ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет инфокоммуникационных технологий

Дисциплина:
«Проектирование и реализация баз данных»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3
«ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В POSTGRE SQL»

Выполнил:
студент группы K32391
Микитчак Иван Михайлович

_____
(подпись)

Проверил:
Говорова Марина Михайловна

_____
(отметка о выполнении)

_____
(подпись)

Санкт-Петербург
2023 г.

Цель работы: овладеть практическими навыками 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.

Выполнение задания

Создание процедур

Формулировка: Создайте процедуру для назначения повышенной стипендии отличникам.
Код:

```
CREATE OR REPLACE PROCEDURE Assign_high_scholarship(session_st
timestamp, session_end timestamp, scholarship_st date, scholarship_end
date)
LANGUAGE 'plpgsql' AS $$
DECLARE student int;
BEGIN
FOR student
IN
SELECT DISTINCT student_id
      FROM st_in_group JOIN (
            SELECT student_personnel_number
            FROM (
                  SELECT *, CASE WHEN mark = '5A' THEN 0 WHEN mark =
                  'Зачёт' THEN 0 ELSE 1 END AS non_excellent
                  FROM attestations
                  ) AS attestations_extended
            GROUP BY student_personnel_number
            HAVING SUM(non_excellent) = 0
            ) AS excellent_pn ON st_in_group.student_personnel_number =
            excellent_pn.student_personnel_number
LOOP
INSERT INTO sch_in_st (scholarship_id, student_id,
            ch_valid_since, sch_valid_till)
VALUES (2, student, scholarship_st, scholarship_end);
COMMIT;
END LOOP;
END;
$$
```

Таблица назначенных стипендий до вызова скрипта:

Data Output Messages Notifications

	sch_in_st_id [PK] integer	scholarship_id integer	student_id integer	sch_valid_since date	sch_valid_till date

Вызываем скрипт:

```
CALL Assign_high_scholarship('2022-05-01 00:00:00+03', '2022-07-01 00:00:00+03', '2022-09-01', '2023-01-01')
```

Результат:

Data Output Messages Notifications

	sch_in_st_id [PK] integer	scholarship_id integer	student_id integer	sch_valid_since date	sch_valid_till date
1	2	2	1	2022-09-01	2023-01-01

Формулировка: Создайте процедуру для перевода студентов на другой курс.
Код:

```
CREATE OR REPLACE PROCEDURE Transfer_students(session_st timestamp, session_end timestamp, from_group int, to_group int, since date, till date)
LANGUAGE 'plpgsql' AS $$
DECLARE spn int;
BEGIN
FOR spn
IN
SELECT student_personnel_number
      FROM (
            SELECT *, CASE WHEN mark = '2A' THEN 1 WHEN mark = 'Незачёт' THEN 1 ELSE 0 END AS fail
            FROM attestations
            WHERE attestation_date > '2022-05-01 00:00:00+03' AND attestation_date < '2022-07-01 00:00:00+03' AND attempt=1
```

```
            AND student_personnel_number IN (
                  SELECT student_personnel_number
                  FROM st_in_group
                  WHERE st_group_id = 1
                  )
            ) AS attestations_extended
            GROUP BY student_personnel_number
            HAVING SUM(fail) = 0
LOOP
UPDATE st_in_group
SET st_group_id = to_group, student_in_group_since=since,
student_in_group_till=till
WHERE student_personnel_number = spn;
END LOOP;
END;
$$
```

Таблица студентов в группе до перевода:

Data Output Messages Notifications

	student_personnel_number [PK] integer	student_id integer	st_group_id integer	student_in_group_since date	student_in_group_till date	student_status character varying (20)
1	333111	1	1	2021-09-01	2022-07-01	Учится
2	333112	2	1	2021-09-01	2022-07-01	Учится
3	333113	3	1	2021-09-01	2022-07-01	Учится
4	333114	4	1	2021-09-01	2022-07-01	Учится
5	333115	5	1	2021-09-01	2022-07-01	Учится
6	333116	6	1	2021-09-01	2022-07-01	Учится
7	333117	7	1	2021-09-01	2022-07-01	Учится
8	333118	8	1	2021-09-01	2022-07-01	Учится
9	333119	9	1	2021-09-01	2022-07-01	Учится
10	333120	10	1	2021-09-01	2022-07-01	Учится

