Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет безопасности информационных технологий

Дисциплина:

«Информационная безопасность баз данных»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

«Функции и триггеры в БД»

Выполнили:

Чу Ван Доан, студент группы N3247

(подпись)

Проверил:

Волков А.Г.

(отметка о выполнении)

(подпись)

Санкт-Петербург

2024 г.

Содержание

1 Функции и триггеры в БД

1.1 Цель работы

Получение навыков написание процедур, функций и триггеров в БД.

1.2 Задание

1.2.1 Написать процедуру, которая выполняет агрегации значений в таблице и обновляет значение в другой таблице. Таким образом, чтобы при запуске пользователем информация в таблице обновлялась и содержала агрегированные значения из другой таблицы**.**

1.2.2 Написать триггер, который будет выполнять действие из 1 пункта автоматически при вставке записи в исходную таблицу. Таким образом, чтобы агрегированная информация всегда была актуальна**.**

1.2.3 Написать триггер, который на основании даты из вставляемой записи, вставлял ее в соответствующую таблицу.

1.2.4 Написать триггер, который при вставке в таблицу, производил подмену вставляемого значения в соответствии с уже существующим словарем.

1.2.5 Написать процедуру выводящую сумму первого, последнего и значений записей в таблице, находящихся в позициях золотого сечения.

1.3 Ход работы

1.3.1 Задание 1

Из лабы 2 у нас есть таблица students:

```
n3247_22=# select * from students;
 id_student | student_name | student_group | student_point
------------+--------------+---------------+---------------
          1 | student_1    | group_1       |            64
          3 | student_3    | group_3       |            99
          4 | student_4    | group_2       |            75
          5 | student_5    | group_4       |           100
          6 | student_6    | group_2       |            69
(5 rows)
```

Рисунок 1 – Таблица students

Создадим таблицу, содержащую среднее значение оценки всех студентов.

```
n3247_22=# create table average (avg_point real);
CREATE TABLE
n3247_22=# insert into average values(0);
INSERT 0 1
n3247_22=# select * from average;
 avg_point
-----------
         0
(1 row)
```

Рисунок 2 – Создание таблицы average

Создадим процедуру, которая определяет среднее значение оценки всех студентов в таблице students и обновляет значение в таблице average.

```
n3247_22=# CREATE OR REPLACE FUNCTION avg_value() RETURNS real AS $$ DECLARE avg_p real;
n3247_22$# BEGIN avg_p := (SELECT avg(student_point) FROM students);
n3247_22$# UPDATE average SET avg_point = avg_p;
n3247_22$# RETURN avg_p;
n3247_22$# END $$ LANGUAGE plpgsql;
CREATE FUNCTION
```

Рисунок 3 – Создание процедуры avg_value()

```
n3247_22=# SELECT * FROM avg_value();
 avg_value
-----------
      81.4
(1 row)

n3247_22=# SELECT * FROM average;
 avg_point
-----------
      81.4
(1 row)
```

Рисунок 4 – Результат процедуры avg_value()

1.3.2 Задание 2

Создадим функцию для триггера:

```
n3247_22=# CREATE OR REPLACE FUNCTION trigger_update_avgvalue() RETURNS trigger AS
$$
DECLARE
avg_p real;
BEGIN
IF NEW.student_point > 60
THEN avg_p := (SELECT avg(student_point) FROM students);
UPDATE average SET avg_point = avg_p;
END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
```

Рисунок 5 – Создание функции для триггера

Создадим оператор определения триггера:

```
n3247_22=# CREATE TRIGGER trigger_keep_avgvalue
AFTER INSERT OR UPDATE ON students FOR EACH ROW
EXECUTE PROCEDURE trigger_update_avgvalue();
CREATE TRIGGER
```

Рисунок 6 – Создание оператора определения триггера

Проверяем триггер путем добавить или обновить записи в таблице students.

```
n3247_22=# INSERT INTO students VALUES (2, 'student_2', 'group_1', 95);
INSERT 0 1
n3247_22=# SELECT * FROM average;
 avg_point
-----------
 83.666664
(1 row)

n3247_22=# UPDATE students SET student_point = 74 WHERE id_student = 2;
UPDATE 1
n3247_22=# SELECT * FROM average;
 avg_point
-----------
 80.166664
(1 row)
```

Рисунок 7 – Проверка триггера

1.3.3 Задание 3

Создадим таблицу, в которой сохранится информации об комнатах.

```
n3247_22=# CREATE TABLE room (id INT PRIMARY KEY, length INT, width INT, height INT, time DATE);
CREATE TABLE
```

