

Министерство науки и высшего образования
Российской Федерации Федеральное государственное
автономное образовательное учреждение высшего
образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

Отчет

по лабораторной работе №3

«ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В PostgreSQL»

по дисциплине «**Базы данных**»

Выполнила: Кириллова В.Е.

Факультет: ИКТ

Группа: K3240

Проверила: Говорова М. М.



УНИВЕРСИТЕТ ИТМО

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

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, SQL Shell (psql).

Практическое задание:

1. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4):
 - Для поиска билетов в заданный пункт назначения.
 - Создание новой кассы продажи билетов.
 - Определить расход топлива по всем маршрутам за истекший месяц.
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.

Вариант 8. БД «Аэропорт»

Описание предметной области: Необходимо обеспечить продажу билетов на нужный рейс, при отсутствии билетов (необходимого количества билетов) предложить билет на ближайший рейс.

БД должна содержать следующий минимальный набор сведений: Бортовой номер самолета. Тип самолета. Количество мест. Страна. Производитель. Грузоподъемность. Скорость. Дата выпуска. Налёт в часах. Дата последнего ремонта. Назначение самолета. Расход топлива. Код экипажа. Паспортные данные членов экипажа. Номер рейса. Дата вылета. Время вылета. Аэропорт вылета. Аэропорт назначения. Расстояние. Транзитные посадки (прилет, вылет, аэропорт, время в аэропорту). ФИО пассажира. Паспортные данные. Номер места. Тип места. Цена билета. Касса продажи билета (возможен электронный билет) (номер и адрес).

ВЫПОЛНЕНИЕ

1) Создать процедуры/функции согласно индивидуальному заданию


1. Создание новой кассы продажи билетов:

- Выполнение функции

Query Editor Query History

```
1 create procedure Создание_кассы
2 (
3     Адрес varchar
4 )
5 LANGUAGE SQL
6 AS $$
7 INSERT INTO Касса VALUES ((select max(Номер_кассы) from Касса), Адрес);
8 $$;
```

- Адреса касс до выполнения процедуры:



Vika/postgres@PostgreSQL 11 ▾

Query Editor

Query History

1



```
select * from Касса
```

Data Output

Explain

Messages

Notifications

	<div>Номер_кассы</div> <div>[PK] integer </div>	<div>Адрес</div> <div>character varying (20) </div>
1	1	ул. Пушкина 1
2	2	ул. Мира 12
3	3	Биржевая линия 14
4	4	ул.Мира 71
5	5	2-я Хабаровская 1
6	6	ул.Ленина 17

- Выполнения процедуры (Добавление нового адреса для кассы):

The screenshot shows a PostgreSQL Query Editor interface. At the top, the user is logged in as 'Vika/postgres@PostgreSQL 11'. Below the toolbar, the 'Query Editor' tab is active, displaying a single SQL statement: `1 call Создание_кассы('ул. Пушкина, 31')`. The 'Messages' tab is selected, showing the output: 'CALL' followed by 'Query returned successfully in 49 msec.'

- Новый список адресов:

The screenshot shows a PostgreSQL Query Editor interface. The 'Query Editor' tab is active, displaying the SQL statement: `1 select * from Касса`. The 'Data Output' tab is selected, showing a table with two columns: 'Номер_кассы [PK] integer' and 'Адрес character varying (20)'. The table contains 7 rows of data.

	Номер_кассы [PK] integer	Адрес character varying (20)
1	1	ул. Пушкина 1
2	2	ул. Мира 12
3	3	Биржевая линия 14
4	4	ул.Мира 71
5	5	2-я Хабаровская 1
6	6	ул.Ленина 17
7	7	ул. Пушкина, 31

2. Для поиска билетов в заданный пункт назначения:

- Создание функции для поиска билетов в пункт назначения:

```
1 CREATE OR REPLACE FUNCTION Поиск_билетов(Пункт_назначения varchar)
2 returns table
3 (
4     Номер_рейса integer,
5     Вылет timestamp,
6     Прилет timestamp,
7     Аэропорт varchar
8 )
9 language plpgsql
10 as $$
11 begin
12     return query
13     (
14         select Рейс.Номер_рейса, (Рейс.Дата_вылета + Время_вылета), (Рейс.Дата_прилета + Время_прилета), Название_аэропорта
15         from Информационное_табло
16         inner join Рейс on Рейс.Номер_рейса = Информационное_табло.Номер_рейса
17         where Страна_город_прибытия = Пункт_назначения
18     );
19 end;
20 $$;
```

Data Output Explain Messages Notifications

CREATE FUNCTION

Query returned successfully in 49 msec.