Вызываем процедуру:

```
CALL Transfer_students('2022-05-01 00:00:00+03', '2022-07-01
00:00:00+03', 1, 2, '2022-09-01', '2023-07-01')
```

Результат:

Data Output Messages Notifications

	student_personnel_number [PK] integer	student_id integer	st_group_id integer	student_in_group_since date	student_in_group_till date	student_status character varying (20)
1	333111	1	2	2022-09-01	2023-07-01	Учится
2	333112	2	2	2022-09-01	2023-07-01	Учится
3	333113	3	2	2022-09-01	2023-07-01	Учится
4	333114	4	2	2022-09-01	2023-07-01	Учится
5	333115	5	2	2022-09-01	2023-07-01	Учится
6	333116	6	2	2022-09-01	2023-07-01	Учится
7	333117	7	2	2022-09-01	2023-07-01	Учится
8	333118	8	1	2021-09-01	2022-07-01	Учится
9	333119	9	1	2021-09-01	2022-07-01	Учится
10	333120	10	2	2022-09-01	2023-07-01	Учится

Формулировка: Создайте процедуру для изменения оценки при пересдаче.
Код:

```
CREATE OR REPLACE PROCEDURE Reattempt(spn int, epn int, dic int, mark
varchar(10), atd timestamp)
LANGUAGE 'plpgsql'
AS $$
DECLARE last_attempt int;
DECLARE last_id int;
BEGIN
SELECT attestation_id INTO last_id FROM attestations ORDER BY
      attestation_id DESC LIMIT 1;

SELECT attempt INTO last_attempt
      FROM attestations
      WHERE student_personnel_number = spn AND
      disc_in_curr_id = dic
      ORDER BY attestation_date DESC LIMIT 1;

INSERT INTO attestations(student_personnel_number,
      educator_personnel_number, disc_in_curr_id, mark, attempt,
      attestation_date, attestation_id)
VALUES (spn, epn, dic, mark, last_attempt + 1, atd, last_id+1);
END;
$$
```

Таблица аттестаций до вызова:

Data Output Messages Notifications

	student_personnel_number integer	educator_personnel_number integer	disc_in_curr_id integer	mark character varying (10)	attestation_id [PK] integer	attempt integer	attestation_date timestamp with time zone
38	333118	111555	4	Незачёт	38	2	2022-08-31 18:00:00+03
39	333118	111222	1	2F	39	3	2022-09-15 18:00:00+03
40	333118	111111	2	2F	40	3	2022-09-20 18:00:00+03
41	333118	111333	3	2F	41	3	2022-09-25 18:00:00+03
42	333118	111555	4	Незачёт	42	3	2022-09-30 18:00:00+03
43	333119	111222	1	3D	43	1	2022-06-01 18:00:00+03
44	333119	111111	2	3D	44	1	2022-06-05 18:00:00+03
45	333119	111333	3	4B	45	1	2022-06-10 18:00:00+03
46	333119	111555	4	Незачёт	46	1	2022-05-30 18:00:00+03
47	333118	111555	4	Зачёт	47	2	2022-08-31 18:00:00+03
48	333120	111222	1	5A	48	1	2022-06-01 18:00:00+03
49	333120	111111	2	4B	49	1	2022-06-05 18:00:00+03
50	333120	111333	3	4C	50	1	2022-06-10 18:00:00+03
51	333120	111555	4	Незачёт	51	1	2022-05-30 18:00:00+03

Вызов процедуры:

```
CALL Reattempt(333115, 111555, 4, 'Зачёт', '2022-08-31 18:00:00+03')
```

