

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

**Факультет инфокоммуникационных
технологий**

Дисциплина:
«Базы данных»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3
«АНАЛИЗ ДАННЫХ. ПОСТРОЕНИЕ ИНФОЛОГИЧЕСКОЙ
МОДЕЛИ ДАННЫХ БД»**

Выполнил:
студент группы К32402
Екушев Владислав
Александрович

_____ (подпись)

Проверил:
Говорова Марина Михайловна

_____ (подпись и наименование)

_____ (подпись)

Санкт-Петербург
2023 г.

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

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, pgadmin 4.

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 работы:

Методы:

1. Для повышения стипендии отличникам на 10%

```
1 CREATE FUNCTION public.increase_scholarship_10_percent(IN scholarship_id integer)
2 RETURNS void
3 LANGUAGE 'plpgsql'
4
5 AS $BODY$
6 declare
7     prev_amount int;
8 begin
9     select amount into prev_amount from scholarship where id=scholarship_id;
10    update scholarship set amount=prev_amount * 1.1 where id=scholarship_id;
11 end;
12 $BODY$;
13
14 ALTER FUNCTION public.increase_scholarship_10_percent(integer)
15 OWNER TO postgres;
```

До:

	id [PK] integer	amount integer	type scholarship_type_enum
1	1	3000	Basic
2	2	4000	Basic
3	3	15000	Increased
4	4	100000	Increased
5	5	200000	Increased

После:

	id [PK] integer	amount integer	type scholarship_type_enum
1	1	3300	Basic
2	2	4000	Basic
3	3	15000	Increased
4	4	100000	Increased
5	5	200000	Increased

2. Для перевода студентов на следующий курс

```
1  -- FUNCTION: public.increase_student_group_course(integer, text)
2
3  -- DROP FUNCTION IF EXISTS public.increase_student_group_course(integer, text);
4
5  CREATE OR REPLACE FUNCTION public.increase_student_group_course(
6      group_id integer,
7      student_group_code text)
8      RETURNS void
9      LANGUAGE 'plpgsql'
10     COST 100
11     VOLATILE PARALLEL UNSAFE
12 AS $BODY$
13 declare
14     prev_curriculum_id integer;
15     prev_year_start integer;
16     prev_year_end integer;
17     new_student_group_id integer;
18 begin
19     select curriculum_id, year_start, year_end into prev_curriculum_id, prev_year_start, prev_year_end
20     from student_group where id=group_id;
21     INSERT INTO public.student_group(
22     curriculum_id, code, year_start, year_end, status)
23     VALUES (prev_curriculum_id, student_group_code, prev_year_start + 1, prev_year_end + 1, 'Formed');
24     select id into new_student_group_id from student_group where student_group.code=student_group_code;
25     update student set student_group_id=new_student_group_id where student_group_id=group_id;
26 end;
27 $BODY$;
28
29 ALTER FUNCTION public.increase_student_group_course(integer, text)
30     OWNER TO postgres;
31
```

До:

	id [PK] integer	curriculum_id integer	code text	year_start integer	year_end integer	status student_group_status_enum
1	4	13	F402	3	2	Formed
2	5	13	F401	3	2	Formed
3	2	4	K32402	2	3	Formed
4	3	4	K32401	2	3	Formed

	id [PK] integer	student_group_id integer	person_id integer	study_start date	study_end date	status student_status_enum
1	2	2	1	2021-09-01	2025-07-01	Studying
2	3	2	2	2021-09-01	2025-07-01	Studying
3	4	2	3	2021-09-01	2025-07-01	Studying
4	5	2	4	2021-09-01	2025-07-01	Studying
5	6	2	5	2021-09-01	2025-07-01	Studying
6	8	3	7	2021-09-01	2025-07-01	Transferred
7	9	3	8	2021-09-01	2025-07-01	Studying
8	10	3	10	2021-09-01	2025-07-01	AcademicLeave
9	11	4	11	2021-09-01	2025-07-01	Studying
10	12	4	12	2021-09-01	2025-07-01	Studying
11	13	4	13	2021-09-01	2025-07-01	Studying
12	14	5	14	2020-09-01	2024-07-01	Studying
13	15	5	15	2020-09-01	2024-07-01	Studying
14	17	2	1	2021-09-01	2025-07-01	Studying
15	7	3	6	2021-09-01	2023-05-29	Dismissed

