МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«МИРЭА – Российский технологический университет» РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

Клиент-серверные системы управления банком данных

Практическая работа 2-5.

Управление контактами с клиентами

Выполнили студенты группы:

БСБО-05-20

Зарин Н. Н.

Луговой И. И.

Москва 2022 г.

Задание на практику:

1. Работа с ER-диаграммой и структурой базы данных
2. Заполнение базы данных, создание ролей, назначение привилегий
3. Автоматизация функционала
4. Реализация индексов, работа с выгрузкой данных

Отчет оформить в формате doc (docx) или pdf и выслать на проверку

Выполнение задания

1. Работа с ER-диаграммой и структурой базы данных
   ER – диаграмма изображена на рис.1

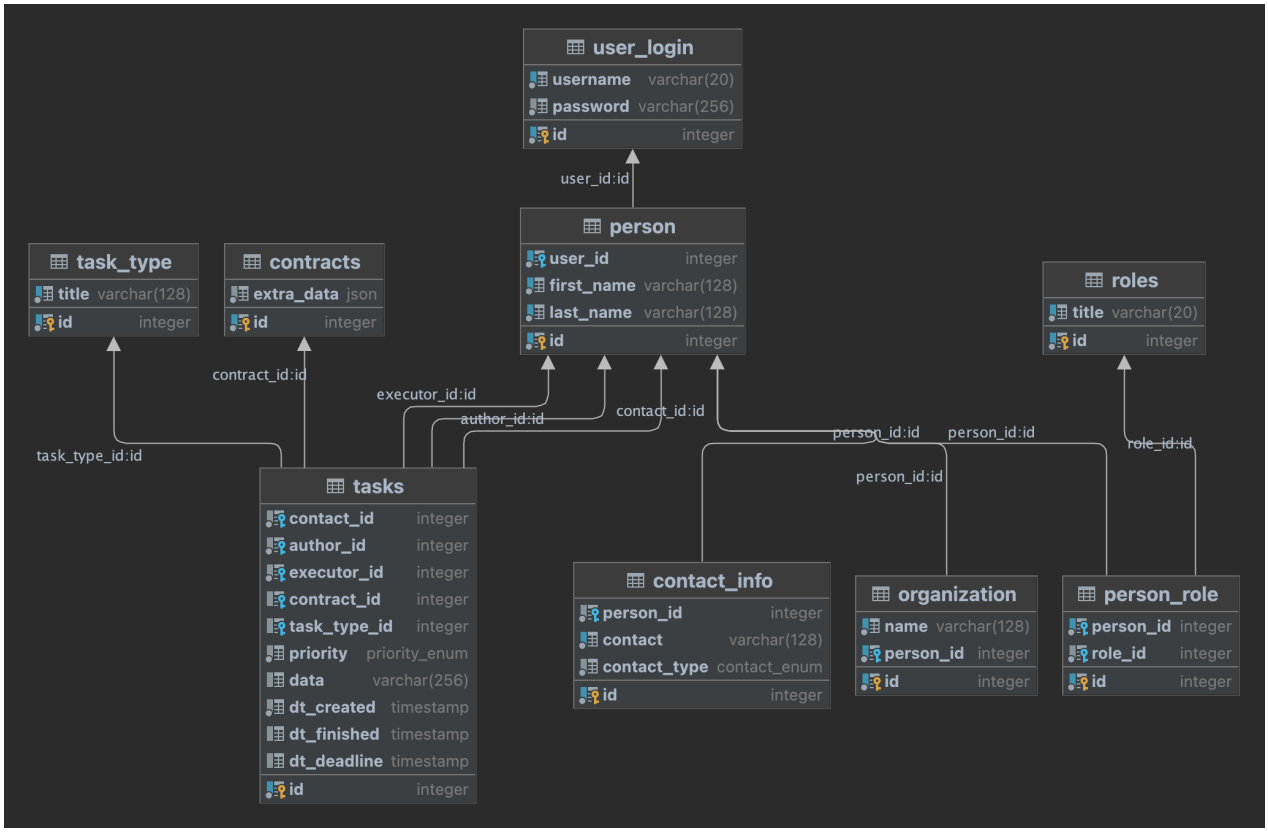

Рис. 1 – ER-диаграмма.

