

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего
образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №5 «процедуры, функции, триггеры в PostgreSQL»

по дисциплине «Проектирование и реализация баз данных»

Автор: Трубников А.П

Факультет: ИКТ

Группа: K3239

Преподаватель: Говорова М.М.



Санкт-Петербург 2024

Цель работы:

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

Практическое задание:

1. Создать процедуры/функции 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.

Задание 1.1 Создание хранимых процедур.

Для снижения цен на книги, которые находятся на базе в количестве, превышающем 1000 штук.

Изначальное заполнение бд:

```
INSERT INTO "Lab3"."Circulation"(  
  "Number of copies", "Balance in stock", "Code ISBN", "Circulation date", "Circulation code", "Price"  
VALUES (1001, 1001, 991, '2024-01-01', 11, 1000); |
```

Команды в sql shell:

```
\c Lab4
```

```
CREATE OR REPLACE FUNCTION reduce_price_if_excess()
```

```
RETURNS TABLE("ID" INT, "Number of copies" INT, "Price" INT) AS $$
```

```
BEGIN
```

```
  RETURN QUERY
```

```
  UPDATE "Lab3"."Circulation"
```

```
  SET "Price" = "Price" * 0.9
```

```
  WHERE "Number of copies" > 1000
```

```
  RETURNING "Circulation code" AS "ID", "Number of copies", "Price";
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

Вызов:

```
SELECT reduce_price_if_excess();
```

Результ

```


1 SELECT ci."Price"
2     FROM "Lab3"."Circulation" ci
3     WHERE ci."Code ISBN" = '991';


```


Data Output


Messages


Notifications
























	Price		
	integer		
1	900		

Задание 1.2 Создание хранимых процедур.

Для ввода новой книги.

```

CREATE OR REPLACE PROCEDURE insert_new_book(
    IN book_title character varying(50),
    IN year_of_writing date,
    IN year_of_publication date,
    IN author_first_name character varying(50),
    IN author_last_name character varying(50),

```

```

    IN author_surname character varying(50),
    IN author_email character varying(50),
    IN book_number integer,
    IN authorship_id integer,
    IN category_name character varying(50),
    IN categorization_id integer
)
LANGUAGE plpgsql
AS $$
DECLARE
    author_code integer;
    category_code integer;
BEGIN
    SELECT "Author code" INTO author_code
    FROM "Lab3"."Author"
    WHERE "First name" = author_first_name
    AND "Last name" = author_last_name
    AND surname = author_surname;

    IF author_code IS NULL THEN
        SELECT MAX("Author code") + 1 INTO author_code FROM "Lab3"."Author";
        INSERT INTO "Lab3"."Author"("Author code", "E-mail", "First name", "Last name",
surname)
        VALUES(author_code, author_email, author_first_name, author_last_name,
author_surname);
    END IF;

    SELECT "Category code" INTO category_code
    FROM "Lab3"."Book category"
    WHERE "Name" = category_name;

    IF category_code IS NULL THEN

```

```
SELECT MAX("Category code") + 1 INTO category_code FROM "Lab3"."Book  
category";
```

```
INSERT INTO "Lab3"."Book category"("Category code", "Name")
```

```
VALUES(category_code, category_name);
```

```
END IF;
```

```
INSERT INTO "Lab3"."Book"("Book number", "Book title", "Year of writing", "Year of  
publication")
```

```
VALUES(book_number, book_title, year_of_writing, year_of_publication);
```

```
INSERT INTO "Lab3"."Authorship"("Number of author", "Authorship ID", "Author code",  
"Book number")
```

```
VALUES(1, authorship_id, author_code, book_number);
```

```
INSERT INTO "Lab3"."Categorization"("Category priority", "Categorization ID", "Book  
number", "Category code")
```

```
VALUES(1, categorization_id, book_number, category_code);
```

```
END;
```

```
$$;
```

Вызов:

```
CALL insert_new_book('Тест названия книги','2022-01-01','2022-01-01','Тест  
Имя автора','Фамилия автора','Отчество автора','Email автора',14,12,'Название  
категории',12);
```

