

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего
образования «Санкт-Петербургский национальный исследовательский
университет информационных технологий, механики и оптики»

Мегафакультет трансляционных информационных технологий
Факультет инфокоммуникационных технологий

Дисциплина: Базы Данных

Отчет по Лабораторной работе №3

«ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В PostgreSQL»

Выполнила: Микулина Алиса Романовна

Группа: К32421, 2 курс

Преподаватель: Говорова Марина
Михайловна

Санкт-Петербург

2023

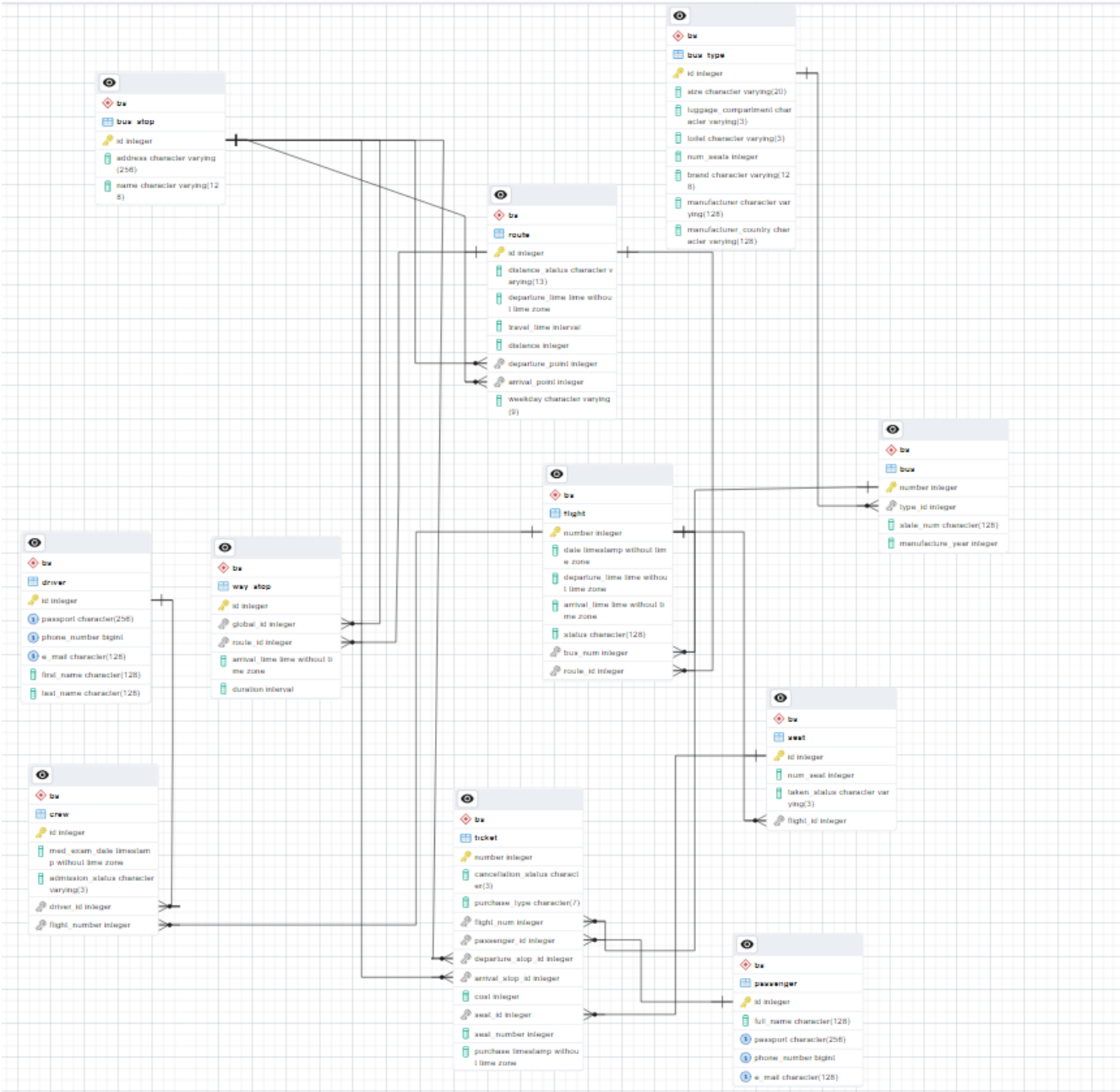
Цель работы: овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

Практическое задание:

1. Создать процедуры/функции согласно индивидуальному заданию и (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.

