

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

## **Отчет**

по лабораторной работе «Процедуры, функции, триггеры в PostgreSQL»  
по дисциплине «Проектирование и реализация баз данных»

Автор: Булыга Е.А.  
Факультет: ИКТ  
Группа: К32421  
Преподаватель: Говорова М.М.

**itmo**

Санкт-Петербург 2023

# Содержание

<b>1</b>	<b>Введение</b>	<b>3</b>
<b>2</b>	<b>Выполнение</b>	<b>4</b>
2.1	Таблица capitalizations . . . . .	4
2.2	Таблица cash_turnover . . . . .	6
2.3	Таблица payments . . . . .	8
2.4	Таблица payments_for_service . . . . .	10
2.5	Таблица remittances . . . . .	12
<b>3</b>	<b>Вывод</b>	<b>16</b>

# 1 Введение

**Цель работы:** овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

**Практическое задание:**

1. Создать процедуры/функции согласно индивидуальному заданию и (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.

## 2 Выполнение

### 2.1 Таблица capitalizations

Триггер на обновление баланса на счете при добавлении записи в таблицу капитализации вклада.

```
CREATE OR REPLACE FUNCTION fn_update_cash_after_capitalization()
RETURNS trigger LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
    UPDATE
        accounts
    SET
        balance = balance + NEW.money
    WHERE
        accounts.account_id =(
            SELECT
                account_id
            FROM
                deposits
            WHERE
                deposits.deposit_id = NEW.deposit_id
        );
    RETURN NEW;
END;
$BODY$;

CREATE TRIGGER update_cash_after_capitalization
AFTER INSERT
ON capitalizations
FOR EACH ROW
EXECUTE FUNCTION fn_update_cash_after_capitalization();
```

Процедура добавления записей в таблицу капитализации.

```
CREATE OR REPLACE PROCEDURE proc_deposits_capitalization()
LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
    i BIGINT;
BEGIN
    FOR i IN (
        SELECT
            deposit_id
        FROM
            deposits
    ) LOOP IF (
        (
            SELECT
                accounts.balance
            FROM
                (
                    SELECT
```

```

        deposits.account_id,
        types_of_deposits.percent
    FROM
        deposits
    LEFT JOIN types_of_deposits
    ON deposits.type_of_deposit_id =
        types_of_deposits.type_of_deposit_id
    WHERE
        deposits.account_id = i
    ) AS tbl1
    LEFT JOIN accounts ON tbl1.account_id = accounts.account_id
    ) > 0
    ) THEN
    INSERT INTO capitalizations (deposit_id, money)
    VALUES
    (
        i,
        (
            SELECT
            (
                accounts.balance * tbl1.percent / 100 / 12
            )
            FROM
            (
                SELECT
                    deposits.account_id,
                    types_of_deposits.percent
                FROM
                    deposits
                LEFT JOIN types_of_deposits
                ON deposits.type_of_deposit_id =
                    types_of_deposits.type_of_deposit_id
                WHERE
                    deposits.deposit_id = i
            ) AS tbl1
            LEFT JOIN accounts ON tbl1.account_id = accounts.account_id
        )
    );
    ELSE
    INSERT INTO capitalizations (deposit_id, money)
    VALUES
    (
        i,
        (
            SELECT
                0.01
            FROM
            (
                SELECT
                    deposits.account_id,
                    types_of_deposits.percent
                FROM
                    deposits
                LEFT JOIN types_of_deposits
                ON deposits.type_of_deposit_id =
                    types_of_deposits.type_of_deposit_id
                WHERE
                    deposits.deposit_id = i
            ) AS tbl1
            LEFT JOIN accounts ON tbl1.account_id = accounts.account_id
        )
    );

```

```

    )
  );
END IF;
END LOOP;
END;
$BODY$;

```

SQL Shell (psql)

```

bankdatabase=# SELECT * FROM accounts ORDER BY account_id;