```

Lab4=# CREATE OR REPLACE PROCEDURE insert_new_book(
Lab4=#     IN book_title character varying(50),
Lab4=#     IN year_of_writing date,
Lab4=#     IN year_of_publication date,
Lab4=#     IN author_first_name character varying(50),
Lab4=#     IN author_last_name character varying(50),
Lab4=#     IN author_surname character varying(50),
Lab4=#     IN author_email character varying(50),
Lab4=#     IN book_number integer,
Lab4=#     IN authorship_id integer,
Lab4=#     IN category_name character varying(50),
Lab4=#     IN categorization_id integer
Lab4=# )
Lab4=# LANGUAGE plpgsql
Lab4=# AS $$
Lab4=# DECLARE
Lab4=#     author_code integer;
Lab4=#     category_code integer;
Lab4=# BEGIN
Lab4=#     SELECT "Author code" INTO author_code
Lab4=#     FROM "Lab3"."Author"
Lab4=#     WHERE "First name" = author_first_name
Lab4=#     AND "Last name" = author_last_name
Lab4=#     AND surname = author_surname;
Lab4=#
Lab4=#     IF author_code IS NULL THEN
Lab4=#         SELECT MAX("Author code") + 1 INTO author_code FROM "Lab3"."Author";
Lab4=#         INSERT INTO "Lab3"."Author"("Author code", "E-mail", "First name", "Last name", surname)
Lab4=#         VALUES(author_code, author_email, author_first_name, author_last_name, author_surname);
Lab4=#     END IF;
Lab4=#
Lab4=#     SELECT "Category code" INTO category_code
Lab4=#     FROM "Lab3"."Book category"
Lab4=#     WHERE "Name" = category_name;
Lab4=#
Lab4=#     IF category_code IS NULL THEN
Lab4=#         SELECT MAX("Category code") + 1 INTO category_code FROM "Lab3"."Book category";
Lab4=#         INSERT INTO "Lab3"."Book category"("Category code", "Name")
Lab4=#         VALUES(category_code, category_name);
Lab4=#     END IF;
Lab4=#
Lab4=#     INSERT INTO "Lab3"."Book"("Book number", "Book title", "Year of writing", "Year of publication")
Lab4=#     VALUES(book_number, book_title, year_of_writing, year_of_publication);
Lab4=#
Lab4=#     INSERT INTO "Lab3"."Authorship"("Number of author", "Authorship ID", "Author code", "Book number")
Lab4=#     VALUES(1, authorship_id, author_code, book_number);
Lab4=#
Lab4=#     INSERT INTO "Lab3"."Categorization"("Category priority", "Categorization ID", "Book number", "Category code")
Lab4=#     VALUES(1, categorization_id, book_number, category_code);
Lab4=# END;
Lab4=# $$;
Lab4=#
Lab4=# CREATE PROCEDURE
Lab4=# CALL insert_new_book('Тест названия книги', '2022-01-01', '2022-01-01', 'Тест Имя автора', 'Фамилия автора', 'Отчество автора', 'Email автора', 14, 12, 'Название категории', 12);
Lab4=#

```

Результат:

Query

Query History

Scratch Pad X

1 SELECT * FROM "Lab3"."Book"

2 ORDER BY "Book number" ASC

Data Output

Messages

Notifications

Book number

Book title

Year of writing

Year of publication

[PK] integer

character varying (50)

date

date

4

7

2000-01-01

2015-10-10

5

9

2006-01-01

2006-02-01

6

10

20024-01-01

2024-02-01

7

11

2002-03-25

2004-07-25

8

12

2022-01-01

2022-02-01

9

13

2022-01-01

2022-02-01

10

14

2022-01-01

2022-01-01

Проектирование баз данных

Базы данных. Язык SQL

Базы данных2.0. Язык SQL

SimpleBD

Новая книга

Новая книга

Тест названия книги

✓ Successfully run. Total query runtime: 96 msec. 10 rows affected. ✕

Задание 1.3 Создание хранимых процедур.

Для ввода нового заказа.

```
CREATE OR REPLACE PROCEDURE insert_new_order(
    IN circulation code integer,
```

```

    IN number_of_copies integer,
    IN first_name_customer character varying(50),
    IN last_name_customer character varying(50),
    IN surname_customer character varying(50),
    IN address_customer character varying(100),
    IN phone_number_customer character varying(50),
    IN first_name_employee character varying(50),
    IN last_name_employee character varying(50),
    IN surname_employee character varying(50),
    IN post_employee character varying(100),
    IN phone_number_employee character varying(50),
    IN act_number integer,
    IN act character varying(100),
    IN prepayment_invoice integer,
    IN order_date date,
    IN order_status character varying(50),
    IN order_time date,
    IN balance_account integer,
    IN payment_state character varying(50)
)
LANGUAGE plpgsql
AS $$
DECLARE
    employee_code integer;
    code_customer integer;
    order_code integer;
    placing_id integer;
BEGIN
    SELECT "Employee code" INTO employee_code

```



```

FROM "Lab3"."Employee"

