**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 «Системы обработки информации и управления»**

Лабораторная работа №5

Курс «Базы данных»

«Использование триггеров и хранимых

процедур в PostgreSQL»

Выполнил:

студент группы ИУ5-45Б

Расулов А. Н.

Проверил:

Маслеников К. Ю.

2024 г.

**Цель лабораторной работы:**

Изучить хранимые процедуры, функции и триггеры в базах данных,

приобрести практические навыки создания хранимых процедур и триггеров в

PostgreSQL.

***Функция для поиска информации по названию компании.***

CREATE FUNCTION find\_company\_info(company\_name\_arg VARCHAR(255))

RETURNS TABLE (

last\_name VARCHAR(255),

first\_name VARCHAR(255),

address VARCHAR(255),

city VARCHAR(255),

index\_code VARCHAR(255),

phone VARCHAR(255),

e\_mail VARCHAR(255)

)

LANGUAGE SQL

AS $$

SELECT last\_name,

first\_name,

address,

city,

index\_code,

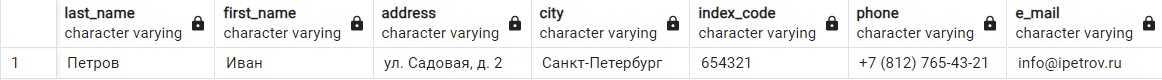
phone,

e\_mail FROM

customers WHERE company\_name = company\_name\_arg;

$$;

SELECT \* FROM find\_company\_info('ИП Петров');



***Функция для поиска товаров по диапазону цен.***

CREATE FUNCTION find\_items\_with\_price(min\_price int, max\_price int)

RETURNS TABLE (

pr\_name VARCHAR(255),

pr\_price numeric,

in\_stock integer,

reorder bool,

description text

)

LANGUAGE SQL

AS $$

SELECT pr\_name,

pr\_price,

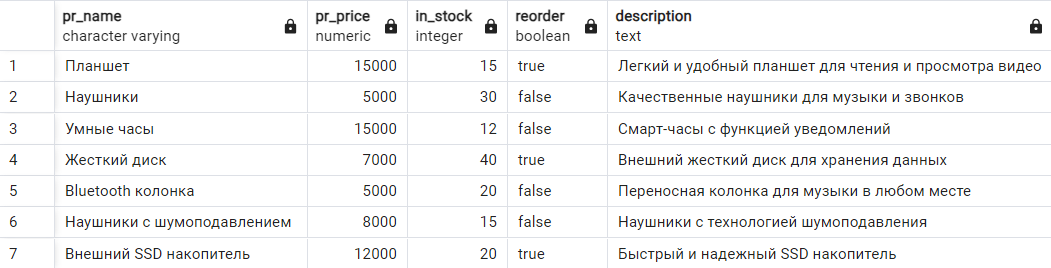
in\_stock,

reorder,

description FROM

products WHERE pr\_price BETWEEN min\_price AND max\_price;

$$;

SELECT \* FROM find\_items\_with\_price (5000, 15000) 

***Функция для поиска заказов по дате заказа и диапазону дат доставки.***

CREATE FUNCTION find\_orders\_by\_dates(or\_date DATE, min\_s\_date DATE DEFAULT '2000-01-01', max\_s\_date DATE DEFAULT '2099-01-01')

RETURNS TABLE (

id\_order integer,

id\_customer integer,

order\_date date,

ship\_date date,

paid\_date date,

status character

)

LANGUAGE SQL

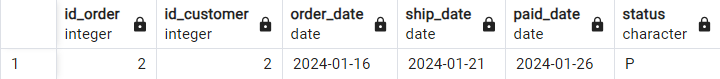
AS $$

SELECT \*FROM orders

WHERE (order\_date = or\_date AND

ship\_date BETWEEN min\_s\_date AND max\_s\_date);

$$;

select \*from find\_orders\_by\_dates ('2024-01-16')

***Функция по заданию варианта.***

1. *Выбрать все сведения о покупателях двух указанных компаний. Номера компаний будут передаваться как параметры.*