```

account_id	user_id	bik	korr_account	inn	kpp	balance	created_at	closed_at	currency_id
1	1001	591731039	346786729259808992	1484612055	268765666	8228	2023-03-20 20:59:45,921653		2
2	1002	567592410	776627963145124192	7693145261	614293120	1140	2023-03-20 20:59:45,921653		2
3	1003	824440147	352748740887908088	1258368255	682661582	2882	2023-03-20 20:59:45,921653		2
4	1004	196242436	227536873680890256	5104831302	571539482	7948	2023-03-20 20:59:45,921653		2
5	1005	715100225	1849582583308082528	1264561546	684749868	8178	2023-03-20 20:59:45,921653		1
6	1006	105239847	41355457544376321	9748776026	283810052	2122	2023-03-20 20:59:45,921653		1
7	1007	929291628	776627963145224192	7059468784	118821657	1362	2023-03-20 20:59:45,921653		1
8	1008	498981421	767788826257377280	8408231192	712134374	0	2023-03-20 20:59:45,921653		1
9	1009	564491538	224519517363562496	4163278577	961499682	4383	2023-03-20 20:59:45,921653		1
10	1001	184741352	776627963145224192	9920565408	579837625	9898	2023-03-20 20:59:45,921653		1
11	1002	292601058	481120053472387200	9411145356	932626750	0	2023-03-20 20:59:45,921653		1
12	1005	915962263	276832715666973696	1555084199	180277873	10	2023-03-20 20:59:45,921653		1
13	1005	182021623	71560208383954496	8031171484	962077887	2429	2023-03-20 20:59:45,921653		3
14	1002	246421530	52473408818322944	7758790848	508106076	0.010260654	2023-03-20 20:59:45,921653		3
15	1003	98510713	54853498839915520	7681555595	924725342	10158.877	2023-03-20 20:59:45,921653		3
16	1003	996388498	14599632215970967	5751792279	595750776	30727.324	2023-01-11 00:00:00		3
17	1004	125499342	774970428748290568	4425798645	339362742	25377.193	2023-03-20 20:59:45,921653		3
18	1004	894645555	53436767934728192	6167898821	784008799	15378.662	2023-03-20 20:59:45,921653		2
19	1004	844284389	75813998917454848	7958829463	985457131	5177.3975	2023-03-20 20:59:45,921653		1

(19 строк)

```

bankdatabase=# SELECT * FROM capitalizations;

```

capitalization_id	deposit_id	created_at	money
1	1	2023-03-21 21:03:48,178912	0.81
2	2	2023-03-21 21:03:48,178912	25
3	3	2023-03-21 21:03:48,178912	125
4	4	2023-03-21 21:03:48,178912	62.5
5	5	2023-03-21 21:03:48,178912	62.5
6	6	2023-03-21 21:03:48,178912	29.166666
7	1	2023-03-21 21:03:52,823346	5.833333e-05
8	3	2023-03-21 21:03:52,823346	25.8625
9	3	2023-03-21 21:03:52,823346	125.520836
10	4	2023-03-21 21:03:52,823346	62.65625
11	5	2023-03-21 21:03:52,823346	62.768048
12	6	2023-03-21 21:03:52,823346	29.336885
13	1	2023-03-21 21:45:36,563468	5.867361e-05
14	2	2023-03-21 21:45:36,563468	25.125156
15	3	2023-03-21 21:45:36,563468	126.04384
16	4	2023-03-21 21:45:36,563468	62.81289
17	5	2023-03-21 21:45:36,563468	63.02192
18	6	2023-03-21 21:45:36,563468	29.507936
42	2	2023-05-05 22:10:42,415969	0.81
43	4	2023-05-05 22:10:42,415969	0.81
44	5	2023-05-05 22:10:42,415969	0.81
45	6	2023-05-05 22:10:42,415969	0.81
46	1	2023-05-05 22:10:42,415969	0.81
47	3	2023-05-05 22:10:42,415969	0.81

(24 строк)

```

bankdatabase=# CALL proc_deposits_capitalization();
CALL
bankdatabase=# SELECT * FROM accounts ORDER BY account_id;

```

Рис. 1: До вызова процедуры

SQL Shell (psql)

```

bankdatabase=# CALL proc_deposits_capitalization();
CALL
bankdatabase=# SELECT * FROM accounts ORDER BY account_id;