WHERE "First name" = first_name_employee
AND "Last name" = last_name_employee
AND "Surname" = surname_employee;

IF employee_code IS NULL THEN

    SELECT MAX("Employee code") + 1 INTO employee_code FROM
"Lab3"."Employee";

    INSERT INTO "Lab3"."Employee"("Employee code", "Post", "First name",
"Surname", "Last name", "Phone number")

    VALUES(employee_code, post_employee, first_name_employee,
surname_employee, last_name_employee, phone_number_employee);

END IF;


SELECT "Code Customer" INTO code_customer
FROM "Lab3"."Customer"

WHERE "First name" = first_name_customer
AND "Last name" = last_name_customer
AND "Surname" = surname_customer;

IF code_customer IS NULL THEN

    SELECT MAX("Code Customer") + 1 INTO code_customer FROM
"Lab3"."Customer";

    INSERT INTO "Lab3"."Customer"("First name", "Code Customer", "Last
name", "Surname", "Address", "Phone number")

    VALUES(first_name_customer, code_customer, last_name_customer,
surname_customer, address_customer, phone_number_customer);

END IF;


SELECT MAX("Order code") + 1 INTO order_code FROM "Lab3"."Order";

```

```
INSERT INTO "Lab3"."Order"("Order code", "Code customer", "Act number",  
"Act", "Prepayment invoice", "Order date", "Order status", "Order time", "Balance  
account", "Payment state", "Employee code")
```

```
VALUES (order_code, code_customer, act_number, act, prepayment_invoice,  
order_date, order_status, order_time, balance_account, payment_state,  
employee_code);
```

```
SELECT MAX("Placing ID") + 1 INTO placing_id FROM "Lab3"."Placing an  
order";
```

```
INSERT INTO "Lab3"."Placing an order"("Number of copies", "Placing ID",  
"Order code", "Circulation code")
```

```
VALUES (number_of_copies, placing_id, order_code, circulation_code);
```

```
END;
```

```
$$;
```

```
Lab4=# CREATE OR REPLACE PROCEDURE insert_new_order(  
Lab4=# IN circulation_code integer,  
Lab4=# IN number_of_copies integer,  
Lab4=# IN first_name_customer character varying(50),  
Lab4=# IN last_name_customer character varying(50),  
Lab4=# IN surname_customer character varying(50),  
Lab4=# IN address_customer character varying(100),  
Lab4=# IN phone_number_customer character varying(50),  
Lab4=# IN first_name_employee character varying(50),  
Lab4=# IN last_name_employee character varying(50),  
Lab4=# IN surname_employee character varying(50),  
Lab4=# IN post_employee character varying(100),  
Lab4=# IN phone_number_employee character varying(50),  
Lab4=# IN act_number integer,  
Lab4=# IN act character varying(100),  
Lab4=# IN prepayment_invoice integer,  
Lab4=# IN order_date date,  
Lab4=# IN order_status character varying(50),  
Lab4=# IN order_time date,  
Lab4=# IN balance_account integer,  
Lab4=# IN payment_state character varying(50)  
Lab4=# )  
Lab4=# LANGUAGE plpgsql  
Lab4=# AS $$  
Lab4=# DECLARE  
Lab4=# employee_code integer;  
Lab4=# code_customer integer;  
Lab4=# order_code integer;  
Lab4=# placing_id integer;  
Lab4=# BEGIN  
Lab4=# SELECT "Employee code" INTO employee_code  
Lab4=# FROM "Lab3"."Employee"  
Lab4=# WHERE "First name" = first_name_employee  
Lab4=# AND "Last name" = last_name_employee  
Lab4=# AND "Surname" = surname_employee;  
Lab4=#  
Lab4=# IF employee_code IS NULL THEN  
Lab4=# SELECT MAX("Employee code") + 1 INTO employee_code FROM "Lab3"."Employee";  
Lab4=# INSERT INTO "Lab3"."Employee"("Employee code", "Post", "First name", "Surname", "Last name", "Phone number")  
Lab4=# VALUES(employee_code, post_employee, first_name_employee, surname_employee, last_name_employee, phone_number_employee);  
Lab4=# END IF;  
Lab4=#  
Lab4=# SELECT "Code Customer" INTO code_customer  
Lab4=# FROM "Lab3"."Customer"  
Lab4=# WHERE "First name" = first_name_customer  
Lab4=# AND "Last name" = last_name_customer  
Lab4=# AND "Surname" = surname_customer;  
Lab4=#  
Lab4=# IF code_customer IS NULL THEN  
Lab4=# SELECT MAX("Code Customer") + 1 INTO code_customer FROM "Lab3"."Customer";  
Lab4=# INSERT INTO "Lab3"."Customer"("First name", "Code Customer", "Last name", "Surname", "Address", "Phone number")  
Lab4=# VALUES(first_name_customer, code_customer, last_name_customer, surname_customer, address_customer, phone_number_customer);  
Lab4=# END IF;  
Lab4=#  
Lab4=# SELECT MAX("Order code") + 1 INTO order_code FROM "Lab3"."Order";  
Lab4=# INSERT INTO "Lab3"."Order"("Order code", "Code customer", "Act number", "Act", "Prepayment invoice", "Order date", "Order status", "Order time", "Balance account", "  
Lab4=# payment state", "Employee code")  
Lab4=# VALUES (order_code, code_customer, act_number, act, prepayment_invoice, order_date, order_status, order_time, balance_account, payment_state, employee_code);  
Lab4=#  
Lab4=# SELECT MAX("Placing ID") + 1 INTO placing_id FROM "Lab3"."Placing an order";  
Lab4=# INSERT INTO "Lab3"."Placing an order"("Number of copies", "Placing ID", "Order code", "Circulation code")  
Lab4=# VALUES (number_of_copies, placing_id, order_code, circulation_code);  
Lab4=# END;  
Lab4=# $$;  
Lab4=# CREATE PROCEDURE
```

Пример выполнения:

```
Lab4=# CALL insert_new_order(
Lab4(#   circulation_code := 3,
Lab4(#   number_of_copies := 2,
Lab4(#   first_name_customer := 'John',
Lab4(#   last_name_customer := 'Doe',
Lab4(#   surname_customer := 'Smith',
Lab4(#   address_customer := '123 Main St',
Lab4(#   phone_number_customer := '555-1234',
Lab4(#   first_name_employee := 'Jane',
Lab4(#   last_name_employee := 'Smith',
Lab4(#   surname_employee := 'Johnson',
Lab4(#   post_employee := 'Менеджер',
Lab4(#   phone_number_employee := '555-5678',
Lab4(#   act_number := 456,
Lab4(#   act := 'Act description',
Lab4(#   prepayment_invoice := 100,
Lab4(#   order_date := '2022-01-15',
Lab4(#   order_status := 'Принят в работу',
Lab4(#   order_time := '2022-01-16',
Lab4(#   balance_account := 500,
Lab4(#   payment_state := 'Paid'
Lab4(# );
ОШИБКА: Недостаточно товара на складе для заказа.
```

```
Lab4=# CALL insert_new_order(
Lab4(#   circulation_code := 7,
Lab4(#   number_of_copies := 2,
Lab4(#   first_name_customer := 'John',
Lab4(#   last_name_customer := 'Doe',
Lab4(#   surname_customer := 'Smith',
Lab4(#   address_customer := '123 Main St',
Lab4(#   phone_number_customer := '555-1234',
Lab4(#   first_name_employee := 'Jane',
Lab4(#   last_name_employee := 'Smith',
Lab4(#   surname_employee := 'Johnson',
Lab4(#   post_employee := 'Менеджер',
Lab4(#   phone_number_employee := '555-5678',
Lab4(#   act_number := 456,
Lab4(#   act := 'Act description',
Lab4(#   prepayment_invoice := 100,
Lab4(#   order_date := '2022-01-15',
Lab4(#   order_status := 'Принят в работу',
Lab4(#   order_time := '2022-01-16',
Lab4(#   balance_account := 500,
Lab4(#   payment_state := 'Paid'
Lab4(# );
CALL
```

Результат:

QueryQuery HistoryScratch Pad X

```
1 SELECT * FROM "Lab3"."Order"
2 ORDER BY "Order code" ASC
```

Data OutputMessagesNotifications

	Order code [PK] integer	Code customer integer	Act number integer	Act character varying (100)	Prepayment invoice integer	Order date date	Order status character varying (50)	Order time date	Balance account integer	Payment state character varying (50)	Employee code integer	
7	9	9	9	Акт	1500	2024-01-01	Принят в работу	2024-02-02	1500	Выполнен	2	
8	10	2	9	Акт	1001	2024-02-01	Принят в работу	2024-02-02	1001	Выполнен	2	
9	11	2	10	Акт	1001	2024-02-01	Принят в работу	2024-02-02	1001	Выполнен	2	
10	12	2	12	Example Act	1100	2022-01-01	Принят в работу	2022-02-02	1100	Paid	2	
11	13	2	13	Акт	1000	2022-01-01	Принят в работу	2022-02-02	1000	Выполнен	2	
12	14	2	14	Example Act	1100	2022-01-01	Принят в работу	2022-02-02	1100	Paid	2	
13	15	1	456	Act description	100	2022-01-15	Принят в работу	2022-01-16	500	Paid	4	

Задание 2 Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL (согласно индивидуальному заданию, часть 5).

Триггер имеет следующие инструкции: Если в содержании заказа новая строка, то триггер должен проверить баланс тиража, если количество штук в заказе меньше баланса на складе, то строка вставляется и баланс на складе уменьшается, на число заказа, если количество в заказе больше чем на складе, то отказ

Код функции и триггера:

```
CREATE OR REPLACE FUNCTION check_copies_balance()

RETURNS TRIGGER AS $$

DECLARE temp_balance INT;
BEGIN

    SELECT "Balance in stock" INTO temp_balance

    FROM "Lab3"."Circulation"

    WHERE "Circulation code" = NEW."Circulation code";

    IF NEW."Number of copies" <= temp_balance THEN

        UPDATE "Lab3"."Circulation"

        SET "Balance in stock" = temp_balance - NEW."Number of copies"

        WHERE "Circulation code" = NEW."Circulation code";

        RETURN NEW;

    ELSE

        RAISE EXCEPTION 'Недостаточно товара на складе для заказа.';

    END IF;

END;

$$ LANGUAGE plpgsql;


CREATE TRIGGER check_copies_balance_trigger

BEFORE INSERT ON "Lab3"."Placing an order"

FOR EACH ROW

EXECUTE FUNCTION check_copies_balance();
```

```

Lab4=# CREATE OR REPLACE FUNCTION check_copies_balance()
Lab4-# RETURNS TRIGGER AS $$
Lab4$# DECLARE temp_balance INT;
Lab4$# BEGIN
Lab4$# SELECT "Balance in stock" INTO temp_balance
Lab4$# FROM "Lab3"."Circulation"
Lab4$# WHERE "Circulation code" = NEW."Circulation code";
Lab4$# IF NEW."Number of copies" <= temp_balance THEN
Lab4$# UPDATE "Lab3"."Circulation"
Lab4$# SET "Balance in stock" = temp_balance - NEW."Number of copies"
Lab4$# WHERE "Circulation code" = NEW."Circulation code";
Lab4$# RETURN NEW;
Lab4$# ELSE
Lab4$# RAISE EXCEPTION 'Недостаточно товара на складе для заказа.';
Lab4$# END IF;
Lab4$# END;
Lab4$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
Lab4=# CREATE TRIGGER check_copies_balance_trigger
Lab4-# BEFORE INSERT ON "Lab3"."Placing an order"
Lab4-# FOR EACH ROW
Lab4-# EXECUTE FUNCTION check_copies_balance();
CREATE TRIGGER

```

Проверка работы триггера:

Создаем новый order и новый placing an order

INSERT INTO "Lab3"."Placing an order"(

"Number of copies", "Placing ID", "Order code", "Circulation code")

VALUES (500, 12, 13, 11);

Проверяем данные.

6	1001	501	991	2024-01-01	11	900
---	------	-----	-----	------------	----	-----

Количество уменьшилось на 500

Создаем новый order и новый placing an order

INSERT INTO "Lab3"."Placing an order"(

"Number of copies", "Placing ID", "Order code", "Circulation code")

VALUES (502, 13, 14, 11);

```
4 INSERT INTO "Lab3"."Placing an order"(  
5   "Number of copies", "Placing ID", "Order code", "Circulation code")  
6   VALUES (502, 13, 14, 11);
```

Data Output [Messages](#) Notifications

ERROR: Недостаточно товара на складе для заказа.
CONTEXT: функция PL/pgSQL check_copies_balance(), строка 13, оператор RAISE

ОШИБКА: Недостаточно товара на складе для заказа.
SQL state: P0001

Товара недостаточно на складе, так как на данный момент там 501 копия, когда в заказе 502

Вывод:

1. Овладение созданием и использованием процедур, функций и триггеров в PostgreSQL позволяет значительно улучшить эффективность работы с базой данных.
2. Применение процедур и функций позволяет уменьшить повторяемость кода, облегчая его поддержку и обновление.
3. Триггеры могут быть использованы для автоматизации определенных действий при изменении данных в базе, что повышает целостность и безопасность информации.
4. При правильном использовании процедур, функций и триггеров в PostgreSQL можно значительно ускорить выполнение запросов и упростить сложные операции с данными.
5. Владение этими инструментами расширяет возможности разработчика при работе с базой данных и повышает профессиональный уровень в области администрирования и разработки баз данных.