2. Заполнение базы данных, создание ролей, назначение привилегий

   2.1 Листинг скрипта для создания БД

```
DROP DATABASE IF EXISTS mirea;

CREATE DATABASE mirea;

CREATE TYPE contact_enum AS ENUM ('email', 'телефон', 'адрес');
CREATE TYPE priority_enum AS ENUM ('низкий', 'средний', 'высокий');

CREATE TABLE IF NOT EXISTS user_login
(
    id SERIAL PRIMARY KEY NOT NULL,
    username VARCHAR(20) NOT NULL UNIQUE,
    password VARCHAR(256) NOT NULL
);

CREATE TABLE IF NOT EXISTS person
(
    id SERIAL PRIMARY KEY NOT NULL UNIQUE,
    user_id INTEGER REFERENCES user_login(id) ON DELETE CASCADE NOT NULL
UNIQUE,
    first_name VARCHAR(128) NOT NULL,
    last_name VARCHAR(128) NOT NULL
);

CREATE TABLE IF NOT EXISTS contact_info
(
```

```
    id SERIAL PRIMARY KEY NOT NULL UNIQUE,
    person_id INTEGER REFERENCES person(id) ON DELETE CASCADE NOT NULL,
    contact VARCHAR(128) NOT NULL UNIQUE,
    contact_type contact_enum NOT NULL
);

CREATE TABLE IF NOT EXISTS roles
(
    id SERIAL PRIMARY KEY NOT NULL UNIQUE,
    title VARCHAR(20) NOT NULL UNIQUE
);

CREATE TABLE IF NOT EXISTS person_role
(
    id SERIAL PRIMARY KEY NOT NULL UNIQUE,
    person_id INTEGER REFERENCES person(id) ON DELETE CASCADE NOT NULL UNIQUE,
    role_id INTEGER REFERENCES roles(id) ON DELETE RESTRICT NOT NULL
);

CREATE TABLE IF NOT EXISTS organization
(
    id SERIAL PRIMARY KEY NOT NULL UNIQUE,
    name VARCHAR(128) NOT NULL UNIQUE,
    person_id INTEGER REFERENCES person(id) ON DELETE CASCADE NOT NULL UNIQUE
);

CREATE TABLE IF NOT EXISTS task_type
(
    id SERIAL PRIMARY KEY NOT NULL UNIQUE,
    title VARCHAR(128) NOT NULL UNIQUE
);

CREATE TABLE IF NOT EXISTS contracts
(
    id SERIAL PRIMARY KEY NOT NULL UNIQUE,
    extra_data json NOT NULL
);

CREATE TABLE IF NOT EXISTS tasks
(
    id SERIAL PRIMARY KEY NOT NULL UNIQUE,
    contact_id INTEGER REFERENCES person(id) ON DELETE CASCADE NOT NULL,
    author_id INTEGER REFERENCES person(id) ON DELETE NO ACTION NOT NULL,
    executor_id INTEGER REFERENCES person(id) ON DELETE NO ACTION NOT NULL,
    contract_id INTEGER REFERENCES contracts(id) ON DELETE CASCADE NULL,
    task_type_id INTEGER REFERENCES task_type(id) ON DELETE NO ACTION NULL,
    priority priority_enum NOT NULL,
    data VARCHAR(256) NULL,
    dt_created TIMESTAMP NOT NULL,
    dt_finished TIMESTAMP NULL,
    dt_deadline TIMESTAMP NULL
);

CREATE FUNCTION get_login_id_by_login(name VARCHAR)
    RETURNS INTEGER
    LANGUAGE plpgsql AS
    $func$
    DECLARE ret INTEGER;
    BEGIN
       SELECT id INTO ret FROM user_login WHERE username = name;
       RETURN ret;
    END
    $func$;
```

```
CREATE FUNCTION get_person_id_by_login(name VARCHAR)
    RETURNS INTEGER
    LANGUAGE plpgsql AS
    $func$
    DECLARE ret INTEGER;
    BEGIN
       SELECT P.id INTO ret FROM user_login ul
       INNER JOIN person p ON ul.id = p.user_id
       WHERE ul.username = name;
       RETURN ret;
    END
    $func$;

CREATE FUNCTION get_role_id(name VARCHAR)
    RETURNS INTEGER
    LANGUAGE plpgsql AS
    $func$
    DECLARE ret INTEGER;
    BEGIN
       SELECT id INTO ret FROM roles WHERE title = name;
       RETURN ret;
    END
    $func$;

CREATE FUNCTION get_task_type_id(name VARCHAR)
    RETURNS INTEGER
    LANGUAGE plpgsql AS
    $func$
    DECLARE ret INTEGER;
    BEGIN
       SELECT id INTO ret FROM task_type WHERE title = name;
       RETURN ret;
    END
    $func$;
```