- Результат выполнения функции для поиска билетов:

Query Editor

Query History

1

select * from Поиск_билетов('Россия, Иркутск')

2

Data Output

Explain

Messages

Notifications

	Номер_рейса integer	Вылет timestamp without time zone	Прилет timestamp without time zone	Аэропорт character varying
1	591	2022-05-28 20:30:00	2022-06-01 00:40:00	Иркутск - 1
2	1000	2022-06-02 00:00:00	2022-06-03 00:00:00	Иркутск - 1

3. Определить расход топлива по всем маршрутам за истекший месяц:

- Создание функции для определения расхода топлива:

```
1 create or replace function Расход_топлива_за_месяц()
2 returns table
3 (
4     Расход double precision
5 )
6 language plpgsql
7 as $$
8 begin
9 return query
10 (
11 select sum(Время_в_воздухе * Расход_топлива) as Расход
12 from
13 (
14     select Время_в_воздухе, Марка_самолета
15     from
16     (
17         select (extract(epoch from (Время_полета - Пересадка)))/3600) as Время_в_воздухе, Бортовой_номер
18         from
19         (
20             select Рейс.Номер_рейса, Рейс.Бортовой_номер, (Дата_прилета + Время_прилета) - (Дата_вылета + Время_вылета) as Время_полета, sum(Дата_и_время_полета - Дата_и_время_прилета) as Время_прилета
21             inner join Рейс on Рейс.Номер_рейса = Информационное_табло.Номер_рейса
22             inner join Транзитные_посадки on Транзитные_посадки.Номер_рейса = Информационное_табло.Номер_рейса
23             where Дата_вылета > ('now'::date - '1 month'::interval)
24             group by Рейс.Номер_рейса, Время_полета
25         )
26         as Таблица
27     )
28     as Список
29     inner join Самолет on Самолет.Бортовой_номер = Список.Бортовой_номер
30 )
31 as Списочек
32 inner join Марка_самолета on Марка_самолета.Марка_самолета = Списочек.Марка_самолета
33 );
34 end;
35 $$;
```

Data Output Explain Messages Notifications

CREATE FUNCTION

Query returned successfully in 42 msec.

- Результат выполнения функции:

Query Editor Query History

```
1 select * from Расход_топлива_за_месяц()
```

Data Output Explain Messages Notifications

	Расход double precision	
1	1472.65	

2) Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL.

1. Триггер добавляет данные в таблицу Билет (добавляет новые билеты) при добавлении нового рейса:

```
Vika/postgres@PostgreSQL 11
Query Editor  Query History
1 create or replace function Добавление_рейса()
2 returns trigger
3 as $$
4 begin
5     if (TG_OP = 'INSERT')
6     then
7         for i in 1..(select Количество_мест from Марка_самолета where Марка_самолета = (select Марка_самолета from Самолет where Борт_номер = (select
8             insert into Билет(Номер_билета, Номер_рейса_б, Номер_места) values((select max(Номер_билета::integer) + 1 from Билет), new.Номер_рейса
9             end loop;
10        return null;
11    end if;
12 end;
13 $$ language 'plpgsql';
14
15 --create trigger Триггер_на_рейс_INSERT
16 --after insert on Рейс
17 --for each row execute procedure Добавление_рейса()
18
```

Data Output Explain Messages Notifications

CREATE FUNCTION

Query returned successfully in 54 msec.

• Добавляем новый рейс под номером 1000:

```
Query Editor  Query History
1 insert into Рейс(Номер_рейса, Номер_самолета, Дата_вылета, Статус, Дата_прилета, Тип_рейса, Борт_номер)
2 values(1000, 1, '2022-06-02', 'Ожидание регистрации', '2022-06-03', 'Пассажирский', 123)

```

Data Output Explain Messages Notifications

INSERT 0 1

Query returned successfully in 99 msec.

