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»**

Автор: Кахикало К.Р.

Факультет: ИКТ

Группа: К3240

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Оглавление

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

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, pgadmin 4.

Практическое задание:

1. Создать процедуры/функции согласно индивидуальному заданию (часть 4).

2.1. Модифицировать триггер (триггерную функцию) на проверку корректности входа и выхода сотрудника (см. Практическое задание 1 Лабораторного практикума (Приложение)) с максимальным учетом «узких» мест некорректных данных по входу и выходу).

2.2. Создать авторский триггер по варианту индивидуального задания.

Вариант 7. БД «Курсы»

Описание предметной области: Сеть учебных подразделений НОУ ДПО занимается организацией внебюджетного образования.

Имеется несколько образовательных программ краткосрочных курсов, предназначенных для определенных специальностей, связанных с программным обеспечением ИТ. Каждый программа имеет определенную длительность и свой перечень изучаемых дисциплин. Одна дисциплина может относиться к нескольким программам. На каждую программу может быть набрано несколько групп обучающихся.

По каждой дисциплине могут проводиться лекционные, лабораторные/практические занятия и практика определенном объеме часов. По каждой дисциплине и практике проводится аттестация в формате экзамен/дифзачет/зачет.

Необходимо хранить информацию по аттестации обучающихся.

Подразделение обеспечивает следующие ресурсы: учебные классы, лекционные аудитории и преподавателей. Необходимо составить расписание занятий.

БД должна содержать следующий минимальный набор сведений: Фамилия слушателя. Имя слушателя. Паспортные данные. Контакты. Код программы. Программа. Тип программы. Объем часов. Номер группы. максимальное количество человек в группе (для набора). Дата начала обучения. Дата окончания обучения. Название дисциплины. Количество часов. Дата занятий. Номер пары. Номер аудитории. Тип аудитории. Адрес площадки. Вид занятий (лекционные, практические или лабораторные). Фамилия преподавателя. Имя и отчество преподавателя. Должность преподавателя. Дисциплины, которые может вести преподаватель.

Задание 1.1 (ЛР 1 БД). Выполните инфологическое моделирование базы данных системы. (Ограничения задать самостоятельно.)

Задание 1.2. Создайте логическую модель БД, используя ИЛМ (задание 1.1). Используйте необходимые средства поддержки целостности данных в СУБД.

Задание 2. Создать запросы:

- Вывести все номера групп и программы, где количество слушателей меньше 10.
- Вывести список преподавателей с указанием количества программ, где они преподавали за истекший год.

- Вывести список преподавателей, которые не проводят занятия на третьей паре ни в один из дней недели.
- Вывести список свободных лекционных аудиторий на ближайший понедельник.
- Вычислить общее количество обучающихся по каждой программе за последний год.
- Вычислить среднюю загруженность компьютерных классов в неделю за последний месяц (в часах).
- Найти самые популярные программы за последние 3 года.

Задание 3. Создать представление:

- для потенциальных слушателей, содержащее перечень специальностей, изучаемых на них дисциплин и количество часов;
- общих доход по каждой программе за последний год.

Задание 4. Создать хранимые процедуры:

- Для получения расписания занятий для групп на определенный день недели.
- Записи на курс слушателя.
- Получения перечня свободных лекционных аудиторий на любой день недели. Если свободных аудиторий не имеется, то выдать соответствующее сообщение.

Задание 5. Создать необходимые триггеры.

Ход работы:

Функция для получения расписания занятий для группы на определенный день недели. Дни недели указываются числами от 0 до 6 включительно.

```
CREATE OR REPLACE FUNCTION get_classes(group_id_param UUID, class_day_of_week
INT)
	RETURNS TABLE(class_id UUID) AS $$
BEGIN
	RETURN QUERY
	SELECT
	class.class_id
	FROM
	class
	INNER JOIN
	course_participant ON class.course_id = course_participant.course_id AND
course_participant.group_id = group_id_param
	WHERE
	EXTRACT(DOW FROM class.date) = class_day_of_week;
END;
$$ LANGUAGE plpgsql;
```