Ход выполнения работы:

Схема логической модели базы данных:



Задание 4

Создать хранимые процедуры:

- Продажи билета.

```
CREATE OR REPLACE PROCEDURE buy_ticket
(new_flight_num integer, new_passenger_id integer, new_departure_stop character varying(256),
new_arrival_stop character varying(256), new_purchase_type character(7), new_num_seat integer)
LANGUAGE SQL
AS $$
    UPDATE bs.seat SET taken_status = 'yes'
    WHERE num_seat = new_num_seat
    AND flight_id = new_flight_num;

    INSERT INTO bs.ticket (purchase_type, flight_num, passenger_id,
                           departure_stop_id, arrival_stop_id, cost,
                           seat_id, purchase)
    VALUES (new_purchase_type, new_flight_num, new_passenger_id,
            (SELECT id FROM bs.bus_stop WHERE address = new_departure_stop),
            (SELECT id FROM bs.bus_stop WHERE address = new_arrival_stop),
            (SELECT cost FROM bs.seat WHERE flight_id = new_flight_num AND num_seat = new_num_seat),
            (SELECT id FROM bs.seat WHERE flight_id = new_flight_num AND num_seat = new_num_seat),
            current_timestamp);
$$;
```

С помощью имеющейся функции просмотра билетов на завтра смотрим, на какие рейсы есть места. Давайте купим билет на первый завтрашний рейс и посмотрим, меняется ли количество доступных билетов на рейс.

До покупки:

	номер_рейса integer	откуда character varying (128)	куда character varying (128)	время_отправления time without time zone	количество_свободных_мест bigint
1	206	Общежитие на Вязьме	Красная площадь	07:00:00	37
2	1690	Общежитие на Вязьме	Общежитие на Белорусской	08:00:00	38
3	1691	Общежитие на Вязьме	Общежитие на Белорусской	10:00:00	36
4	76	Красная площадь	Общежитие на Вязьме	12:00:00	40
5	1692	Общежитие на Вязьме	Общежитие на Белорусской	12:00:00	28
6	1693	Общежитие на Вязьме	Общежитие на Белорусской	14:00:00	29
7	1694	Общежитие на Вязьме	Общежитие на Белорусской	16:00:00	36
8	1695	Общежитие на Вязьме	Общежитие на Белорусской	18:00:00	36
9	1696	Общежитие на Вязьме	Общежитие на Белорусской	20:00:00	31
10	1697	Общежитие на Вязьме	Общежитие на Белорусской	22:00:00	35

Количество доступных билетов изменилось, значит, билет продан!!!

	номер_рейса integer	откуда character varying (128)	куда character varying (128)	время_отправления time without time zone	количество_свободных_мест bigint
1	206	Общежитие на Вязьме	Красная площадь	07:00:00	36
2	1690	Общежитие на Вязьме	Общежитие на Белорусской	08:00:00	38
3	1691	Общежитие на Вязьме	Общежитие на Белорусской	10:00:00	36
4	76	Красная площадь	Общежитие на Вязьме	12:00:00	40
5	1692	Общежитие на Вязьме	Общежитие на Белорусской	12:00:00	28
6	1693	Общежитие на Вязьме	Общежитие на Белорусской	14:00:00	29
7	1694	Общежитие на Вязьме	Общежитие на Белорусской	16:00:00	36
8	1695	Общежитие на Вязьме	Общежитие на Белорусской	18:00:00	36
9	1696	Общежитие на Вязьме	Общежитие на Белорусской	20:00:00	31
10	1697	Общежитие на Вязьме	Общежитие на Белорусской	22:00:00	35

