

Санкт-Петербургский Политехнический Университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

ОТЧЕТ
по лабораторной работе
«Триггеры, вызовы процедур»
Базы данных

Работу выполнил студент

группа 43501/3 Дьячков В.В.

Работу принял преподаватель

_____ Мясов А.В.

Санкт-Петербург

2018

Содержание

1	Цель работы	3
2	Программа работы	3
3	Хранимые процедуры	3
3.1	Автоматическое заполнение ключевого поля	3
3.2	Контроль целостности данных в подчиненной таблице	4
3.3	Расчет стоимости бронирования	5
3.4	Проверка доступности номера	6
4	Выводы	7

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. Хранимые процедуры

3.1. Автоматическое заполнение ключевого поля

Создадим триггер для автоматического заполнения ID в таблице `room_type`. Внутри триггера, для наглядности, будем брать следующее значение последовательности `room_type_id_seq` и прибавлять 100.

```
1 CREATE OR REPLACE FUNCTION room_type_primary_key()
2     RETURNS TRIGGER
3 LANGUAGE plpgsql
4 AS $$
5 BEGIN
6     SELECT nextval('room_type_id_seq') + 100 INTO new.id;
7     RETURN new;
8 END;
9 $$;
10
11 CREATE TRIGGER room_type_primary_key
12     BEFORE INSERT
13     ON room_type
14     FOR EACH ROW EXECUTE PROCEDURE room_type_primary_key();
15
16 SELECT *
17 FROM room_type
18 WHERE id = (SELECT MAX(id) FROM room_type);
19 -- id, type, capacity, description
20 -- 9, Studio, 4, A room with a studio bed - a couch that can be converted into a bed
21
```

```

22 INSERT INTO room_type
23 VALUES (DEFAULT, 'Twin', 2, 'A room with two beds');
24
25 SELECT *
26 FROM room_type
27 WHERE id = (SELECT MAX(id) FROM room_type);
28 -- id, type, capacity, description
29 -- 110, Twin, 2, A room with two beds

```

Листинг 1: primary_key_trigger.sql

Видно, что триггер сработал и заполнил значение ключевого поля.

3.2. Контроль целостности данных в подчиненной таблице

Контроль целостности внешних ключей в подчиненных таблицах обеспечивается ограничениями `constraints` внешних ключей.

```

1 UPDATE room SET id = 5009 WHERE id = 1212
2 -- [23503] ERROR: update or delete on table "room" violates foreign key constraint "
   reservation_room_id_fkey" on table "reservation"
3 -- Detail: Key (id)=(1212) is still referenced from table "reservation".
4
5 DELETE FROM room WHERE id = 1212
6 -- [23503] ERROR: update or delete on table "room" violates foreign key constraint "
   reservation_room_id_fkey" on table "reservation"
7 -- Detail: Key (id)=(1212) is still referenced from table "reservation".

```

Листинг 2: foreign_key_trigger.sql

Тем не менее, можно создать триггер для контроля логической целостности данных. Создадим триггер, который будет удалять отзыв о бронировании при появлении записи об отмене этого заказа (добавлении записи в таблицу `cancellation`).

```

1 CREATE OR REPLACE FUNCTION review_cancellation()
2     RETURNS TRIGGER
3 LANGUAGE plpgsql
4 AS $$
5 BEGIN
6     DELETE FROM review
7     WHERE review.reservation_id = new.reservation_id;
8     RETURN new;
9 END;
10 $$;
11
12 CREATE TRIGGER review_cancellation
13     AFTER INSERT
14     ON cancellation
15     FOR EACH ROW EXECUTE PROCEDURE review_cancellation();

```

Листинг 3: review_cancellation.sql

Попробуем добавить бронирование и отзыв к нему. После этого добавим запись об отмене бронирования и убедимся, что соответствующий отзыв был автоматически удален.

```

1 INSERT INTO reservation
2 VALUES (DEFAULT, 3, 1, '2017-01-01', '2017-01-02', 1234, TRUE);
3
4 SELECT * FROM reservation
5 WHERE id = (SELECT MAX(id) FROM reservation);
6 -- id,      room_id, user_id, from,      to,      price,      is_paid
7 -- 100044, 3,      1,      2017-01-01, 2017-01-02, "$2,951.00", true
8
9 INSERT INTO review
10 VALUES (DEFAULT, 100044, 'Advantages', 'Disadvantages', 5);
11
12 SELECT * FROM review
13 WHERE reservation_id = 100044;
14 -- id,      reservation_id, advantages, disadvantages, rating
15 -- 30001, 100044,      Advantages, Disadvantages, 5
16
17 INSERT INTO cancellation
18 VALUES (DEFAULT, 100044, 'COMPLETED');
19
20 SELECT * FROM review
21 WHERE reservation_id = 100044;
22 -- id,      reservation_id, advantages, disadvantages, rating

```

Листинг 4: review_cancellation_example.sql

Стоит отметить, что в реальной системе вместо удаления, скорее всего, использовалась бы пометка отзыва специальным флагом, который бы указывал на то, что этот отзыв не надо учитывать при расчете различных показателей (например, средней оценки отеля).

3.3. Расчет стоимости бронирования

Задание: Расчет стоимости бронирования по данным стоимости номеров в разные моменты времени

С помощью функции `generate_series` сгенерируем набор дат, на которые забронирован номер (исключим последний, так как будем считать что день выезда не входит в стоимость бронирования).

