



**Министерство науки и высшего образования
Российской Федерации Федеральное
государственное бюджетное образовательное
учреждение высшего образования «Московский
государственный
технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)**

**Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»**

Курс «Базы данных»

Отчет по лабораторной работе №1-6

**Выполнил:
студент группы ИУ5-41Б
Кашима А.**

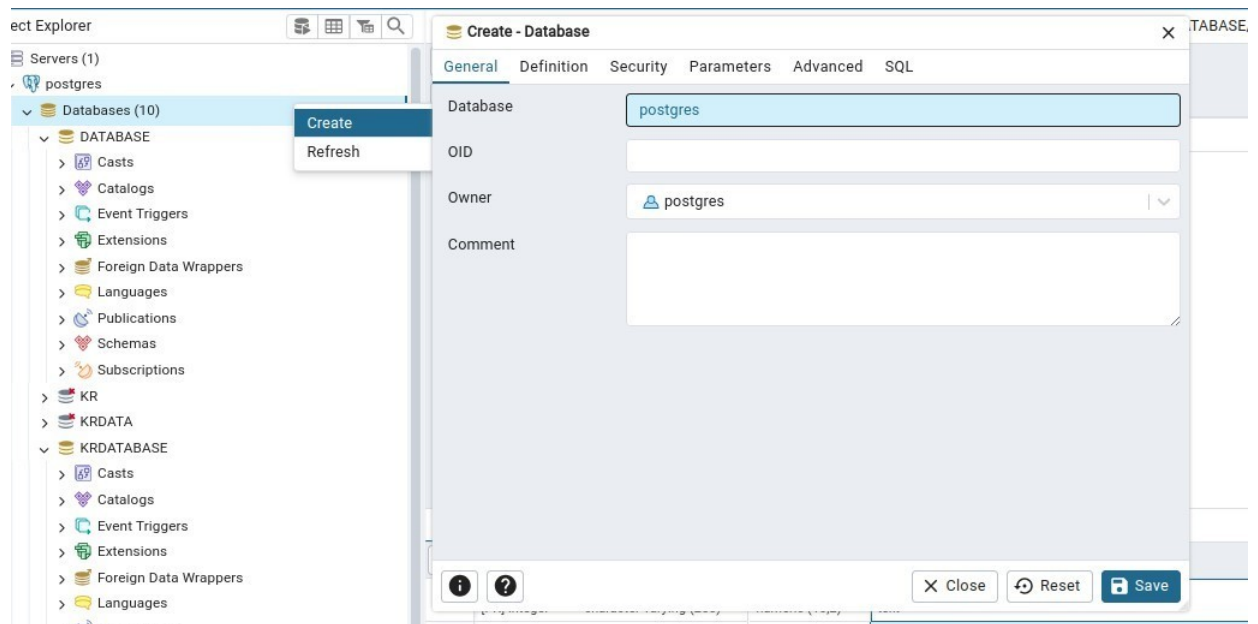
**Проверил:
преподаватель каф. ИУ5
Ковалева Н.А.**

Москва, 2024 г.

Цель лабораторной работы1

Изучить основные принципы организации PostgreSQL, получить теоретические и практические навыки создания базы данных в СУБД PostgreSQL, изучить основные понятия и операторы, научиться работать в среде pgAdmin, преобразовать базу данных MS Access в базу PostgreSQL, сформировать знания и умения по программированию на языке SQL, приобрести практические навыки работы со средствами языка SQL для обновления, удаления и вставки данных в БД.

Создание БД в среде pgAdmin



Создание таблиц:

- С помощью запроса

```
CREATE TABLE products (  
    product_id SERIAL PRIMARY KEY,  
    product_name VARCHAR(255) NOT NULL,  
    price DECIMAL(10, 2) NOT NULL,  
    description TEXT  
);
```

```
CREATE TABLE customers (  
    customer_id SERIAL PRIMARY KEY,  
    customer_name VARCHAR(255) NOT NULL,  
    email VARCHAR(255) NOT NULL,  
    phone VARCHAR(20),  
    address TEXT  
);
```

```
CREATE TABLE items (  
    item_id SERIAL PRIMARY KEY,  
    order_id INT NOT NULL,  
    product_id INT NOT NULL,  
    quantity INT NOT NULL,  
    price_per_unit DECIMAL(10, 2) NOT NULL,  
    FOREIGN KEY (order_id) REFERENCES orders(order_id),  
    FOREIGN KEY (product_id) REFERENCES products(product_id)  
);
```

```
CREATE TABLE orders (  
    order_id SERIAL PRIMARY KEY,  
    customer_id INT NOT NULL,  
    order_date DATE NOT NULL,  
    ship_date DATE,  
    paid_date DATE,  
    status VARCHAR(50) NOT NULL,  
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)  
);  
|
```

Изменение таблицы с помощью ALTER TABLE

```
29  
30 ALTER TABLE customers  
31 DROP CONSTRAINT unique_email;  
32
```

Data Output	Messages	Notifications
-------------	----------	---------------

ALTER TABLE

Query returned successfully in 60 msec.

Создание ограничения на уникальность одному столбцу

```
22
23
24 ALTER TABLE customers
25 ADD CONSTRAINT unique_email UNIQUE (email);
26
27
```

Data Output Messages Notifications

ALTER TABLE

Query returned successfully in 82 msec.

Создание значения по умолчанию

```
30
31 ALTER TABLE products
32 ALTER COLUMN price SET DEFAULT 0.00;
33
34
```

Data Output Messages Notifications

ALTER TABLE

Query returned successfully in 48 msec.

Заполнение таблиц базы данных Пример запроса для вставки данных:

INSERT INTO products (product_name, price, description, quantity) VALUES ('Product 3', 29.99, 'Description for Product 3')

Заполненные таблицы:

59
60
61
62

select * from products

Data OutputMessagesNotifications

	product_id [PK] integer	product_name character varying (255)	price numeric (10,2)	description text
1	2	Product 3	29.99	Description of Product 3
2	3	Product 4	39.99	Description of Product 4
3	4	Product 4	15.99	Description of Product 4
4	5	Product 5	25.99	Description of Product 5
5	1	Product 2	24.99	EXPIRED
6	6	Product 6	19.99	Description of Product 6
7	7	Product 7	34.99	Description of Product 7
8	8	Product 8	42.99	Description of Product 8
9	9	Product 6	19.99	Description of Product 6
10	10	Product 7	34.99	Description of Product 7

62
63
64

select * from orders

Data OutputMessagesNotifications

	order_id [PK] integer	customer_id integer	order_date date	ship_date date	paid_date date	status character
1	2	2	2024-02-20	[null]	[null]	P
2	3	1	2024-02-21	[null]	[null]	P
3	1	1	2024-02-19	[null]	[null]	P
4	4	1	2024-02-22	2024-02-23	2024-02-24	S
5	5	2	2024-02-23	2024-02-24	2024-02-25	S
6	6	1	2024-02-24	2024-02-25	2024-02-26	P
7	7	3	2024-02-25	2024-02-26	2024-02-27	P
8	8	2	2024-02-26	2024-02-27	2024-02-28	S
9	9	1	2024-02-27	2024-02-28	2024-02-29	P
10	10	3	2024-02-28	2024-02-29	2024-03-01	S

64 **select** * **from** items

Data Output Messages Notifications

	item_id [PK] integer	order_id integer	product_id integer	quantity integer	price_per_unit numeric (10,2)
1	7	1	2	2	19.99
2	8	1	3	1	29.99
3	23	1	4	3	15.99
4	24	2	5	2	25.99
5	25	2	3	1	29.99
6	26	3	1	4	12.99
7	27	3	2	2	19.99

66 **select** * **from** customers

Data Output Messages Notifications

	customer_id [PK] integer	customer_name character varying (255)	email character varying (255)	phone character varying (20)	address text
1	1	John Doe	john@example.com	123-456-7890	123 Main St
2	2	John Doe	john@example.com	123-456-7890	123 Main St
3	3	Jane Smith	jane@example.com	456-789-0123	456 Elm St
4	4	Alice Johnson	alice@example.com	789-012-3456	789 Oak St
5	5	Customer 4	customer4@example.com	123-456-7890	123 Maple St
6	6	Customer 5	customer5@example.com	456-789-0123	456 Birch St
7	7	Customer 6	customer6@example.com	789-012-3456	789 Pine St
8	8	Customer 7	customer7@example.com	123-456-7890	123 Cedar St
9	9	Customer 8	customer8@example.com	456-789-0123	456 Walnut St
10	10	Customer 9	customer9@example.com	789-012-3456	789 Cherry St

Обновление данных в таблице

```
68 update customers set customer_name='ahmedkashima' where customer_id = 1
69
```

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 477 msec.