	number [PK] integer	cancellation_ character (3)	purchase_type character (7)	flight_num integer	passenger_id integer	departure_sta integer	arrival_stop_i integer	cost integer	seat_id integer	seat_number integer	purchase timestamp without time
13	6268	no	online	206	8290	143	144	4000	12637	25	2022-11-01 10:10:00
14	6269	yes	online	206	1869	143	144	4000	12639	27	2022-11-01 10:10:00
15	6270	no	offline	206	9913	143	144	4000	12643	31	2022-11-01 10:10:00
16	6271	yes	offline	206	811	143	144	4000	12646	34	2022-11-01 10:10:00
17	6272	no	online	206	8864	143	144	4000	12650	38	2022-11-01 10:10:00
18	6273	no	offline	206	1801	143	144	4000	12651	39	2022-11-01 10:10:00
19	6274	yes	online	206	46	143	144	4000	12652	40	2022-11-01 10:10:00
20	6275	no	offline	206	7999	143	144	4000	12653	41	2022-11-01 10:10:00
21	6276	yes	online	206	5401	143	144	4000	12657	45	2022-11-01 10:10:00
22	6277	yes	online	206	9257	143	144	4000	12660	48	2022-11-01 10:10:00
23	6278	yes	online	206	1844	143	144	4000	12661	49	2022-11-01 10:10:00
24	6279	yes	online	206	2870	143	144	4000	12666	54	2022-11-01 10:10:00
25	6280	yes	offline	206	2473	143	144	4000	12667	55	2022-11-01 10:10:00
26	6281	yes	offline	206	7233	143	144	4000	12668	56	2022-11-01 10:10:00
27	6282	no	offline	206	9430	143	144	4000	12672	60	2022-11-01 10:10:00
28	6283	yes	offline	206	5182	143	144	4000	12673	61	2022-11-01 10:10:00
29	6284	yes	offline	206	705	143	144	4000	12674	62	2022-11-01 10:10:00
30	6285	no	offline	206	8851	143	144	4000	12676	64	2022-11-01 10:10:00
31	6286	yes	offline	206	7360	143	144	4000	12679	67	2022-11-01 10:10:00
32	6287	no	online	206	547	143	144	4000	12680	68	2022-11-01 10:10:00
33	6288	yes	offline	206	2106	143	144	4000	12681	69	2022-11-01 10:10:00
34	61813	[null]	online	206	4906	143	144	4000	12614	2	2023-05-22 01:37:07...
Total rows: 34 of 34 Query complete 00:00:00.072 Ln 1, Col 4											

- Возврата билета.

```
CREATE OR REPLACE PROCEDURE return_ticket
```

```
(replace_ticket_num integer)
```

```
LANGUAGE SQL
```

```
AS $$
```

```
    UPDATE bs.ticket SET cancellation_status = 'yes'
```

```
    WHERE number = replace_ticket_num;
```

```
    UPDATE bs.seat SET taken_status = 'no'
```

```
    WHERE id = (SELECT seat_id FROM bs.ticket WHERE number = replace_ticket_num);
```

```
$$;
```

Вернем тот же билет, что купили в прошлый раз, чтобы проверить работу процедуры.

```
CALL return_ticket (61813)
```

Доступных билетов снова стало 37, а сам билет стал помечен, как отмененный!!!

	номер_рейса integer	откуда character varying (128)	куда character varying (128)	время_отправления time without time zone	количество_свободных_мест bigint
1	206	Общежитие на Вязьме	Красная площадь	07:00:00	37
2	1690	Общежитие на Вязьме	Общежитие на Белорусской	08:00:00	38
3	1691	Общежитие на Вязьме	Общежитие на Белорусской	10:00:00	36
4	76	Красная площадь	Общежитие на Вязьме	12:00:00	40
5	1692	Общежитие на Вязьме	Общежитие на Белорусской	12:00:00	28
6	1693	Общежитие на Вязьме	Общежитие на Белорусской	14:00:00	29
7	1694	Общежитие на Вязьме	Общежитие на Белорусской	16:00:00	36
8	1695	Общежитие на Вязьме	Общежитие на Белорусской	18:00:00	36
9	1696	Общежитие на Вязьме	Общежитие на Белорусской	20:00:00	31
10	1697	Общежитие на Вязьме	Общежитие на Белорусской	22:00:00	35

