МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет безопасности информационных технологий

Направление подготовки: 10.03.01 Информационная безопасность

Образовательная программа: "Информационная безопасность / Information security"

Дисциплина:

«*Информационная безопасность баз данных*»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

«Функции и триггеры в БД»

Выполнил студент:

N3246 / ИББД N23 1.3

Суханкулиев Мухаммет /_____

ФИО *Подпись*

Проверила:

Карманова Наталия Андреевна /_____

ФИО *Подпись*

_____

Отметка о выполнении (один из вариантов: отлично, хорошо, удовлетворительно, зачтено)

_____

Дата

Санкт-Петербург

2025 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы – Получение навыков написания процедур, функций и триггеров в БД.

Для достижения поставленной цели необходимо решить следующие **задачи**:

1. Написать процедуру, которая выполняет агрегацию 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.
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.
3. Написать триггер, который на основании даты из вставляемой записи, вставлял ее в соответствующую таблицу.
4. Написать триггер, который при вставке в таблицу, производил подмену вставляемого значения в соответствии с существующим словарем.
5. Реализовать триггер, который использует по крайней мере 2–3 специальных переменных (NEW, OLD, TG_OP и др.).

1 ФУНКЦИИ И ТРИГГЕРЫ В БД

Будет использована база данных со второй лабораторной работы (`marvel`).

1.1 Ход работы

1.1.1 Написание процедуры

Напишем процедуру, которая будет считать количество супергероев в каждой команде и обновлять таблицу `teams`, добавляя в неё это значение.

Создадим новый столбец в `teams` для хранения количества героев:

```
ALTER TABLE teams ADD COLUMN hero_count INT DEFAULT 0;

ALTER TABLE
Query returned successfully in 78 msec.
```

Создадим процедуру для обновления этого значения:

```
CREATE OR REPLACE FUNCTION update_hero_count()
RETURNS VOID AS $$
BEGIN
    UPDATE teams
    SET hero_count = (
        SELECT COUNT(*) FROM heroes WHERE heroes.team_id = teams.id
    );
END;
$$ LANGUAGE plpgsql;

CREATE FUNCTION
Query returned successfully in 98 msec.
```

Запуск процедуры вручную:

```
SELECT update_hero_count();

Successfully run. Total query runtime: 55 msec. 1 rows affected.
```

1.1.2 Написание триггера 1

Теперь создадим триггер, который будет автоматически обновлять количество героев в teams при добавлении нового героя.

Создадим триггерную функцию:

```
CREATE OR REPLACE FUNCTION trigger_update_hero_count()
RETURNS TRIGGER AS $$
BEGIN
    PERFORM update_hero_count();
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE FUNCTION
Query returned successfully in 62 msec.
```

Привяжем триггер к таблице `heroes`:

```
CREATE TRIGGER hero_insert_trigger
AFTER INSERT OR DELETE ON heroes
FOR EACH STATEMENT
EXECUTE FUNCTION trigger_update_hero_count();

CREATE TRIGGER
Query returned successfully in 81 msec.
```

Запуск: добавляем нового героя в `heroes`, после чего `teams.hero_count` автоматически обновится.

```
INSERT INTO heroes (id, name, real_name, team_id, debut_year)
VALUES (6, 'Thor', 'Thor Odinson', 1, 1962);

INSERT 0 1 Query returned successfully in 103 msec.

marvel=# select * from teams;
 id |          name           |       base       | hero_count
----+-------------------------+------------------+------------
  1 | Avengers                | Stark_Tower      |          3
  2 | X-Men                   | Xavier_Institute |          0
  3 | Guardians_of_the_Galaxy | Space            |          1
  5 | S.H.I.E.L.D.            | Helicarrier      |          0
  4 | Fantastic_Four          | Baxter_Building  |          0
(5 строк)
```

1.1.3 Написание триггера 2

Допустим, у нас есть таблицы `heroes_20th_century` и `heroes_21st_century`, и мы хотим, чтобы новые записи распределялись по ним в зависимости от года дебюта.