Удаление данных из таблиц

```
68 delete from customers where customer_id = 11|
69
```

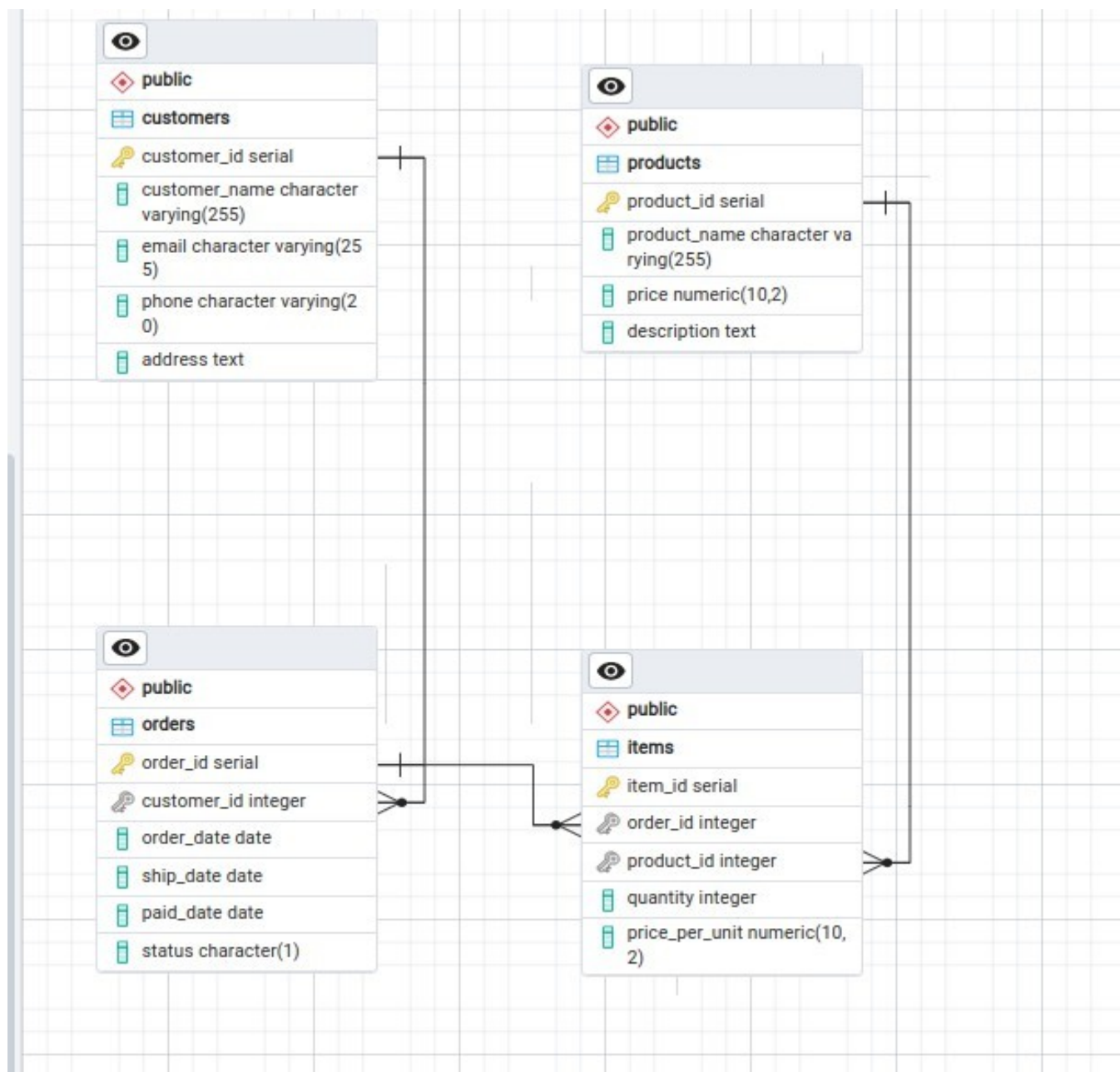
Data Output Messages Notifications

NOTICE: John Doe

DELETE 1

Query returned successfully in 79 msec.

Создание диаграммы БД



Выводы

Изучены основные принципы работы PostgreSQL и разработки в среде pgAdmin; изучены основные операторы и понятия, база данных из MS Access преобразована в базу данных PostgreSQL; сформированы умения по программированию на языке SQL, приобретены практические умения по работе со средствами вставки, обновления и удаления данных в БД.

Цель лабораторной работы2:

Сформировать знания и умения по программированию на языке SQL, приобрести практические навыки работы со средствами языка SQL для выборки и редактирования данных в БД.

а. запрос, выбирающий все данные из таблицы;

```
59
60 select * from products
61
```

	product_id [PK] integer	product_name character varying (255)	price numeric (10,2)	description text
1	2	Product 3	29.99	Description of Product 3
2	3	Product 4	39.99	Description of Product 4
3	4	Product 4	15.99	Description of Product 4
4	5	Product 5	25.99	Description of Product 5
5	1	Product 2	24.99	EXPIRED
6	6	Product 6	19.99	Description of Product 6
7	7	Product 7	34.99	Description of Product 7
8	8	Product 8	42.99	Description of Product 8
9	9	Product 6	19.99	Description of Product 6
10	10	Product 7	34.99	Description of Product 7

б. запрос, выбирающий данные из некоторых столбцов таблицы;

```
62 select product_name, price from products
```

	product_name character varying (255)	price numeric (10,2)
1	Product 3	29.99
2	Product 4	39.99
3	Product 4	15.99
4	Product 5	25.99
5	Product 2	24.99
6	Product 6	19.99
7	Product 7	34.99
8	Product 8	42.99
9	Product 6	19.99
10	Product 7	34.99

с. запрос с использованием сортировки данных;

```
62 select product_name, price from products order by price desc
```

	product_name character varying (255)	price numeric (10,2)
1	Product 10	68.99
2	Product 9	48.99
3	Product 8	42.99
4	Product 8	42.99
5	Product 8	42.99
6	Product 4	39.99
7	Product 7	34.99
8	Product 7	34.99
9	Product 7	34.99
10	Product 3	29.99

д. запрос с использованием ограничения на выборку данных;

```
62 select product_name, price from products order by price desc limit 5
```

	product_name character varying (255)	price numeric (10,2)
1	Product 10	68.99
2	Product 9	48.99
3	Product 8	42.99
4	Product 8	42.99
5	Product 8	42.99

е. запрос с использованием операторов сравнения;

63
64
65

```
select product_name, price from products where price::numeric::int > 45 order by price::numeric::int asc;
```

Data OutputMessagesNotifications

	product_name character varying (255)	price numeric (10,2)
1	Product 9	48.99
2	Product 10	68.99

ф. запрос с использованием оператора BETWEEN;