```

account_id	user_id	bik	korr_account	inn	kpp	balance	created_at	closed_at	currency_id
1	1001	591731039	346786729259808992	1484612055	268765666	8228	2023-03-20 20:59:45,921653		2
2	1002	567592410	776627963145124192	7693145261	614293120	1140	2023-03-20 20:59:45,921653		2
3	1003	824440147	352748740887908088	1258368255	682661582	2882	2023-03-20 20:59:45,921653		2
4	1004	196242436	227536873680890256	5104831302	571539482	7948	2023-03-20 20:59:45,921653		2
5	1005	715100225	1849582583308082528	1264561546	684749868	8178	2023-03-20 20:59:45,921653		1
6	1006	105239847	41355457544376321	9748776026	283810052	2122	2023-03-20 20:59:45,921653		1
7	1007	929291628	776627963145224192	7059468784	118821657	1362	2023-03-20 20:59:45,921653		1
8	1008	498981421	767788826257377280	8408231192	712134374	0	2023-03-20 20:59:45,921653		1
9	1009	564491538	224519517363562496	4163278577	961499682	4383	2023-03-20 20:59:45,921653		1
10	1001	184741352	776627963145224192	9920565408	579837625	9898	2023-03-20 20:59:45,921653		1
11	1002	292601058	481120053472387200	9411145356	932626750	0	2023-03-20 20:59:45,921653		1
12	1005	915962263	276832715666973696	1555084199	180277873	10	2023-03-20 20:59:45,921653		1
13	1005	182021623	71560208383954496	8031171484	962077887	2429	2023-03-20 20:59:45,921653		3
14	1002	246421530	52473408818322944	7758790848	508106076	0.020260654	2023-03-20 20:59:45,921653		3
15	1003	98510713	54853498839915520	7681555595	924725342	10158.887	2023-03-20 20:59:45,921653		3
16	1003	996388498	14599632215970967	5751792279	595750776	30727.324	2023-01-11 00:00:00		3
17	1004	125499342	774970428748290568	4425798645	339362742	25377.203	2023-03-20 20:59:45,921653		3
18	1004	894645555	53436767934728192	6167898821	784008799	15378.672	2023-03-20 20:59:45,921653		2
19	1004	844284389	75813998917454848	7958829463	985457131	5177.407	2023-03-20 20:59:45,921653		1

(19 строк)

```

bankdatabase=# SELECT * FROM capitalizations;

```

capitalization_id	deposit_id	created_at	money
1	1	2023-03-21 21:03:48,178912	0.81
2	2	2023-03-21 21:03:48,178912	25
3	3	2023-03-21 21:03:48,178912	125
4	4	2023-03-21 21:03:48,178912	62.5
5	5	2023-03-21 21:03:48,178912	62.5
6	6	2023-03-21 21:03:48,178912	29.166666
7	1	2023-03-21 21:03:52,823346	5.833333e-05
8	2	2023-03-21 21:03:52,823346	25.8625
9	3	2023-03-21 21:03:52,823346	125.520836
10	4	2023-03-21 21:03:52,823346	62.65625
11	5	2023-03-21 21:03:52,823346	62.768048
12	6	2023-03-21 21:03:52,823346	29.336885
13	1	2023-03-21 21:45:36,563468	5.867361e-05
14	2	2023-03-21 21:45:36,563468	25.125156
15	3	2023-03-21 21:45:36,563468	126.04384
16	4	2023-03-21 21:45:36,563468	62.81289
17	5	2023-03-21 21:45:36,563468	63.02192
18	6	2023-03-21 21:45:36,563468	29.507936
54	2	2023-05-05 23:26:47,698632	0.81
55	4	2023-05-05 23:26:47,698632	0.81
56	5	2023-05-05 23:26:47,698632	0.81
57	6	2023-05-05 23:26:47,698632	0.81
58	1	2023-05-05 23:26:47,698632	0.81
59	3	2023-05-05 23:26:47,698632	0.81

(24 строк)

```

bankdatabase=#

```

Рис. 2: После вызова процедуры

## 2.2 Таблица cash\_turnover

Создадим триггер, который проверяет наличие необходимой суммы на балансе, если происходит снятие.

```

CREATE OR REPLACE FUNCTION
  fn_check_cash_before_turnover()
  RETURNS trigger
  LANGUAGE 'plpgsql'
AS $BODY$
  BEGIN
    IF (
      SELECT
        balance
      FROM
        accounts
      WHERE
        account_id = (
          SELECT
            account_id
          FROM
            cards
          WHERE
            card_number = NEW.card_number
        )
    ) < NEW.money AND NEW.in_out = '0' THEN
      RETURN NULL;
    END IF;
    RETURN NEW;
  END;
$BODY$;

CREATE TRIGGER check_cash_before_cash_turnover
  BEFORE INSERT
  ON cash_turnover
  FOR EACH ROW
  EXECUTE FUNCTION fn_check_cash_before_turnover();

