Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(Университет ИТМО)

Факультет **Инфокоммуникационных технологий**

Образовательная программа **Мобильные и сетевые технологии**

Направление подготовки (специальность) **09.03.03 Прикладная информатика**

О Т Ч Е Т

по дисциплине «Проектирование и реализация баз данных»

на тему: процедуры, функции, триггеры в PostgreSQL

Обучающийся: Олейникова Полина Леонидовна, K32402

Преподаватель: Говорова М. М.

Дата 14.05.2023

Санкт-Петербург 2023

ВВЕДЕНИЕ

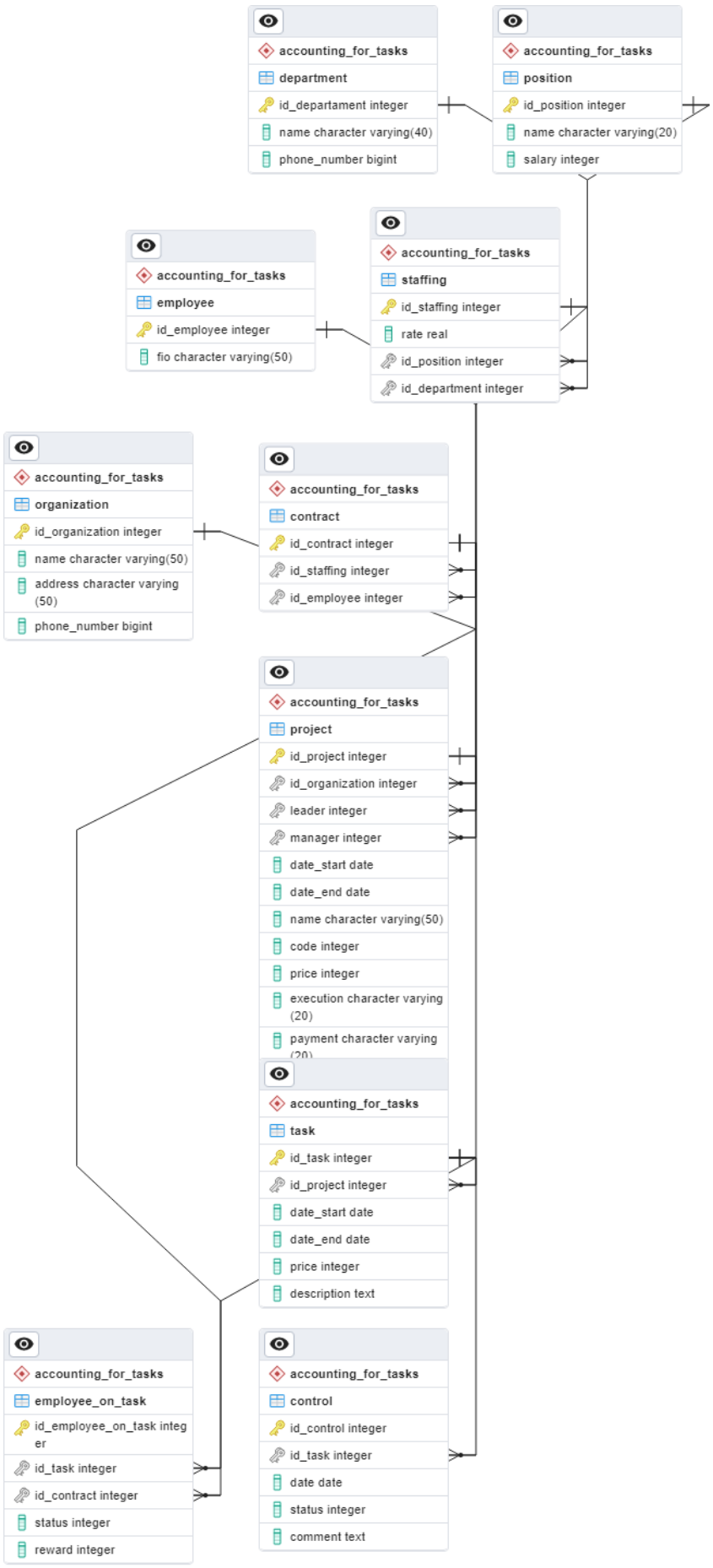
Цель работы: овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

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.

