

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

****

Институт комплексной безопасности и специального приборостроения

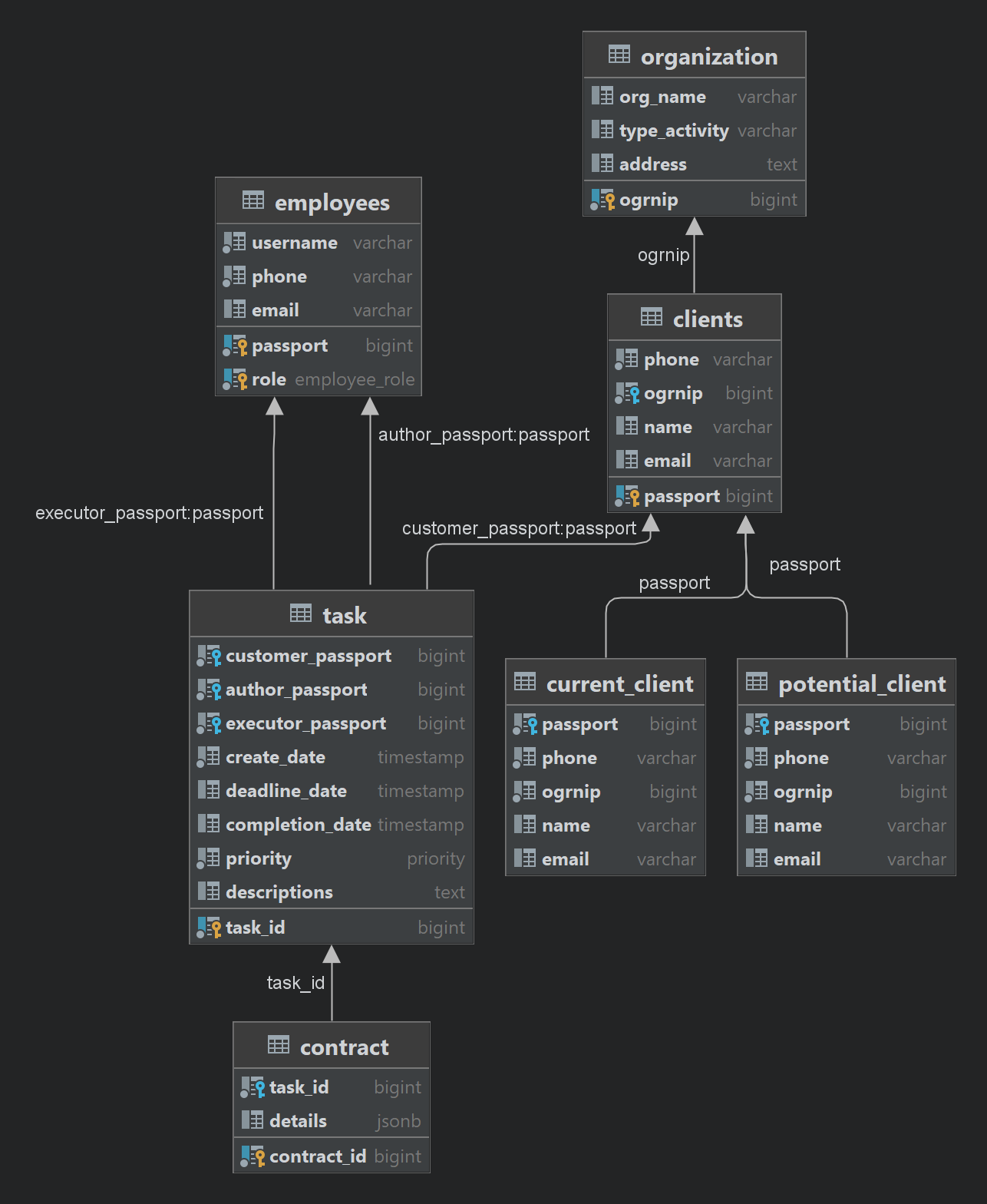
Кафедра КБ-14 «Интеллектуальные системы информационной безопасности»