CREATE FUNCTION find\_customers\_with\_no\_orders() RETURNS TABLE (

id\_customer INTEGER,

company\_name VARCHAR(255),

last\_name VARCHAR(255),

first\_name VARCHAR(255),

address VARCHAR(255),

city VARCHAR(255),

index\_code VARCHAR(255),

phone VARCHAR(255),

e\_mail VARCHAR(255)

) AS

$$

SELECT c.id\_customer,

c.company\_name,

c.last\_name,

c.first\_name,

c.address,

c.city,

c.index\_code,

c.phone,

c.email

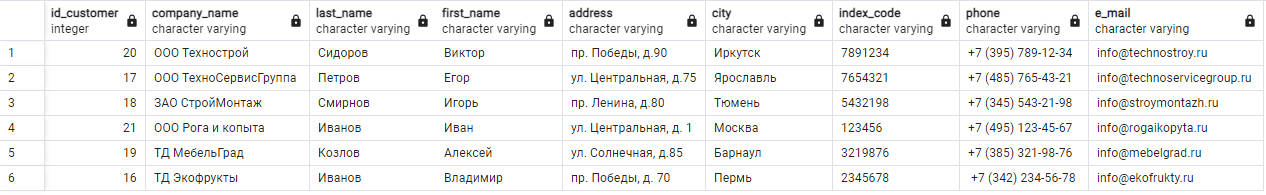
FROM orders o

RIGHT JOIN customers c ON o.id\_customer = c.id\_customer

WHERE o.id\_order is NULL;

$$ LANGUAGE sql;

select \* from find\_customers\_with\_no\_orders()



1. *Подсчитать сумму сделок каждого покупателя из определенного города. Название города вводится как параметр.*

CREATE OR REPLACE FUNCTION get\_spent\_by\_customer\_from\_city(arg\_city VARCHAR(20))

RETURNS TABLE (

company\_name VARCHAR(255),

spent INTEGER

) AS

$$

SELECT c.company\_name, SUM(i.quantity \* p.pr\_price) as total

FROM orders o

LEFT JOIN customers c ON o.id\_customer = c.id\_customer

RIGHT JOIN items i ON o.id\_order = i.id\_order

JOIN products p ON i.id\_product = p.id\_product

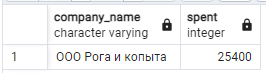
WHERE c.city = arg\_city

GROUP BY c.company\_name

ORDER BY total;

$$ LANGUAGE sql;

select \* from get\_spent\_by\_customer\_from\_city('Москва')



***Создать триггер INSERT.***

CREATE FUNCTION update\_stock() RETURNS TRIGGER AS $$

DECLARE

product\_quantity INT;

BEGIN

SELECT in\_stock INTO product\_quantity FROM products WHERE products.id\_product = NEW.id\_product;

IF product\_quantity >= NEW.quantity THEN

UPDATE products SET in\_stock = product\_quantity - NEW.quantity WHERE products.id\_product = NEW.id\_product;

ELSE

RAISE EXCEPTION 'Недостаточно товара с ID %', NEW.id\_product;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

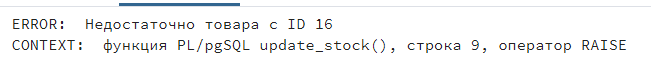
CREATE TRIGGER insert\_item\_trigger

AFTER INSERT ON items

FOR EACH ROW

EXECUTE FUNCTION update\_stock();

INSERT INTO items (id\_order, id\_product, quantity, total)

values (5, 16, 70, 100)

***Создать триггер DELETE.***

CREATE FUNCTION delete\_item() RETURNS TRIGGER AS $$

DECLARE

being\_shipped BOOL;

BEGIN

SELECT EXISTS (

SELECT 'Что-то' FROM orders WHERE orders.id\_item = OLD.id\_item AND orders.ship\_date IS NULL AND orders.status = 'A'

) INTO being\_shipped;

IF being\_shipped IS TRUE THEN

RAISE EXCEPTION 'Нельзя удалить товар так как он доставляется';

END IF;

RETURN NULL;

END;

$$ LANGUAGE plpgsql;