66
67

```
select product_name, price from products where price between 30 and 50 order by price asc;
```

Data OutputMessagesNotifications

	product_name character varying (255)	price numeric (10,2)
1	Product 7	34.99
2	Product 7	34.99
3	Product 7	34.99
4	Product 4	39.99
5	Product 8	42.99
6	Product 8	42.99
7	Product 8	42.99
8	Product 9	48.99

і. запрос с использованием предиката IS NULL;

```
1 SELECT p.product_id, p.product_name
2 FROM products p
3 LEFT JOIN items oi ON p.product_id = oi.product_id
4 WHERE oi.product_id IS NULL
5 ORDER BY p.product_id;
6
```

Data Output Messages Notifications

	product_id [PK] integer	product_name character varying (255)
1	6	Product 6
2	7	Product 7
3	8	Product 8
4	9	Product 6
5	10	Product 7
6	11	Product 8
7	12	Product 6
8	13	Product 7
9	14	Product 8
10	15	Product 9
11	16	Product 10

ј. запрос с использованием агрегатных функций;

```
68 SELECT
69     SUM(items.quantity) AS total_quantity_sold,
70     SUM(items.price_per_unit * items.quantity) AS total_revenue,
71     (SELECT AVG(products.price - items.price_per_unit)
72      FROM items
73      JOIN products ON items.product_id = products.product_id) AS avg_profit_per_item
74 FROM items;
75
```

Data Output Messages Notifications

	total_quantity_sold bigint	total_revenue numeric	avg_profit_per_item numeric
1	15	291.85	7.4285714285714286

задание по вариантам 25 - Получить список заказов, фамилии, телефоны и адреса покупателей, которые совершили заказ с <любая дата> по <любая дата>. Список отсортировать по дате заказа.

9 SELECT o.order_id, c.customer_name, c.phone, c.address, o.order_date
10 FROM orders o
11 JOIN customers c ON o.customer_id = c.customer_id
12 WHERE o.order_date BETWEEN '2024-02-20' AND '2024-02-25' ORDER BY o.order_date;
13
14

Data OutputMessagesNotifications

order_id

integer

customer_name

character varying (255)

phone

character varying (20)

address

text


order_date

date

1	2	John Doe	123-456-7890	123 Main St	2024-02-20
2	3	ahmedkashima	123-456-7890	123 Main St	2024-02-21
3	4	ahmedkashima	123-456-7890	123 Main St	2024-02-22
4	5	John Doe	123-456-7890	123 Main St	2024-02-23
5	6	ahmedkashima	123-456-7890	123 Main St	2024-02-24
6	7	Jane Smith	456-789-0123	456 Elm St	2024-02-25

задание по вариантам 25-Получить информацию о покупателе (фамилия, адрес, телефон, дата заказа) с максимальной суммой заказа.

```
14 WITH order_totals AS (  
15     SELECT o.order_id, o.customer_id, o.order_date,  
16           SUM(oi.quantity * oi.price_per_unit) AS order_total  
17     FROM orders o  
18     JOIN items oi ON o.order_id = oi.order_id  
19     GROUP BY o.order_id, o.customer_id, o.order_date  
20 ),  
21 max_order AS (  
22     SELECT customer_id, order_date, order_total  
23     FROM order_totals  
24     ORDER BY order_total DESC  
25     LIMIT 1  
26 )  
27 SELECT c.customer_name, c.address, c.phone, mo.order_date  
28 FROM max_order mo  
29 JOIN customers c ON mo.customer_id = c.customer_id;  
30  
31  
32
```

Data Output Messages Notifications				
				
	customer_name character varying (255) 🔒	address text 🔒	phone character varying (20) 🔒	order_date date 🔒
1	ahmedkashima	123 Main St	123-456-7890	2024-02-19

Выводы

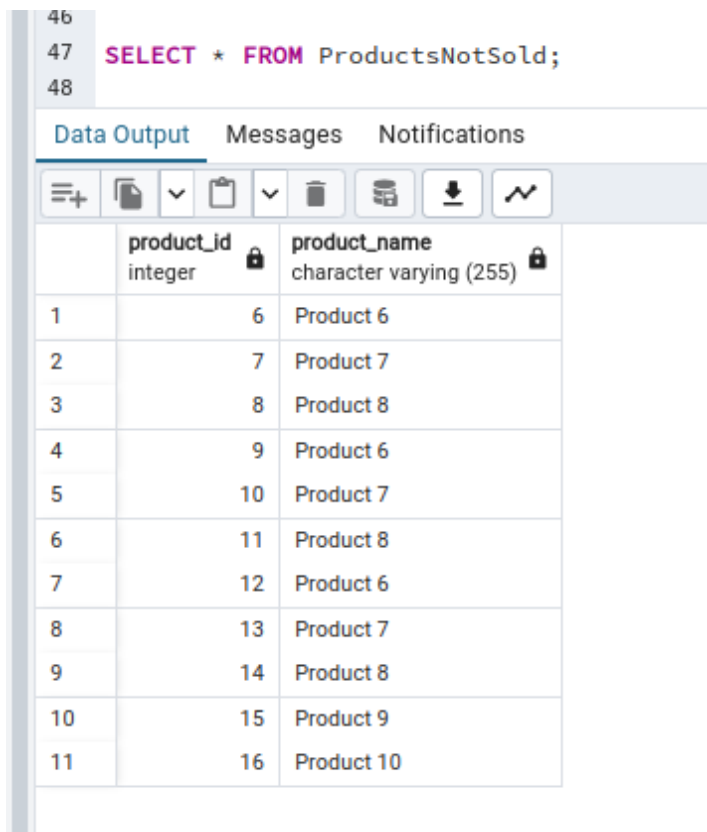
Знания и умения по программированию на языке SQL сформированы, приобретены практические навыки по работе со средствами языка SQL для выборки и редактирования данных в БД.

Цель лабораторной работы3

Изучить механизм использования представлений и предоставления прав в PostgreSQL, получить практически навыки создания представлений в среде PostgreSQL.

1. Создать любое простое представление и запросить с помощью него данные.

```
CREATE VIEW ProductsNotSold AS
SELECT p.product_id, p.product_name
FROM products p
WHERE NOT EXISTS
( SELECT 1
  FROM items oi
  WHERE p.product_id = oi.product_id
)
ORDER BY p.product_id ASC;
```



The screenshot shows a PostgreSQL query editor with a SQL query on line 47 and its results displayed in a table below. The table has two columns: 'product_id' (integer) and 'product_name' (character varying (255)). The results are ordered by product_id.

	product_id integer	product_name character varying (255)
1	6	Product 6
2	7	Product 7
3	8	Product 8
4	9	Product 6
5	10	Product 7
6	11	Product 8
7	12	Product 6
8	13	Product 7
9	14	Product 8
10	15	Product 9
11	16	Product 10

2. Проверить соответствие данных прямым запросом.

```
49 SELECT p.product_id, p.product_name
50 FROM products p
51 WHERE NOT EXISTS (
52     SELECT 1
53     FROM items oi
54     WHERE p.product_id = oi.product_id
55 )
56 ORDER BY p.product_id ASC;
57
```

Data Output Messages Notifications

	product_id [PK] integer	product_name character varying (255)
1	6	Product 6
2	7	Product 7
3	8	Product 8
4	9	Product 6
5	10	Product 7
6	11	Product 8
7	12	Product 6
8	13	Product 7
9	14	Product 8
10	15	Product 9
11	16	Product 10

3. Изменить созданное представление с помощью команды ALTER VIEW, добавив псевдонимы полям.

```
57
58 alter view ProductsNotSold rename column product_id to id;
```

Data Output Messages Notifications

ALTER VIEW

Query returned successfully in 38 msec.