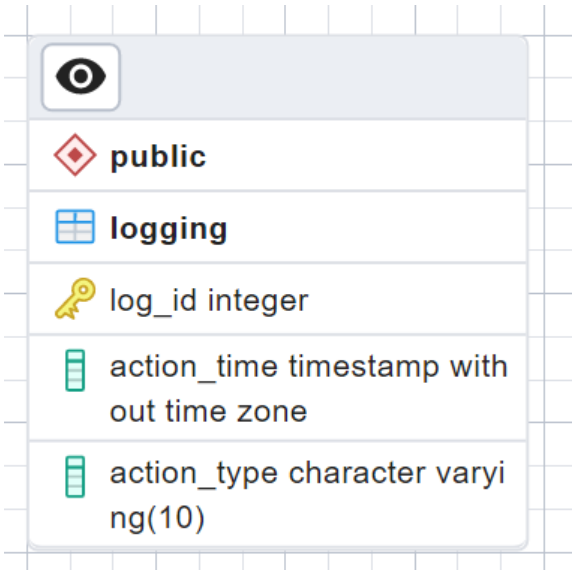
Результат:

Data Output Messages Notifications

	student_personnel_number integer	educator_personnel_number integer	disc_in_curr_id integer	mark character varying (10)	attestation_id [PK] integer	attempt integer	attestation_date timestamp with time zone
39	333118	111222	1	2F	39	3	2022-09-15 18:00:00+03
40	333118	111111	2	2F	40	3	2022-09-20 18:00:00+03
41	333118	111333	3	2F	41	3	2022-09-25 18:00:00+03
42	333118	111555	4	Незачёт	42	3	2022-09-30 18:00:00+03
43	333119	111222	1	3D	43	1	2022-06-01 18:00:00+03
44	333119	111111	2	3D	44	1	2022-06-05 18:00:00+03
45	333119	111333	3	4B	45	1	2022-06-10 18:00:00+03
46	333119	111555	4	Незачёт	46	1	2022-05-30 18:00:00+03
47	333118	111555	4	Зачёт	47	2	2022-08-31 18:00:00+03
48	333120	111222	1	5A	48	1	2022-06-01 18:00:00+03
49	333120	111111	2	4B	49	1	2022-06-05 18:00:00+03
50	333120	111333	3	4C	50	1	2022-06-10 18:00:00+03
51	333120	111555	4	Незачёт	51	1	2022-05-30 18:00:00+03
52	333115	111555	4	Зачёт	52	2	2022-08-31 18:00:00+03

Создание триггера на логирование действий

Мы будем логировать действия, которые пользователь совершает в таблице attestations. Для этого мы заведём таблицу logging. Ниже приведена схема таблицы logging:



Создадим триггерную функцию для реагирования на действия в таблице attestations:

```
CREATE OR REPLACE FUNCTION log_action() RETURNS TRIGGER AS
$log_action$
DECLARE cur_t timestamp;
BEGIN
SELECT CURRENT_TIMESTAMP INTO cur_t;
IF (TG_OP = 'INSERT') THEN
           INSERT INTO logging (action_time, action_type)
VALUES (cur_t, 'INSERT');
ELSIF (TG_OP = 'UPDATE') THEN
           INSERT INTO logging (action_time, action_type)
VALUES (cur_t, 'UPDATE');
        ELSIF (TG_OP = 'DELETE') THEN
           INSERT INTO logging (action_time, action_type)
VALUES (cur_t, 'DELETE');
END IF;
RETURN NULL;
END;
$log_action$
LANGUAGE plpgsql;
```

Теперь создадим триггер:

```
CREATE TRIGGER Action_log
AFTER INSERT OR UPDATE OR DELETE
ON attestations
```

```
EXECUTE PROCEDURE Log_action();
```

Проверим действие триггера, совершив действия в таблице attestations. Мы сделаем следующие вызовы:

```
INSERT INTO attestations
VALUES (333111, 111111, 1, '5A', 52, 1, now());

UPDATE attestations
SET mark = '3D'
WHERE attestation_id=52;

DELETE FROM attestations
WHERE attestation_id=52;
```

Таблица logging до вызовов:

Data Output Messages Notifications

	log_id [PK] integer	action_time timestamp without time zone	action_type character varying (10)

Таблица logging после вызовов:

Data Output Messages Notifications

	log_id [PK] integer	action_time timestamp without time zone	action_type character varying (10)
1	16	2023-06-05 17:37:42.470813	INSERT
2	17	2023-06-05 17:37:42.470813	UPDATE
3	18	2023-06-05 17:37:42.470813	DELETE

Выводы

Я овладел практическими навыками создания и использования процедур, функций и триггеров в базе данных PostgreSQL.