Индивидуальное задание: Вариант 4. БД «Учет выполнения заданий»

Схема логической модели базы данных:


accounting_for_tasks
department
id_departament integer
name character varying(40)
phone_number bigint
accounting_for_tasks
position
id_position integer
name character varying(20)
salary integer
accounting_for_tasks
employee
id_employee integer
fio character varying(50)
accounting_for_tasks
staffing
id_staffing integer
rate real
id_position integer
id_department integer
accounting_for_tasks
organization
id_organization integer
name character varying(50)
address character varying (50)
phone_number bigint
accounting_for_tasks
contract
id_contract integer
id_staffing integer
id_employee integer
accounting_for_tasks
project
id_project integer
id_organization integer
leader integer
manager integer
date_start date
date_end date
name character varying(50)
code integer
price integer
execution character varying (20)
payment character varying (20)
accounting_for_tasks
task
id_task integer
id_project integer
date_start date
date_end date
price integer
description text
accounting_for_tasks
employee_on_task
id_employee_on_task integer
id_task integer
id_contract integer
status integer
reward integer
accounting_for_tasks
control
id_control integer
id_task integer
date date
status integer
comment text

1 Выполнение

1.1 Процедуры\функции

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

```
Accounting for tasks=# select * from accounting_for_tasks.position;
 id_position |     name     | salary
-------------+--------------+--------
           1 | программист  | 150000
           2 | руководитель | 255000
           3 | лаборант     |  55125
(3 строки)
```

```
create procedure accounting_for_tasks.salary_increase() AS $$ BEGIN
update
  accounting_for_tasks.position
set
  salary = salary * 1.05
where
  id_position in (
    select
      id_position
    from
      accounting_for_tasks.control
      join accounting_for_tasks.task using(id_task)
      join accounting_for_tasks.employee_on_task using(id_task)
      join accounting_for_tasks.contract using(id_contract)
      join accounting_for_tasks.staffing using(id_staffing)
      join accounting_for_tasks.position using(id_position)
    where
      control.status = 100
      and task.date_end - control.date > 3
  );
END;
$$ LANGUAGE plpgsql;
```

```
Accounting for tasks=# create procedure accounting_for_tasks.salary_increase() AS $$ BEGIN
Accounting for tasks$# update
Accounting for tasks$#   accounting_for_tasks.position
Accounting for tasks$# set
Accounting for tasks$#   salary = salary * 1.05
Accounting for tasks$# where
Accounting for tasks$#   id_position in (
Accounting for tasks$#     select
Accounting for tasks$#       id_position
Accounting for tasks$#     from
Accounting for tasks$#       accounting_for_tasks.control
Accounting for tasks$#       join accounting_for_tasks.task using(id_task)
Accounting for tasks$#       join accounting_for_tasks.employee_on_task using(id_task)
Accounting for tasks$#       join accounting_for_tasks.contract using(id_contract)
Accounting for tasks$#       join accounting_for_tasks.staffing using(id_staffing)
Accounting for tasks$#       join accounting_for_tasks.position using(id_position)
Accounting for tasks$#     where
Accounting for tasks$#       control.status = 100
Accounting for tasks$#       and task.date_end - control.date > 3
Accounting for tasks$#   );
Accounting for tasks$# END;
Accounting for tasks$# $$ LANGUAGE plpgsql;
CREATE PROCEDURE
```

```
Accounting for tasks=# call accounting_for_tasks.salary_increase();
CALL
Accounting for tasks=# select * from accounting_for_tasks.position;
 id_position |     name     | salary
-------------+--------------+--------
           1 | программист  | 150000
           2 | руководитель | 255000
           3 | лаборант     |  57881
(3 строки)
```

