

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

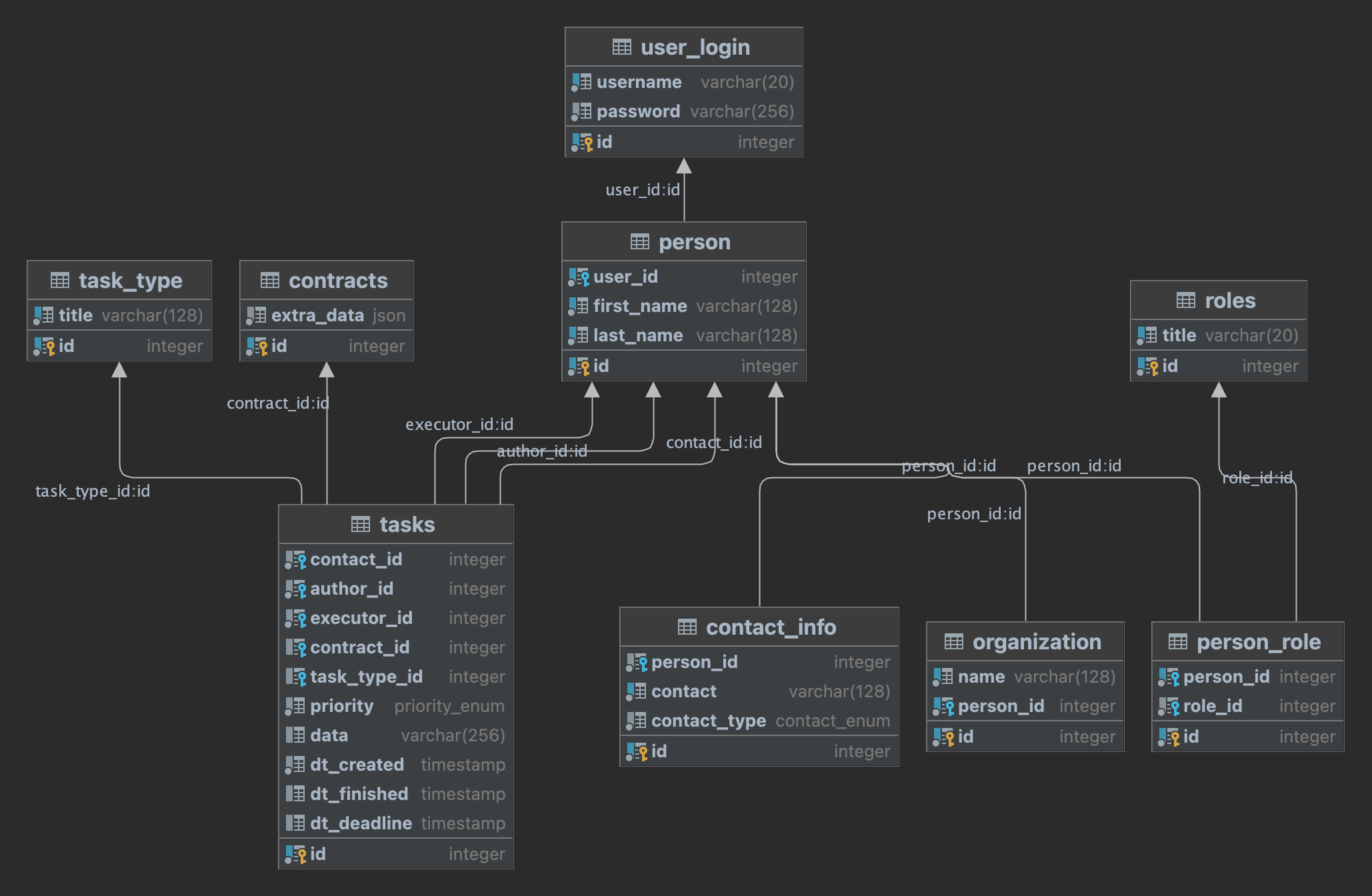


Рис. 1 – ER-диаграмма.

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;

