Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Отчёт по лабораторной работе

Дисциплина: Базы данных

Тема: Триггеры, вызовы процедур

Выполнили студенты гр. 43501/3 Крылов И.С.

Преподаватель Мяснов А.В

«_____» _____2019 г.

Санкт-Петербург
2019

1 Цель работы.

Познакомить студентов с возможностями реализации более сложной обработки данных на стороне сервера с помощью хранимых процедур и триггеров.

2 Программа работы

1. Создание двух триггеров: один триггер для автоматического заполнения ключевого поля, второй триггер для контроля целостности данных в подчиненной таблице при удалении/изменении записей в главной таблице.

2. Создание триггера в соответствии с индивидуальным заданием, полученным у преподавателя.

3. Создание триггера в соответствии с индивидуальным заданием, вызывающего хранимую процедуру.

4. Выкладывание скрипта с созданными сущностями в GitLab.

5. Демонстрация результатов преподавателю.

3 Ход работы

В соответствии с индивидуальным заданием было создано два триггера. База была заполненна тестовыми данными с помощью генератора.

3.1 Индивидальное задание

Триггеры:

1. При создании новой игры проверять нет ли игры у того же разрабочика с названием, отличающимся по дистанции Левенштейна более, чем на 3. Если есть - выбрасывать исключение и не давать добавлять.

2. При формировании аппаратных требований к игре не должно быть ситуации, чтобы минимальные требования превышали оптимальные. Если производися попытка добавить или изменить требования так, что может привести к невыполнению требования, то выбрасывать исключение.

3.2 Добавление новой игры

Был разработан sql-скрипт, создающий триггер, который активируется на добавление нового элемента в таблицу game и проверяет наличие игры того же разработчика с названием, отличающимся по дистанции Левенштейна менее чем на 3. Если такая игра присутсвует - выбрасывается исключение о невозможности добавить запись в таблицу.

```
CREATE OR REPLACE FUNCTION game_name_leven_check()
    returns trigger
LANGUAGE plpgsql
AS $$
DECLARE
    conflicts varchar [];
BEGIN
    conflicts := ARRAY(
        SELECT name FROM game WHERE (new.id_developer =
   id_developer) AND (levenshtein(new.name :: text, name :: text)
    > 3)
    );
    IF array_length(conflicts, 1) > 0 THEN
        RAISE EXCEPTION 'Levenshtein value is greater than 3.
   Conflicts: %', conflicts;
    END IF;
    RETURN new;
END;
$$;

CREATE TRIGGER game_name_leven_check
    BEFORE INSERT OR UPDATE
    ON game
    FOR EACH ROW EXECUTE PROCEDURE game_name_leven_check();
```

Созданный триггер был проверен на нескольких запросах. Можно утверждать, что триггер работает правильно.

3.3 Добавление новых системных требований

Проверяем наличие ситуации когда минимальные требования превышают оптимальные. При обнаружении такой ситуации выбрасываем исключение.

```
CREATE OR REPLACE FUNCTION sys_req_check()
    returns trigger
LANGUAGE plpgsql
AS $$
DECLARE
    conflicts BIGINT [];
BEGIN
    conflicts := ARRAY(
        SELECT id_system_requirements from system_requirements
   where (select cpu.cores from minimal_requirements join cpu
```

```
    using (id_cpu) where id_minimal_requirements = new.
    id_minimal_requirements) > (select cpu.cores from
    optimal_requirements join cpu using (id_cpu) where
    id_optimal_requirements = new.id_optimal_requirements)
10    );
11    IF array_length(conflicts, 1) > 0 THEN
12        RAISE EXCEPTION 'Incorrect format: %', conflicts;
13    END IF;
14    RETURN new;
15 END;
16 $$;
17
18 CREATE TRIGGER sys_req_check
19    BEFORE INSERT OR UPDATE
20    ON system_requirements
21    FOR EACH ROW EXECUTE PROCEDURE sys_req_check();
```

4 Вывод

В ходе лабораторной работы были улучшены навыки работы с sql и получены навыки разработки триггеров. Решения типа хранимых процедур и триггеров, позволяют перенести часть вычислений на сторону sql сервера. Учитывая, что он с подобными задачами справляется быстрее и при этом затрачивая меньшие ресурсы подобная практика считается хорошей. Главным недостатоком использования триггеров является то, что про них можно забыть и тогда разобраться, что происходит в базе данных может оказаться трудным.