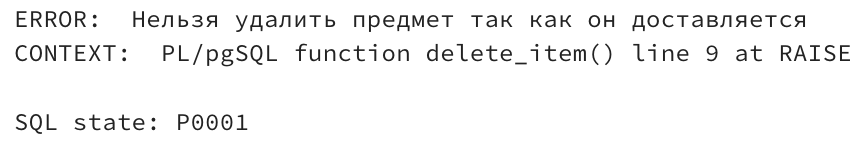
CREATE TRIGGER delete\_item\_trigger

BEFORE DELETE ON items

FOR EACH ROW

EXECUTE FUNCTION delete\_item();

delete from items where id\_order = 16



***Создать триггер UPDATE.***

CREATE FUNCTION update\_items\_total()

RETURNS TRIGGER AS $$

BEGIN

UPDATE items

SET total = items.quantity \* NEW.pr\_price

WHERE id\_product = NEW.id\_product;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

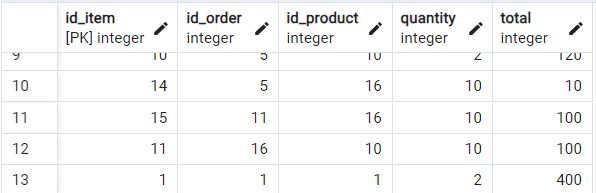
CREATE TRIGGER product\_update\_trigger

AFTER UPDATE ON products

FOR EACH ROW EXECUTE FUNCTION update\_items\_total();

update products

set pr\_price = 200 where id\_product = 1



*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:*

CREATE OR REPLACE FUNCTION delete\_product()

RETURNS TRIGGER

AS $$

BEGIN

DELETE FROM items WHERE id\_product = OLD.id;

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER delete\_product\_trigger

BEFORE DELETE ON products

FOR EACH ROW

EXECUTE FUNCTION delete\_product();

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

CREATE OR REPLACE FUNCTION on\_alter\_or\_drop\_table() RETURNS event\_trigger

LANGUAGE plpgsql

AS $$

BEGIN

RAISE EXCEPTION 'Cannot alter or drop table';

END;

$$;

CREATE EVENT TRIGGER alter\_table\_block

ON table\_rewrite

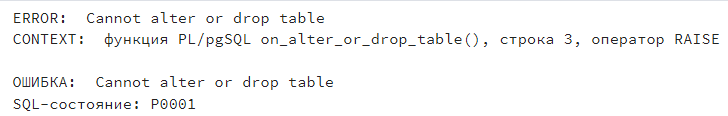
EXECUTE PROCEDURE on\_alter\_or\_drop\_table();

CREATE EVENT TRIGGER drop\_table\_block

ON sql\_drop

EXECUTE PROCEDURE on\_alter\_or\_drop\_table();

drop table test



*Функции row\_number(), rank (), dense\_rank()*

SELECT

id\_product,

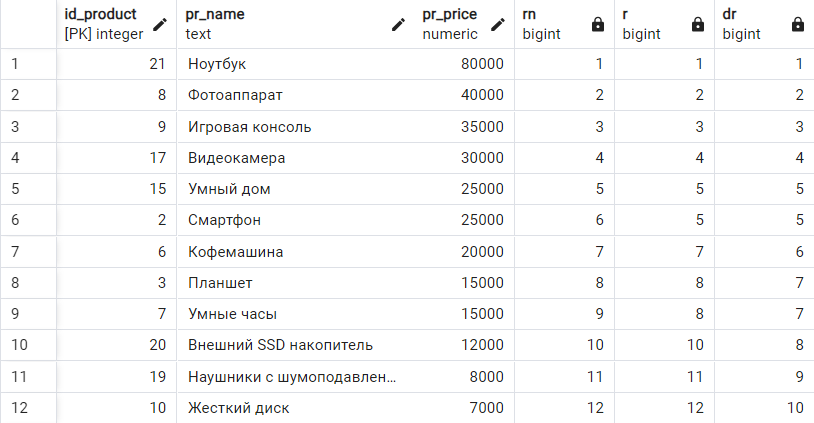
pr\_name,

pr\_price,

ROW\_NUMBER() OVER (ORDER BY pr\_price DESC) AS rn,

RANK() OVER (ORDER BY pr\_price DESC) AS r,

DENSE\_RANK() OVER (ORDER BY pr\_price DESC) AS dr

FROM products; 