```

Также добавим триггер, обновляющий баланс на счете в зависимости от действия: снятие или внесение.

```

CREATE OR REPLACE FUNCTION fn_update_cash_after_turnover()
  RETURNS trigger
  LANGUAGE 'plpgsql'
AS $BODY$
  BEGIN
    IF NEW.in_out = '1' THEN
      UPDATE
        accounts
      SET
        balance = balance + NEW.money
      WHERE
        account_id = (
          SELECT
            account_id
          FROM
            cards
          WHERE
            card_number = NEW.card_number
        );
    ELSE
      UPDATE
        accounts

```

```

SET
    balance = balance - NEW.money
WHERE
    account_id = (
        SELECT
            account_id
        FROM
            cards
        WHERE
            card_number = NEW.card_number
    );
END IF;
RETURN NEW;
END;
$BODY$;

CREATE TRIGGER update_cash_after_turnover
AFTER INSERT
ON cash_turnover
FOR EACH ROW
EXECUTE FUNCTION fn_update_cash_after_turnover();

```

```

bankdatabase=# SELECT accounts.account_id, balance, card_number FROM cards LEFT JOIN accounts ON cards.account_id = accounts.account_id ORDER BY accounts.account_id;
 account_id | balance | card_number
-----
1 | 8228 | 4858443003042444
2 | 2140 | 4788927954158345
3 | 2882 | 4712227257286898
4 | 7948 | 4111194292302436
5 | 8178 | 5582651815776708
6 | 2122 | 6011008321403328
7 | 1362 | 5222830188437785
8 | 0 | 5577386779412118
9 | 4383 | 4976693980889907
10 | 9898 | 4325492597540086
11 | 0 | 4404394795473009
12 | 10 | 4308268238089647
13 | 2429 | 5481196642315710
(13 строк)

bankdatabase=# INSERT INTO cash_turnover (card_number, in_out, nfc, address, money) VALUES (4788927954158345, '0', '0', '500-820 Fermentum Ave', 3000);
INSERT 0 0

```

Рис. 3: Неудачная попытка добавить запись о снятии наличных

```

bankdatabase=# SELECT accounts.account_id, balance, card_number FROM cards LEFT JOIN accounts ON cards.account_id = accounts.account_id ORDER BY accounts.account_id;
 account_id | balance | card_number
-----
1 | 8228 | 4858443003042444
2 | 2140 | 4788927954158345
3 | 2882 | 4712227257286898
4 | 7948 | 4111194292302436
5 | 8178 | 5582651815776708
6 | 2122 | 6011008321403328
7 | 1362 | 5222830188437785
8 | 0 | 5577386779412118
9 | 4383 | 4976693980889907
10 | 9898 | 4325492597540086
11 | 0 | 4404394795473009
12 | 10 | 4308268238089647
13 | 2429 | 5481196642315710
(13 строк)

bankdatabase=# INSERT INTO cash_turnover (card_number, in_out, nfc, address, money) VALUES (4788927954158345, '0', '0', '500-820 Fermentum Ave', 3000);
INSERT 0 0

```

Рис. 4: Удачная попытка добавить запись о снятии наличных, а затем - внесении их обратно

## 2.3 Таблица payments

Создадим триггер, проверяющий наличие достаточной суммы для проведения операции оплаты в магазинах. Этот же триггер будет использоваться в операциях платы за обслуживание, т.к. в них используется номер карты.



```

CREATE OR REPLACE FUNCTION
  fn_check_cash_with_card_number()
  RETURNS trigger
  LANGUAGE 'plpgsql'
AS $BODY$
  BEGIN
    IF (
      SELECT
        balance
      FROM
        accounts
      WHERE
        account_id = (
          SELECT
            account_id
          FROM
            cards
          WHERE
            card_number = NEW.card_number
        )
    ) < NEW.money THEN
      RETURN NULL;
    END IF;
    RETURN NEW;
  END;
$BODY$;

CREATE TRIGGER check_cash_before_payment
  BEFORE INSERT
  ON cash_turnover
  FOR EACH ROW
  EXECUTE FUNCTION fn_check_cash_with_card_number();

