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

Факультет инфокоммуникационных технологий

Дисциплина:
«Базы данных»

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

Выполнила:
студентка группы K32421
Панкова Кристина
Сергеевна

(подпись)

Проверила:
Говорова Марина Михайловна

(отметка о выполнении)

(подпись)

Санкт-Петербург
2023 г.

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.

Индивидуальное задание: Вариант 4

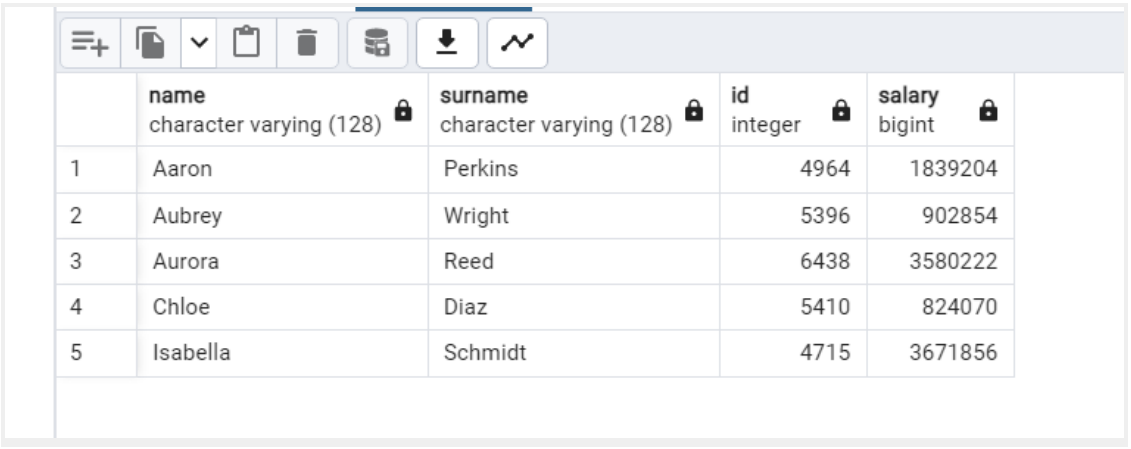
Задание 4. Создать хранимые процедуры:

1. **Для повышения оклада сотрудников, выполнивших задания с трехдневным опережением графика на заданный процент.**

```
create or replace procedure raise_salary(in prcnt integer)
language plpgsql as $$
begin
   update tasks.employment_tab
        set salary = salary * (1 + prcnt / 100)
    where id in (
         select distinct empl.id
           from tasks.employment_tab empl
            join tasks.tasks_tab task on task.employment_id = empl.id
          where task.due_date = task.check_date + 3 -- задание проверено на 3 дня раньше заданного + статус "завершено"
            and task.status = 'Done'
      );
end;
$$;
```

Проверим через следующий запрос:

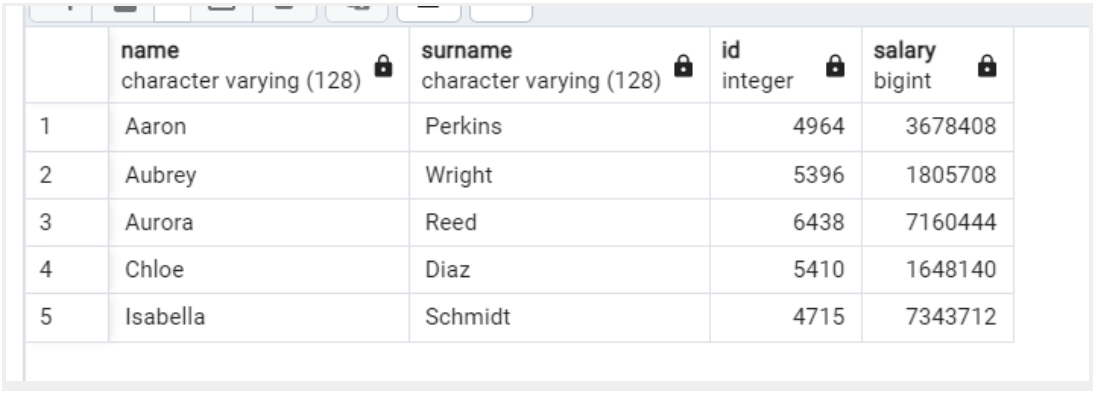
```
select distinct emp.name, emp.surname, empl.id, empl.salary
 from tasks.employment_tab empl
 join tasks.employees_tab emp on emp.id = empl.employee_id
 join tasks.tasks_tab task on task.employment_id = empl.id
where task.due_date = task.check_date + 3
  and task.status = 'Done';
```