Список сгенерированных дат соединим (`JOIN`) с таблицей цен по условию: ID комнат совпадают и сгенерированная дата попадает в интервал дат нужной цены. Тогда сумма цен (`price`) будет являться рассчитанной ценой данного бронирования.

```

1 CREATE OR REPLACE FUNCTION calculate_price()
2   RETURNS trigger
3   LANGUAGE plpgsql
4   AS $$
5   BEGIN
6     SELECT SUM(price.price) INTO new.price
7     FROM generate_series(new."from", new."to" - INTERVAL '1 day', '1 day') AS date
8     JOIN price ON price.room_id = 1 AND date BETWEEN "from" AND "to";
9     RETURN new;
10  END;
11  $$;
12

```

```

13 CREATE TRIGGER calculate_price
14     BEFORE INSERT OR UPDATE
15     ON reservation
16     FOR EACH ROW EXECUTE PROCEDURE calculate_price();

```

Листинг 5: calculate_price.sql

Попробуем добавить бронирование, не указывая цену, причем для наглядности возьмем даты бронирования на границе период действия цен.

```

1 SELECT * FROM price
2 WHERE room_id = 1;
3 -- id,      from,      to,      price,      room_id
4 -- 35019, 2017-01-01, 2017-11-08, "$2,951.00", 1
5 -- 35020, 2017-11-09, 2018-04-26, "$2,666.00", 1
6 -- 35021, 2018-04-27, 2019-01-26, "$6,319.00", 1
7
8 INSERT INTO reservation
9 VALUES (DEFAULT, 1, 1, '2018-04-20', '2018-04-30', NULL, TRUE);
10
11 SELECT * FROM reservation
12 WHERE id = (SELECT MAX(id) FROM reservation);
13 -- id,      room_id, user_id, from,      to,      price,      is_paid
14 -- 100031, 1,      1,      2018-04-20, 2018-04-30, "$37,619.00", true

```

Листинг 6: calculate_price_example.sql

Видно, что цена 10-дневного бронирования была рассчитана верно:

$$7 \text{ дней} \cdot \$2,666 + 3 \text{ дня} \cdot \$6,319 = \$37,619,00.$$

3.4. Проверка доступности номера

Задание: При добавлении бронирования проверять доступность номера, в случае недоступности – выбрасывать исключение.

Для проверки доступности номера на весь период бронирования выполним подзапрос для поиска конфликтующих бронирований (с тем же номером и указанным условием на даты). После этого, если список конфликтующих бронирований не пуст, то бросим исключение, содержащее поясняющее сообщение и список.

```

1 CREATE OR REPLACE FUNCTION reservation_validate_availability()
2     RETURNS trigger
3 LANGUAGE plpgsql
4 AS $$
5 DECLARE
6     conflicts BIGINT [];
7 BEGIN
8     conflicts := ARRAY(
9         SELECT id
10        FROM reservation
11        WHERE id != new.id AND room_id = new.room_id
12              AND "to" > new."from" AND "from" < new."to"
13    );
14     IF array_length(conflicts, 1) > 0 THEN
15         RAISE EXCEPTION 'Room is not available. Conflicts: %', conflicts;
16     END IF;

```

```

17 RETURN new;
18 END;
19 $$;
20
21 CREATE TRIGGER reservation_validate_availability
22 BEFORE INSERT OR UPDATE
23 ON reservation
24 FOR EACH ROW EXECUTE PROCEDURE reservation_validate_availability();

```

Листинг 7: reservation_validate_availability.sql

Попробуем добавить бронирование, конфликтующее по датам с бронированием, созданным в предыдущем пункте. Для наглядности попробуем несколько разных комбинаций: новое начинается до предыдущего и кончается в середине предыдущего; новое начинается до предыдущего и кончается после предыдущего и т.д.

```

1 SELECT * FROM reservation
2 WHERE id = (SELECT MAX(id) FROM reservation);
3 -- id,      room_id, user_id, from,      to,      price,      is_paid
4 -- 100031, 1,      1,      2018-04-20, 2018-04-30, "$37,619.00", true
5
6 INSERT INTO reservation
7 VALUES (DEFAULT, 1, 1, '2018-04-01', '2018-04-25', NULL, TRUE);
8 -- [P0001] ERROR: Room is not available. Conflicts: {100031}
9 -- Where: PL/pgSQL function reservation_validate_availability() line 14 at RAISE
10
11 INSERT INTO reservation
12 VALUES (DEFAULT, 1, 1, '2018-04-01', '2018-05-31', NULL, TRUE);
13 -- [P0001] ERROR: Room is not available. Conflicts: {26126,75734,100031}
14 -- Where: PL/pgSQL function reservation_validate_availability() line 14 at RAISE
15
16 INSERT INTO reservation
17 VALUES (DEFAULT, 1, 1, '2018-04-25', '2018-05-31', NULL, TRUE);
18 -- [P0001] ERROR: Room is not available. Conflicts: {26126,75734,100031}
19 -- Where: PL/pgSQL function reservation_validate_availability() line 14 at RAISE
20
21 INSERT INTO reservation
22 VALUES (DEFAULT, 1, 1, '2018-04-25', '2018-04-26', NULL, TRUE);
23 -- [P0001] ERROR: Room is not available. Conflicts: {100031}
24 -- Where: PL/pgSQL function reservation_validate_availability() line 14 at RAISE

```

Листинг 8: reservation_validate_availability_example.sql

Видно, что добавление конфликтующих бронирований было отклонено.

4. Выводы

В процессе выполнения данной работы:

- создан триггер для автоматического заполнения ключевого поля и сохранения целостности в связанной таблице;
- создан триггер для автоматического расчета стоимости бронирования;
- создан триггер для автоматической проверки доступности номера на указанные даты при добавлении нового бронирования.