```

Также создадим триггер, обновляющий информацию о балансе после внесения записи о трате в магазине.

```

CREATE OR REPLACE FUNCTION fn_update_cash_after_payment()
  RETURNS trigger
  LANGUAGE 'plpgsql'
AS $BODY$
  BEGIN
    UPDATE
      accounts
    SET
      balance = balance - NEW.money
    WHERE
      account_id = (
        SELECT
          account_id
        FROM
          cards
        WHERE
          card_number = NEW.card_number
      );
    RETURN NEW;
  END;
$BODY$;

```

```

CREATE TRIGGER update_cash_after_payment
AFTER INSERT
ON payments
FOR EACH ROW
EXECUTE FUNCTION fn_update_cash_after_payment();

```

```

bankdatabase=# SELECT accounts.account_id, balance, card_number FROM cards LEFT JOIN accounts ON cards.account_id = accounts.account_id ORDER BY accounts.account_id;
 account_id | balance | card_number
-----
1 | 8228 | 4858443003042444
2 | 2140 | 4788927954158345
3 | 2882 | 4712227257286898
4 | 7948 | 4111194292302436
5 | 8178 | 5582651815776708
6 | 2122 | 6011008321403328
7 | 1362 | 5222830188437785
8 | 0 | 5577386779412118
9 | 4383 | 497669390888907
10 | 9898 | 4325492597540086
11 | 0 | 4404394795473009
12 | 10 | 4308268238089647
13 | 2429 | 5481196642315710
(13 строк)

bankdatabase=# INSERT INTO payments (retail_outlet_id, card_number, money) VALUES (4, 5481196642315710, 3000);
INSERT 0 0
bankdatabase=# INSERT INTO payments (retail_outlet_id, card_number, money) VALUES (4, 5481196642315710, 1500);
INSERT 0 1
bankdatabase=# SELECT accounts.account_id, balance, card_number FROM cards LEFT JOIN accounts ON cards.account_id = accounts.account_id ORDER BY accounts.account_id;
 account_id | balance | card_number
-----
1 | 8228 | 4858443003042444
2 | 2140 | 4788927954158345
3 | 2882 | 4712227257286898
4 | 7948 | 4111194292302436
5 | 8178 | 5582651815776708
6 | 2122 | 6011008321403328
7 | 1362 | 5222830188437785
8 | 0 | 5577386779412118
9 | 4383 | 497669390888907
10 | 9898 | 4325492597540086
11 | 0 | 4404394795473009
12 | 10 | 4308268238089647
13 | 929 | 5481196642315710
(13 строк)

```

Рис. 5: Пример внесения записей в таблицу payments

## 2.4 Таблица payments\_for\_service

Добавим триггер на проверку достаточной суммы на балансе по функции, написанной в предыдущем пункте.

```

CREATE TRIGGER check_cash_before_service_fee
BEFORE INSERT
ON payments_for_service
FOR EACH ROW
EXECUTE FUNCTION fn_check_cash_with_card_number();

```

Создадим триггер для обновления информации о балансе при добавлении записи в таблицу о платах за обслуживание.

```

CREATE OR REPLACE FUNCTION fn_update_cash_after_service_fee()
RETURNS trigger
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
    UPDATE
    accounts
    SET
    balance = balance - NEW.money
    WHERE

```

```

        account_id = (
            SELECT
                account_id
            FROM
                cards
            WHERE
                card_number = NEW.card_number
        );
    RETURN NEW;
END;
$BODY$;

CREATE TRIGGER update_cash_after_service_fee
AFTER INSERT
ON payments_for_service
FOR EACH ROW
EXECUTE FUNCTION fn_update_cash_after_service_fee();

```

И также создадим процедуру, которая будет проходить по всем картам в базе, у которых плата за обслуживание отлична от нуля, и снимать платеж за обслуживание.

```

CREATE OR REPLACE PROCEDURE proc_payment_for_service()
LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
    i BIGINT;
BEGIN
    FOR i IN (
        SELECT
            card_number
        FROM
            cards
        WHERE
            type_of_card_id IN (
                SELECT
                    type_of_card_id
                FROM
                    types_of_cards
                WHERE
                    service_fee <> 0
            )
    ) LOOP INSERT INTO payments_for_service (card_number, money)
VALUES
    (
        i,
        (
            SELECT
                service_fee
            FROM
                cards
            LEFT JOIN types_of_cards
            ON cards.type_of_card_id = types_of_cards.type_of_card_id
            WHERE
                cards.card_number = i
        )
    );
END LOOP;