**Клиент-серверные системы управления банком данных**

**Практическая работа 2-5.**

**Управление контактами с клиентами**

1. **Практическая работа №2-5**
   1. **Работа с ER-диаграммой и структурой базы данных**



**Скрипт schema.sql:**

CREATE SCHEMA IF NOT EXISTS company;  
  
CREATE TABLE IF NOT EXISTS company.organization(  
 ogrnip bigserial primary key UNIQUE NOT NULL,  
 org\_name varchar,  
 type\_activity varchar,  
 address text  
);  
  
CREATE TABLE IF NOT EXISTS company.clients  
(  
 passport bigserial primary key UNIQUE NOT NULL,  
 phone varchar NOT NULL,  
 ogrnip bigserial references company.organization(ogrnip) NOT NULL,  
 "name" varchar,  
 email varchar  
);  
  
CREATE TABLE IF NOT EXISTS company.employees (  
 passport bigserial UNIQUE NOT NULL,  
 username varchar NOT NULL,  
 "role" employee\_role NOT NULL,  
 phone varchar NOT NULL,  
 email varchar,  
 PRIMARY KEY (passport, "role")  
);  
  
CREATE TABLE IF NOT EXISTS company.task (  
 task\_id bigserial primary key UNIQUE NOT NULL,  
 customer\_passport bigserial references company.clients(passport) NOT NULL,  
 author\_passport bigserial references company.employees(passport) NOT NULL,  
 executor\_passport bigserial references company.employees(passport) NOT NULL,  
 create\_date timestamp without time zone NOT NULL,  
 deadline\_date timestamp without time zone,  
 completion\_date timestamp without time zone,  
 priority priority NOT NULL,  
 descriptions text  
);  
  
CREATE TABLE IF NOT EXISTS company.contract (  
 contract\_id bigserial primary key UNIQUE NOT NULL,  
 task\_id bigserial references company.task(task\_id) NOT NULL,  
 details jsonb  
);  
  
CREATE TABLE IF NOT EXISTS company.current\_client(  
 passport bigserial references company.clients(passport) NOT NULL  
) INHERITS(company.clients);  
  
CREATE TABLE IF NOT EXISTS company.potential\_client (  
 passport bigserial references company.clients(passport) NOT NULL  
) INHERITS(company.clients);

**Скрипт types.sql:**

DO $$  
BEGIN  
 IF NOT *EXISTS* (SELECT 1 FROM pg\_type WHERE typname = 'employee\_role') THEN  
 CREATE TYPE employee\_role as ENUM ('manager', 'ranker');  
 END IF;  
 IF NOT *EXISTS* (SELECT 1 FROM pg\_type WHERE typname = 'priority') THEN  
 CREATE TYPE priority as ENUM ('low', 'medium', 'high');  
 END IF;  
END$$;

* 1. **Заполнение базы данных, создание ролей, назначение привилегий**
     1. **Заполнение базы данных**

**Скрипт insert.sql:**

INSERT INTO company.organization  
VALUES (4565465468789, 'GroupIB', 'Информационная безопасность', 'г.Москва, чистые пруды 1'),  
 (1232487855544, 'Kaspersky', 'ИнфоБез', 'г.Москва, ленинградское щоссе 1'),  
 (4565712855898, 'Yandex', 'Разработка ИС', 'г.Москва, лубянка 20');  
  
INSERT INTO company.clients  
VALUES (1234567899, '+7456789451', 4565465468789, 'Иван Иванович', 'qwe@gmail.com'),  
 (9874561233, '+7445289452', 4565465468789, 'Сергей Петрович', 'ert@gmail.com'),  
 (4567891322, '+7456799453', 4565465468789, 'Алексей Николаевич', 'yui@gmail.com'),  
  