2.2 Листинг скрипта для заполнения базы данных

```
INSERT INTO roles(title)
    VALUES
        ('Администратор'),
        ('Менеджер'),
        ('Сотрудник'),
        ('Клиент');

INSERT INTO user_login(username, password)
    VALUES
        ('admin', sha256('mj4ir9^*&$gdhfb87&#(*b')),
        ('manager1', sha256('f6$*86%^&*$fgyu^*&^*%!')),
        ('manager2', sha256('tv^%C%^B&^8t*6rB^&Dd8D8D')),
        ('worker1', sha256('g67F%V&&(*B^&V*&)DV$$S$')),
        ('worker2', sha256('V^%BF%^#C%^DB#&D^*#&FD(@5d')),
        ('client1', sha256('H^&VG%^D$%VTV67v56C%DD$#X')),
        ('client2', sha256('^&FD&^F#DF^*IGD*MBD*G^S&F^')),
        ('client3', sha256('nyBY&G^&V67f^%^FE^&F#6digf6')),
        ('client4', sha256('67v56DC%^*#^&f56f5C84g7v8b'));

INSERT INTO person(user_id, first_name, last_name)
    VALUES
        (get_login_id_by_login('admin'), 'Билли', 'Херингтон'),
        (get_login_id_by_login('manager1'), 'Тодд', 'Говард'),
        (get_login_id_by_login('manager2'), 'Жак', 'Фреско'),
        (get_login_id_by_login('worker1'), 'Илья', 'Луговой'),
```

```
        (get_login_id_by_login('worker2'), 'Никита', 'Зарин'),
        (get_login_id_by_login('client1'), 'Дмитрий', 'Пучков'),
        (get_login_id_by_login('client2'), 'Максим', 'Кац'),
        (get_login_id_by_login('client3'), 'Рулон', 'Обоев'),
        (get_login_id_by_login('client4'), 'Ушат', 'Помоев');

INSERT INTO person_role(person_id, role_id)
    VALUES
        (get_person_id_by_login('admin'), get_role_id('Администратор')),
        (get_person_id_by_login('manager1'), get_role_id('Менеджер')),
        (get_person_id_by_login('manager2'), get_role_id('Менеджер')),
        (get_person_id_by_login('worker1'), get_role_id('Сотрудник')),
        (get_person_id_by_login('worker2'), get_role_id('Сотрудник')),
        (get_person_id_by_login('client1'), get_role_id('Клиент')),
        (get_person_id_by_login('client2'), get_role_id('Клиент')),
        (get_person_id_by_login('client3'), get_role_id('Клиент')),
        (get_person_id_by_login('client4'), get_role_id('Клиент'));

INSERT INTO contact_info(person_id, contact, contact_type)
    VALUES
        (get_person_id_by_login('admin'), '+77777777777', 'телефон'),
        (get_person_id_by_login('admin'), 'master@dungeon.com', 'email'),
        (get_person_id_by_login('admin'), 'Москва, Темный переулок, дом 69',
'адрес'),
        (get_person_id_by_login('manager1'), 'info@bethsoft.com', 'email'),
        (get_person_id_by_login('manager1'), '237207, Тульская область, город
Подольск, наб. Ломоносова, 07', 'адрес'),
        (get_person_id_by_login('manager2'), '478611, Смоленская область,
город Луховицы, бульвар Балканская, 98', 'адрес'),
        (get_person_id_by_login('worker1'), '+79152979221', 'телефон'),
        (get_person_id_by_login('worker1'), 'ilya.l2013@yandex.ru', 'email'),
        (get_person_id_by_login('worker2'), '+79153682126', 'телефон'),
        (get_person_id_by_login('worker2'), 'nikita0zrz@gmail.com', 'email'),
        (get_person_id_by_login('worker2'), 'Барвиха, дом 6', 'адрес'),
        (get_person_id_by_login('client1'), '+79097146321', 'email'),
        (get_person_id_by_login('client1'), '+79187122692', 'телефон'),
        (get_person_id_by_login('client1'), '870701, Пензенская область, город
Дорохово, бульвар Сталина, 12', 'адрес'),
        (get_person_id_by_login('client2'), 'rosemary64@gmail.com', 'email'),
        (get_person_id_by_login('client2'), '+79036362845', 'телефон'),
