ЛАБОРАТОРНАЯ РАБОТА №3

ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В POSTGRESQL

Выполнил: РЫБАЛКО ОЛЕГ ДМИТРИЕВИЧ К32392

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

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, SQL Shell (psql).

Практическое задание:

1. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4).
2. Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL (согласно индивидуальному заданию, часть 5). Допустимо создать универсальный триггер или отдельные триггеры на логирование действий.

Задание 1:

1. Создание хранимой процедуры для получения расписания занятий для групп на определенный день недели.

```
CREATE OR REPLACE FUNCTION GetScheduleByDayOfWeek(groupId INT, dayOfWeek INT)
  RETURNS TABLE (Идентификатор INTEGER)
AS $$
BEGIN
      RETURN QUERY
      SELECT Идентификатор
      FROM Занятие as a
      INNER JOIN Группа as b
      ON a.ИдентификаторГруппы = b.Идентификатор
      WHERE extract(isodow from a.ДатаПроведения) = dow
      AND b.Идентификатор = groupId;
END
$$ LANGUAGE plpgsql;
```

2. Создание хранимой процедуры для записи слушателя на курс

```
CREATE OR REPLACE PROCEDURE EnrollListenerToCourse(
    IN listenerId INTEGER,
    IN courseId INTEGER
)
AS $$
BEGIN
          INSERT INTO Обучается (ИдентификаторГруппы, ДатаНачалаОбучения,
ДатаОкончанияОбучения, Статус, ИдентификаторСлушателя)
      VALUES (
          (SELECT Идентификатор FROM Группа WHERE КодНабора = courseId),
          (SELECT ДатаНачала FROM НаборНаПрограмму WHERE Код = courseId),
          (SELECT ДатаОкончания FROM НаборНаПрограмму WHERE Код = courseId),
          'Учится',
          listenerId
      );
END
$$ LANGUAGE plpgsql;
```

3. Получения перечня свободных лекционных аудиторий на любой день недели. Если свободных аудиторий не имеется, то выдать соответствующее сообщение.

```
CREATE OR REPLACE PROCEDURE GetFreeLectureRoomsByDayOfWeek(IN dayOfWeek INT)
AS $$
DECLARE
    freeRooms TEXT[];
BEGIN
    SELECT ARRAY_AGG(Аудитория.Номер)
    INTO freeRooms
    FROM Аудитория
    WHERE Аудитория.Номер NOT IN (
        SELECT DISTINCT Занятие.НомерАудитории
        FROM Занятие
        WHERE EXTRACT(DOW FROM Занятие.ДатаПроведения) = dayOfWeek
    );

    IF freeRooms IS NULL OR array_length(freeRooms, 1) = 0 THEN
        RAISE NOTICE 'Свободных аудиторий не найдено';
    ELSE
        RAISE NOTICE 'Свободные аудитории: %', freeRooms;
    END IF;
END
$$ LANGUAGE plpgsql;
```

Задание 2.

```
CREATE OR REPLACE FUNCTION LogEvent()
  RETURNS TRIGGER AS $$
DECLARE
    tableName TEXT;
BEGIN
    tableName := TG_TABLE_NAME;

    IF TG_OP = 'INSERT' THEN
        INSERT INTO Лог (Таблица, Событие, Время)
        VALUES (tableName, 'Вставка', CURRENT_TIMESTAMP);
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO Лог (Таблица, Событие, Время)
        VALUES (tableName, 'Удаление', CURRENT_TIMESTAMP);
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO Лог (Таблица, Событие, Время)
        VALUES (tableName, 'Редактирование', CURRENT_TIMESTAMP);
    END IF;

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

```
CREATE OR REPLACE FUNCTION CreateLogTrigger()
  RETURNS VOID AS $$
DECLARE
    tableName TEXT;
    triggerName TEXT;
BEGIN
    FOR tableName IN SELECT table_name FROM information_schema.tables WHERE
table_schema = 'public' AND table_type = 'BASE TABLE' LOOP
        triggerName := 'LogEventTrigger_' || tableName;
        EXECUTE 'CREATE TRIGGER ' || triggerName || '
            AFTER INSERT OR DELETE OR UPDATE
            ON ' || tableName || '
            FOR EACH ROW
            EXECUTE FUNCTION LogEvent()';
    END LOOP;
END;
$$ LANGUAGE plpgsql;

SELECT CreateLogTrigger();
DROP TRIGGER LogEventTrigger_Лог ON Лог;
```

Таблица	Событие	Время
Обучается	Вставка	2023-05-24 21:47:27.044492
Занятие	Вставка	2023-05-24 21:51:26.373817
Занятие	Вставка	2023-05-24 21:51:34.704473
Занятие	Удаление	2023-05-24 21:52:06.503109
Занятие	Редактирование	2023-05-24 21:52:15.806375

Вывод:

После выполнения данной лабораторной работы я стал лучше понимать, как работают процедуры и триггеры в базе данных PostgreSQL. Также, я на практике смог применить свои знания, создав несколько процедур и триггеров для своей базы данных