CREATE OR REPLACE FUNCTION instead\_of\_delete\_products()

RETURNS TRIGGER AS $$

BEGIN

RAISE EXCEPTION 'Deleting products is not allowed!';

END;

$$ LANGUAGE plpgsql;

***Триггер замещения***

CREATE OR REPLACE FUNCTION instead\_of\_delete\_orders\_by\_customers()

RETURNS TRIGGER AS $$

BEGIN

RAISE EXCEPTION 'Deleting orders by customers is not allowed!';

END;

$$ LANGUAGE plpgsql;

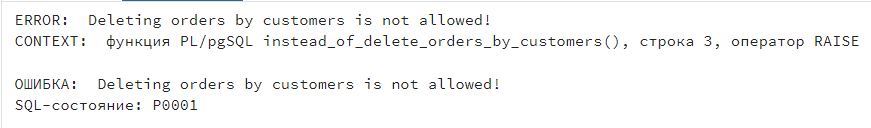
CREATE TRIGGER instead\_of\_delete\_orders\_by\_customers

INSTEAD OF DELETE ON orders\_by\_customers

FOR EACH ROW

EXECUTE FUNCTION instead\_of\_delete\_orders\_by\_customers();

delete from orders\_by\_customers where id\_order = 1



***По названию поставщика выдать названия всех товаров, поставленные данным поставщиком, с указанием для каждого товара даты последней поставки. Использовать явный курсор***

DO $$

DECLARE

companyname TEXT := 'ИП Петров'; -- Замените 'Название компании' на имя нужной компании

cur CURSOR FOR SELECT p.pr\_name, o.order\_date

FROM products p

INNER JOIN items i ON p.id\_product = i.id\_product

INNER JOIN orders o ON i.id\_order = o.id\_order

INNER JOIN customers c ON o.id\_customer = c.id\_customer

WHERE c.company\_name = companyname

ORDER BY o.order\_date DESC;

rec RECORD;

BEGIN

OPEN cur;

LOOP

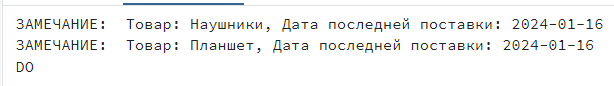
FETCH cur INTO rec;

EXIT WHEN NOT FOUND;

RAISE NOTICE 'Товар: %, Дата последней поставки: %', rec.pr\_name, rec.order\_date;

END LOOP;

CLOSE cur;

END $$; 

***По паре дат выдать таблицу товаров, по которым были операции в этот******период с указанием оборота товара в две колонки: приход, расход.******Использовать неявный курсор.***

DO $$

DECLARE

start\_date DATE := '2024-01-14';

end\_date DATE := '2024-01-20';

rec RECORD;

BEGIN

FOR rec IN SELECT p.pr\_name, o.order\_date, o.ship\_date

FROM products p

INNER JOIN items i ON p.id\_product = i.id\_product

INNER JOIN orders o ON i.id\_order = o.id\_order

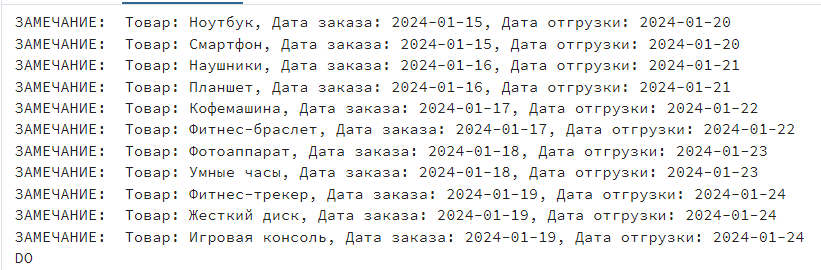
WHERE o.order\_date BETWEEN start\_date AND end\_date

ORDER BY o.order\_date, o.ship\_date

LOOP

RAISE NOTICE 'Товар: %, Дата заказа: %, Дата отгрузки: %', rec.pr\_name, rec.order\_date, rec.ship\_date;

END LOOP;

END $$; 