	name character varying (128)	surname character varying (128)	id integer	salary bigint
1	Aaron	Perkins	4964	1839204
2	Aubrey	Wright	5396	902854
3	Aurora	Reed	6438	3580222
4	Chloe	Diaz	5410	824070
5	Isabella	Schmidt	4715	3671856

До процедуры

```
call raise_salary(100);
```



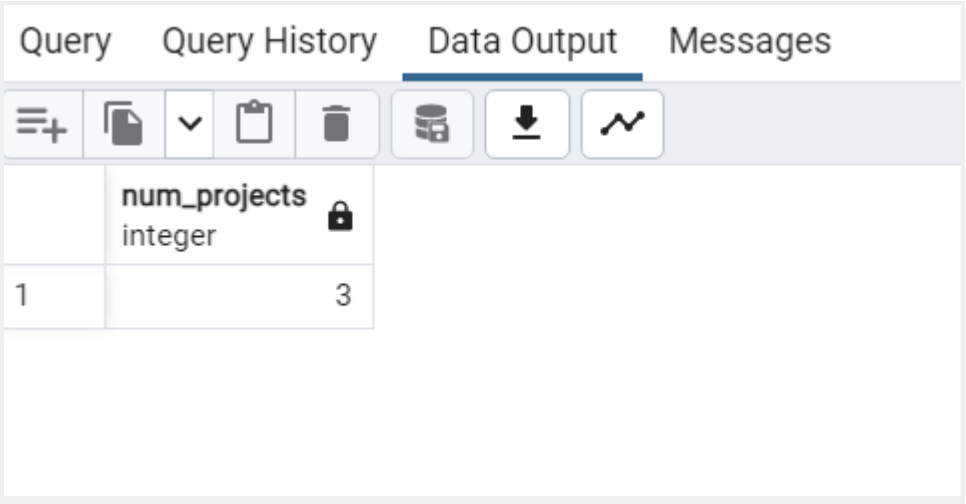
	name character varying (128)	surname character varying (128)	id integer	salary bigint
1	Aaron	Perkins	4964	3678408
2	Aubrey	Wright	5396	1805708
3	Aurora	Reed	6438	7160444
4	Chloe	Diaz	5410	1648140
5	Isabella	Schmidt	4715	7343712

После процедуры

2. **Для вычисления количества проектов, в выполнении которых участвует сотрудник.**

```
create or replace function num_projects(empl_id integer)
returns integer as $$
declare
  vProjectCount integer;
begin
  select count(distinct proj.id) into vProjectCount
    from tasks.employees_tab empl
    join tasks.employment_tab emp on emp.employee_id = empl.id
    left join tasks.projects_tab proj on proj.id = task.project_id
    left join tasks.tasks_tab task on task.employment_id = emp.id
    left join tasks.steps_tab step on step.employment_id =  emp.id
   where empl.id = empl_id;
  return vProjectCount;
end;
$$ language plpgsql;

select num_projects(10);
```



Результат выполнения

3. **Для поиска номера телефона сотрудника (телефон установлен в каждом отделе).**

Т.к. сотрудник может работать в различных отделах (частичная занятость), для удобства функция возвращает тип table, чтобы получить информацию об представляющем сотрудника отделе и возможности представлять все возможные телефонные номера как таблицу.

```
create or replace function get_phone_number(empl_id integer)
returns table(dep_name varchar, phone_num varchar) as $$
  select dep.name, dep.phone_number
    from tasks.employees_tab empl
      join tasks.employment_tab emp on emp.employee_id = empl.id
      join tasks.departments_tab dep on dep.id = emp.department_id
  where empl.id = empl_id;
$$ language sql;

select (get_phone_number(10)).dep_name, (get_phone_number(10)).phone_num;
```

	dep_name character varying	phone_num character varying
1	Legal	82530373692
2	Human Resources	86932135029
3	Sales	86961261737