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

```
CREATE FUNCTION accounting_for_tasks.emp_project(emp int)
RETURNS TABLE (count bigint) AS $$ BEGIN
    RETURN QUERY
    SELECT Count(id_project)
    FROM   (SELECT id_project,
                   id_contract
            FROM   accounting_for_tasks.employee_on_task
                   JOIN accounting_for_tasks.task using(id_task)
                   JOIN accounting_for_tasks.project using(id_project)
            GROUP  BY id_project,
                      id_contract
            HAVING id_contract = emp) AS emp_pr
    GROUP  BY id_contract;
    END;
    $$ LANGUAGE plpgsql;
```

```
Accounting for tasks=# CREATE FUNCTION accounting_for_tasks.emp_project(emp int) RETURNS TABLE (count bigint) AS $$ BEGI
N
Accounting for tasks$# RETURN QUERY
Accounting for tasks$# SELECT Count(id_project)
Accounting for tasks$# FROM   (SELECT id_project,
Accounting for tasks$#                id_contract
Accounting for tasks$#         FROM   accounting_for_tasks.employee_on_task
Accounting for tasks$#                JOIN accounting_for_tasks.task using(id_task)
Accounting for tasks$#                JOIN accounting_for_tasks.project using(id_project)
Accounting for tasks$#         GROUP  BY id_project,
Accounting for tasks$#                   id_contract
Accounting for tasks$#         HAVING id_contract = emp) AS emp_pr
Accounting for tasks$# GROUP  BY id_contract;
Accounting for tasks$# END;
Accounting for tasks$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
```

```
Accounting for tasks=# select * from accounting_for_tasks.emp_project(1);
 count
-------
     2
(1 строка)
```

3) Для поиска номера телефона сотрудника.

CREATE FUNCTION accounting_for_tasks.emp_phone_number(emp int) RETURNS TABLE (emp_phone bigint) AS $$ BEGIN
RETURN QUERY
SELECT phone_number
FROM accounting_for_tasks.contract
JOIN accounting_for_tasks.staffing using(id_staffing)
JOIN accounting_for_tasks.department using(id_department)
WHERE id_contract = emp;
END;
$$ LANGUAGE plpgsql;

```
Accounting for tasks=# CREATE FUNCTION accounting_for_tasks.emp_phone_number(emp int) RETURNS TABLE (emp_phone bigint) A
S $$ BEGIN
Accounting for tasks$# RETURN QUERY
Accounting for tasks$# SELECT phone_number
Accounting for tasks$# FROM   accounting_for_tasks.contract
Accounting for tasks$#        JOIN accounting_for_tasks.staffing using(id_staffing)
Accounting for tasks$#        JOIN accounting_for_tasks.department using(id_department)
Accounting for tasks$# WHERE  id_contract = emp;
Accounting for tasks$# END;
Accounting for tasks$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
Accounting for tasks=# select * from accounting_for_tasks.emp_phone_number(1);
  emp_phone
------------
 89128967277
(1 строка)
```

1.2 Триггер для логирования событий вставки, удаления, редактирования

CREATE TRIGGER t_employee AFTER INSERT OR UPDATE OR DELETE ON accounting_for_tasks.employee FOR EACH ROW EXECUTE PROCEDURE accounting_for_tasks.add_to_log ();

```
CREATE OR REPLACE FUNCTION accounting_for_tasks.add_to_log()
RETURNS TRIGGER AS $$
DECLARE
   mstr varchar(30);
   astr varchar(100);
   retstr varchar(254);
BEGIN
   IF    TG_OP = 'INSERT' THEN
      astr = NEW;
      mstr := 'Add data ';
      retstr := mstr||astr;
      INSERT INTO accounting_for_tasks.logs(text, added, table_name) values
(retstr,NOW(), TG_TABLE_NAME);
      RETURN NEW;
   ELSIF TG_OP = 'UPDATE' THEN
      astr = NEW;
      mstr := 'Update data ';
      retstr := mstr||astr;
      INSERT INTO accounting_for_tasks.logs(text, added, table_name) values
(retstr,NOW(), TG_TABLE_NAME);
      RETURN NEW;
   ELSIF TG_OP = 'DELETE' THEN
      astr = OLD;
      mstr := 'Remove data ';
      retstr := mstr || astr;
      INSERT INTO accounting_for_tasks.logs(text, added, table_name) values
(retstr,NOW(), TG_TABLE_NAME);
      RETURN OLD;
   END IF;
END;
$$ LANGUAGE plpgsql;
```

