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: Гнеушев В. А.

Факультет: ИКТ

Группа: К3239

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Оглавление

Цель работы

Овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

Практическое задание

Вариант 1 (max - 6 баллов)

1. Создать процедуры/функции 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.

Вариант 2 (max - 8 баллов)

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

2.1. Модифицировать триггер (триггерную функцию) на проверку корректности входа и выхода сотрудника

2.2. Создать авторский триггер по варианту индивидуального задания.

Описание проекта

В этом отчете я буду описывать функции и триггер для спроектированной и созданной мной на работе базе данных.

Система, для которой создавалась база данных – это обучающая платформа, интегрированная в Telegram бота. У пользователей платформы есть 3 роли – ученик, учитель и администратор.

Ученики получают доступ к образовательной платформе, оплатив обучение и авторизовавшись в Telegram боте при помощи почты, указанной при оплате.

К отчету прилагается модель .”erwin” базы данных, с которой я работал.

Функции

Функция 1 – Получение суммарной чистой прибыли ученика за весь курс

```
CREATE OR REPLACE FUNCTION student_net_income(telegram_id VARCHAR)
RETURNS DECIMAL AS
$code$
    SELECT COALESCE(SUM(investment - revenue), 0)
    FROM user_report
    WHERE telegram_user_id = student_net_income.telegram_id;
$code$
LANGUAGE SQL;
```

```
postgres=# select id, student_net_income(id) as income from telegram_user order by income desc;
     id      | income
-------------+--------
 243816260   |    550
 987654321   |      0
 818181818   |      0
 243816261   |      0
 243816262   |      0
 243816263   |      0
 243816264   |      0
 243816265   |      0
 243816266   |      0
 243816267   |      0
 243816268   |      0
 243816269   |      0
 2438162601  |      0
 2438162602  |      0
 2438162603  |      0
 2438162604  |      0
 2438162605  |      0
 2438162606  |      0
 2438162607  |      0
 2438162608  |      0
 2438162609  |      0
 24381626011 |      0
 24381626012 |      0
```

Функция 2 – Получение количества покупок курса

```sql
CREATE OR REPLACE FUNCTION amount_course_purchases() RETURNS
INTEGER AS
$code$
DECLARE
    total_count INTEGER;
BEGIN
    SELECT COUNT(email) INTO total_count
    FROM user_status_by_email
    WHERE status = 'active';

    RETURN total_count;
END;
$code$
LANGUAGE plpgsql;
```

```
postgres=# select amount_course_purchases();
 amount_course_purchases
-------------------------
                       2
(1 row)
```

Функция 3 – Удаление пользователя из админов (обнуление привилегий)

```
CREATE OR REPLACE PROCEDURE remove_admin_permissions(IN telegram_id
VARCHAR) AS
$code$
BEGIN
    UPDATE telegram_user SET role = NULL WHERE id = telegram_id and
role = 'admin';
END;
$code$
LANGUAGE plpgsql;
```

```
postgres=# select id, role from telegram_user where id = '243816260';
    id     | role
-----------+-------
 243816260 | admin
(1 row)


postgres=# call remove_admin_permissions('243816260');
CALL
postgres=# select id, role from telegram_user where id = '243816260';
    id     | role
-----------+------
 243816260 |
(1 row)
```

Триггер

Триггер будет отслеживать изменения в таблице админов “admin”.
Для этого создадим таблицу с логами этих изменений под названием “admin_changes_logs”.

```
CREATE TABLE admin_changes_logs(
    event_at timestamp default (now() at time zone 'utc'),
    telegram_user_id VARCHAR(256),
    text TEXT,
    primary key ("event_at", "telegram_user_id"),
    foreign key ("telegram_user_id") references "telegram_user"
("id")
);
```

И создадим функцию “create_admin_table_changes_log”, добавляющую логи в эту таблицу при изменениях в таблице “admin”.

```
CREATE OR REPLACE FUNCTION create_admin_table_changes_log()
RETURNS TRIGGER AS
$code$
```

```
DECLARE
    mstr varchar(30);
    astr varchar(100);
    retstr varchar(254);
BEGIN
    IF    TG_OP = 'INSERT' THEN
        astr = NEW.telegram_user_id;
        mstr := 'Add new admin ';
        retstr := mstr||astr;
        INSERT INTO admin_changes_logs("telegram_user_id", "text")
values (astr, retstr);
        RETURN NEW;
    ELSIF TG_OP = 'UPDATE' THEN
        astr = NEW.telegram_user_id;
        mstr := 'Update admin ';
        retstr := mstr||astr;
        INSERT INTO admin_changes_logs("telegram_user_id", "text")
values (astr, retstr);
        RETURN NEW;
    ELSIF TG_OP = 'DELETE' THEN
        astr = OLD.telegram_user_id;
        mstr := 'Remove admin ';
        retstr := mstr || astr;
        INSERT INTO admin_changes_logs("telegram_user_id", "text")
values (astr, retstr);
        RETURN OLD;
    END IF;
END;
$code$
LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER admin_changes_logging
 AFTER INSERT OR UPDATE OR DELETE ON admin
 FOR EACH ROW
EXECUTE PROCEDURE create_admin_table_changes_log();
```

Сам триггер будет называться “admin_changes_logging” и срабатывать при добавлениях/изменениях/удалениях строк в таблице “admin”.

```
postgres=# select * from admin_changes_logs;
 event_at | telegram_user_id | text
----------+------------------+------
(0 rows)


postgres=# insert into admin("telegram_user_id") values ('123');
INSERT 0 1
postgres=# select * from admin_changes_logs;
          event_at          | telegram_user_id |       text
----------------------------+------------------+------------------
 2024-01-16 03:16:13.519117 | 123              | Add new admin 123
(1 row)
```

```
postgres=# delete from admin;
DELETE 1
postgres=# select * from admin_changes_logs;
          event_at          | telegram_user_id |       text
----------------------------+------------------+-------------------
 2024-01-16 03:18:16.079883 | 123              | Add new admin 123
 2024-01-16 03:18:31.692162 | 123              | Remove admin 123
(2 rows)
```

Вывод

В ходе данной лабораторной работы я научился создавать и использовать функции и триггеры в PostgreSQL.