Результат выполнения

Часть 2. Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных

Создание универсальной таблицы для логирования и функции:

```
create table global_log_tab(
  id             serial primary key,
  tab_name     varchar,
  tab_schema varchar,
  row_id         bigint,
  operation     varchar(1),
  operation_time timestamp
);

create or replace function logger() returns trigger as $$
declare
   oper varchar(1);
   row_id integer;
begin
   if  TG_OP = 'INSERT' then
       row_id = NEW.id;
       oper := 'I';
       INSERT INTO global_log_tab(tab_name, tab_schema, row_id, operation,
operation_time)
               values (TG_TABLE_NAME, TG_TABLE_SCHEMA, row_id, oper, now());
       return new;
  elsif TG_OP = 'UPDATE' then
       row_id = NEW.id;
       oper := 'U';
       INSERT INTO global_log_tab(tab_name, tab_schema, row_id, operation,
operation_time)
               values (TG_TABLE_NAME, TG_TABLE_SCHEMA, row_id, oper, now());
       return new;
  elsif TG_OP = 'DELETE' then
       row_id = OLD.id;
       oper := 'D';
       INSERT INTO global_log_tab(tab_name, tab_schema, row_id, operation,
operation_time)
               values (TG_TABLE_NAME, TG_TABLE_SCHEMA, row_id, oper, now());
       return old;
   end if;
end;
$$ language plpgsql;
```

Создадим процедуру для автоматизации создания лог-триггеров для таблиц в нашей базе - теперь можно легко настраивать логирование для пачек новых таблиц:

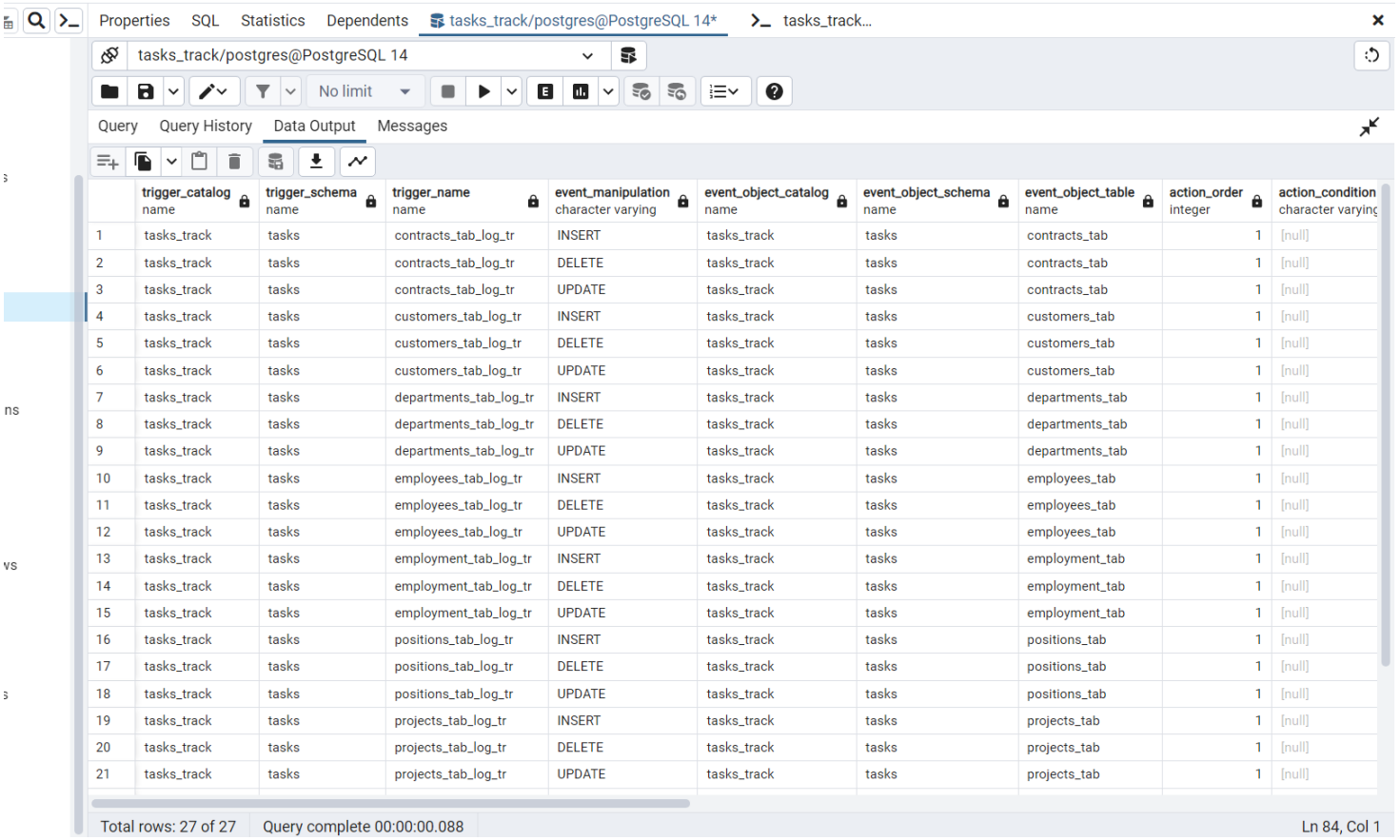
```
create or replace procedure create_triggers() AS $$
declare
  rec record;
begin
  for rec in
    select t.table_schema, t.table_name
      from information_schema.tables t
        left join information_schema.triggers tr
```

```
        on tr.event_object_table = t.table_name -- джойним чтобы проверять наличие
триггеров
   where t.table_schema = 'tasks'
     and t.table_type = 'BASE TABLE'
     and tr.event_object_table is null -- условие, чтобы не пересоздавать уже
созданные триггеры
 loop
   RAISE NOTICE 'Currently creating trigger for table  %.% ',
   quote_ident(rec.table_schema),
   quote_ident(rec.table_name);

   EXECUTE format('create trigger %I_log_tr after insert or update or
                        delete on %I.%I for each row execute procedure logger();',
                        rec.table_name, rec.table_schema, rec.table_name);
 end loop;
END;
$$ LANGUAGE plpgsql;

call create_triggers();

select * from information_schema.triggers;
```