	number [PK] integer	cancellation_status character (3)	purchase_type character (7)	flight_num integer	passenger_id integer	departure_stop_id integer	arrival_stop_id integer	cost integer	seat_id integer	si ir
14	6269	yes	online	206	1869	143	144	4000	12639	
15	6270	no	offline	206	9913	143	144	4000	12643	
16	6271	yes	offline	206	811	143	144	4000	12646	
17	6272	no	online	206	8864	143	144	4000	12650	
18	6273	no	offline	206	1801	143	144	4000	12651	
19	6274	yes	online	206	46	143	144	4000	12652	
20	6275	no	offline	206	7999	143	144	4000	12653	
21	6276	yes	online	206	5401	143	144	4000	12657	
22	6277	yes	online	206	9257	143	144	4000	12660	
23	6278	yes	online	206	1844	143	144	4000	12661	
24	6279	yes	online	206	2870	143	144	4000	12666	
25	6280	yes	offline	206	2473	143	144	4000	12667	
26	6281	yes	offline	206	7233	143	144	4000	12668	
27	6282	no	offline	206	9430	143	144	4000	12672	
28	6283	yes	offline	206	5182	143	144	4000	12673	
29	6284	yes	offline	206	705	143	144	4000	12674	
30	6285	no	offline	206	8851	143	144	4000	12676	
31	6286	yes	offline	206	7360	143	144	4000	12679	
32	6287	no	online	206	547	143	144	4000	12680	
33	6288	yes	offline	206	2106	143	144	4000	12681	
34	61813	yes	online	206	4906	143	144	4000	12614	

- Добавления нового рейса.

Помимо добавления рейса, создадим автоматически добавление мест на этот рейс.

```
CREATE OR REPLACE FUNCTION add_flight
(new_date timestamp, new_bus_num integer, new_route_id integer, new_ticket_cost integer)
RETURNS VOID AS $$
DECLARE
    cnt integer = 1;
BEGIN
    INSERT INTO bs.flight (date, departure_time, arrival_time, bus_num, route_id)
    VALUES (new_date,
            (SELECT departure_time FROM bs.route WHERE id = new_route_id),
            (SELECT departure_time FROM bs.route WHERE id = new_route_id),
            new_bus_num, new_route_id);

    WHILE cnt <= (SELECT num_seats FROM bs.bus_type type
        JOIN bs.bus bus ON bus.type_id = type.id
        WHERE bus.number = new_bus_num)
    LOOP
        INSERT INTO bs.seat (num_seat, taken_status, flight_id, cost)
        VALUES (cnt, 'no',
            (SELECT number FROM bs.flight WHERE date = new_date AND route_id = new_route_id),
            new_ticket_cost);

        cnt = cnt + 1;
    END LOOP;
END;
$$ LANGUAGE plpgsql;
```

Попробуем добавить новый рейс на 2023-06-12, на утренний автобус.






```
SELECT add_flight('2023-06-12', 1267, 22, 100)
```

	number [PK] integer	date timestamp without time zone	departure_time time without time zone	arrival_time time without time zone	status character (128)	bus_num integer	route_id integer
1	1805	2023-06-12 00:00:00	08:00:00	08:00:00	[null]	1267	22

Рейс добавился!! Теперь смотрим, добавились ли места:

```
SELECT * FROM bs.seat WHERE flight_id = 1805
```

50 мест добавлено, значит, все работает!

	id [PK] integer 	num_seat integer 	taken_status character varying (3) 	flight_id integer 	cost integer 
1	123775	1	no	1805	100
2	123776	2	no	1805	100
3	123777	3	no	1805	100
4	123778	4	no	1805	100
5	123779	5	no	1805	100
6	123780	6	no	1805	100
7	123781	7	no	1805	100
8	123782	8	no	1805	100
9	123783	9	no	1805	100
10	123784	10	no	1805	100
11	123785	11	no	1805	100
12	123786	12	no	1805	100
13	123787	13	no	1805	100
14	123788	14	no	1805	100
15	123789	15	no	1805	100
16	123790	16	no	1805	100
17	123791	17	no	1805	100
18	123792	18	no	1805	100
19	123793	19	no	1805	100
20	123794	20	no	1805	100
21	123795	21	no	1805	100
22	123796	22	no	1805	100
Total rows: 50 of 50		Query complete 00:00:00.068			