Рисунок 8 – Создание таблицы room

Создадим функцию для триггера:

```
n3247_22=# CREATE OR REPLACE FUNCTION check_time_room() RETURNS TRIGGER AS
$$
BEGIN
EXECUTE 'CREATE TABLE IF NOT EXISTS date_' || TO_CHAR(NEW.time, 'YYYY_MM_DD') ||
' (id INT PRIMARY KEY, length INT, width INT, height INT)';
EXECUTE 'INSERT INTO date_' || TO_CHAR(NEW.time, 'YYYY_MM_DD') || ' (id, length, width, height) VALUES ($1, $2, $3, $4)'
USING NEW.id, NEW.length, NEW.width, NEW.height;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
```

Рисунок 9 – Создание функции триггера

Создадим оператор определения триггера:

```
n3247_22=# CREATE OR REPLACE TRIGGER trigger_check_room
n3247_22-# AFTER INSERT OR UPDATE ON room FOR EACH ROW
n3247_22-# EXECUTE PROCEDURE check_time_room();
CREATE TRIGGER
```

Рисунок 10 – Создание оператора определения триггера

Проверяем триггер путем добавить или обновить записи в таблице room:

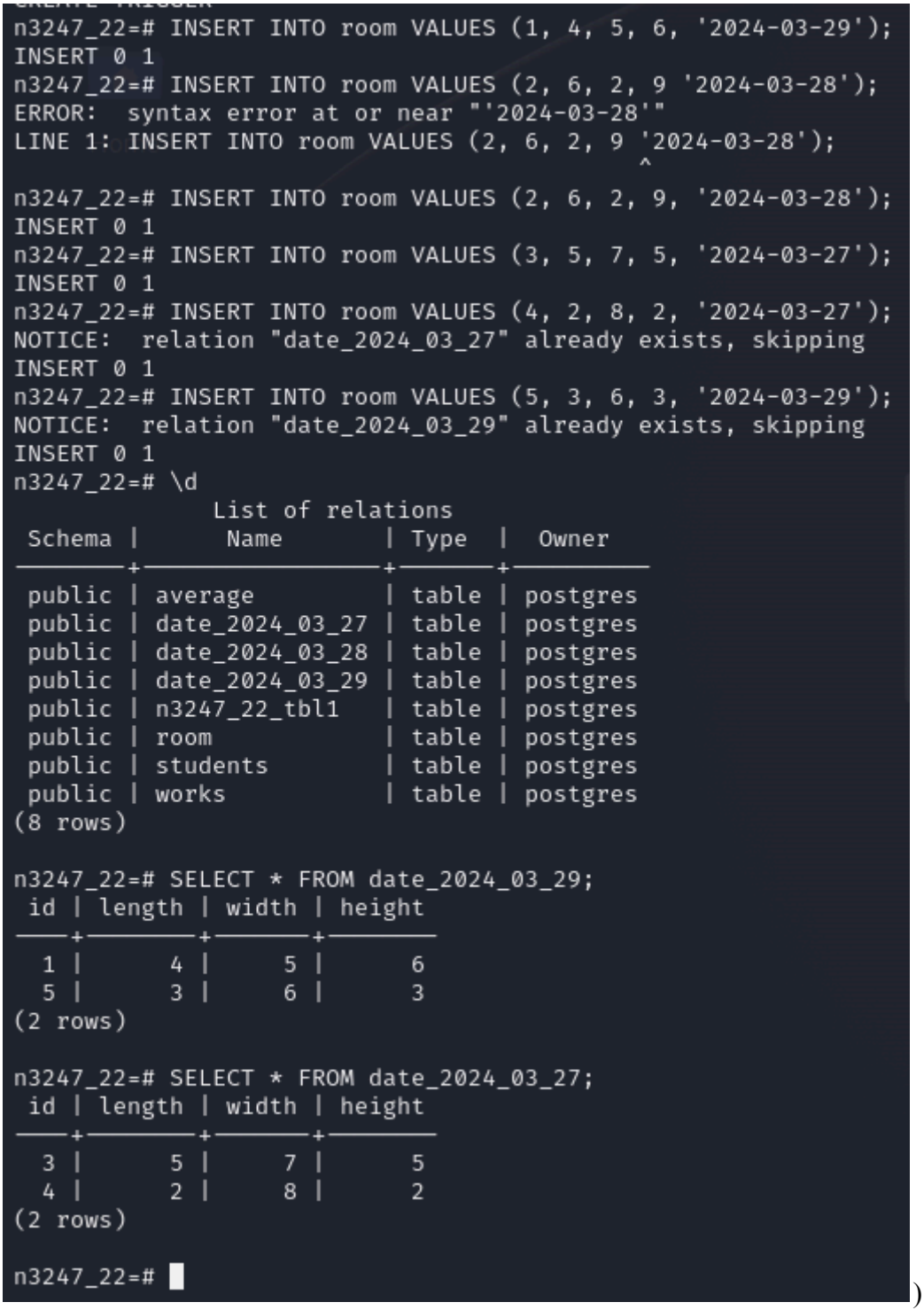

Рисунок 11 – Проверка триггера

1.3.4 Задание 4

Создадим словарь:

```
n3247_22=# CREATE TABLE dictionary (id INT PRIMARY KEY, english VARCHAR(100), russian VARCHAR(100));
CREATE TABLE
n3247_22=# INSERT INTO dictionary VALUES (1, 'table', 'стол'), (2, 'school', 'школа'), (3, 'teacher', 'учитель'), (4, 'pen', 'ручка');
INSERT 0 4
n3247_22=# SELECT * FROM dictionary;
 id | english | russian
----+---------+---------
  1 | table   | стол
  2 | school  | школа
  3 | teacher | учитель
  4 | pen     | ручка
(4 rows)

n3247_22=#
```

Рисунок 12 – Создание словари

Создадим таблицу russian_to_english:

```
n3247_22=# CREATE TABLE russian_to_english (id BIGSERIAL, word VARCHAR(100));
CREATE TABLE
```

Рисунок 13 – Создание таблицы russian_to_english

Создадим функцию для триггера и оператор определения триггера:

```
n3247_22=# CREATE OR REPLACE FUNCTION translate_word() RETURNS trigger AS
$$
BEGIN
IF NEW.word IN (SELECT russian FROM dictionary)
THEN NEW.word = (SELECT english FROM dictionary
WHERE dictionary.russian = NEW.word);