Properties SQL Statistics Dependents tasks_track/postgres@PostgreSQL 14* tasks_track...

tasks_track/postgres@PostgreSQL 14

No limit

Query Query History Data Output Messages

	trigger_catalog name	trigger_schema name	trigger_name name	event_manipulation character varying	event_object_catalog name	event_object_schema name	event_object_table name	action_order integer	action_condition character varying
1	tasks_track	tasks	contracts_tab_log_tr	INSERT	tasks_track	tasks	contracts_tab	1	[null]
2	tasks_track	tasks	contracts_tab_log_tr	DELETE	tasks_track	tasks	contracts_tab	1	[null]
3	tasks_track	tasks	contracts_tab_log_tr	UPDATE	tasks_track	tasks	contracts_tab	1	[null]
4	tasks_track	tasks	customers_tab_log_tr	INSERT	tasks_track	tasks	customers_tab	1	[null]
5	tasks_track	tasks	customers_tab_log_tr	DELETE	tasks_track	tasks	customers_tab	1	[null]
6	tasks_track	tasks	customers_tab_log_tr	UPDATE	tasks_track	tasks	customers_tab	1	[null]
7	tasks_track	tasks	departments_tab_log_tr	INSERT	tasks_track	tasks	departments_tab	1	[null]
8	tasks_track	tasks	departments_tab_log_tr	DELETE	tasks_track	tasks	departments_tab	1	[null]
9	tasks_track	tasks	departments_tab_log_tr	UPDATE	tasks_track	tasks	departments_tab	1	[null]
10	tasks_track	tasks	employees_tab_log_tr	INSERT	tasks_track	tasks	employees_tab	1	[null]
11	tasks_track	tasks	employees_tab_log_tr	DELETE	tasks_track	tasks	employees_tab	1	[null]
12	tasks_track	tasks	employees_tab_log_tr	UPDATE	tasks_track	tasks	employees_tab	1	[null]
13	tasks_track	tasks	employment_tab_log_tr	INSERT	tasks_track	tasks	employment_tab	1	[null]
14	tasks_track	tasks	employment_tab_log_tr	DELETE	tasks_track	tasks	employment_tab	1	[null]
15	tasks_track	tasks	employment_tab_log_tr	UPDATE	tasks_track	tasks	employment_tab	1	[null]
16	tasks_track	tasks	positions_tab_log_tr	INSERT	tasks_track	tasks	positions_tab	1	[null]
17	tasks_track	tasks	positions_tab_log_tr	DELETE	tasks_track	tasks	positions_tab	1	[null]
18	tasks_track	tasks	positions_tab_log_tr	UPDATE	tasks_track	tasks	positions_tab	1	[null]
19	tasks_track	tasks	projects_tab_log_tr	INSERT	tasks_track	tasks	projects_tab	1	[null]
20	tasks_track	tasks	projects_tab_log_tr	DELETE	tasks_track	tasks	projects_tab	1	[null]
21	tasks_track	tasks	projects_tab_log_tr	UPDATE	tasks_track	tasks	projects_tab	1	[null]

