

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	3
2	Программа работы	3
3	Триггер для автоматического заполнения ключевого поля	3
4	Триггер для контроля целостности данных	4
5	Триггер, вызывающий ХП при добавлении новой подписки пользователю	5
6	Триггер, реализующий проверку на дубли при добавлении или изменении подписок	5
7	Выводы	6

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.

```
1 CREATE OR REPLACE FUNCTION language_pk()  
2   RETURNS TRIGGER  
3   LANGUAGE plpgsql  
4 AS  
5 $$  
6 BEGIN  
7   SELECT nextval('language_id_seq') INTO NEW.id;  
8   RETURN NEW;  
9 END;  
10 $$;  
11  
12 CREATE TRIGGER language_pk  
13   BEFORE INSERT  
14   ON language  
15   FOR EACH ROW  
16   EXECUTE PROCEDURE language_pk();  
17  
18 SELECT *  
19 FROM language  
20 WHERE id = (SELECT MAX(id)  
21             FROM language);  
22 -- id | name  
23 -----  
24 -- 3  | es-ES  
25  
26 INSERT INTO language  
27 VALUES (DEFAULT, 'it-IT');  
28  
29 SELECT *  
30 FROM language  
31 WHERE id = (SELECT MAX(id)  
32             FROM language);  
33 -- id | name  
34 -----
```

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.

```

1 CREATE TABLE IF NOT EXISTS player
2 (
3   id SERIAL PRIMARY KEY,
4   name TEXT NOT NULL
5 );
6
7 CREATE TABLE IF NOT EXISTS team
8 (
9   id SERIAL PRIMARY KEY,
10  name TEXT NOT NULL,
11  captain_id INTEGER NOT NULL REFERENCES player
12 );
13
14 CREATE OR REPLACE FUNCTION on_player_removed()
15 RETURNS TRIGGER
16 LANGUAGE plpgsql
17 AS
18 $$
19 BEGIN
20   IF (TG_OP = 'DELETE') THEN
21     DELETE FROM team WHERE captain_id = OLD.id;
22     RETURN OLD;
23   END IF;
24   RETURN NULL;
25 END;
26 $$;
27
28 CREATE TRIGGER on_player_removed
29 BEFORE UPDATE OR DELETE
30 ON player
31 FOR EACH ROW
32 EXECUTE PROCEDURE on_player_removed();
33
34 INSERT INTO player (name)
35 VALUES ('Rashford');
36
37 INSERT INTO team (name, captain_id)
38 VALUES ('Manchester United', 1);
39
40 SELECT *
41 FROM team;
42
43  id | name | captain_id
44  ---|-----|-----
45  1 | Manchester United | 1
46
47 DELETE
48 FROM player
49 where name = 'Rashford';
50
51 SELECT *
52 FROM team;
53
54 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.

```
1 CREATE OR REPLACE FUNCTION on_subscription_added()
2   RETURNS TRIGGER
3   LANGUAGE plpgsql
4 AS
5 $$
6 BEGIN
7   PERFORM up_to_n_movie_recommendations_for_user(10, NEW.user_id);
8   RETURN NEW;
9 END;
10 $$;
11
12 CREATE TRIGGER on_subscription_added
13   AFTER INSERT
14   ON subscription
15   FOR EACH ROW
16 EXECUTE PROCEDURE on_subscription_added();
17
18 INSERT INTO subscription (user_id, started, expires, autorenewable, payment)
19 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.

```
1 CREATE OR REPLACE FUNCTION check_for_duplicates()
2   RETURNS TRIGGER
3   LANGUAGE plpgsql
4 AS
5 $$
6 DECLARE
7   subscription_user_id BIGINT;
8   subscription_id      BIGINT;
9 BEGIN
10  subscription_user_id := (SELECT user_id
11                           FROM subscription
12                           WHERE id = NEW.subscription_id
13                           LIMIT 1);
14  subscription_id :=
15    (SELECT s.id
16     FROM subscription s
17     JOIN subscription_series_season sss
18     ON s.user_id = subscription_user_id
```

```

19         AND s.id = sss.subscription_id
20         WHERE sss.series_season_id = NEW.series_season_id
21         LIMIT 1);
22 IF subscription_id IS NOT NULL THEN
23     RAISE EXCEPTION 'User (id=%) already has subscription (id=%) which contains series season
24     (id=%)', subscription_user_id, subscription_id, NEW.series_season_id;
25 END IF;
26 RETURN NEW;
27 $$;
28
29 CREATE TRIGGER check_for_duplicates
30 BEFORE INSERT OR UPDATE
31 ON subscription_series_season
32 FOR EACH ROW
33 EXECUTE PROCEDURE check_for_duplicates();
34
35 SELECT *
36 FROM subscription_series_season
37 LIMIT 1;
38
39 -- subscription_id | series_season_id
40 -----
41 --          26      |          1123
42
43 SELECT user_id
44 FROM subscription
45 WHERE id = 26;
46
47 -- user_id
48 -----
49 --    8733
50
51 INSERT INTO subscription (user_id, started, expires, autorenewable, payment)
52 VALUES (8733, now(), now(), TRUE, '$55.00');
53
54 SELECT *
55 FROM subscription
56 WHERE user_id = 8733
57     AND payment = '$55.00';
58 --   id   | user_id | started      | expires      | autorenewable | payment
59 -----
60 -- 160331 |   8733  | '2019-01-15' | '2019-01-15' | TRUE          | '$55.00'
61
62 INSERT INTO subscription_series_season (subscription_id, series_season_id)
63 VALUES (160331, 1123);
64
65 -- ERROR: User (id=8733) already has subscription (id=18) which contains series season (id
66 -- =1123)
67 -- 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.