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: Космач М.Р.

Факультет: ИКТ

Группа: К3239

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Оглавление

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:

Вариант 2 (max - 8 баллов)

1. Создать процедуры/функции согласно индивидуальному заданию (часть 4).
2. Создать авторский триггер по варианту индивидуального задания.

Выполнение

Создайте хранимые процедуры:

1. **О текущей сумме вклада и сумме начисленного за месяц процента для заданного клиента;**

```
CREATE OR REPLACE FUNCTION bank.get_deposit_info(client_id INTEGER) RETURNS TABLE (
    id_client INTEGER,
    full_name CHARACTER VARYING(50),
    deposit_sum bigint,
    deposit_perc_sum bigint
) AS
$$ BEGIN RETURN QUERY
SELECT client.id_client, client.full_name, SUM(initial_deposit_amount + deposit_amount), SUM(payments) FROM Bank.deposit
JOIN Bank.accruals ON deposit.contract_number = accruals.contract_number
JOIN Bank.client ON deposit.id_client = client.id_client
WHERE deposit.actual_date_of_closing_deposit IS NULL AND deposit.id_client = client_id
GROUP BY client.id_client, client.full_name;
END;
$$
```

LANGUAGE plpgsql

```
bank=# CREATE OR REPLACE FUNCTION bank.get_deposit_info(client_id INTEGER) RETURNS TABLE (
bank(# id_client INTEGER,
bank(# full_name CHARACTER VARYING(50),
bank(# deposit_sum integer,                                                                    t
bank(# deposit_perc_sum INTEGER
bank(# ) AS
bank-# $$ BEGIN RETURN QUERY
bank$# SELECT client.id_client, client.full_name, SUM(initial_deposit_amount + deposit_amount), SUM(payments) F
bank$# JOIN Bank.accruals ON deposit.contract_number = accruals.contract_number
bank$# JOIN Bank.client ON deposit.id_client = client.id_client
bank$# WHERE deposit.actual_date_of_closing_deposit IS NULL AND deposit.id_client = client_id
bank$# GROUP BY client.id_client, client.full_name;
bank$# END;
bank$# $$
bank-# LANGUAGE plpgsql
bank-# ;
CREATE FUNCTION                                                                                R
```

```
bank=# SELECT * FROM bank.get_deposit_info(2);
 id_client | full_name | deposit_sum | deposit_perc_sum
-----------+-----------+-------------+------------------
         2 | Jane Doe  |       70000 |            10000
(1 строка)
```

- **Найти клиента банка, имеющего максимальное количество кредитовна текущий день;**

```
CREATE OR REPLACE FUNCTION bank.find_max_credits() RETURNS TABLE (
        id_client integer,
        full_name CHARACTER VARYING(50),
        credit_count bigint
) AS
$$ BEGIN RETURN QUERY
SELECT client.id_client, client.full_name, COUNT(credit.contract_number) FROM Bank.credit
JOIN Bank.client ON credit.id_client = client.id_client
WHERE credit.actual_loan_closing_date IS NULL AND credit.planned_loan_closing_date >
CURRENT_DATE AND credit.id_client = client.id_client
GROUP BY client.id_client, client.full_name
HAVING COUNT(credit.contract_number) >= (
        SELECT DISTINCT COUNT(credit.contract_number) FROM Bank.credit
        JOIN Bank.client ON credit.id_client = client.id_client
WHERE credit.actual_loan_closing_date IS NULL AND credit.planned_loan_closing_date >
CURRENT_DATE AND credit.id_client = client.id_client
GROUP BY client.id_client
);
END;
$$
LANGUAGE plpgsql;
```

```
bank=# CREATE OR REPLACE FUNCTION bank.find_max_credits() RETURNS TABLE (
bank(# id_client INTEGER,
bank(# full_name CHARACTER VARYING(50),
bank(# credit_count integer
bank(# ) AS
bank-# $$ BEGIN RETURN QUERY
bank$# SELECT client.id_client, client.full_name, COUNT(credit.contract_number) FROM Bank.credit
bank$# JOIN Bank.client ON deposit.id_client = client.id_client
bank$# WHERE deposit.actual_loan_closing_date IS NULL AND credit.planned_loan_closing_date > CURRENT_DATE AND d
eposit.id_client = client_id
bank$# GROUP BY client.id_client, client.full_name
bank$# HAVING COUNT(credit.contract_number) >= (
bank$# SELECT COUNT(credit.contract_number) FROM Bank.credit
bank$# JOIN Bank.client ON deposit.id_client = client.id_client
bank$# WHERE deposit.actual_loan_closing_date IS NULL AND credit.planned_loan_closing_date > CURRENT_DATE AND d
eposit.id_client = client_id
bank$# );
bank$# END;
bank$# $$
bank-# LANGUAGE plpgsql;
CREATE FUNCTION
```

```
bank=# SELECT * FROM bank.find_max_credits();
 id_client |    full_name    | credit_count
-----------+-----------------+--------------
         1 | John Smith      |            1
         3 | Michael Johnson |            1
```

