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: К3239

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.

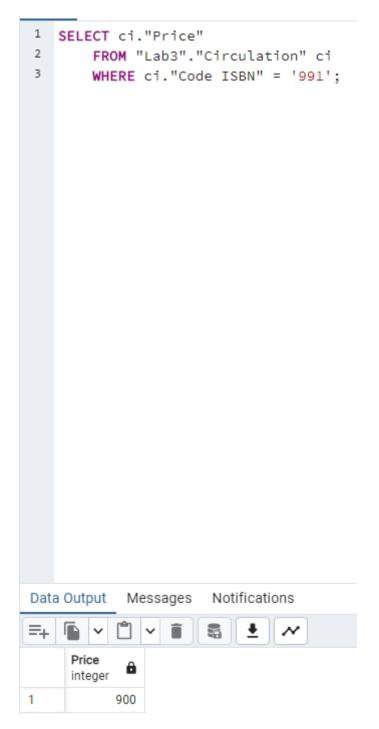
Задание 1.1 Создание хранимых процедур.

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

Изначальное заполнение бд:

Результ

```
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();
```



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
```

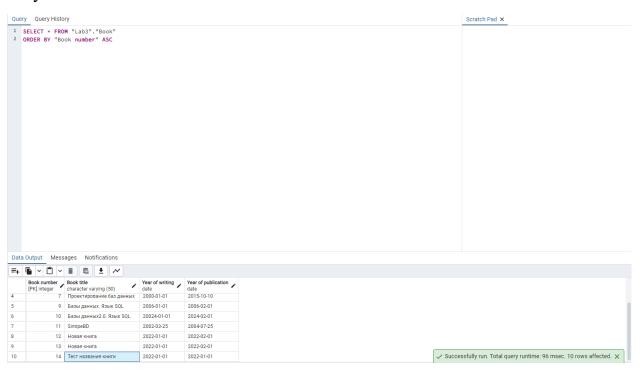
```
SELECT MAX("Category code") + 1 INTO category code FROM "Lab3"."Book
category";
    INSERT INTO "Lab3". "Book category" ("Category code", "Name")
    VALUES(category code, category name);
  END IF;
  INSERT INTO "Lab3". "Book" ("Book number", "Book title", "Year of writing", "Year of
publication")
  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('Tecт названия книги','2022-01-01','2022-01-01','Tecт Имя автора','Фамилия автора','Отчество автора','Email автора',14,12,'Название категории',12);

```
LANGE OF CARTE ON REPLACE PROCEDURE insert_new_book(

LANGE IN year_of_molicities atts,
LANGE IN year_of_molicities,
LANGE IN year
```

Результат:



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;

\$\$;

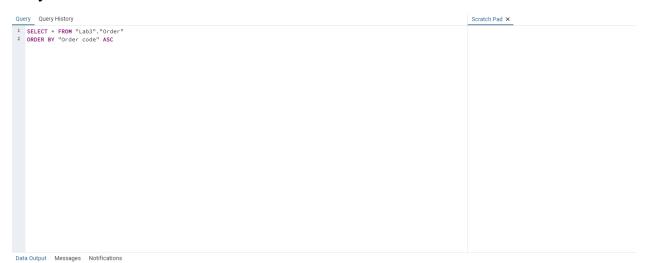
```
Space (CART ON SENACE PROCEDUME Assert, new_neter(
Labe(s) 11 (cricalisin_code integer,
Labe(s) 12 (cricalisin_code integer,
Labe(s) 13 (cricalisin_code integer,
Labe(s) 14 (cricalisin_code integer,
Labe(s) 15 (cricalisin_code character varying(s)),
Labe(s) 18 (cricalisin_code character varying(s)),
Labe(s) 19 (cricalisin_code character varying(s)),
Labe(s) 10 (cricalisin_code character varying(s),
Labe(s) 10 (cric
```

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 := 'John',
Lab4(# last_name_customer := 'Smith',
Lab4(# address_customer := 'Sist_n',
Lab4(# phone_number_customer := 'Sist_n',
Lab4(# phone_number_customer := 'Sist_n',
Lab4(# last_name_employee := 'Jane',
Lab4(# last_name_employee := 'Jane',
Lab4(# last_name_employee := 'Smith',
Lab4(# opst_employee := 'Sist_n',
Lab4(# opst_employee := 'Sist_n',
Lab4(# act_number := 456,
Lab4(# act_number := 456,
Lab4(# act_number := 456,
Lab4(# cris_ntame_employee := 'Johnson',
Lab4(# act_number := 2022-01.15',
Lab4(# act_number := 2022-01.15',
Lab4(# act_number := 2022-01.16',
Lab4(# act_number := 2022-01.16',
Lab4(# blab4(# act_number := 2022-01.16',
Lab4(# blab4(# act_number := 2022-01.16',
Lab4(# blab4(# act_number := 2022-01.16',
Lab4(# act_number := 2022-01.16',
Lab4(
```

```
Lab4=# CALL insert_new_order(
Lab4(# circulation_code := 7,
Lab4(# number_of_copies := 2,
Lab4(# first_name_customer := 'John',
Lab4(# last_name_customer := 'John',
Lab4(# last_name_customer := 'Smith',
Lab4(# surname_customer := '33 Nain_St',
Lab4(# phone_number_customer := '355-1234',
Lab4(# first_name_employee := 'Johnson',
Lab4(# last_name_employee := 'Smith',
Lab4(# last_name_employee := 'Simith',
Lab4(# post_employee := 'Mbhegaxep',
Lab4(# post_employee := 'Mbhegaxep',
Lab4(# act_number_employee := '1655-5678',
Lab4(# act_number := '486,
Lab4(# act_number := '846,
Lab4(# act_number := '846,
Lab4(# order_state := 'Paid',
Lab4(# order_state := 'Paid',
Lab4(# order_state := 'Paid',
Lab4(# order_state := 'Paid',
Lab4(# payment_state := 'Paid',
Lab4(# );
CALL
```

Результат:



=+											
	Order code [PK] integer	Code customer /	Act number /	Act character varying (100)	Prepayment invoice integer	Order date /	Order status character varying (50)	Order time /	Balance account integer	Payment state character varying (50)	Employee code integer
7	9	9	9	AKT	1500	2024-01-01	Принят в работу	2024-02-02	1500	Выполнен	2
8	10	2	9	AKT	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.



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);

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

VALUES (502, 13, 14, 11);

Data Output Messages Notifications

ERROR: Hegocratowno товара на складе для заказа.

CONTEXT: функция PL/pgSQL check_copies_balance(), строка 13, оператор RAISE

OUMDEKA: Недостаточно товара на складе для заказа.
SQL state: P8001
```

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.