```

```
END;
$BODY$;
```

```
bankdatabase=# SELECT account_id, balance, card_number, service_fee FROM (SELECT accounts.account_id, balance, card_number, type_of_card_id FROM cards LEFT JOIN accounts ON cards.account_id = accounts.account_id
) AS tbl1 LEFT JOIN types_of_cards ON tbl1.type_of_card_id = types_of_cards.type_of_card_id ORDER BY account_id;
```

account_id	balance	card_number	service_fee
1	8078	4858443083042444	150
2	2100	4788927954158345	40
3	2882	4712227257286898	0
4	7948	4111194292302436	0
5	7178	5582651815776788	1000
6	1272	6811088321403328	850
7	1362	5222830188437785	0
8	0	5577386779412118	60
9	3533	4976683980888987	850
10	9858	4325492597540886	40
11	0	4404394795473809	150
12	10	4308268238889647	200
13	889	5481196642315710	40

(13 строк)

```
bankdatabase=# CALL proc_payment_for_service();
CALL
bankdatabase=# SELECT account_id, balance, card_number, service_fee FROM (SELECT accounts.account_id, balance, card_number, type_of_card_id FROM cards LEFT JOIN accounts ON cards.account_id = accounts.account_id
) AS tbl1 LEFT JOIN types_of_cards ON tbl1.type_of_card_id = types_of_cards.type_of_card_id ORDER BY account_id;
```

account_id	balance	card_number	service_fee
1	7928	4858443083042444	150
2	2060	4788927954158345	40
3	2882	4712227257286898	0
4	7948	4111194292302436	0
5	6178	5582651815776788	1000
6	422	6811088321403328	850
7	1362	5222830188437785	0
8	0	5577386779412118	60
9	2633	4976683980888987	850
10	9818	4325492597540886	40
11	0	4404394795473809	150
12	10	4308268238889647	200
13	849	5481196642315710	40

(13 строк)

Рис. 6: До и после внесения записей в таблицу payments\_for\_service

## 2.5 Таблица remittances

Создадим триггер, проверяющий наличие необходимой суммы на счете отправителя.

```
CREATE OR REPLACE FUNCTION fn_check_cash_with_account_id()
    RETURNS trigger
    LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
    IF (
        SELECT
            balance
        FROM
            accounts
        WHERE
            account_id = NEW.out_account
    ) < NEW.money THEN
        RETURN NULL;
    END IF;
    RETURN NEW;
END;
$BODY$;
```

```
CREATE TRIGGER check_cash_before_remittance
    BEFORE INSERT
    ON remittances
    FOR EACH ROW
    EXECUTE FUNCTION fn_check_cash_with_account_id();
```

И создадим триггер, вносящий изменения в баланс на счете как отправителя, так и получателя.

```

CREATE OR REPLACE FUNCTION
    fn_update_cash_after_remittance()
    RETURNS trigger LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
    UPDATE
        accounts
    SET
        balance = balance + NEW.money
    WHERE
        accounts.account_id = NEW.in_account;
    UPDATE
        accounts
    SET
        balance = balance - NEW.money
    WHERE
        accounts.account_id = NEW.out_account;
    RETURN NEW;
END;
$BODY$;

CREATE TRIGGER update_cash_after_remittance
    AFTER INSERT
    ON remittances
    FOR EACH ROW
    EXECUTE FUNCTION fn_update_cash_after_remittance();

```

```

bankdatabase=# SELECT account_id, balance FROM accounts ORDER BY account_id;
 account_id | balance
-----
          1 |    7928
          2 |    2060
          3 |    2882
          4 |    7948
          5 |    6178
          6 |     422
          7 |    1362
          8 |         0
          9 |    2683
         10 |    9818
         11 |         0
         12 |         0
         13 |     849
         14 | 0.020260654
         15 | 10150.887
         16 | 30757.334
         17 | 25377.203
         18 | 15370.672
         19 | 5177.407
(19 строк)

bankdatabase=# INSERT INTO remittances (in_account, out_account, money) VALUES (5, 8, 1000);
INSERT 0 1
bankdatabase=# INSERT INTO remittances (in_account, out_account, money) VALUES (8, 5, 1000);
INSERT 0 1
bankdatabase=# SELECT account_id, balance FROM accounts ORDER BY account_id;
 account_id | balance
-----
          1 |    7928
          2 |    2060
          3 |    2882
          4 |    7948
          5 |    5178
          6 |     422
          7 |    1362
          8 |    1000
          9 |    2683
         10 |    9818
         11 |         0
         12 |         0
         13 |     849
         14 | 0.020260654
         15 | 10150.887
         16 | 30757.334
         17 | 25377.203
         18 | 15370.672
         19 | 5177.407
(19 строк)