END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
```

```
n3247_22=# CREATE TRIGGER trigger_translate_word
BEFORE INSERT ON russian_to_english FOR EACH ROW
EXECUTE PROCEDURE translate_word();
CREATE TRIGGER
```

Рисунок 14 – Создание таблицы russian_to_english и триггера на ней

Проверяем триггер путем добавить или обновить записи в таблице russian_to_english.

```
n3247_22=# INSERT INTO russian_to_english (word) VALUES ('школа');
INSERT 0 1
n3247_22=# SELECT * FROM russian_to_english;
 id |  word
----+--------
  1 | school
(1 row)

n3247_22=# INSERT INTO russian_to_english (word) VALUES ('ручка');
INSERT 0 1
n3247_22=# SELECT * FROM russian_to_english;
 id |  word
----+--------
  1 | school
  2 | pen
(2 rows)
```

Рисунок 15 – Проверка триггера

1.3.5 Задание 5

Создадим таблицу student_point и вставим записи в нее:

```
n3247_22=# CREATE TABLE student_point (id BIGSERIAL PRIMARY KEY, point INT);
CREATE TABLE
n3247_22=# INSERT INTO student_point (point) VALUES (78), (93), (65), (96), (74), (95), (67);
INSERT 0 7
n3247_22=# SELECT * FROM student_point;
 id | point
----+-------
  1 |    78
  2 |    93
  3 |    65
  4 |    96
  5 |    74
  6 |    95
  7 |    67
(7 rows)
```

Рисунок 16 – – Создание таблицы и вставка записей

Напишем процедуру выводящую сумму первого, последнего и значений записей в таблице, находящихся в позициях золотого сечения.

```
n3247_22=# CREATE FUNCTION total() RETURNS INT AS
$$
DECLARE
sum_ INT;
count_ INT;
gold_index INT;
BEGIN
SELECT COUNT(*) INTO count_ FROM student_point;
gold_index := ROUND(count_ * 0.618);
SELECT SUM(point) INTO sum_ FROM student_point
WHERE id = 1 OR id = count_ OR id = gold_index;
RETURN sum_;
END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
n3247_22=# SELECT total();
 total
-------
   241
(1 row)
```

Рисунок 17 – Создание процедуры total

Заключение

В ходе выполнения лабораторной работы был изучен теоретический материал по функциям и триггерам в БД на языке SQL. Приобретенные знания были применены на практике в СУБД PostgreSQL.