- Найти клиентов банка, не имеющих задолженности по кредитам.

```
CREATE OR REPLACE FUNCTION bank.find_not_debtor() RETURNS TABLE (
        id_client integer,
        full_name CHARACTER VARYING(50)
) AS
$$ BEGIN RETURN QUERY
SELECT DISTINCT client.id_client, client.full_name FROM Bank.credit
JOIN Bank.client ON credit.id_client = client.id_client
JOIN Bank.payments_timetable ON credit.contract_number = payments_timetable.contract_number
WHERE credit.actual_loan_closing_date IS NULL AND credit.planned_loan_closing_date >
CURRENT_DATE;
END;
$$
LANGUAGE plpgsql;
```

```
bank=# CREATE OR REPLACE FUNCTION bank.find_not_debtor() RETURNS TABLE (
bank(# id_client integer,
bank(# full_name CHARACTER VARYING(50)
bank(# ) AS
bank-# $$ BEGIN RETURN QUERY
bank$# SELECT DISTINCT client.id_client, client.full_name FROM Bank.credit
bank$# JOIN Bank.client ON credit.id_client = client.id_client
bank$# JOIN Bank.payments_timetable ON credit.contract_number = payments_timetable.contract_number
bank$# WHERE credit.actual_loan_closing_date IS NULL AND credit.planned_loan_closing_date > CURRENT_DATE;
bank$# END;
bank$# $$
bank-# LANGUAGE plpgsql;
CREATE FUNCTION
```

```
bank=# SELECT * FROM bank.find_not_debtor();
 id_client | full_name
-----------+------------
         1 | John Smith
```

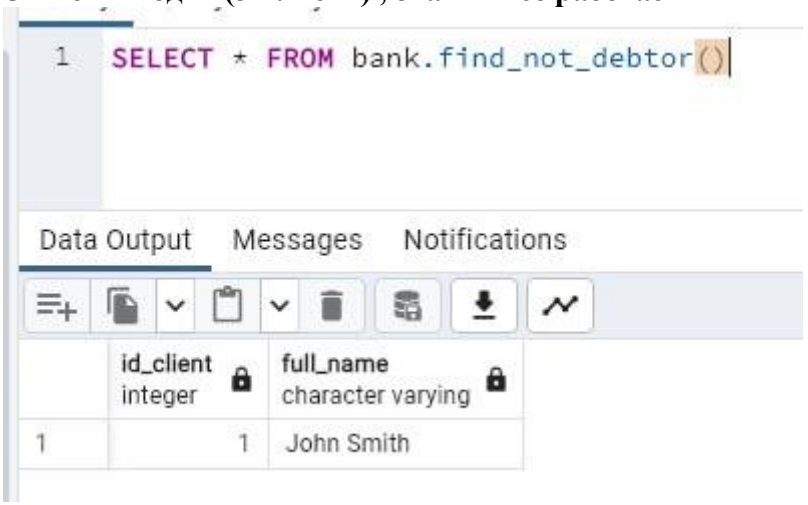
Проверка :
вот допустим у нас есть 3 клиент, у которого 1 кредит закрыт, а второй не закрыт и долги по нему есть

	imber	loan_date date	actual_loan_closing_date date	planned_loan_closing_date date	initial_loan_amount integer	current_debt integer	id_employee integer	id_client integer	currency_code integer	credit_code integer	status character var
1	1	2022-03-25	2024-01-18	2024-12-25	20000	0	1	1	1	1	Finished
2	2	2022-06-09	2023-01-04	2023-02-09	70000	10000	2	1	2	2	Active
3	3	2022-07-15	2022-10-15	2022-10-15	15000	1000	3	2	3	3	Active
4	4	2022-08-28	2022-10-28	2022-10-28	500000	0	4	2	4	4	Finished
5	5	2022-10-11	2022-12-11	2022-12-11	1000	0	5	3	5	5	Finished
6	6	2023-11-30	[null]	2024-12-18	50000	6000	2	3	1	1	Active

Data Output Messages Notifications

	ID_payments_timetable [PK] integer	payment_amount integer	actual_date_payments date	planned_data_payment date	contract_number integer
1	8	2000	[null]	2024-01-01	6
2	1	7000	2024-01-18	2024-07-26	1
3	2	30000	[null]	2024-09-09	2
4	3	14000	2022-09-15	2022-09-15	3
5	4	500000	[null]	2022-10-28	4
6	6	6500	2022-12-26	2022-12-26	1
7	7	30000	2023-01-09	2023-01-09	2

Он не выводит (3 клиент) , значит все работает



Создайте необходимые триггеры:

- Триггер для автоматического обновления статуса и долга по

кредиту после внесённой платы

```
  CREATE OR REPLACE FUNCTION proccess_timetable()

RETURNS TRIGGER AS $$

BEGIN

  IF (SELECT current_debt FROM bank.credit WHERE
credit.contract_number = NEW.contract_number) - NEW.payment_amount
<= 0 THEN

    UPDATE bank.credit

    SET status = 'Finished'

    WHERE credit.contract_number = NEW.contract_number;



    UPDATE bank.credit

    SET current_debt = 0

    WHERE credit.contract_number = NEW.contract_number;



        UPDATE bank.credit

    SET actual_loan_closing_date = CURRENT_DATE
```

```
    WHERE credit.contract_number = NEW.contract_number;


        ELSE

        UPDATE bank.credit

    SET current_debt = (SELECT current_debt FROM bank.credit WHERE
credit.contract_number = NEW.contract_number) - NEW.payment_amount

    WHERE credit.contract_number = NEW.contract_number;


  END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;


CREATE OR REPLACE TRIGGER proccess_payment

AFTER UPDATE OF actual_date_payments ON bank.payments_timetable

FOR EACH ROW

EXECUTE FUNCTION proccess_timetable();
```

```
bank=# CREATE OR REPLACE FUNCTION proccess_timetable()
bank-# RETURNS TRIGGER AS $$
bank$# BEGIN
bank$#   IF (SELECT current_debt FROM bank.credit) - NEW.payment_amount <= 0 THEN
bank$#     UPDATE bank.credit
bank$#     SET status = 'Finished'
bank$#     WHERE credit.contract_number = NEW.contract_number;
bank$#
bank$#     UPDATE bank.credit
bank$#     SET current_debt = 0
bank$#     WHERE credit.contract_number = NEW.contract_number;
bank$#
bank$# ELSE
bank$# UPDATE bank.credit
bank$#     SET current_debt = (SELECT current_debt FROM bank.credit) - NEW.payment_amount
bank$#     WHERE credit.contract_number = NEW.contract_number;
bank$#
bank$#   END IF;
bank$#   RETURN NEW;
bank$# END;
bank$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
bank=#
bank=# CREATE OR REPLACE TRIGGER proccess_payment
bank-# AFTER UPDATE OF actual_date_payments ON bank.payments_timetable
bank-# FOR EACH ROW
bank-# EXECUTE FUNCTION proccess_timetable();
CREATE TRIGGER
```

Query | Query History

```
1  SELECT * FROM bank.credit
2  ORDER BY contract_number ASC
```

Data Output | Messages | Notifications

	contract_number [PK] integer	loan_date date	actual_loan_closing_date date	planned_loan_closing_date date	initial_loan_amount integer	current_debt integer	id_employee integer	id_client integer	currency_code integer	credit_code integer	status character varying
1	1	2022-03-25	[null]	2024-12-25	20000	7000	1	1	1	1	Active
2	2	2022-06-09	2023-01-04	2023-02-09	70000	10000	2	1	2	2	Active
3	3	2022-07-15	2022-10-15	2022-10-15	15000	1000	3	2	3	3	Active
4	4	2022-08-28	2022-10-28	2022-10-28	500000	0	4	2	4	4	Finished
5	5	2022-10-11	2022-12-11	2022-12-11	1000	0	5	3	5	5	Finished
6	6	2023-11-30	[null]	2024-12-18	50000	6000	2	3	1	1	Active

	ID_payments_timetable [PK] integer	payment_amount integer	actual_date_payments date	planned_data_payment date	contract_number integer
1	1	7000	2024-01-18	2022-07-26	1
2	2	30000	2022-09-09	2022-09-09	2
3	3	14000	2022-09-15	2022-09-15	3
4	4	500000	[null]	2022-10-28	4
5	6	6500	2022-12-26	2022-12-26	1
6	7	30000	2023-01-09	2023-01-09	2

```
1  SELECT * FROM bank.credit
2  ORDER BY contract_number ASC
```

Data Output | Messages | Notifications

	contract_number [PK] integer	loan_date date	actual_loan_closing_date date	planned_loan_closing_date date	initial_loan_amount integer	current_debt integer	id_employee integer	id_client integer	currency_code integer	credit_code integer	status character varying
1	1	2022-03-25	2024-01-18	2024-12-25	20000	0	1	1	1	1	Finished
2	2	2022-06-09	2023-01-04	2023-02-09	70000	10000	2	1	2	2	Active
3	3	2022-07-15	2022-10-15	2022-10-15	15000	1000	3	2	3	3	Active
4	4	2022-08-28	2022-10-28	2022-10-28	500000	0	4	2	4	4	Finished
5	5	2022-10-11	2022-12-11	2022-12-11	1000	0	5	3	5	5	Finished
6	6	2023-11-30	[null]	2024-12-18	50000	6000	2	3	1	1	Active

Вывод

В ходе лабораторной работы была освоена работа с процедурами и триггерами.