После:

	id [PK] integer	curriculum_id integer	code text	year_start integer	year_end integer	status student_group_status_enum
1	4	13	F402	3	2	Formed
2	5	13	F401	3	2	Formed
3	2	4	K32402	2	3	Formed
4	3	4	K32401	2	3	Formed
5	10	4	K3301	3	4	Formed

	id [PK] integer	student_group_id integer	person_id integer	study_start date	study_end date	status student_status_enum
1	8	3	7	2021-09-01	2025-07-01	Transferred
2	9	3	8	2021-09-01	2025-07-01	Studying
3	10	3	10	2021-09-01	2025-07-01	AcademicLeave
4	11	4	11	2021-09-01	2025-07-01	Studying
5	12	4	12	2021-09-01	2025-07-01	Studying
6	13	4	13	2021-09-01	2025-07-01	Studying
7	14	5	14	2020-09-01	2024-07-01	Studying
8	15	5	15	2020-09-01	2024-07-01	Studying
9	7	3	6	2021-09-01	2023-05-29	Dismissed
10	2	10	1	2021-09-01	2025-07-01	Studying
11	3	10	2	2021-09-01	2025-07-01	Studying
12	4	10	3	2021-09-01	2025-07-01	Studying
13	5	10	4	2021-09-01	2025-07-01	Studying
14	6	10	5	2021-09-01	2025-07-01	Studying
15	17	10	1	2021-09-01	2025-07-01	Studying

3. Для изменения оценки при успешной пересдаче экзамена

При всех использованных попытках выбрасывается ошибка:

	id [PK] integer	student_id integer	teacher_id integer	edu_program_discipline_id integer	score integer	attempt integer	date date
1	12	7	5	3	2	3	[null]

```
1 SELECT change_discipline_score(  
2     7,  
3     3,  
4     5,  
5     '2023-06-12'  
6 );
```

Data Output Messages Notifications

ERROR: All attempts where used on this discipline by student
CONTEXT: PL/pgSQL function change_discipline_score(integer,integer,integer,date) line 15 at RAISE
SQL state: P0001

Если меняется оценка на непроходной балл, выбрасывается ошибка:

Query Query History

```
1 SELECT change_discipline_score(  
2     2,  
3     3,  
4     2,  
5     '2023-06-12'  
6 );
```

Data Output Messages Notifications

ERROR: Function argument `score` is invalid: should be greater than 2
CONTEXT: PL/pgSQL function change_discipline_score(integer,integer,integer,date) line 21 at RAISE
SQL state: P0001

Если студент уже получил проходной балл за экзамен, то выбрасывается ошибка:

Query Query History

```
1 SELECT change_discipline_score(  
2     2,  
3     3,  
4     5,  
5     '2023-06-12'  
6 );
```

Data Output Messages Notifications

ERROR: Student already passed that exam
CONTEXT: PL/pgSQL function change_discipline_score(integer,integer,integer,date) line 26 at RAISE
SQL state: P0001

Правильный запрос

До:

Query

Query History

1

```
select * from score where score=2 and attempt=1
```

Data Output

Messages

Notifications

	id [PK] integer	student_id integer	teacher_id integer	edu_program_discipline_id integer	score integer	attempt integer	date date
1	2	3	2	3	2	1	[null]
2	4	5	2	3	2	1	[null]

Запрос:

1

```
SELECT change_discipline_score(
```

2

```
    5,
```

3

```
    3,
```

4

```
    5,
```

5

```
    '2023-06-12'
```

6

```
);
```

Data Output

Messages

Notifications

change_discipline_score

void

1	
---	--

После:

1

```
select * from score
```

Data Output

Messages

Notifications

	id [PK] integer	student_id integer	teacher_id integer	edu_program_discipline_id integer	score integer	attempt integer	date date
1	1	2	2	3	4	1	[null]
2	5	6	4	3	5	1	[null]
3	6	7	5	3	0	1	[null]
4	2	3	2	3	2	1	[null]
5	3	4	2	3	3	2	[null]
6	4	5	2	3	2	1	[null]
7	9	7	5	3	4	2	[null]
8	7	8	5	3	4	1	[null]
9	8	8	6	3	3	1	[null]
10	11	7	5	3	2	2	[null]
11	12	7	5	3	2	3	[null]
12	13	2	3	3	5	2	2023-01-01
13	14	5	2	3	5	2	2023-06-12

Триггеры

1. Для логирования событий CUD в БД на таблице `scholarship_assignment`:

```
1 CREATE TYPE public.cud_operation_enum AS ENUM
2   ('insert', 'update', 'delete');
3
4 ALTER TYPE public.cud_operation_enum
5   OWNER TO postgres;
6
7 create table scholarship_assignment_logs(
8   id serial primary key,
9   op_type cud_operation_enum not null,
10  op_timestamp timestamp,
11  affected_id int null,
12  affected_person_id int null,
13  affected_scholarship_id int null
14 );
15
16 CREATE FUNCTION public.scholarship_assignment_log()
17 RETURNS trigger
18 LANGUAGE 'plpgsql'
19 NOT LEAKPROOF
20 AS $BODY$
21 declare
22   op_type cud_operation_enum;
23 begin
24   if tg_op='INSERT' then
25     op_type := 'insert';
26     insert into scholarship_assignment_logs(op_type, op_timestamp, affected_id, affected_person_id, affected_scholarship_id)
27     values (op_type, now(), new.id, new.person_id, new.scholarship_id);
28     return new;
29   end if;
30   if tg_op='UPDATE' then
31     op_type := 'update';
32     insert into scholarship_assignment_logs(op_type, op_timestamp, affected_id, affected_person_id, affected_scholarship_id)
33     values (op_type, now(), old.id, old.person_id, old.scholarship_id);
34     return new;
35   end if;
36   if tg_op='DELETE' then
37     op_type := 'delete';
38     insert into scholarship_assignment_logs(op_type, op_timestamp, affected_id, affected_person_id, affected_scholarship_id)
39     values (op_type, now(), old.id, old.person_id, old.scholarship_id);
40     return old;
41   end if;
42 end;
43 $BODY$;
44
45 ALTER FUNCTION public.scholarship_assignment_log()
46   OWNER TO postgres;
47
48 create trigger scholarship_assignment_log_trigger
49 after insert or update or delete on scholarship_assignment
50 for each row execute procedure scholarship_assignment_log();
```

Вставка:

	id [PK] integer	op_type cud_operation_enum	op_timestamp timestamp without time zone	affected_id integer	affected_person_id integer	affected_scholarship_id integer
1	1	insert	2023-06-23 06:36:52.456606	9	7	1

Обновление:

	id [PK] integer	op_type cud_operation_enum	op_timestamp timestamp without time zone	affected_id integer	affected_person_id integer	affected_scholarship_id integer
1	1	insert	2023-06-23 06:36:52.456606	9	7	1
2	2	update	2023-06-23 06:38:07.86315	9	7	1

Удаление:

	id [PK] integer	op_type cud_operation_enum	op_timestamp timestamp without time zone	affected_id integer	affected_person_id integer	affected_scholarship_id integer
1	1	insert	2023-06-23 06:36:52.456606	9	7	1
2	2	update	2023-06-23 06:38:07.86315	9	7	1
3	3	delete	2023-06-23 06:39:12.758792	9	7	2

Выводы

В ходе лабораторной работы получилось овладеть навыками создания хранимых процедур и триггеров в PostgreSQL, были созданы функции и триггеры согласно индивидуальному заданию варианта 3.

Чтобы создать универсальный триггер для логирования данных можно воспользоваться переменными **old::text** и **new::text**, которые отдадут текстовую репрезентацию вставляемых/изменяемых/удаляемых данных.

Также можно использовать переменную **TG_TABLE_NAME**, которая содержит в себе имя таблицы, которая участвует в операции. Таким образом, можно создать таблицу с логами, которая будет содержать в себе операцию (**TG_OP**), имя таблицы (**TG_TABLE_NAME**), старые данные (**old::text**) и новые данные (**new::text**).