4. Изменить запрос созданного представления с помощью команды CREATE OR REPLACE VIEW. Создать представление с опцией WITH CHECK OPTION.

```
1 create or replace view dealdate as
2 select deal_id, product_id, deal_price, deal_quantity, deal_date from deal
3 where deal_date < '19/05/2022' order by deal_id desc with cascaded check option;
4
```

Data Output Сообщения Notifications

CREATE VIEW

Запрос завершён успешно, время выполнения: 68 msec.

6. Удалить представление.

```
76
77 drop view ProductsNotSold
```

Data Output Messages Notifications

DROP VIEW

Query returned successfully in 163 msec.

8. Создать роль Test_creator без права входа в систему, но с правом создания БД и ролей.

```
1 create role test_creator with nologin createdb createrole
2
```

Data Output Messages Notifications

CREATE ROLE

Query returned successfully in 48 msec.

9. Создать пользователя user1 с правом входа в систему. Убедиться, что user1 не может создать БД.

```
1 create role user1 with createdb login password '1'
```

Data Output	Messages	Notifications
CREATE ROLE		
Query returned successfully in 47 msec.		

10. Включить пользователя user1 в группу Test_creator.

```
3 create database try;
```

Data Output	Messages	Notifications
CREATE DATABASE		
Query returned successfully in 109 msec.		

```
5 alter group test_creator add user user1
```

Data Output	Messages	Notifications
ALTER ROLE		
Query returned successfully in 62 msec.		

12. Создать роли без права создания таблицы и с правом создания таблицы, последовательно проверить работу ролей.

Создадим роль без права создания таблиц:

```
11  
12 CREATE ROLE role_without_create;  
13
```

Data Output	Messages	Notifications
CREATE ROLE		
Query returned successfully in 81 msec.		

Создадим роль с правом создания таблиц:

```
13  
14 CREATE ROLE role_with_create WITH CREATEDB;  
15
```

Data Output Messages Notifications

CREATE ROLE

Query returned successfully in 95 msec.

CREATE TABLE test_table_role1 (id SERIAL PRIMARY KEY, name TEXT);

```
16 CREATE TABLE test_table_role1 (id SERIAL PRIMARY KEY, name TEXT);  
17  
18
```

Data Output Messages Notifications

ERROR: relation "test_table_role1" already exists

SQL state: 42P07

13. Добавить к роли право на любые действия с таблицей, проверить работу прав.

Добавление прав к ролям:

```
18 GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO role_without_create;  
19
```

Data Output Messages Notifications

GRANT

Query returned successfully in 43 msec.

```
20 CREATE TABLE test_table_role1_1 (id SERIAL PRIMARY KEY, name TEXT);  
21
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 67 msec.

проверить работу прав.

```
22 SELECT * FROM test_table_role1_1;
23
```

Data Output Messages Notifications



id	name
[PK] integer	text

14. Удалить право вставки в таблицу, проверить работу прав.

```
25 REVOKE INSERT ON ALL TABLES IN SCHEMA public FROM role_without_create;
26
```

Data Output Messages Notifications

REVOKE

Query returned successfully in 108 msec.

```
27 INSERT INTO role_without_create (column1) VALUES ('value1');
28
29
```

Data Output Messages Notifications

ERROR: relation "role_without_create" does not exist
LINE 27: INSERT INTO role_without_create (column1) VALUES ('value1');
 ^

SQL state: 42P01
Character: 681

Выводы:

Изучен механизм использования представлений и предоставления прав в PostgreSQL, получены практические навыки создания представления в PostgreSQL.

Цель лабораторной работы:4

Соединение с базой данных

Перейдите в режим редактора исходных файлов.

В файл .pro добавьте подключение библиотеки для работы с базой данных:

```
1 #-----
2 #
3 # Project created by QtCreator 2024-03-15T22:48:25
4 #
5 #-----
6
7 QT      += core gui sql
8
9 greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
10
11 TARGET = untitled1
12 TEMPLATE = app
13
14
15 SOURCES += main.cpp\
16          widget.cpp
17
18 HEADERS  += widget.h
19
20 FORMS    += widget.ui
21
```

Отображение таблицы;

	Abbr	Title	City	INN
1	ABC	ABC Corporati...	Moscow	12345678901234567000
2	DEF	DEF Ltd.	Novosibirsk	24680135792468013000
3	niggaa	boya	unknow	8990
4	its me	notha	belgoard	8098
5	INNA	nothingDD	moscow	78787
6	JKL	JKL Industries	Kazanehaha3	213
7	kashima	niger	moscow	101019
8	kashimaAHMED	zarbah	moscowQQw	1010191121200

SELECT-запрос, осуществляющий выборку данных:

```
//  
// Создать объект запроса с привязкой к установленному соединению  
QSqlQuery query(dbconn);  
// Создать строку запроса на выборку данных  
QString sqlstr = "select * from org";  
// Выполнить запрос и поверить его успешность  
if( !query.exec(sqlstr) )  
{  
    QMessageBox::critical(this,"Error", query.lastError().text());  
    return;  
}  
// Если запрос активен (успешно завершен),  
// то вывести сообщение о прочитанном количестве строк в окно вывода  
// и установить количество строк для компонента таблицы
```

Выводы:

Научились создавать на языке C++ с помощью библиотек Qt оконное приложение для работы с базой данных.

Цель лабораторно работ5:

изучить хранимые процедуры и триггеры в базах данных, приобрести практические навыки создания хранимых процедур и триггеров в среде PostgreSQL.

а. Пример из теоретической части:

```
1 CREATE OR REPLACE PROCEDURE InsertNewProducts()  
2 LANGUAGE SQL  
3 AS $$  
4 INSERT INTO products (product_name, price, description)  
5 VALUES  
6 ('Product 6', 19.99, 'Description of Product 6'),  
7 ('Product 7', 34.99, 'Description of Product 7'),  
8 ('Product 8', 42.99, 'Description of Product 8');  
9 $$;  
10
```

Data Output Messages Notifications

CREATE PROCEDURE

Query returned successfully in 57 msec.

CALL InsertNewProducts();

1. функция

а. Пример из теоретической части:

1)

```
CREATE OR REPLACE FUNCTION OrderPrice(x NUMERIC)
RETURNS TABLE(order_id INT, total NUMERIC) AS $$
BEGIN
    RETURN
    QUERY SELECT
        o.order_id,
        SUM(i.quantity * p.price) AS total
    FROM
        orders o
    JOIN
        items i ON o.order_id = i.order_id
    JOIN
        products p ON i.product_id = p.product_id
    GROUP BY
        o.order_id
    HAVING
        SUM(i.quantity * p.price) > x
    ORDER BY
        o.order_id;
END;
$$ LANGUAGE plpgsql;
```

```

18
19 CREATE OR REPLACE FUNCTION OrderPrice(x NUMERIC)
20 RETURNS TABLE(order_id INT, total NUMERIC) AS $$
21 BEGIN
22     RETURN QUERY
23     SELECT
24         o.order_id,
25         SUM(i.quantity * p.price) AS total
26     FROM
27         orders o
28     JOIN
29         items i ON o.order_id = i.order_id
30     JOIN
31         products p ON i.product_id = p.product_id
32     GROUP BY
33         o.order_id
34     HAVING
35         SUM(i.quantity * p.price) > x
36     ORDER BY
37         o.order_id;
38 END;
39 $$ LANGUAGE plpgsql;
40

```

Эта функция рассчитывает общую цену для каждого заказа, умножив количество заказанных товаров на соответствующую цену за единицу, а затем суммируя эти значения. Он вернет order_id и общую цену для ордеров, общая цена которых превышает входной параметр x.

