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

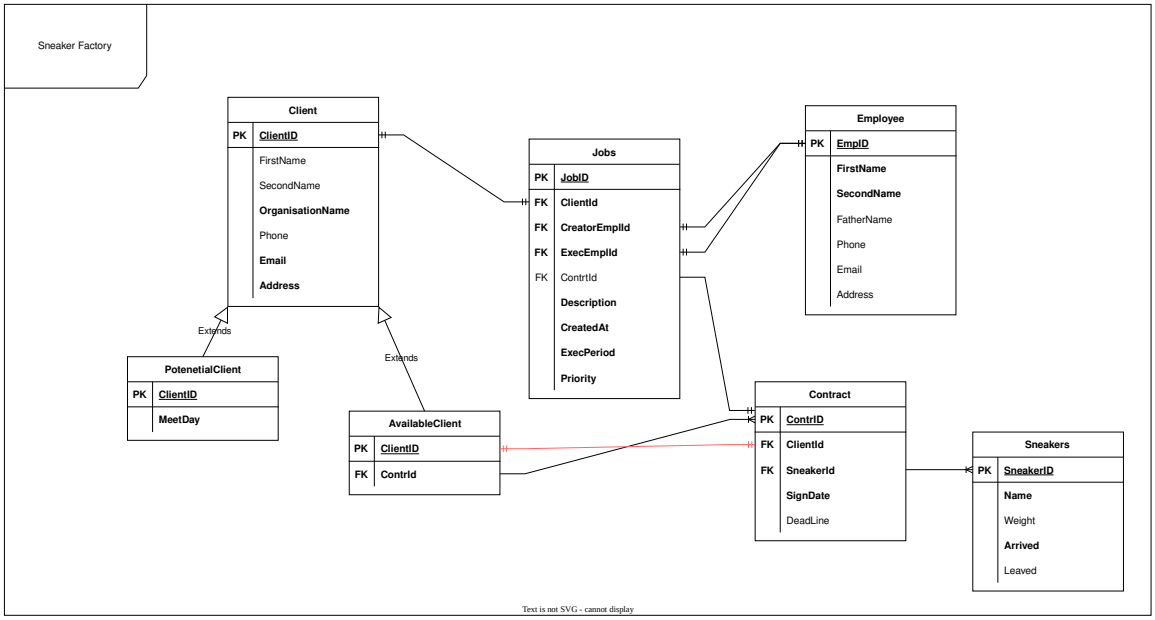
Практическая работа 2-5.

Управление контактами с клиентами

1. Практическая работа №2

1.1 Работа с ER-диаграммой и структурой базы данных

По условию задания требуется создать свою базу данных, используя систему управления базами данных «PostgreSQL». В качестве условного заказчика был взят завод по производству кроссовок(«Sneaker Factory»). В связи с тем, что кроссовки, как объект, имеют свои физические особенности таблица «Sneakers» имеет дополнительные атрибуты присущие кроссовкам в реальном мире.