SQL Shell (psql)

```
Accounting for tasks$# DECLARE
Accounting for tasks$#     mstr varchar(30);
Accounting for tasks$#     astr varchar(100);
Accounting for tasks$#     retstr varchar(254);
Accounting for tasks$# BEGIN
Accounting for tasks$#     IF    TG_OP = 'INSERT' THEN
Accounting for tasks$#         astr = NEW;
Accounting for tasks$#         mstr := 'Add data ';
Accounting for tasks$#         retstr := mstr||astr;
Accounting for tasks$#         INSERT INTO accounting_for_tasks.logs(text, added, table_name) values (retstr,NOW(), TG_T
ABLE_NAME);
Accounting for tasks$#         RETURN NEW;
Accounting for tasks$#     ELSIF TG_OP = 'UPDATE' THEN
Accounting for tasks$#         astr = NEW;
Accounting for tasks$#         mstr := 'Update data ';
Accounting for tasks$#         retstr := mstr||astr;
Accounting for tasks$#         INSERT INTO accounting_for_tasks.logs(text, added, table_name) values (retstr,NOW(), TG_T
ABLE_NAME);
Accounting for tasks$#         RETURN NEW;
Accounting for tasks$#     ELSIF TG_OP = 'DELETE' THEN
Accounting for tasks$#         astr = OLD;
Accounting for tasks$#         mstr := 'Remove data ';
Accounting for tasks$#         retstr := mstr || astr;
Accounting for tasks$#         INSERT INTO accounting_for_tasks.logs(text, added, table_name) values (retstr,NOW(), TG_T
ABLE_NAME);
Accounting for tasks$#         RETURN OLD;
Accounting for tasks$#     END IF;
Accounting for tasks$# END;
Accounting for tasks$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
```

```
Accounting for tasks=# CREATE TRIGGER t_employee AFTER INSERT OR UPDATE OR DELETE ON accounting_for_tasks.employee FOR E
ACH ROW EXECUTE PROCEDURE accounting_for_tasks.add_to_log ();
CREATE TRIGGER
Accounting for tasks=#
```

```
Accounting for tasks=# insert into accounting_for_tasks.employee (id_employee, fio) values (5, 'Дмитриев Дмитрий Сергеев
ич');
INSERT 0 1
Accounting for tasks=# update  accounting_for_tasks.employee set fio='Сергеев Дмитрий Сергеевич' where id_employee=5;
UPDATE 1
Accounting for tasks=# delete from accounting_for_tasks.employee where id_employee=5
Accounting for tasks-# ;
DELETE 1
Accounting for tasks=# select * from accounting_for_tasks.logs;
                  text                   |           added            | table_name
-----------------------------------------+----------------------------+------------
 Add data (5,"Дмитриев Дмитрий Сергеевич")   | 2023-05-14 16:22:07.160223 | employee
 Update data (5,"Сергеев Дмитрий Сергеевич") | 2023-05-14 16:23:43.665128 | employee
 Remove data (5,"Сергеев Дмитрий Сергеевич") | 2023-05-14 16:24:21.428622 | employee
(3 строки)
```

ЗАКЛЮЧЕНИЕ

В данной работе мною были изучены функции и процедуры, а также созданы триггеры для корректного хранения данных.