```

47
48 SELECT * FROM OrderPrice(15)
49

```

Data Output Messages Notifications

	order_id integer	total numeric
1	1	147.94
2	2	91.97
3	3	159.94

2)

CREATE OR REPLACE PROCEDURE

MeanValue (

Value1 REAL DEFAULT

0, Value2 REAL

DEFAULT 0, Value3

REAL DEFAULT 0,

INOUT outputers REAL DEFAULT 0

)

LANGUAGE plpgsql

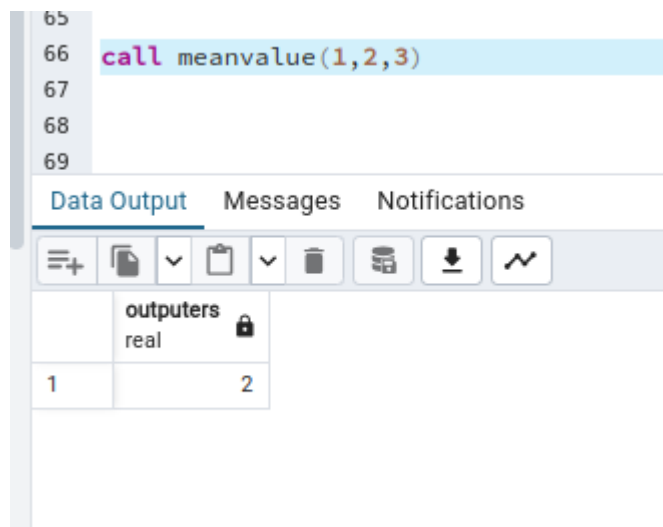
AS \$\$

BEGIN

SELECT (Value1 + Value2 + Value3) / 3 INTO outputers;

END; \$\$;

```
50
51 CREATE OR REPLACE PROCEDURE
52 MeanValue (
53     Value1 REAL DEFAULT 0,
54     Value2 REAL DEFAULT 0,
55     Value3 REAL DEFAULT 0,
56     INOUT outputers REAL DEFAULT 0
57 )
58 LANGUAGE plpgsql
59 AS $$
60 BEGIN
61     SELECT (Value1 + Value2 + Value3) / 3 INTO outputers;
62 END; $$;
63
64
65
```



с. Хранимая процедура для поиска по диапазону цен.

```
CREATE OR REPLACE FUNCTION GetProductsByPriceRange(min_price NUMERIC, max_price
NUMERIC)
RETURNS TABLE(product_name VARCHAR, price MONEY) AS $$
BEGIN
    RETURN QUERY
    SELECT
        products.product_name,
        products.price::MONEY -- Cast the price column to MONEY type
    FROM
        products
    WHERE
        products.price BETWEEN min_price AND max_price;
END;
$$ LANGUAGE plpgsql;
```

```

7 CREATE OR REPLACE FUNCTION GetProductsByPriceRange(min_price NUMERIC, max_price NUMERIC)
8 RETURNS TABLE(product_name VARCHAR, price MONEY) AS $$
9 BEGIN
10     RETURN QUERY
11     SELECT
12         products.product_name,
13         products.price::MONEY -- Cast the price column to MONEY type
14     FROM
15         products
16     WHERE
17         products.price BETWEEN min_price AND max_price;
18 END;
19 $$ LANGUAGE plpgsql;

```

```

100
101
102 SELECT * FROM GetProductsByPriceRange(15.99, 24.99);
103
104

```

Data Output Messages Notifications



	product_name character varying	price money
1	Product 4	\$15.99
2	Product 2	\$24.99

d. Хранимая процедура для поиска заказов по дате заказа и диапазону дат заказа. ,,

```

CREATE OR REPLACE FUNCTION GetOrdersByDateRange(start_date DATE, end_date
DATE)
RETURNS TABLE(order_id INT, customer_name CHAR, order_date DATE)
AS $$
BEGIN
    RETURN QUERY
    SELECT
        o.order_id,
        c.customer_name::CHAR, -- Explicitly cast to CHAR type
        o.order_date
    FROM
        orders o
    JOIN
        customers c ON o.customer_id = c.customer_id
    WHERE
        o.order_date BETWEEN start_date AND end_date;
END;
$$ LANGUAGE plpgsql;

```

```

119
120 -- CREATE OR REPLACE FUNCTION GetOrdersByDateRange(start_date DATE, end_date DATE)
121 -- RETURNS TABLE(order_id INT, customer_name CHAR, order_date DATE)
122 -- AS $$
123 -- BEGIN
124 --     RETURN QUERY
125 --     SELECT
126 --         o.order_id,
127 --         c.customer_name::CHAR, -- Explicitly cast to CHAR type
128 --         o.order_date
129 --     FROM
130 --         orders o
131 --     JOIN
132 --         customers c ON o.customer_id = c.customer_id
133 --     WHERE
134 --         o.order_date BETWEEN start_date AND end_date;
135 -- END;
136 -- $$ LANGUAGE plpgsql;
137
138
139 SELECT * FROM GetOrdersByDateRange('2024-02-01', '2024-02-22');
140
141
142

```

Data Output Messages Notifications



	order_id integer	customer_name character	order_date date
1	4	J	2024-02-22
2	1	J	2024-02-19
3	3	J	2024-02-21
4	2	J	2024-02-20

е. По заданию: варианта

1) получить сгруппированный по городу список с информацией (№заказа, дата заказа, дата доставки) за интервал временной. Отсортировать список по дате доставки. Интервал вводятся как параметры.

```
DROP FUNCTION IF EXISTS GetOrdersByCityAndDateRange(date, date);
```

```
CREATE OR REPLACE FUNCTION GetOrdersByCityAndDateRange(start_date  
DATE, end_date DATE)
```

```
RETURNS TABLE(city TEXT, order_id INT, order_date DATE, ship_date  
DATE)
```

```
AS $$
```

```
BEGIN
```

```
    RETURN
```

```
    QUERY SELECT
```

```
        c.address AS
```

```
        city, o.order_id,
```

```
        o.order_date,
```

```
        o.ship_date
```

```
    FROM
```

```
        orders o
```

```
    JOIN
```

```
        customers c ON o.customer_id = c.customer_id
```

```
    WHERE
```

```
        o.ship_date BETWEEN start_date AND end_date
```

```
    ORDER BY
```

```
        o.ship_date;
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

```
--- - SELECT * FROM GetOrdersByCityAndDateRange('2024-02-01', '2024-02-  
28');
```

```

144 CREATE OR REPLACE FUNCTION GetOrdersByCityAndDateRange(start_date DATE, end_date DATE)
145 RETURNS TABLE(city TEXT, order_id INT, order_date DATE, ship_date DATE)
146 AS $$
147 BEGIN
148     RETURN QUERY
149     SELECT
150         c.address AS city,
151         o.order_id,
152         o.order_date,
153         o.ship_date
154     FROM
155         orders o
156     JOIN
157         customers c ON o.customer_id = c.customer_id
158     WHERE
159         o.ship_date BETWEEN start_date AND end_date
160     ORDER BY
161         o.ship_date;
162 END;
163 $$ LANGUAGE plpgsql;
164
165
166 SELECT * FROM GetOrdersByCityAndDateRange('2024-02-01', '2024-02-28');
167

```

Data Output Messages Notifications



	city text	order_id integer	order_date date	ship_date date
1	123 Main St	4	2024-02-22	2024-02-23
2	123 Main St	5	2024-02-23	2024-02-24
3	123 Main St	6	2024-02-24	2024-02-25
4	456 Elm St	7	2024-02-25	2024-02-26
5	123 Main St	8	2024-02-26	2024-02-27
6	123 Main St	9	2024-02-27	2024-02-28

2) подсчитать количество заказов по городам

```
SELECT
    c.address AS city,
    COUNT(o.order_id) AS order_count
FROM
    orders o
JOIN
    customers c ON o.customer_id = c.customer_id
GROUP BY
    c.address;
```

```
168
169 SELECT
170     c.address AS city,
171     COUNT(o.order_id) AS order_count
172 FROM
173     orders o
174 JOIN
175     customers c ON o.customer_id = c.customer_id
176 GROUP BY
177     c.address;
178
179
```

Data Output Messages Notifications



	city text	order_count bigint
1	456 Elm St	3
2	123 Main St	10

создать функция

```
CREATE FUNCTION calculate_average_price()  
RETURNS NUMERIC  
LANGUAGE SQL  
AS $$  
    SELECT AVG(price) FROM products;  
$$;
```

вызывать функция

```
SELECT calculate_average_price();
```

```
38  
39 -- CREATE FUNCTION calculate_average_price()  
40 -- RETURNS NUMERIC  
41 -- LANGUAGE SQL  
42 -- AS $$  
43 --     SELECT AVG(price) FROM products;  
44 -- $$;  
45  
46  
47 SELECT calculate_average_price();  
48
```

Data Output Messages Notifications

	calculate_average_price numeric
1	30.7757142857142857

This function calculates the average price of all products in your products table. You can call this function to get the average price.