Задание 5

- Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных

Создаем таблицу, в которой будут храниться данные о всех изменениях:

```
CREATE TABLE logs (  
    id serial PRIMARY KEY,  
    event_type varchar(20) NOT NULL,  
    table_name varchar(20) NOT NULL,  
    record_id integer NOT NULL,  
    event_time timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP  
)
```

Дальше создаем функцию, которая будет выполняться при срабатывании триггера:

```
CREATE OR REPLACE FUNCTION log_event()  
RETURNS TRIGGER AS $$  
BEGIN  
    IF TG_OP = 'INSERT' THEN  
        INSERT INTO logs (event_type, table_name, record_id)  
        VALUES ('INSERT', TG_TABLE_NAME, NEW.id);  
    ELSIF TG_OP = 'UPDATE' THEN  
        INSERT INTO logs (event_type, table_name, record_id)  
        VALUES ('UPDATE', TG_TABLE_NAME, NEW.id);  
    ELSIF TG_OP = 'DELETE' THEN  
        INSERT INTO logs (event_type, table_name, record_id)  
        VALUES ('DELETE', TG_TABLE_NAME, OLD.id);  
    END IF;  
    RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;
```

Далее создаем триггеры на таблицы, для которых нужно логировать события:

```
CREATE TRIGGER log_changes
AFTER INSERT OR UPDATE OR DELETE
ON bs.bus
FOR EACH ROW
EXECUTE PROCEDURE log_event();
```

Проверяем работу триггеров.

INSERT:

```
INSERT INTO bs.driver (passport, phone_number, e_mail, first_name, last_name)
VALUES ('0000 000000', 89036417779, 'mrjoulin@yandex.ru', 'Matthew', 'Ivanov')
```

Новая запись в таблице водителей появилась

	id [PK] integer	passport character (256)	phone_number bigint	e_mail character (128)	first_name character (128)	last_name character (128)
1	1037	0000 000000 ...	89036417779	mrjoulin@yandex.ru ...	Matthew ...	Ivanov ...

```
SELECT * FROM logs
```

В таблице логов также появилась информация!!!

	id [PK] integer	event_type character varying (20)	table_name character varying (20)	record_id integer	event_time timestamp without time zone
1	1	INSERT	driver	1037	2023-05-22 05:23:08.972071

UPDATE:

Поменяем номер телефона

```
UPDATE bs.driver SET phone_number = 89198358347 WHERE first_name = 'Matthew' AND last_name = 'Ivanov'
```

Запись изменена:

	id [PK] integer	passport character (256)	phone_number bigint	e_mail character (128)	first_name character (128)	last_name character (128)
1	1037	0000 000000 ...	89198358347	mrjoulin@yandex.ru ...	Matthew ...	Ivanov ...

В таблице логов тоже отобразилось изменение:

	id [PK] integer	event_type character varying (20)	table_name character varying (20)	record_id integer	event_time timestamp without time zone
1	1	INSERT	driver	1037	2023-05-22 05:23:08.972071
2	2	UPDATE	driver	1037	2023-05-22 05:28:02.974549

DELETE:

Удалим добавленного водителя из базы

DELETE FROM bs.driver WHERE first_name = 'Matthew' AND last_name = 'Ivanov'

Запись удалена

Query Query History Data Output Messages Notifications

DELETE 1

Query returned successfully in 55 msec.

Запись об этом также есть и в логах

	id [PK] integer	event_type character varying (20)	table_name character varying (20)	record_id integer	event_time timestamp without time zone
1	1	INSERT	driver	1037	2023-05-22 05:23:08.972071
2	2	UPDATE	driver	1037	2023-05-22 05:28:02.974549
3	3	DELETE	driver	1037	2023-05-22 05:30:40.756201

Вывод:

В процессе выполнения лабораторной работы я овладела навыками создания процедур, функций и триггеров с помощью PostgreSQL.