САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ПЕТРА ВЕЛИКОГО

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе №6

Дисциплина: «Базы данных»

Тема: «SQL-программирование: триггеры, вызовы процедур»

Выполнил студент гр. 43501/3 _____ А.Ю. Ламтев
(подпись)

Преподаватель _____ А.В. Мяснов
(подпись)

«___» _____ 2019 г.

Санкт-Петербург
2019

Содержание

1. Цели работы

Познакомиться 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.

3. Триггер для автоматического заполнения ключевого поля

В листинге 1 представлен триггер, автоматически заполняющий поле с первичным ключом при добавлении записи в таблицу `language`, а также продемонстрирована его работа.

```
CREATE OR REPLACE FUNCTION language_pk()
  RETURNS TRIGGER
  LANGUAGE plpgsql
AS
$$
BEGIN
  SELECT nextval('language_id_seq') INTO NEW.id;
  RETURN NEW;
END;
$$;

CREATE TRIGGER language_pk
  BEFORE INSERT
  ON language
  FOR EACH ROW
EXECUTE PROCEDURE language_pk();

SELECT *
FROM language
WHERE id = (SELECT MAX(id)
            FROM language);
-- id | name
----------------
-- 3  | es-ES

INSERT INTO language
VALUES (DEFAULT, 'it-IT');

SELECT *
FROM language
WHERE id = (SELECT MAX(id)
            FROM language);
-- id | name
----------------
```

```
-- 4  | it-IT
```

Листинг 1: pk-trigger.sql

4. Триггер для контроля целостности данных

В листинге 2 представлен триггер, который позволяет поддерживать целостность данных в подчиненной таблице при удалении записей в главной таблице.

```
CREATE TABLE IF NOT EXISTS player
(
  id    SERIAL PRIMARY KEY,
  name TEXT NOT NULL
);

CREATE TABLE IF NOT EXISTS team
(
  id          SERIAL PRIMARY KEY,
  name        TEXT    NOT NULL,
  captain_id INTEGER NOT NULL REFERENCES player
);

CREATE OR REPLACE FUNCTION on_player_removed()
  RETURNS TRIGGER
  LANGUAGE plpgsql
AS
$$
BEGIN
  IF (TG_OP = 'DELETE') THEN
    DELETE FROM team WHERE captain_id = OLD.id;
    RETURN OLD;
  END IF;
  RETURN NULL;
END;
$$;

CREATE TRIGGER on_player_removed
  BEFORE UPDATE OR DELETE
  ON player
  FOR EACH ROW
EXECUTE PROCEDURE on_player_removed();

INSERT INTO player (name)
VALUES ('Rashford');

INSERT INTO team (name, captain_id)
VALUES ('Manchester United', 1);

SELECT *
FROM team;
-- id|          name       | captain_id
-------------------------------------------
-- 1 | Manchester United |          1

DELETE
FROM player
where name = 'Rashford';

SELECT *
FROM team;

-- Empty table
```

Листинг 2: fk-consistency-trigger.sql

После удаления записи из таблицы `player`, первичный ключ которой

является внешним ключом для таблицы `team`, также удаляется соответствующая запись из таблицы `team`.

5. Триггер, вызывающий ХП при добавлении новой подписки пользователю

Задание: При добавлении новой подписки пользователю вызвать процедуру, которая на основании данных о подписках пользователя и о популярности фильмов по категориям формирует список предложений для заданного пользователя. Данная процедура `up_to_n_movie_recommendations_for_user(n SMALLINT, user_id BIGINT)` была разработана в рамках предыдущей лабораторной работы.

```
CREATE OR REPLACE FUNCTION on_subscription_added()
  RETURNS TRIGGER
  LANGUAGE plpgsql
AS
$$
BEGIN
  PERFORM up_to_n_movie_recommendations_for_user(10, NEW.user_id);
  RETURN NEW;
END;
$$;

CREATE TRIGGER on_subscription_added
  AFTER INSERT
  ON subscription
  FOR EACH ROW
EXECUTE PROCEDURE on_subscription_added();

INSERT INTO subscription (user_id, started, expires, autorenewable, payment)
VALUES (123, now(), now(), TRUE, '$55.0');
```

Листинг 3: new-subscription-trigger.sql

6. Триггер, реализующий проверку на дубли при добавлении или изменении подписок

Задание: Реализовать проверку на дубли при добавлении или изменении подписок. В случае дубля выбрасывать исключение.

```
CREATE OR REPLACE FUNCTION check_for_duplicates()
  RETURNS TRIGGER
  LANGUAGE plpgsql
AS
$$
DECLARE
  subscription_user_id BIGINT;
  subscription_id      BIGINT;
BEGIN
  subscription_user_id := (SELECT user_id
                           FROM subscription
                           WHERE id = NEW.subscription_id
                           LIMIT 1);
  subscription_id :=
      (SELECT s.id
       FROM subscription s
              JOIN subscription_series_season sss
                   ON s.user_id = subscription_user_id
```

```
19          WHERE sss.series_season_id = NEW.series_season_id
20          LIMIT 1);
21    IF subscription_id IS NOT NULL THEN
22      RAISE EXCEPTION 'User (id=%) already has subscription (id=%) which contains series season
        (id=%)', subscription_user_id, subscription_id, NEW.series_season_id;
23    END IF;
24    RETURN NEW;
25  END;
26  $$;
27
28  CREATE TRIGGER check_for_duplicates
29    BEFORE INSERT OR UPDATE
30    ON subscription_series_season
31    FOR EACH ROW
32  EXECUTE PROCEDURE check_for_duplicates();
33
34  SELECT *
35  FROM subscription_series_season
36  LIMIT 1;
37
38  -- subscription_id | series_season_id
39  ------------------------------------
40  --        26         |       1123
41
42  SELECT user_id
43  FROM subscription
44  WHERE id = 26;
45
46  -- user_id
47  ------------
48  --  8733
49
50  INSERT INTO subscription (user_id, started, expires, autorenewable, payment)
51  VALUES (8733, now(), now(), TRUE, '$55.00');
52
53  SELECT *
54  FROM subscription
55  WHERE user_id = 8733
56    AND payment = '$55.00';
57  --    id    | user_id |    started    |    expires    | autorenewable | payment
58  ------------------------------------------------------------------------------
59  -- 160331 |    8733   | '2019-01-15' | '2019-01-15'|      TRUE       | '$55.00'
60
61  INSERT INTO subscription_series_season (subscription_id, series_season_id)
62  VALUES (160331, 1123);
63
64  -- ERROR: User (id=8733) already has subscription (id=18) which contains series season (id
        =1123)
65  -- Where: PL/pgSQL function check_for_duplicates() line 18 at RAISE
```

Листинг 4: check-for-duplicates.sql

Проверка на дубли осуществляется следующим образом. При добавлении в таблицу `subscription_series_season` новой пары «подписка – сезон сериала» происходит проверка всех подписок данного пользователя на наличие данного сезона сериала.

7. Выводы

В результате работы был изучен принцип работы триггеров, а также синтаксис, который позволяет их создавать, и затем было создано несколько триггеров, срабатывающих при добавлении, обновлении или удалении записей, удовлетворяющих определенным условиям для определенных таблиц.