***Увеличить стоимость всех операций в заданном году на 10%. Использовать******неявный курсор.***

DO $$

DECLARE

target\_date DATE := '2023-01-13';

rec RECORD;

BEGIN

FOR rec IN SELECT i.iditem, i.total

FROM items i

INNER JOIN orders o ON i.idorder = o.idorder

WHERE o.orderdate = target\_date

LOOP

UPDATE items

SET total = total \* 1.1

WHERE iditem = rec.iditem;

END LOOP;

END $$;

***По названию покупателя выбрать все его операции и за каждую 3 операцию******определить бонус в 5% от стоимости операции. Результатом процедуры******должна быть таблица со следующими столбцами: Дата |сумма операции|******бонус***

CREATE OR REPLACE PROCEDURE apply\_bonus\_to\_customer(customer\_name TEXT)

LANGUAGE plpgsql

AS $$

DECLARE

rec RECORD;

operation\_count INTEGER := 0;

BEGIN

FOR rec IN SELECT o.orderdate, i.total

FROM orders o

INNER JOIN items i ON o.idorder = i.idorder

INNER JOIN customers c ON o.idcustomers = c.idcustomers

WHERE c.companyname = customer\_name

ORDER BY o.orderdate

LOOP

operation\_count := operation\_count + 1;

IF operation\_count % 3 = 0 THEN

UPDATE items

SET total = total \* 1.05

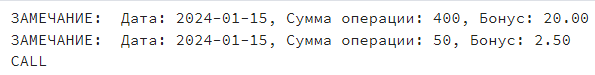
WHERE idorder = (SELECT o.idorder FROM orders o WHERE o.orderdate = rec.orderdate);

END IF;

RAISE NOTICE 'Дата: %, Сумма операции: %, Бонус: %', rec.orderdate, rec.total, (rec.total \* 0.05);

END LOOP;

END $$;

call apply\_bonus\_to\_customer('ООО Рога и копыта') 

***Для заданного периода времени подсчитать промежуточные итоги по каждому товару, с которым были операции в этот период. Итоги по товару должны иметь такой вид: Название товара | Дата операции| количество | промежуточный итог = итог за предыдущую дату +(-)количество***

DO $$

DECLARE

rec record;

prev\_quantity integer := 0;

prev\_date date := '2024-01-15';

prev\_product\_id integer := 0;

BEGIN

FOR rec IN

SELECT o.order\_date, i.id\_product, p.pr\_name, i.quantity

FROM orders o

JOIN items i ON o.id\_order = i.id\_order

JOIN products p ON i.id\_product = p.id\_product

WHERE o.order\_date BETWEEN '2024-01-15' AND '2024-01-16'

ORDER BY i.id\_product, o.order\_date

LOOP

IF rec.id\_product = prev\_product\_id THEN

IF rec.order\_date = prev\_date THEN

prev\_quantity := prev\_quantity + rec.quantity;

ELSE

RAISE NOTICE 'Название товара: %, Дата операции: %, Количество: %, Промежуточный итог: %', rec.pr\_name, prev\_date, prev\_quantity, prev\_quantity;

prev\_quantity := rec.quantity;

prev\_date := rec.order\_date;

END IF;

ELSE

RAISE NOTICE 'Название товара: %, Дата операции: %, Количество: %, Промежуточный итог: %', rec.pr\_name, prev\_date, prev\_quantity, prev\_quantity;

prev\_quantity := rec.quantity;

prev\_date := rec.order\_date;

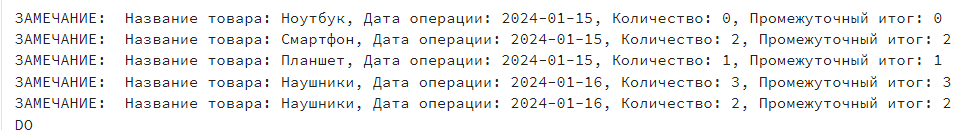
prev\_product\_id := rec.id\_product;

END IF;

END LOOP;

RAISE NOTICE 'Название товара: %, Дата операции: %, Количество: %, Промежуточный итог: %', rec.pr\_name, prev\_date, prev\_quantity, prev\_quantity;

END;

$$ LANGUAGE plpgsql;**