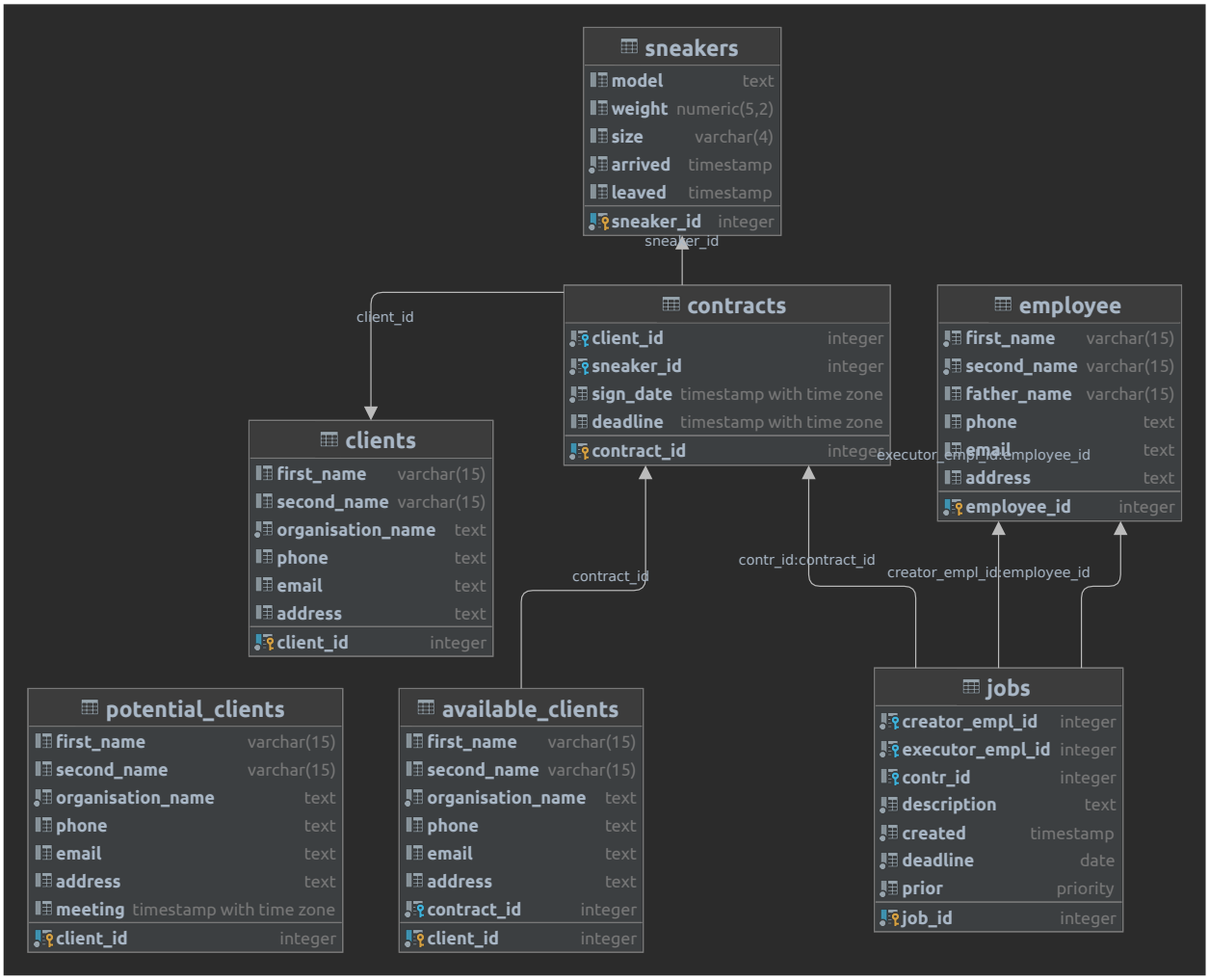
Фигура 1: Рис. 1.1 UML набросок-диаграмма, созданная в draw.io

База данных, а также таблицы с колонками, основными и внешними ключами, ограничениями и т.д. создаются при помощи таких команд:

Листинг 1.1 Создание базы данных

```
CREATE DATABASE Sneaker_Factory;
CREATE TABLE Clients(
    client_id INTEGER PRIMARY KEY,
    first_name VARCHAR(15),
    second_name VARCHAR(15),
    organisation_name TEXT NOT NULL,
    phone TEXT,
    email TEXT,
    address TEXT
);
```

```sql
-- Создание 2ух групп пользователей, наследуемых от Клиента
CREATE TABLE Potential_Clients(
    meeting TIMESTAMPTZ
) INHERITS (Clients);
CREATE TABLE Available_Clients() INHERITS (Clients);

CREATE TABLE Sneakers(
    sneaker_id INTEGER PRIMARY KEY,
    model text,
    weight numeric(5, 2),
    size varchar(4),
    arrived timestamp DEFAULT current_timestamp NOT NULL,
    leaved timestamp
);

CREATE TABLE Contracts(
    contract_id integer PRIMARY KEY,
    client_id integer REFERENCES clients(client_id) NOT NULL,
    sneaker_id integer REFERENCES sneakers(sneaker_id) NOT NULL,
    sign_date timestamptz NOT NULL,
    deadline timestamptz
);

ALTER TABLE Available_Clients ADD COLUMN contract_id INTEGER NOT NULL;
ALTER TABLE Available_Clients ADD CONSTRAINT contract_ref_constr FOREIGN KEY
(contract_id) references Contracts (contract_id);

CREATE TABLE Employee(
    employee_id integer PRIMARY KEY,
    first_name VARCHAR(15) NOT NULL,
    second_name VARCHAR(15) NOT NULL,
    father_name VARCHAR(15),
    address TEXT
);

CREATE TYPE priority AS ENUM ('low','medium','high');

CREATE TABLE Jobs(
    job_id INTEGER PRIMARY KEY,
    creator_empl_id INTEGER REFERENCES Employee(employee_id) NOT NULL,
    executor_empl_id INTEGER REFERENCES Employee(employee_id) NOT NULL,
    contr_id INTEGER REFERENCES Contracts(contract_id),
    description text NOT NULL,
    created timestamp default current_timestamp NOT NULL,
    deadline date NOT NULL,
    prior priority NOT NULL);
```



Фигура 2: Рис 1.2 созданная DataGrip ER-диаграмма полученной базы данных

Проблема деления клиентов на 2 группы была решена при помощи наследования, имеющегося в PostgreSQL. Для определения приоритета задания была создана специальная структура данных основанная на перечислении («Enum») с такими значениями, как «Low» - «Низкий», «Medium» - «Средний», «High» - «Высокий» для последующего выполнения задания связанного с определением приоритета задачи.