3. Создать триггер INSERT

```
CREATE OR REPLACE FUNCTION update_product_stock()
RETURNS TRIGGER AS $$
BEGIN
    -- Decrease the stock quantity of the ordered product
    UPDATE products
    SET stock_quantity = stock_quantity - NEW.quantity
    WHERE product_id = NEW.product_id;

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER update_stock_on_order
AFTER INSERT ON items
FOR EACH ROW
EXECUTE FUNCTION update_product_stock();
```

```
55
56 CREATE OR REPLACE FUNCTION update_product_stock()
57 RETURNS TRIGGER AS $$
58 BEGIN
59     -- Decrease the stock quantity of the ordered product
60     UPDATE products
61     SET stock_quantity = stock_quantity - NEW.quantity
62     WHERE product_id = NEW.product_id;
63
64     RETURN NULL;
65 END;
66 $$ LANGUAGE plpgsql;
67
68 CREATE TRIGGER update_stock_on_order
69 AFTER INSERT ON items
70 FOR EACH ROW
71 EXECUTE FUNCTION update_product_stock();
72
73
```

4. Создать триггер DELETE

```
CREATE OR REPLACE FUNCTION deletefn() RETURNS TRIGGER AS $$  
DECLARE
```

```
    x varchar := 'John Doe';
```

```
BEGIN
```

```
    RAISE NOTICE '%', x;
```

```
    RETURN NULL;
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE TRIGGER delete_tr
```

```
AFTER DELETE ON customers
```

```
EXECUTE PROCEDURE deletefn();
```

```
83  
84 CREATE OR REPLACE FUNCTION deletefn() RETURNS TRIGGER AS $$  
85 DECLARE  
86     x varchar := 'John Doe';  
87 BEGIN  
88     RAISE NOTICE '%', x;  
89     RETURN NULL;  
90 END;  
91 $$ LANGUAGE plpgsql;  
92  
93 CREATE OR REPLACE TRIGGER delete_tr  
94 AFTER DELETE ON customers  
95 EXECUTE PROCEDURE deletefn();  
96  
97
```

Data Output Messages Notifications

CREATE TRIGGER

Query returned successfully in 50 msec.

Создать триггер UPDATE:

```
28
29 CREATE OR REPLACE FUNCTION products_update_trigger()
30 RETURNS TRIGGER AS $$
31 BEGIN
32     -- Вставляем обновленные данные в таблицу products_audit
33     INSERT INTO products_audit (product_id, old_product_name, new_product_name, old_price, new_price, old_description, new_description,
34     VALUES (OLD.product_id, OLD.product_name, NEW.product_name, OLD.price, NEW.price, OLD.description, NEW.description, NOW());
35
36     RETURN NEW;
37 END;
38 $$ LANGUAGE plpgsql;
39
40 CREATE TRIGGER update_products_trigger
41 AFTER UPDATE ON products
42 FOR EACH ROW
43 EXECUTE FUNCTION products_update_trigger();
44
45
```

Data Output Messages Notifications

CREATE TRIGGER

Query returned successfully in 118 msec.

```
45
46 UPDATE products
47 SET product_name = 'New Product Name'
48 WHERE product_id = 1;
49
```

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 55 msec.

6. Создать триггер, который при удалении записи из таблицы Products сначала

удаляет все связанные с ней записи из таблицы Items, а затем удаляет саму запись из таблицы Products.

Допустим, у вас есть две таблицы: товары и заказы, и вы хотите удалить все заказы, связанные с удаленным товаром, прежде чем удалять сам товар. Вот как можно создать такой триггер:

```
63
64 CREATE OR REPLACE FUNCTION delete_orders_for_product()
65 RETURNS TRIGGER AS $$
66 BEGIN
67     -- Delete all orders related to the deleted product
68     DELETE FROM orders WHERE product_id = OLD.product_id;
69
70     RETURN OLD;
71 END;
72 $$ LANGUAGE plpgsql;
73
74 CREATE TRIGGER delete_orders_for_product_trigger
75 BEFORE DELETE ON products
76 FOR EACH ROW
77 EXECUTE FUNCTION delete_orders_for_product();
78
```

Data Output Messages Notifications

CREATE TRIGGER

Query returned successfully in 44 msec.

6. Создать триггер, с использованием временной таблицы NEW.

```
80 CREATE OR REPLACE FUNCTION before_insert_products_trigger()
81 RETURNS TRIGGER AS $$
82 BEGIN
83     -- Выводим новые значения перед вставкой в таблицу
84     RAISE NOTICE 'New product_id: %, product_name: %, price: %', NEW.product_id, NEW.product_name, NEW.price;
85
86     -- Возвращаем NEW, чтобы продолжить вставку
87     RETURN NEW;
88 END;
89 $$ LANGUAGE plpgsql;
90
91 CREATE TRIGGER before_insert_products_trigger
92 BEFORE INSERT ON products
93 FOR EACH ROW
94 EXECUTE FUNCTION before_insert_products_trigger();
95
96
```

Data Output Messages Notifications

CREATE TRIGGER

Query returned successfully in 43 msec.

```
96
97 INSERT INTO products (product_name, price, description)
98 VALUES ('New Product', 29.99, 'Description of the new product');
99 |
```

Data Output Messages Notifications

NOTICE: New product_id: 33, product_name: New Product, price: 29.99

INSERT 0 1

Query returned successfully in 213 msec.

7. Создать триггер DDL, который предотвратит удаление или изменение таблиц в базе данных.

Запрет всех команд работы с таблицами

```
102 CREATE OR REPLACE FUNCTION prevent_ddl_changes()
103 RETURNS event_trigger AS $$
104 BEGIN
105     IF (TG_OP = 'DROP TABLE' OR TG_OP = 'ALTER TABLE') THEN
106         RAISE EXCEPTION 'Changes to tables are not allowed';
107     END IF;
108 END;
109 $$ LANGUAGE plpgsql;
110
111 CREATE EVENT TRIGGER prevent_ddl_trigger
112 ON ddl_command_start
113 EXECUTE FUNCTION prevent_ddl_changes();
114
115
```

Data Output Messages Notifications

CREATE EVENT TRIGGER

Query returned successfully in 141 msec.

```
118 DROP TABLE products_audit;
119
120
```

Data Output Messages Notifications

ERROR: column "tg_op" does not exist

LINE 1: (TG_OP = 'DROP TABLE' OR TG_OP = 'ALTER TABLE')
 ^

QUERY: (TG_OP = 'DROP TABLE' OR TG_OP = 'ALTER TABLE')

CONTEXT: PL/pgSQL function prevent_ddl_changes() line 3 at IF

SQL state: 42703

Выводы:

Изучены хранимые процедуры и триггеры в базах данных, приобретены практические навыки создания хранимых процедур и триггеров в среде PostgreSQL.