        (get_person_id_by_login('client2'), '038442, Белгородская область,
город Талдом, проезд Косиора, 42', 'адрес'),
        (get_person_id_by_login('client3'), 'kyra.conn@gmail.com', 'email'),
        (get_person_id_by_login('client3'), '+79163963611', 'телефон'),
        (get_person_id_by_login('client3'), '016809, Смоленская область, город
Домодедово, пр. Домодедовская, 10', 'адрес'),
        (get_person_id_by_login('client4'), '247021, Калининградская область,
город Орехово-Зуево, въезд Будапештсткая, 77', 'адрес'),
        (get_person_id_by_login('client4'), '+79008683644', 'телефон');

INSERT INTO organization(name, person_id)
    VALUES
        ('Dungeon Inc.', get_person_id_by_login('client1')),
        ('Umbrella Corp.', get_person_id_by_login('client2')),
        ('Gym Inc.', get_person_id_by_login('client3')),
        ('LZ Software', get_person_id_by_login('client4'));

INSERT INTO task_type(title)
    VALUES
        ('Отправка оборудования'),
        ('Ремонт оборудования'),
        ('Установка оборудования'),
        ('Звонок'),
```

```
        ('Диагностика'),
        ('Встреча с клиентом');

INSERT INTO contracts(extra_data)
    VALUES
        ('{"Серийный номер": "ZM9RM646873865783", "Производитель": "Toshiba",
"Окончание гарантии": "01-01-2023"}'),
        ('{"Серийный номер": "VT6ED019075942003", "Производитель": "Bosch",
"Точка назначения": "Метро Румянцево, KFC"}'),
        ('{"Серийный номер": ["UN7LG794060753856", "ML6EV666885361170"],
"Производитель": "Apple", "Кол-во": 2}'),
        ('{"Серийный номер": "VL1EX420482601375", "Производитель": "Samsung",
"Окончание гарантии": "01-05-2022"}'),
        ('{"Серийный номер": "XS5LE011324719550", "Производитель":
"Huawei"}');

INSERT INTO tasks(contact_id, author_id, executor_id, contract_id,
task_type_id, priority, data, dt_created, dt_finished, dt_deadline)
    VALUES
        (get_person_id_by_login('client1'),
         get_person_id_by_login('manager1'),
         get_person_id_by_login('worker1'),
         1,
         get_task_type_id('Ремонт оборудования'),
         'средний',
         NULL,
         now()::timestamp,
         NULL,
         now()::timestamp + INTERVAL '7 days'),
        (get_person_id_by_login('client2'),
         get_person_id_by_login('manager2'),
         get_person_id_by_login('manager2'),
         5,
         get_task_type_id('Диагностика'),
         'низкий',
         'Не заряжается устройство',
         now()::timestamp,
         NULL,
         now()::timestamp + INTERVAL '1 days'),
        (get_person_id_by_login('client3'),
         get_person_id_by_login('manager1'),
         get_person_id_by_login('worker2'),
         2,
         get_task_type_id('Отправка оборудования'),
         'высокий',
         NULL,
         now()::timestamp,
         NULL,
         now()::timestamp + INTERVAL '2 hours'),
        (get_person_id_by_login('client3'),
         get_person_id_by_login('manager2'),
         get_person_id_by_login('worker2'),
         3,
         get_task_type_id('Установка оборудования'),
         'низкий',
         'Установить обновление ПО',
         now()::timestamp,
         NULL,
         now()::timestamp + INTERVAL '1 day'),
        (get_person_id_by_login('client4'),
         get_person_id_by_login('manager1'),
         get_person_id_by_login('worker1'),
         NULL,
```

```
        get_task_type_id('Встреча с клиентом'),
        'средний',
        'Обсудить поставку микрочипов',
        now()::timestamp,
        NULL,
        now()::timestamp + INTERVAL '30 minutes');
```