2. Практическая работа №2

2.1 Заполнение базы данных, создание ролей, назначение привилегий

Для последующего тестирования работоспособности созданной базы данных требуется заполнить её потенциальными данными. В ходе выполнения работы было принято решение добавить всем первичным ключам ограничение «Generated as identity», что является современной аналогии псевдотипу «Serial», с целью добавления автоинкрементации и правильной индексации полей при помощи команды:

Листинг 2.1. Ограничение первичных ключей:

```
ALTER TABLE название_таблицы
    ALTER COLUMN название_столбца_id ADD GENERATED ALWAYS AS IDENTITY;
```

В качестве минимального требуемого количества строк было принято решение выбрать 3 строки, заполненные информации в каждой из таблиц. Для этого потребуется выполнить такие команды:

Листинг 2.2. Заполнение всех таблиц:

```
--Sneakers
INSERT INTO sneakers(model, weight, size, arrived, leaved)
    VALUES ('Ортопедические', 200.00, '42', DEFAULT, null);
INSERT INTO sneakers(model, weight, size, arrived, leaved)
    VALUES ('Ultra Force X4', 269.52, '41', now(), null);
INSERT INTO sneakers(model, weight, size, arrived, leaved)
    VALUES ('Ортопедические', 300, '39', '2022-05-20 15:43', null);

--Employee
INSERT INTO employee(first_name, second_name, father_name, phone, email,
address)
    VALUES ('Дмитрий', 'Лакутин', 'Алексеевич', '+79775179280',
'haydurb@mail.com', 'Московская обл. г.Ногинск ул.Пушкина д.15 кв.123');
INSERT INTO employee(first_name, second_name, father_name, phone, email,
address)
    VALUES ('Sharil', 'Lebowski', null, '+1 720-409-2532',
'wheresthemoney@gmail.com', 'c. USA st. Texas s. 1638 Brentwood Drive ');
INSERT INTO employee(first_name, second_name, father_name, phone, email,
address)
    VALUES ('Олег', 'Харьков', 'Владимирович', '89265334529', null, null);

--Potential Clients
INSERT INTO potential_clients(first_name,
                              second_name,
```

```
                                organisation_name,
                                phone,
                                email,
                                address,
                                meeting)
    VALUES ('Сергей',
            'Комаров',
            'ИП Комаровский бутик',
            '8(969)2012298',
            'komarov@yandex.ru',
            'Нижний Новгород Тюхачевского д.56',
            '2023-03-15 10:00');
INSERT INTO potential_clients(first_name,
                                second_name,
                                organisation_name,
                                phone,
                                email,
                                address,
                                meeting)
    VALUES (null,
            null,
            'Shabudabu Shop',
            '3(8585)775-27-01 ',
            'shabudabu0shop@gmail.com',
            '4786 Lyric Causeway Suite 291 West Irma, NM 84308-2385',
            '2027-06-29 02:30 +9:30');
INSERT INTO potential_clients(first_name,
                                second_name,
                                organisation_name,
                                phone,
                                email,
                                address,
                                meeting)
    VALUES (null,
            null,
            'SneakerTops',
            null,
            null,
            null,
            null);

--Available Clients
INSERT INTO available_clients(first_name,
                                second_name,
                                organisation_name,
                                phone,
                                email,
                                address,
                                contract_id)
    VALUES ('Nickel',
            'Willson',
```

```
            'AdiPro',
            '+1(800)-1453-27-25',
            'adiProContact@yahoo.com',
            'Armin-Schumacher-Allee 65c 2125 Horgen',
            null);
INSERT INTO available_clients(first_name,
                              second_name,
                              organisation_name,
                              phone,
                              email,
                              address,
                              contract_id)
    VALUES ('Евгений',
            null,
            'ОАО "Свій Шоп"',
            null,
            'evgenyContShop@ask.com',
            '23398, Київська область, місто Київ, пл. Б. Грінченка, 37 ',
            null);
INSERT INTO available_clients(first_name,
                              second_name,
