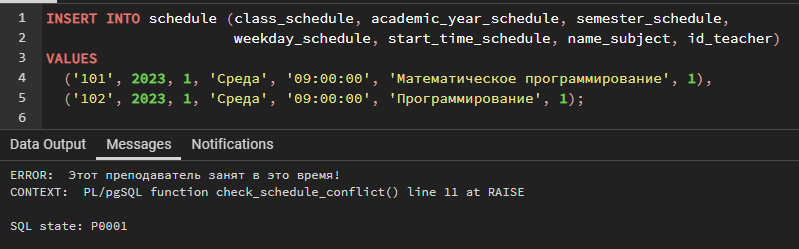
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER schedule\_conflict\_trigger

BEFORE INSERT OR UPDATE ON schedule

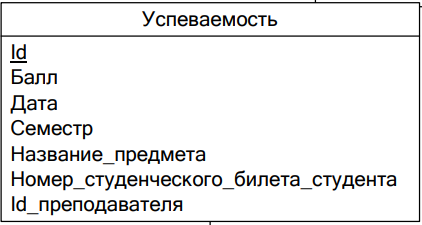
FOR EACH ROW EXECUTE FUNCTION check\_schedule\_conflict();

Теперь при попытке добавить в расписание запись о преподавателе, который уже ведет занятия в это время, вызовется эксепшен:

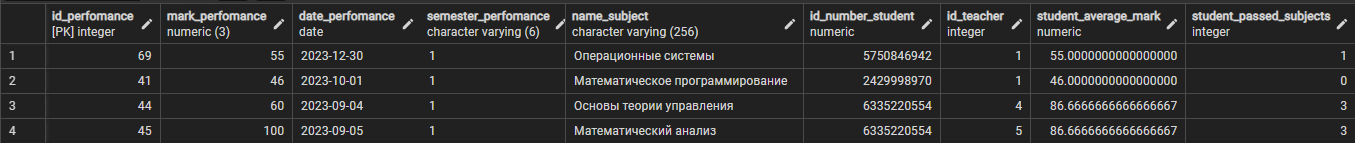


3.2 Средний балл студента

При каждой оценке, добавляемой в таблицу успеваемости (perfomance), автоматически пересчитывать средний балл студента.



До SQL-запроса (обратите внимание на первую запись):



Триггер:

CREATE OR REPLACE FUNCTION update\_student\_average\_mark() RETURNS TRIGGER AS $$

DECLARE

is\_recursive\_update BOOLEAN;

BEGIN

-- Проверяем, был ли триггер вызван ранее в рамках текущей транзакции

SELECT (pg\_trigger\_depth() > 1) INTO is\_recursive\_update;

-- Если это не рекурсивный вызов триггера, выполняем обновление

IF NOT is\_recursive\_update THEN

UPDATE perfomance

SET student\_average\_mark = (

SELECT AVG(mark\_perfomance)

FROM perfomance p

WHERE p.id\_number\_student = NEW.id\_number\_student

)

WHERE perfomance.id\_number\_student = NEW.id\_number\_student;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

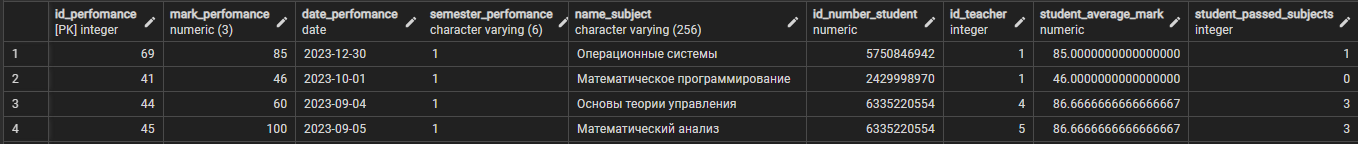
CREATE OR REPLACE TRIGGER update\_average\_mark\_trigger

AFTER INSERT OR UPDATE ON perfomance

FOR EACH ROW

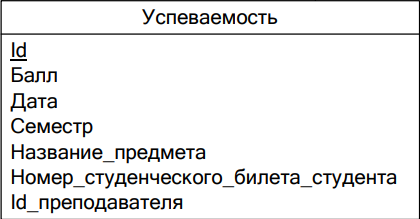
EXECUTE FUNCTION update\_student\_average\_mark();

После SQL-запроса (на обновление оценки с 55 до 85 баллов):

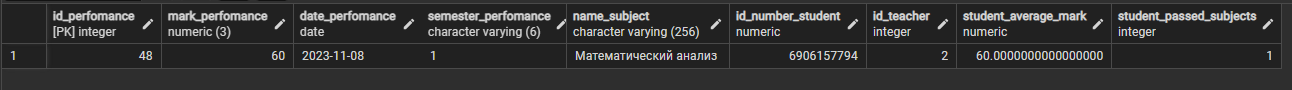


4. Хранимая процедура

Данная процедура используется для того, чтобы занести в БД запись о новой оценке студента. Благодаря чему, помимо добавления самой оценки, автоматически обновится еще и средний балл студента по этому предмету, а также общее число сданных предметов.



До:



CREATE OR REPLACE PROCEDURE update\_student\_performance(

IN p\_id\_number\_student NUMERIC(10),

IN p\_name\_subject VARCHAR(256),

IN p\_mark\_perfomance NUMERIC,

IN p\_semester\_perfomance VARCHAR(6),

IN p\_id\_teacher INT

)

AS $$

BEGIN

-- 1. Добавление новой оценки за экзамен для студента

INSERT INTO perfomance (id\_number\_student, name\_subject, mark\_perfomance, date\_perfomance, semester\_perfomance, id\_teacher)

VALUES (p\_id\_number\_student, p\_name\_subject, p\_mark\_perfomance,

CURRENT\_DATE, p\_semester\_perfomance, p\_id\_teacher);

-- 2. Обновление среднего балла для студента

UPDATE perfomance

SET

student\_average\_mark = (

SELECT AVG(mark\_perfomance)

FROM perfomance p

WHERE p.id\_number\_student = p\_id\_number\_student

AND p.name\_subject = p\_name\_subject

),

student\_passed\_subjects = (

SELECT COUNT(DISTINCT name\_subject)

FROM perfomance p

WHERE p.id\_number\_student = p\_id\_number\_student

AND mark\_perfomance >= 53

)

WHERE EXISTS (

SELECT 1

FROM perfomance p

WHERE p.id\_number\_student = p\_id\_number\_student

AND p.name\_subject = p\_name\_subject

AND p.mark\_perfomance >= 53

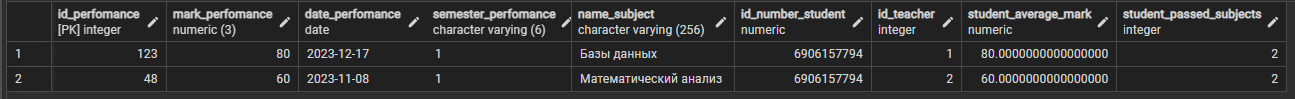
);

END;

$$ LANGUAGE plpgsql;

== CALL update\_student\_performance(6906157794, 'Базы данных', 80, '1', 1); ==

После:



И еще одна оценка:

== CALL update\_student\_performance(6906157794, 'Базы данных', 90, '1', 2); ==