2.3 Листинг скрипта для создания ролей, назначения привилегий и создания политик

```
DROP ROLE IF EXISTS manager;
DROP ROLE IF EXISTS worker;
DROP ROLE IF EXISTS administrator;

CREATE ROLE manager;
CREATE ROLE worker;
CREATE ROLE administrator WITH LOGIN ENCRYPTED PASSWORD 'dungeon_master69';

GRANT ALL ON ALL TABLES IN SCHEMA public TO administrator;
GRANT ALL ON ALL FUNCTIONS IN SCHEMA public TO administrator;
GRANT ALL ON ALL PROCEDURES IN SCHEMA public TO administrator;

ALTER TABLE tasks ENABLE ROW LEVEL SECURITY;

GRANT SELECT ON contracts, tasks, contact_info, task_type, person,
    organization, user_login, person_role, roles TO manager, worker;

GRANT INSERT ON tasks, contact_info, contracts, person, organization
    TO manager;

GRANT UPDATE ON tasks, contact_info, contracts, person, organization
    TO manager;

GRANT UPDATE(dt_finished, priority) ON tasks TO worker;

CREATE POLICY select_tasks ON tasks FOR SELECT TO manager, worker
USING
(
    (SELECT username FROM user_login WHERE user_login.id = tasks.author_id) =
current_user
        OR
    (SELECT username FROM user_login WHERE user_login.id = tasks.executor_id)
= current_user
);

CREATE POLICY select_for_managers ON tasks FOR SELECT TO manager
USING
(
    (SELECT title FROM roles WHERE roles.id = (SELECT role_id FROM person_role
WHERE person_id = tasks.author_id)) = 'Сотрудник'
);

CREATE POLICY access_all ON tasks FOR ALL TO administrator
USING (true) WITH CHECK (true);

CREATE POLICY insert_task ON tasks FOR INSERT TO manager
WITH CHECK (true);

CREATE POLICY update_task ON tasks FOR UPDATE TO manager
USING (true) WITH CHECK(tasks.dt_finished IS NULL AND ((SELECT username FROM
user_login WHERE user_login.id = author_id) = current_user));

CREATE POLICY update_state ON tasks FOR UPDATE TO manager, worker
```

```
USING (dt_finished IS NULL) WITH CHECK((dt_finished IS NOT NULL)
    AND ((SELECT username FROM user_login WHERE user_login.id = author_id) =
current_user
    OR (SELECT username FROM user_login WHERE user_login.id = executor_id) =
current_user)
    );
```