Создадим таблицы:

```
CREATE TABLE heroes_20th_century AS TABLE heroes WITH NO DATA;
CREATE TABLE heroes_21st_century AS TABLE heroes WITH NO DATA;

CREATE TABLE AS
Query returned successfully in 58 msec.
```

Напишем триггерную функцию:

```
CREATE OR REPLACE FUNCTION trigger_distribute_heroes()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.debut_year < 2000 THEN
        INSERT INTO heroes_20th_century VALUES (NEW.*);
    ELSE
        INSERT INTO heroes_21st_century VALUES (NEW.*);
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE FUNCTION
Query returned successfully in 49 msec.
```

Применим триггер:

```
CREATE TRIGGER distribute_hero_trigger
```

```
AFTER INSERT ON heroes
FOR EACH ROW
EXECUTE FUNCTION trigger_distribute_heroes();

CREATE TRIGGER
Query returned successfully in 44 msec.
```

Попробуем «стриггерить»:

```
INSERT INTO heroes (id, name, real_name, team_id, debut_year)
VALUES (7, 'Black Panther', 'TChalla', 1, 1966);

INSERT INTO heroes (id, name, real_name, team_id, debut_year)
VALUES (8, 'Ms. Marvel', 'Kamala Khan', 3, 2013);

INSERT 0 1 Query returned successfully in 73 msec.

marvel=# SELECT * FROM heroes_20th_century;
 id |     name      |  real_name   | team_id | debut_year
----+---------------+--------------+---------+------------
  6 | Thor          | Thor Odinson |       1 |       1962
  7 | Black Panther | TChalla      |       1 |       1966
(2 строки)

marvel=# SELECT * FROM heroes_21st_century;
 id |    name    |  real_name  | team_id | debut_year
----+------------+-------------+---------+------------
  8 | Ms. Marvel | Kamala Khan |       3 |       2013
(1 строка)
```

1.1.4 Написание триггера 3

Допустим, у нас есть таблица powers, где названия способностей должны храниться в правильном формате, и мы хотим, чтобы при вставке, например, `spider_sense` автоматически заменялось на `Spider Senses`.

Создадим таблицу-словарь:

```
CREATE TABLE power_dict (
    incorrect_name TEXT PRIMARY KEY,
    correct_name TEXT NOT NULL
);

INSERT INTO power_dict VALUES ('spider_sense', 'Spider Senses');

CREATE TABLE
Query returned successfully in 80 msec.
```

Напишем триггерную функцию:

```
CREATE OR REPLACE FUNCTION trigger_correct_power_name()
RETURNS TRIGGER AS $$
DECLARE
    corrected_name TEXT;
BEGIN
    SELECT correct_name INTO corrected_name FROM power_dict
    WHERE incorrect_name = NEW.name;

    IF corrected_name IS NOT NULL THEN
        NEW.name := corrected_name;
    END IF;
```

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

CREATE FUNCTION
Query returned successfully in 53 msec.
```

Создадим триггер:

```
CREATE TRIGGER correct_power_name_trigger
BEFORE INSERT ON powers
FOR EACH ROW
EXECUTE FUNCTION trigger_correct_power_name();

CREATE TRIGGER
Query returned successfully in 49 msec.
```

Попробуем «стриггерить»:

```
INSERT INTO powers (id, name) VALUES (8, 'spider_sense');

INSERT 0 1 Query returned successfully in 47 msec.

marvel=# SELECT * FROM powers;
 id |     name
----+--------------
  1 | Genius
  2 | Billionaire
  3 | Playboy
  4 | Superpower
  5 | Regeneration
  6 | Accurate
  7 | Spider_senses
  8 | Spider Senses
(8 строк)
```

1.1.5 Реализация триггера 4

Этот триггер будет логировать изменения в таблице heroes в таблицу heroes_log.

Создадим таблицу логов:

```
CREATE TABLE heroes_log (
    log_id SERIAL PRIMARY KEY,
    hero_id INT,
    operation TEXT,
    old_name TEXT,
    new_name TEXT,
    change_time TIMESTAMP DEFAULT now()
);

CREATE TABLE
Query returned successfully in 51 msec.
```

Напишем триггерную функцию:

```
CREATE OR REPLACE FUNCTION trigger_log_hero_changes()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'UPDATE' THEN
        INSERT INTO heroes_log (hero_id, operation, old_name, new_name)
        VALUES (OLD.id, 'UPDATE', OLD.name, NEW.name);
```

```
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO heroes_log (hero_id, operation, old_name)
        VALUES (OLD.id, 'DELETE', OLD.name);
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE FUNCTION
Query returned successfully in 54 msec.
```

Создадим триггер:

```
CREATE TRIGGER log_hero_changes_trigger
AFTER UPDATE OR DELETE ON heroes
FOR EACH ROW
EXECUTE FUNCTION trigger_log_hero_changes();

CREATE TRIGGER
Query returned successfully in 70 msec.
```

Попробуем «стриггерить»:

```
UPDATE heroes SET name = 'Iron Man Mk II' WHERE id = 1;

UPDATE 1 Query returned successfully in 44 msec.
```

```
marvel=# SELECT * FROM heroes_log;
 log_id | hero_id | operation | old_name |    new_name    |        change_time
--------+---------+-----------+----------+----------------+---------------------------
      1 |       1 | UPDATE    | Iron_Man | Iron Man Mk II | 2025-03-31 21:00:29.807017
(1 строка)
```

```
DELETE FROM heroes WHERE id = 2;

DELETE 1 Query returned successfully in 47 msec.
```

```
marvel=# SELECT * FROM heroes_log;
 log_id | hero_id | operation |    old_name     |    new_name    |        change_time
--------+---------+-----------+-----------------+----------------+---------------------------
      1 |       1 | UPDATE    | Iron_Man        | Iron Man Mk II | 2025-03-31 21:00:29.807017
      2 |       2 | DELETE    | Captain_America |                | 2025-03-31 21:01:15.313875
(2 строки)
```

ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы были изучены и реализованы функции и триггеры в базе данных PostgreSQL. Разработана процедура для агрегирования данных и их обновления в таблице, а также триггеры, выполняющие автоматическое обновление информации, распределение данных по таблицам, подмену значений на основе словаря и логирование изменений.

Работа позволила получить практические навыки написания хранимых функций и триггеров, закрепить понимание механизма их работы, а также освоить использование специальных переменных. Реализованные примеры продемонстрировали, как можно автоматизировать обработку данных, обеспечивая целостность и актуальность информации в базе данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. НИУ ИТМО Методические указания по проведению лабораторной работы №3 – Функции и триггеры в БД: электронный ресурс – 2025. – URL: https://docs.google.com/document/d/1kqAYB0UlQE2FlESJ6gzO2SCr4_MY-8kb
2. Основы технологий баз данных: учебное пособие / Б. А. Новиков, Е. А. Горшкова, Н. Г. Графеева; под ред. Е. В. Рогова. — 2-е изд. — М.: ДМК Пресс, 2020. — 582 с. – URL: https://drive.google.com/file/d/1TjYbunEjxsbovBiHeYOzBuZrOFonlIRk/view?usp=drive_link
3. Базы данных: Учебник для высших учебных заведений / Под ред. проф. А. Д. Хомоненко. — 6-е изд., доп. - СПб.: КОРОНА-Век, 2009. – 736 с. – URL: https://drive.google.com/file/d/1zIOuO6vdQvb_aVUHGAiucK5MPciUswGf/view?usp=drive_link
4. Документация PostgreSQL: Documentation: 9.5: PostgreSQL 9.5.25 Documentation – электронный ресурс – 2025. – URL: https://www.postgresql.org/docs/9.5/index.html