```

Рис. 7: Пример попыток добавления записей о переводах

Также мы бы хотели ограничить возможность переводов со вкладами согласно их типу. Создадим триггер, проверяющий возможность снятия или пополнения счета вклада.

```

CREATE OR REPLACE FUNCTION
    fn_check_access_to_update()
    RETURNS trigger
    LANGUAGE 'plpgsql'
AS $BODY$
BEGIN

```

```

IF NEW.out_account IN (
    SELECT
        account_id
    FROM
        deposits
)
AND (
    SELECT
        removable
    FROM
        deposits
    LEFT JOIN types_of_deposits
    ON deposits.type_of_deposit_id =
        types_of_deposits.type_of_deposit_id
    WHERE
        account_id = NEW.out_account
) = '1'
OR NEW.in_account IN (
    SELECT
        account_id
    FROM
        deposits
)
AND (
    SELECT
        refillable
    FROM
        deposits
    LEFT JOIN types_of_deposits
    ON deposits.type_of_deposit_id =
        types_of_deposits.type_of_deposit_id
    WHERE
        account_id = NEW.in_account
) = '1' THEN RETURN NEW;
END IF;
RETURN NULL;
END;
$BODY$;

```

```

bankdatabase=# SELECT accounts.account_id, balance, removable, refillable FROM (SELECT account_id, removable, refillable FROM deposits LEFT JOIN types_of_deposits ON deposits.type_of_deposit_id = types_of_deposits.type_of_deposit_id) AS tbl1 LEFT JOIN accounts ON tbl1.account_id = accounts.account_id ORDER BY accounts.account_id;
 account_id | balance | removable | refillable 
-----+-----+-----+-----
    14 | 0.020260654 | t | t 
    15 | 10150.887 | f | f 
    16 | 29757.334 | t | t 
    17 | 25377.203 | f | f 
    18 | 15378.672 | t | t 
    19 | 5177.407 | t | t 
(6 строк)

bankdatabase=# INSERT INTO remittances (in_account, out_account, money) VALUES (15, 1, 1000);
INSERT 0 0
bankdatabase=# INSERT INTO remittances (in_account, out_account, money) VALUES (1, 15, 1000);
INSERT 0 0
bankdatabase=# INSERT INTO remittances (in_account, out_account, money) VALUES (16, 1, 1000);
INSERT 0 1
bankdatabase=# SELECT accounts.account_id, balance, removable, refillable FROM (SELECT account_id, removable, refillable FROM deposits LEFT JOIN types_of_deposits ON deposits.type_of_deposit_id = types_of_deposits.type_of_deposit_id) AS tbl1 LEFT JOIN accounts ON tbl1.account_id = accounts.account_id ORDER BY accounts.account_id;
 account_id | balance | removable | refillable 
-----+-----+-----+-----
    14 | 0.020260654 | t | t 
    15 | 10150.887 | f | f 
    16 | 30757.334 | t | t 
    17 | 25377.203 | f | f 
    18 | 15378.672 | t | t 
    19 | 5177.407 | t | t 
(6 строк)

bankdatabase=# INSERT INTO remittances (in_account, out_account, money) VALUES (1, 16, 1000);
INSERT 0 1
bankdatabase=# SELECT accounts.account_id, balance, removable, refillable FROM (SELECT account_id, removable, refillable FROM deposits LEFT JOIN types_of_deposits ON deposits.type_of_deposit_id = types_of_deposits.type_of_deposit_id) AS tbl1 LEFT JOIN accounts ON tbl1.account_id = accounts.account_id ORDER BY accounts.account_id;
 account_id | balance | removable | refillable 
-----+-----+-----+-----
    14 | 0.020260654 | t | t 
    15 | 10150.887 | f | f 
    16 | 29757.334 | t | t 
    17 | 25377.203 | f | f 
    18 | 15378.672 | t | t 
    19 | 5177.407 | t | t 
(6 строк)

```

Рис. 8: Попытки совершить переводы с вкладами

### 3 Вывод

В процессе выполнения лабораторной работы были освоены практические навыки создания и использования функций, процедур и триггеров.