Total rows: 27 of 27 Query complete 00:00:00.088 Ln 84, Col 1

Триггеры для всех таблиц БД

```
68
69  insert into tasks.projects_tab(id, status, project_name, manager_id, supervisor_id, customer_id, project_cost,
70                                 contract_date, contract_finish_date, beginning_date, finish_date)
71                                 values (1966, 'Done', 'project 1966', 5514, 6912,15,7833258,
72                                 '2011-03-15', '2011-06-15', '2011-03-15', '2011-07-15');
73  insert into tasks.contracts_tab(project_id, contract_date, payment_date, cost, status)
74  values(1966, '2011-06-15', '2011-07-15', 3459319, 'Fully_payed');
75  insert into tasks.tasks_tab(id, status, project_id, employment_id, name,
76                              due_date, task_cost, check_date)
77                              values (39341, 'Not_started', 1966,
78                              5844, 'task 21','2010-01-11',696360,'2011-06-15');
79  insert into tasks.tasks_tab(id, status, project_id, employment_id, name, due_date, task_cost, check_date)
80  values (41308, 'Frozen', 1966, 6338, 'task 22','2010-01-05',864858,'2011-06-15');
81  insert into tasks.steps_tab(task_id, employment_id, status, start_date, end_date, name, step_cost)
82  values (41308, 4542, 'Frozen', '2011-05-14', null, 'step 0', 97353);
83  insert into tasks.steps_tab(task_id, employment_id, status, start_date, end_date, name, step_cost)
84  values (41308, 6476, 'Frozen', '2010-01-15', null, 'step 1', 97073);
85  update tasks.projects_tab set finish_date = now() where id = 1966;
86  delete from tasks.steps_tab where task_id = 41308;
87
88  select * from global_log_tab;
89
```

Сделаем несколько операций и проверим наличие логирования

	id [PK] integer	tab_name character varying	tab_schema character varying	row_id bigint	operation character varying (1)	operation_time timestamp without time zone
1	1	projects_tab	tasks	1966	I	2023-05-19 01:19:47.093249
2	2	contracts_tab	tasks	1865	I	2023-05-19 01:19:47.093249
3	3	tasks_tab	tasks	39341	I	2023-05-19 01:19:47.093249
4	4	tasks_tab	tasks	41308	I	2023-05-19 01:19:47.093249
5	5	steps_tab	tasks	16786	I	2023-05-19 01:19:47.093249
6	6	steps_tab	tasks	16787	I	2023-05-19 01:19:47.093249
7	7	projects_tab	tasks	1966	U	2023-05-19 01:21:31.661009
8	8	steps_tab	tasks	16786	D	2023-05-19 01:21:35.45796
9	9	steps_tab	tasks	16787	D	2023-05-19 01:21:35.45796

Таблица global_log_tab

Вывод:

В процессе выполнения данной лабораторной работы я овладела практическими навыками создания и использования процедур, функций и триггеров в базе данных PostgreSQL.