Процедура для записи слушателя на курс. Происходит поиск подходящей группы для клиента и его записывает в группу с наименьшим количеством участников.

```
CREATE OR REPLACE PROCEDURE assign_group_to_course(client_id UUID,
course_id_param UUID)
AS $$
DECLARE
	selected_group_id UUID;
	group_end_date DATE;
BEGIN
	SELECT g.group_id, g.end_date INTO selected_group_id, group_end_date
	FROM "group" g
	INNER JOIN course_participant cp ON g.group_id = cp.group_id AND cp.course_id =
course_id_param
	LEFT JOIN student s ON g.group_id = s.group_id
	WHERE g.start_date <= CURRENT_DATE AND g.end_date >= CURRENT_DATE
	GROUP BY g.group_id, g.end_date
	ORDER BY COUNT(s.student_id) ASC
	LIMIT 1;

	IF selected_group_id IS NULL THEN
	RAISE EXCEPTION 'No suitable group found for the course.';
	END IF;

	INSERT INTO student (student_id, client_id, group_id, start_date, finish_date,
status_in_group)
	VALUES (gen_random_uuid(), client_id, selected_group_id, CURRENT_DATE,
group_end_date, 'учится');
END;
$$ LANGUAGE plpgsql;
```

Функция для получения перечня свободных лекционных аудиторий на любой день недели. Если свободных аудиторий не имеется, то выдать соответствующее сообщение. Возвращает свободные аудитории по времени, дате и типу, так как одна и та же аудитория в один день недели может быть занята, а в другой свободна.

```
CREATE OR REPLACE FUNCTION get_free_auditory(date DATE, type AUDITORY_TYPE
DEFAULT 'лекционная', start_order INT, end_order INT)
	RETURNS TABLE(classroom_id UUID) AS $$
BEGIN
	RETURN QUERY
	SELECT
	classroom.classroom_id
	FROM
	classroom
	WHERE
		classroom.type = type
	AND NOT EXISTS (
	SELECT 1
	FROM class
	WHERE class.classroom_id = classroom.classroom_id
	AND class.date = date
	AND class.classes_order_number BETWEEN start_order AND end_order
	);
END;
$$ LANGUAGE plpgsql;
```

В рамках четвёртой лабораторной работы были созданы триггеры для проверки того что аудитория может вместить все студентов, что аудитория свободно в определённый момент, что учитель может проводить занятия и так далее.

Проверка соответствия дат в группе и потоке.

```
CREATE OR REPLACE FUNCTION check_group_dates()
RETURNS TRIGGER AS $$
DECLARE
  enrollment_start_date DATE;
  enrollment_end_date DATE;
BEGIN
SELECT start_date, end_date INTO enrollment_start_date, enrollment_end_date
FROM enrollment
WHERE enrollment_id = NEW.enrollment_id;


IF NEW.start_date < enrollment_start_date THEN
        RAISE EXCEPTION 'Group start date must be on or after the enrollment start date.';
END IF;
IF NEW.end_date > enrollment_end_date THEN
        RAISE EXCEPTION 'Group end date must be on or before the enrollment end date.';
END IF;


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

Тоже самое, но для студента.

```
CREATE OR REPLACE FUNCTION check_student_dates()
RETURNS TRIGGER AS $$
DECLARE
  group_start_date DATE;
  group_end_date DATE;
BEGIN
SELECT start_date, end_date INTO group_start_date, group_end_date
FROM enrollment
WHERE enrollment_id = NEW.enrollment_id;


IF NEW.start_date < group_start_date THEN
        RAISE EXCEPTION 'Student start date must be on or after the group start date.';
END IF;
IF NEW.end_date > group_end_date THEN
        RAISE EXCEPTION 'Student end date must be on or before the group end date.';
END IF;


RETURN NEW;
END;
```

$$ LANGUAGE plpgsql;

Проверка типов оценок. Чтобы нельзя была перепутать и поставить оценка, которая должна была бы быть в зачете в экзамен и наоборот.