Цель лабораторной работы:6

Изучить базовые понятия и типы резервного копирования баз данных, получить практический навык создания резервной копии базы данных в PostgreSQL, а также ее восстановления.

1) Импорт данных из .csv файла в БД:

Text Import - [person_1.csv]

Import

Character set: Unicode (UTF-8)

Locale: Default - English (USA)

From row: 1

Separator Options

Fixed width

Separated by

Tab

Comma

Semicolon

Space

Other

Merge delimiters

Trim spaces

String delimiter: "

Other Options

Format quoted field as text

Detect special numbers

Evaluate Formulas

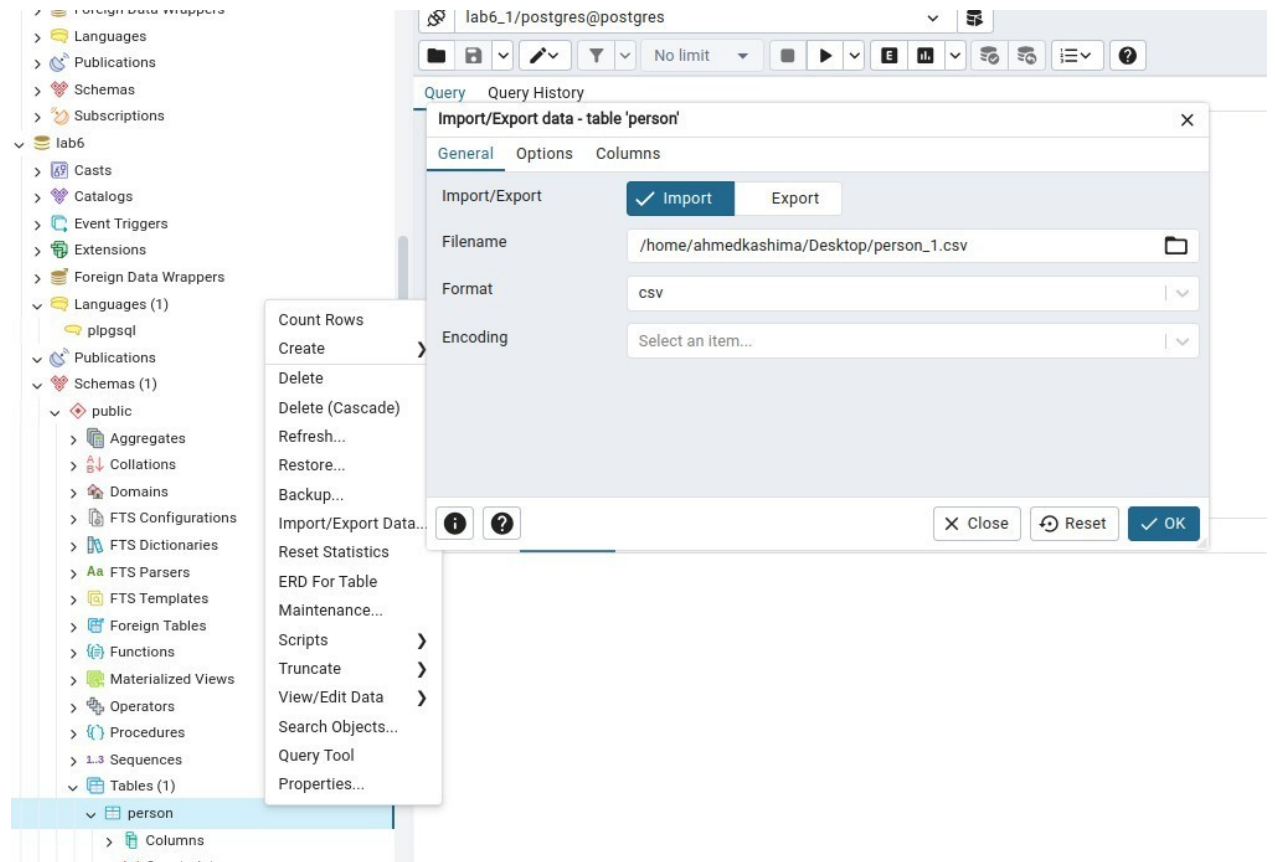
Detect scientific notation

Fields

Column type:

	Standard	Standard	Standard	Standard	Standard	Standard
1	user_index	user_id	first_name	last_name	sex	email
2	1	U001	John	Doe	Male	john.doe@example.com
3	2	U002	Jane	Smith	Female	jane.smith@example.com
4	3	U003	Alice	Johnson	Female	alice.johnson@example.
5	4	U004	Bob	Brown	Male	bob.brown@example.com
6	5	U005	Charlie	Davis	Non-binary	charlie.davis@example.

С помощью графического интерфейса



С помощью команды

```
ahmedkashima@ahmedkashima-Latitude-E5470: ~/psql
lab6_1=# \dt
          List of relations
 Schema | Name  | Type  | Owner
-----+-----+-----+-----
 public | person | table | postgres
(1 row)

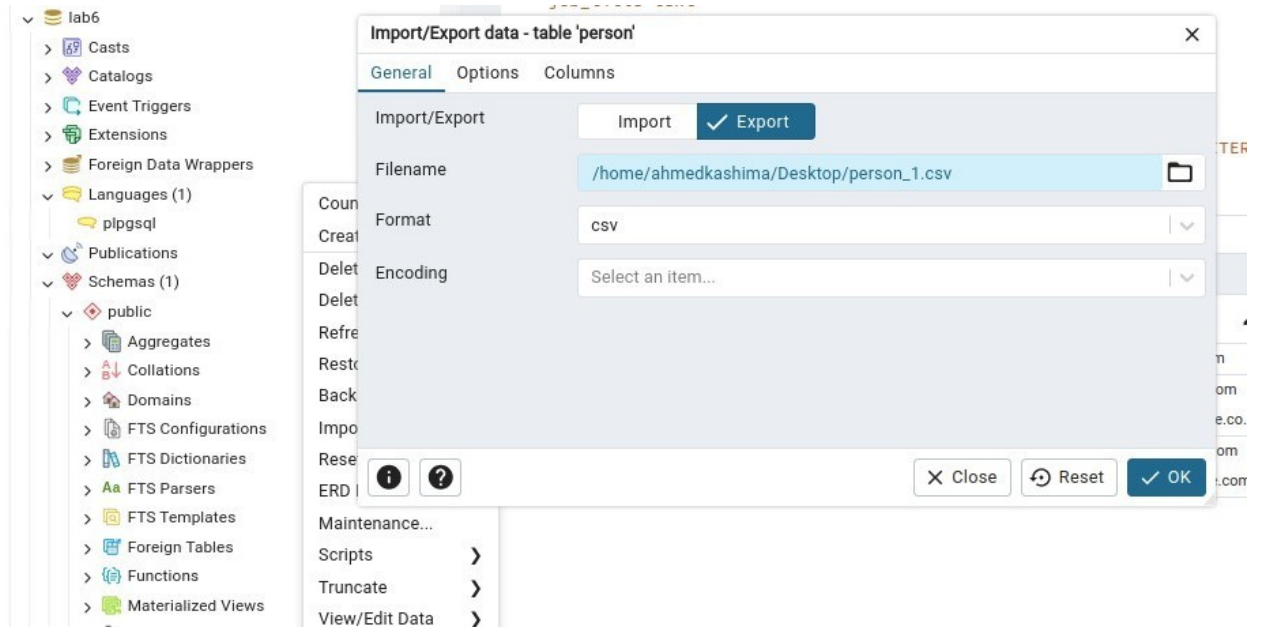
lab6_1=# select * from person
lab6_1-# \copy person from '/home/ahmedkashima/Desktop/person_1.csv' DELIMITER ',' CSV HEADER;
COPY 5
lab6_1-#
```

```
ahmedkashima@ahmedkashima-Latitude-E5470: ~/psql
user_index | user_id | first_name | last_name | sex | email | phone | birth_day | job_title
-----+-----+-----+-----+-----+-----+-----+-----+-----
1 | U001 | John | Doe | Male | john.doe@example.com | 123-456-7890 | 1980-01-15 00:00:00 | Software Engineer
2 | U002 | Jane | Smith | Female | jane.smith@example.com | 234-567-8901 | 1990-02-25 00:00:00 | Data Scientist
3 | U003 | Alice | Johnson | Female | alice.johnson@example.com | 345-678-9012 | 1985-03-30 00:00:00 | Project Manager
4 | U004 | Bob | Brown | Male | bob.brown@example.com | 456-789-0123 | 1975-04-10 00:00:00 | Product Manager
5 | U005 | Charlie | Davis | Non-binary | charlie.davis@example.com | 567-890-1234 | 2000-05-20 00:00:00 | UX Designer
(5 rows)

(END)
```