3. Автоматизация функционала

3.1 Листинг скрипта для автоматизации функционала

```
CREATE FUNCTION check_task_completed()
    RETURNS TRIGGER
    AS
    $update_task_trigger$
    BEGIN
        IF (OLD.dt_finished IS NOT NULL) THEN
        RAISE EXCEPTION 'Задание уже выполнено';
        END IF;
        RETURN NEW;
    END;
    $update_task_trigger$ LANGUAGE plpgsql;
CREATE TRIGGER update_task_trigger
    BEFORE UPDATE ON tasks
    FOR EACH ROW
        EXECUTE PROCEDURE check_task_completed();

CREATE EXTENSION pg_cron;
SELECT cron.schedule('0 0 * * *',
    $$DELETE FROM tasks WHERE dt_finished < (now() - interval '12 month'$$);
```

4. Реализация индексов, работа с выгрузкой данных

4.1 Листинг скрипта для создания индексов

```
CREATE INDEX person_name_index ON person(first_name, last_name);
CREATE INDEX person_login_index ON user_login(username);
CREATE INDEX person_contact_index ON contact_info(contact);

CREATE INDEX organization_name_index ON organization(name);
CREATE INDEX organization_delegate_index ON organization(person_id);
```

4.2 Листинг скрипта для выгрузки данных

```
CREATE OR REPLACE FUNCTION create_report(worker INTEGER, start_date DATE,
end_date DATE)
RETURNS TABLE (total_count INTEGER, finished_in_time INTEGER, finished_late
INTEGER, not_finished INTEGER, not_finished_overdue INTEGER)
AS $body$
DECLARE
    total_count INTEGER;
    finished_in_time INTEGER;
    finished_late INTEGER;
    not_finished INTEGER;
    not_finished_overdue INTEGER;
BEGIN
    start_date = start_date::TIMESTAMP;
    end_date = (end_date + INTERVAL '1 day' - INTERVAL '1
microsecond')::TIMESTAMP;
    total_count := (SELECT count(*) FROM tasks -- Всего задач созданных в
данный период
```

```
                                    WHERE (executor_id = worker)
                                      AND (dt_created BETWEEN start_date AND
end_date));
    finished_in_time := (SELECT count(*) FROM tasks -- Задач, завершенных
вовремя
                                    WHERE (executor_id = worker)
                                      AND (dt_created BETWEEN start_date
AND end_date)
                                      AND (dt_finished BETWEEN start_date
AND end_date)
                                      AND (dt_deadline >= dt_finished));
    finished_late := (SELECT count(*) FROM tasks -- Задач, завершенных с
опозданием
                                    WHERE (executor_id = worker)
                                      AND (dt_created BETWEEN start_date
AND end_date)
                                      AND (dt_finished BETWEEN start_date
AND end_date)
                                      AND (dt_deadline <= dt_finished));
    not_finished := (SELECT count(*) FROM tasks -- Задач, еще не завершенных
                                    WHERE (executor_id = worker)
                                      AND (dt_created BETWEEN start_date
AND end_date)
                                      AND (dt_finished IS NULL)
                                      AND (dt_deadline >= end_date));
    not_finished_overdue := (SELECT count(*) FROM tasks -- Задач, еще не
завершенных, уже с опозданием
                                    WHERE (executor_id = worker)
                                      AND (dt_created BETWEEN start_date
AND end_date)
                                      AND (dt_finished IS NULL)
                                      AND (dt_deadline <= end_date));
    RETURN QUERY SELECT total_count, finished_in_time, finished_late,
not_finished, not_finished_overdue;
END $body$
LANGUAGE plpgsql;
```

4.2 Сохранение отчета в CSV

```
\copy (SELECT * FROM create_report(_person_id_, _start_date_, _end_date_)) TO
'_path_' DELIMITER ',' CSV HEADER;
```

```
psql -h localhost -p 5432 -U postgres mirea
psql (14.5)
Введите "help", чтобы получить справку.

mirea=# \copy (select * from create_report(get_person_id_by_login('worker1'), (current_date - INTERVAL '23 days')::DATE, (current_date)::DATE)) TO '/home/plmr0/kek.csv' DELIMITER '
,' CSV HEADER;
COPY 1
mirea=#
```

A	B	C	D	E
total_count	finished_in_time	finished_late	not_finished	not_finished_overdue
2	0	0	1	1