```
CREATE OR REPLACE FUNCTION check_mark()
RETURNS TRIGGER AS $$
DECLARE
  programm_discipline_id UUID;
  discipline_attestation_type ATTESTATION_TYPE;
BEGIN
SELECT discipline_id INTO programm_discipline_id
FROM programm_element
WHERE programm_element_id = NEW.programm_element_id;


SELECT attestation_type INTO discipline_attestation_type
FROM academic_discipline
WHERE discipline_id = programm_discipline_id;


IF NEW.mark IS NULL THEN
 RETURN NEW;
END IF;
IF discipline_attestation_type = 'зачёт' AND NEW.mark NOT IN ('yes', 'no') THEN
 RAISE EXCEPTION 'If attestation type is зачёт then allowed marks are only yes and no';
END IF;
IF discipline_attestation_type IN ('дифференциальный','экзамен') AND NEW.mark NOT IN
('2', '3', '4', '5') THEN
 RAISE EXCEPTION 'If attestation type is дифференциальный or экзамен then allowed
marks are only 2, 3, 4, 5';
END IF;


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

```
CREATE OR REPLACE FUNCTION check_result_date()
RETURNS TRIGGER AS $$
DECLARE
   student_start_date DATE;
BEGIN
SELECT start_date INTO student_start_date
FROM student
WHERE student_id = NEW.student_id;

IF NEW.date < student_start_date THEN
       RAISE EXCEPTION 'Result date must be on or after the student start
date.';
END IF;
```

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

CREATE OR REPLACE FUNCTION check_result_attempts()
RETURNS TRIGGER AS $$
BEGIN
IF NEW.attempt = 0 THEN
   RETURN NEW;
END IF;

IF EXISTS (SELECT 1 FROM result WHERE programm_element_id =
NEW.programm_element_id AND student_id = NEW.student_id AND attempt =
NEW.attempt - 1) THEN
   RETURN NEW;
END IF;

RAISE EXCEPTION 'Previous attempt is not exists.';
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION check_class()
RETURNS TRIGGER AS $$
DECLARE
   auditory_type AUDITORY_TYPE;
   discipline_discipline_id UUID;
   group_start_date DATE;
   groups_with_late_start INT;
   classroom_capacity INT;
   total_students INT;
BEGIN

SELECT type INTO auditory_type
FROM classroom
WHERE classroom_id = NEW.classroom_id;

IF auditory_type <> 'лекционная' and NEW.type = 'лекция' THEN
  RAISE EXCEPTION 'Classroom type is wrong for this type of class.';
END IF;

SELECT discipline_id INTO discipline_discipline_id
FROM programm_element
WHERE programm_element_id = (
   SELECT programm_element_id
   FROM course
   WHERE course_id = NEW.course_id
);

IF NOT EXISTS (SELECT 1 FROM teaching_permit WHERE discipline_id =
discipline_discipline_id AND (NEW.type <> 'лекция' OR is_lecture_allowed =
TRUE) LIMIT 1) THEN
   RAISE EXCEPTION 'Selected teacher cant be assigned to this class.';
END IF;

SELECT COUNT(*) INTO groups_with_late_start
FROM "group"
WHERE group_id IN (
   SELECT group_id
   FROM course_participant
   WHERE course_id = NEW.course_id
```

```
) AND start_date > NEW.date;

IF groups_with_late_start <> 0 THEN
  RAISE EXCEPTION 'Some of groups has start date after class date.';
END IF;

SELECT capacity INTO classroom_capacity
FROM classroom
WHERE classroom_id = NEW.classroom_id;

SELECT COUNT(*) INTO total_students
FROM student
WHERE group_id IN (
   SELECT group_id
   FROM course_participant
   WHERE course_id = NEW.course_id AND status_in_group = 'учится'
);

IF classroom_capacity < total_students THEN
   RAISE EXCEPTION 'Classroom is too small to fit all students.';
END IF;

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

Вывод

В ходе лабораторной работы я научился создавать триггеры, процедуры и функции.