2) Экспорт данных из БД в .csv файл:

С помощью графического интерфейса



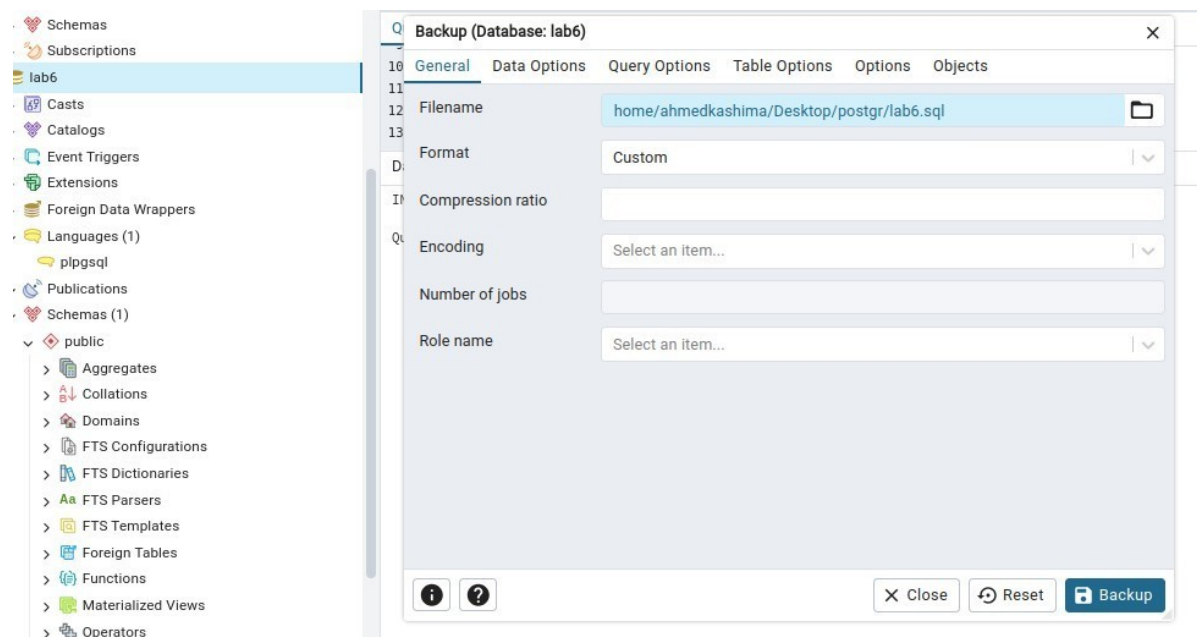
С помощью команды

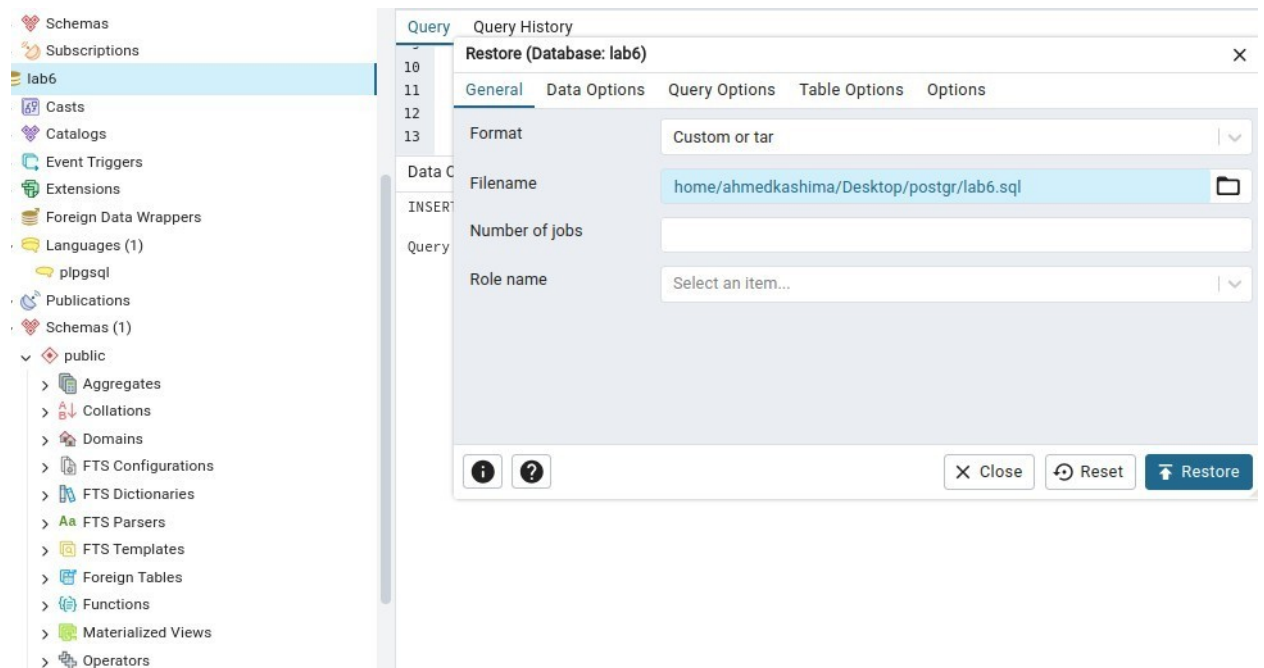
```
ahmedkashima@ahmedkashima-Latitude-E5470: ~/psql
lab6_1=# \copy person to '/home/ahmedkashima/Desktop/person_1.csv' DELIMITER ',' CSV HEADER;;
COPY 5
lab6_1=#
```

3) Бэкап и восстановление с помощью pg_dump.exe

```
ahmedkashima@ahmedkashima-Latitude-E5470:~/psql$ pg_dump -U postgres -W -h localhost -F c -b -v  
-f /home/ahmedkashima/Desktop/postgr/lab6.backup lab6  
Password:  
pg_dump: last built-in OID is 16383  
pg_dump: reading extensions  
pg_dump: identifying extension members  
pg_dump: reading schemas  
pg_dump: reading user-defined tables  
pg_dump: reading user-defined functions  
pg_dump: reading user-defined types  
pg_dump: reading procedural languages  
pg_dump: reading user-defined aggregate functions  
pg_dump: reading user-defined operators  
pg_dump: reading user-defined access methods  
pg_dump: reading user-defined operator classes  
pg_dump: reading user-defined operator families  
pg_dump: reading user-defined text search parsers
```

Графическим способом:





Вывод:

Изучены базовые понятия и типы резервного копирования баз данных, получен практический навык создания резервной копии базы данных PostgreSQL, а также ее восстановления.