 (8988798455, '+7456789451', 1232487855544, 'Григорий Григорьевич', 'ryy@gmail.com'),  
 (7897546455, '+7445289452', 1232487855544, 'Александр Александрович', 'lkl@gmail.com'),  
 (6786431323, '+7891235458', 1232487855544, 'Федор Федорович', 'zxc@gmail.com'),  
  
 (6548787454, '+1235487899', 4565712855898, 'Егор Григорьевич', 'sdfskjl@gmail.com'),  
 (1387861654, '+8779545457', 4565712855898, 'Ярослав Александрович', 'cvsdf@gmail.com'),  
 (9879889846, '+1354578455', 4565712855898, 'Святослав Федорович', 'qwasdw@gmail.com');  
  
  
INSERT INTO company.employees  
VALUES (1111111111, 'Bob', 'manager'::employee\_role, '+7456789451', 'Bob@gmail.com'),  
 (2222222222, 'Tom', 'manager'::employee\_role, '+7165464575', 'Tom@gmail.com'),  
  
 (3333333333, 'Alice', 'ranker'::employee\_role, '+7456789457', 'Alice@gmail.com'),  
 (4444444444, 'Jack', 'ranker'::employee\_role, '+5464484615', 'Jack@gmail.com'),  
 (5555555555, 'Alex', 'ranker'::employee\_role, '+7167846512', 'Alex@gmail.com');  
  
INSERT INTO company.employees  
VALUES (6666666666, 'postgres', 'manager'::employee\_role, '+7456789451', 'postgres@gmail.com');  
  
INSERT INTO company.task  
VALUES (0, 1234567899, 3333333333, 1111111111, '2022-09-02 04:05:06'::timestamp,  
 '2022-09-18 04:05:06'::timestamp, null, 'low'::priority, 'телефонный звонок'),  
 (1, 9874561233, 4444444444, 2222222222, '2022-09-08 04:05:06'::timestamp,  
 '2022-09-25 04:05:06'::timestamp, null, 'high'::priority, 'визит');

INSERT INTO company.current\_client   
 (SELECT *\** FROM company.clients as ccl  
 WHERE (SELECT ct.customer\_passport   
 FROM company.task as ct   
 WHERE customer\_passport =   
 ccl.**passport** and   
 deadline\_date > *LOCALTIMESTAMP* or  
 deadline\_date is null) = ccl.passport);  
  
INSERT INTO company.potential\_client   
 (SELECT *\** FROM company.clients as ccl  
 WHERE (SELECT ct.customer\_passport   
 FROM company.task as ct  
 WHERE customer\_passport = ccl.**passport** and   
 deadline\_date < *LOCALTIMESTAMP* or   
 completion\_date < *LOCALTIMESTAMP*) = ccl.passport);

**Скрипт functions.sql для polices.sql:**

CREATE OR REPLACE FUNCTION *get\_passport\_byCurrentUser*() RETURNS bigint AS  
$BODY$  
BEGIN  
 RETURN (SELECT passport FROM company.employees WHERE passport = *current\_user*);  
END  
$BODY$  
LANGUAGE plpgsql;  
  
CREATE OR REPLACE FUNCTION *get\_manager\_passport\_byCurrentUser*() RETURNS bigint AS  
$BODY$  
BEGIN  
 RETURN (SELECT passport FROM company.employees WHERE passport = *current\_user* and role = 'manager');  
END  
$BODY$  
LANGUAGE plpgsql;  
  
CREATE OR REPLACE FUNCTION *get\_ranker\_passport\_byCurrentUser*() RETURNS bigint AS  
$BODY$  
BEGIN  
 RETURN (SELECT passport FROM company.employees WHERE passport = *current\_user* and role = 'ranker');  
END  
$BODY$  
LANGUAGE plpgsql;  
  