• Видим, что появился рейс под номером 1000:

Номер_рейса [PK] integer	Номер_самолета integer	Дата_вылета date	Статус character varying (20)	Дата_прилета date	Тип_рейса character varying (20)	Борт_номер integer
1	591	1 2022-05-28	В пути	2022-06-01	Пассажирский	123
2	1713	4 2022-05-28	Ожидание регистрации	2022-05-28	Пассажирский	123456
3	2368	3 2022-05-31	Завершен	2022-06-01	Пассажирский	12345
4	2637	5 2022-05-29	Ожидание регистрации	2022-05-29	Пассажирский	1234567
5	4679	2 2022-05-30	Завершен	2022-05-31	Пассажирский	1234
6	7375	6 2022-05-28	Ожидание регистрации	2022-05-29	Пассажирский	12345678
7	1000	1 2022-06-02	Ожидание регистрации	2022-06-03	Пассажирский	123

- В таблице Билет появились новые билеты на рейс 1000:

Query Editor Query History

1 **SELECT** * **FROM** Билет

Data Output Explain Messages Notifications

	Номер_билета [PK] character varying (20)	Номер_рейса_б integer	Номер_места integer	Цена_ integers
32	41	1000	12	
33	42	1000	13	
34	43	1000	14	
35	44	1000	15	
36	45	1000	16	
37	46	1000	17	
38	47	1000	18	
39	48	1000	19	

- Триггер обновляет данные в таблице Билет при изменении самолета в рейсе (если самолет меняется на больший – добавляются новые билеты, при изменении самолета на меньший – у «лишних» билетов ставится “null” в месте):

Query Editor Query History

```

1 create or replace function Обновление_рейса()
2 returns trigger
3 as $$
4 begin
5     if (TG_OP = 'UPDATE')
6     then
7         if ((select Количество_мест from Марка_самолета where Марка_самолета = (select Марка_самолета from Самолет where Бортс
8         then
9             update Билет
10            set Номер_места = null where (Номер_места > (select Количество_мест from Марка_самолета where Марка_самолета = (se
11            return new;
12        elsif ((select Количество_мест from Марка_самолета where Марка_самолета = (select Марка_самолета from Самолет where Бс
13        then
14            for i in (select max(Номер_места::integer) + 1 from Билет where Номер_рейса_б = new.Номер_рейса)..(select Количес
15            insert into Билет(Номер_билета, Номер_рейса_б, Номер_места) values((select max(Номер_билета::integer) + 1 from
16            end loop;
17        return new;
18    end if;
19    end if;
20 end;
21 $$ language 'plpgsql';
22
23 --create trigger Триггер_на_рейс_UPDATE
24 --after update on Рейс
25 --for each row execute procedure Обновление_рейса()
26
27


```

Data Output Explain Messages Notifications

CREATE FUNCTION

Query returned successfully in 36 msec.

- Обновляем рейс – меняем самолет на меньший:


 Vlka/postgres@PostgreSQL 11 ▾

[Query Editor](#)
[Query History](#)

```


1  update Рейс
2  set Бортной_номер = 12345 where Номер_рейса = 1000
  
```

[Data Output](#)
[Explain](#)
[Messages](#)
[Notifications](#)

UPDATE 1

Query returned successfully in 66 msec.

- Видим, что бортовой номер изменился:


 Vlka/postgres@PostgreSQL 11 ▾

[Query Editor](#)
[Query History](#)

```

1  SELECT * FROM Рейс
  
```

[Data Output](#)
[Explain](#)
[Messages](#)
[Notifications](#)

	Номер_рейса [PK] integer	Номер_самолета integer	Дата_вылета date	Статус character varying (20)	Дата_прилета date	Тип_рейса character varying (20)	Бортной_номер integer
1	591	1	2022-05-28	В пути	2022-06-01	Пассажирский	123
2	1713	4	2022-05-28	Ожидание регистрации	2022-05-28	Пассажирский	123456
3	2368	3	2022-05-31	Завершен	2022-06-01	Пассажирский	12345
4	2637	5	2022-05-29	Ожидание регистрации	2022-05-29	Пассажирский	1234567
5	4679	2	2022-05-30	Завершен	2022-05-31	Пассажирский	1234
6	7375	6	2022-05-28	Ожидание регистрации	2022-05-29	Пассажирский	12345678
7	1000	1	2022-06-02	Ожидание регистрации	2022-06-03	Пассажирский	12345

- Видим, что у билетов 210-219 в номере места null:

Vika/postgres@PostgreSQL 11

Query Editor Query History

1 **SELECT** * **FROM** Билет

Data Output Explain Messages Notifications

	Номер_билета [PK] character varying (20)	Номер_рейса_б integer	Номер_места integer	
189	178	1000	149	
190	179	1000	150	
191	180	1000	151	
192	181	1000	152	
193	182	1000	153	
194	183	1000	154	
195	184	1000	155	
196	185	1000	156	
197	186	1000	157	
198	187	1000	158	
199	188	1000	159	
200	189	1000	160	
201	210	1000	[null]	
202	211	1000	[null]	
203	212	1000	[null]	
204	213	1000	[null]	
205	214	1000	[null]	
206	215	1000	[null]	
207	216	1000	[null]	
208	217	1000	[null]	
209	218	1000	[null]	
210	219	1000	[null]	

3) Триггер удаляет соответствующие билеты из таблицы Билет при удалении рейса:

Vika/postgres@PostgreSQL 11

Query Editor Query History

```

1 create or replace function Удаление_рейс()
2 returns trigger
3 as $$
4 begin
5     if (TG_OP = 'DELETE')
6     then
7         delete from Билет where Номер_рейса_б = old.Номер_рейса;
8         return old;
9     end if;
10 end;
11 $$ language 'plpgsql';
12
13 --create trigger Триггер_на_рейс_Delete
14 --before delete on Рейс
15     --for each row execute procedure Удаление_рейс()

```

Data Output Explain Messages Notifications

CREATE FUNCTION

Query returned successfully in 49 msec.

- Удаляем рейс 1000:

Viika/postgres@PostgreSQL 11

Query Editor Query History

```
1 delete from Рейс where Номер_рейса = 1000
```

Data Output Explain Messages Notifications

DELETE 1

Query returned successfully in 68 msec.

- Видим, что билетов на рейс номер 1000 теперь нет:

Viika/postgres@PostgreSQL 11

Query Editor Query History

```
1 SELECT * FROM Билет
```

Data Output Explain Messages Notifications

	Номер_билета [PK] character varying (20)	Номер_рейса_б integer	Номер_места integer	Цена_билета integer	Место_прибытия character varying (20)	Место_отправления character varying (20)	Тип_места character varyi
1	12	2368	23	12000	Сосновоборск	Токио	эконом
2	11	4679	22	15000	Железнодорожск	Нефтекамск	эконом
3	13	591	14	15000	Иркутск	Париж	эконом+
4	14	591	25	10000	Иркутск	Париж	эконом
5	10	591	21	10000	Иркутск	Париж	эконом
6	15	2368	3	10000	Сосновоборск	Токио	эконом
7	16	591	45	10000	Иркутск	Париж	эконом
8	17	4679	34	10000	Железнодорожск	Нефтекамск	эконом
9	18	4679	31	10000	Железнодорожск	Нефтекамск	эконом
10	19	2368	15	10000	Сосновоборск	Токио	эконом
11	20	591	17	10000	Иркутск	Париж	эконом
12	21	591	18	10000	Иркутск	Париж	эконом
13	22	2368	16	10000	Сосновоборск	Токио	эконом
14	23	591	24	10000	Иркутск	Париж	эконом
15	24	4679	30	10000	Железнодорожск	Нефтекамск	эконом
16	26	2368	41	10000	Сосновоборск	Токио	эконом
17	25	2368	39	10000	Сосновоборск	Токио	эконом
18	27	591	29	10000	Иркутск	Париж	эконом
19	28	4679	19	10000	Железнодорожск	Нефтекамск	эконом
20	29	2368	11	10000	Сосновоборск	Токио	эконом

- Рейс удален:

Query Editor

Query History

1

SELECT * FROM Рейс

Data Output

Explain

Messages

Notifications

	Номер_рейса [PK] integer	Номер_самолета integer	Дата_вылета date	Статус character varying (20)	Дата_прилета date	Тип_рейса character varying (20)	Бортовой_номер integer
1	591	1	2022-05-28	В пути	2022-06-01	Пассажирский	123
2	1713	4	2022-05-28	Ожидание регистрации	2022-05-28	Пассажирский	123456
3	2368	3	2022-05-31	Завершен	2022-06-01	Пассажирский	12345
4	2637	5	2022-05-29	Ожидание регистрации	2022-05-29	Пассажирский	1234567
5	4679	2	2022-05-30	Завершен	2022-05-31	Пассажирский	1234
6	7375	6	2022-05-28	Ожидание регистрации	2022-05-29	Пассажирский	12345678

ВЫВОДЫ

В ходе выполнения работы, были освоены практические навыки создания представлений и запросов на выборку данных PostgreSQL, использования подзапросов при модификации данных и индексов.