CREATE OR REPLACE FUNCTION *is\_not\_author*(input\_task\_id bigint) RETURNS boolean AS  
$BODY$  
BEGIN  
 IF (SELECT *count*(*\**) FROM company.task) > 0 THEN  
 IF (SELECT 1 FROM company.task as ct1  
 WHERE (SELECT executor\_passport  
 FROM company.task as ct2  
 WHERE ct2.task\_id = input\_task\_id) = ct1.author\_passport) is true THEN  
 RETURN 1;  
 ELSE  
 RETURN 0;  
 END IF;  
 END IF;  
END;  
$BODY$  
LANGUAGE plpgsql;

**Скрипт polices.sql:**

*--Админ имеет доступ к специальным функциям, например, может изменить автора задания или внести изменения в завершенное задание.*GRANT ALL PRIVILEGES ON SCHEMA company to admin;  
  
DO $$  
BEGIN  
 IF (*current\_user* = 'admin') THEN  
 IF NOT *EXISTS* (SELECT 1 FROM pg\_roles WHERE rolname = 'manager') THEN  
 CREATE ROLE manager NOLOGIN NOSUPERUSER;  
 END IF;  
 IF NOT *EXISTS* (SELECT 1 FROM pg\_roles WHERE rolname = 'ranker') THEN  
 CREATE ROLE ranker NOLOGIN NOSUPERUSER;  
 END IF;  
 END IF;  
END$$;  
  
DO $$  
BEGIN  
 IF (*current\_user* = 'admin') THEN  
 IF NOT *EXISTS* (SELECT 1 FROM pg\_group WHERE groname = 'staff') THEN  
 CREATE GROUP staff;  
 CREATE USER Bob WITH LOGIN ROLE manager IN GROUP staff;  
 CREATE USER Tom WITH LOGIN ROLE manager IN GROUP staff;  
 CREATE USER Alice WITH LOGIN ROLE manager IN GROUP staff;  
 CREATE USER Jack WITH LOGIN ROLE ranker IN GROUP staff;  
 CREATE USER Alex WITH LOGIN ROLE ranker IN GROUP staff;  
 END IF;  
 END IF;  
END$$;  
  
*--Менеджеры назначают задания себе или кому-либо из рядовых сотрудников*GRANT SELECT, INSERT, UPDATE ON TABLE company.task to manager;  
  
 *--Рядовые сотрудники не могут назначать задания*GRANT SELECT ON TABLE company.task to ranker;  
  
*--Менеджеры назначают задания себе или кому-либо из рядовых сотрудников  
--Исполнителем задания может быть сотрудник, не являющийся автором: is\_not\_author(input\_task\_id)*CREATE POLICY manager\_update\_tasks ON company.task AS PERMISSIVE FOR UPDATE TO manager USING  
 (executor\_passport = *get\_manager\_passport\_bycurrentuser*()  
 or author\_passport = *get\_manager\_passport\_bycurrentuser*())  
 with check (completion\_date is null and *is\_not\_author*(task\_id) is true);  
  
*--Помечать задание как выполненное и указывать дату завершения может ... исполнитель задания.*CREATE POLICY ranker\_update\_tasks ON company.task AS PERMISSIVE FOR UPDATE TO ranker USING  
 (executor\_passport = *get\_ranker\_passport\_bycurrentuser*()) with check (completion\_date is null);

*--Просматривать задание, автором которого является менеджер, может либо автор, либо исполнитель задания*CREATE POLICY manager\_select\_tasks ON company.task AS PERMISSIVE FOR SELECT TO manager USING  
 (executor\_passport = *get\_manager\_passport\_bycurrentuser*()  
 or author\_passport = *get\_manager\_passport\_bycurrentuser*());  
  
CREATE POLICY ranker\_select\_tasks ON company.task AS PERMISSIVE FOR SELECT TO ranker USING  
 (executor\_passport = *get\_ranker\_passport\_bycurrentuser*());

* 1. **Автоматизация функционала**
  2. **Реализация индексов, работа с выгрузкой данных**