                              organisation_name,
                              phone,
                              email,
                              address,
                              contract_id)
    VALUES (null,
            null,
            'Retro Sneaker',
            null,
            '9.125.57.184',
            '606700, Свердловская область, город Коломна, бульвар Косиора,
76',
            null);

--Contracts
INSERT INTO contracts(client_id, sneaker_id, sign_date, deadline)
    VALUES ((SELECT client_id FROM available_clients WHERE organisation_name =
'ОАО "Свій Шоп"'),
            (SELECT sneaker_id FROM sneakers WHERE model = 'Ultra Force X4'),
            NOW(),
            NOW() + INTERVAL '30' DAY );
INSERT INTO contracts(client_id, sneaker_id, sign_date, deadline)
    VALUES ((SELECT client_id FROM available_clients WHERE organisation_name =
'AdiPro'),
            (SELECT sneaker_id FROM sneakers WHERE model = 'Ортопедические'
AND size = '39'),
            NOW() - INTERVAL '42' DAY ,
            NOW() + INTERVAL '73' DAY );
INSERT INTO contracts(client_id, sneaker_id, sign_date, deadline)
```

```
    VALUES ((SELECT client_id FROM available_clients WHERE organisation_name =
'Retro Sneaker'),
            (SELECT sneaker_id FROM sneakers WHERE model = 'Ultra Force X4'),
            NOW(),
            null);

--Возвращение к Available_Clients с целью добавления контрактов
UPDATE available_clients SET contract_id = 1 WHERE client_id = (SELECT
client_id FROM contracts WHERE contracts.contract_id = 1);
UPDATE available_clients SET contract_id = 2 WHERE client_id = (SELECT
client_id FROM contracts WHERE contracts.contract_id = 2);
UPDATE available_clients SET contract_id = 3 WHERE client_id = (SELECT
client_id FROM contracts WHERE contracts.contract_id = 3);

--Jobs
INSERT INTO jobs(creator_empl_id, executor_empl_id, contr_id, description,
deadline, prior)
    VALUES ((SELECT employee_id FROM employee WHERE employee_id = 1),
            (SELECT employee_id FROM employee WHERE employee_id = 2),
            (SELECT contract_id FROM contracts WHERE contract_id = 2),
            'Send email to ' || (SELECT email
                                 FROM available_clients
                                 WHERE client_id = (SELECT contracts.client_id
                                                    FROM contracts
                                                    WHERE contracts.contract_id
= 2)),
            NOW() + INTERVAL '3' HOUR,
            'high');
INSERT INTO jobs(creator_empl_id, executor_empl_id, contr_id, description,
deadline, prior)
    VALUES ((SELECT employee_id FROM employee WHERE employee_id = 3),
            (SELECT employee_id FROM employee WHERE employee_id = 3),
            NULL,
            'Create new tasks for ' || (SELECT email FROM employee WHERE
employee_id = 2),
            NOW() + INTERVAL '4' HOUR,
            'medium');
INSERT INTO jobs(creator_empl_id, executor_empl_id, contr_id, description,
deadline, prior)
    VALUES ((SELECT employee_id FROM employee WHERE employee_id = 1),
            (SELECT employee_id FROM employee WHERE employee_id = 3),
            (SELECT contract_id FROM contracts WHERE contract_id = 1),
            'Предложи ' || (SELECT email
                      FROM available_clients
                      WHERE client_id = (SELECT contracts.client_id
                                         FROM contracts
                                         WHERE contracts.contract_id = 1)) ||
' новый контракт.',
            NOW() + INTERVAL '9' HOUR,
            'low');
```

Для того, чтобы разграничить права доступа и изменения данных потребовалось создать роли (роль представляет собой сущность, которая может владеть объектами базы данных, схожа с понятием «пользователь»), при помощи:

Листинг 2.3 Роль:

```
CREATE ROLE name [ [ WITH ] option [ ... ] ]
where option can be:
      SUPERUSER | NOSUPERUSER
    | CREATEDB | NOCREATEDB
    | CREATEROLE | NOCREATEROLE
    | INHERIT | NOINHERIT
    | LOGIN | NOLOGIN
    | REPLICATION | NOREPLICATION
    | BYPASSRLS | NOBYPASSRLS
    | CONNECTION LIMIT connlimit
    | [ ENCRYPTED ] PASSWORD 'password' | PASSWORD NULL
    | VALID UNTIL 'timestamp'
    | IN ROLE role_name [, ...]
    | IN GROUP role_name [, ...]
    | ROLE role_name [, ...]
    | ADMIN role_name [, ...]
    | USER role_name [, ...]
    | SYSID uid
```

Выполнение задачи просмотра, изменения и создания определенных данных только выделенному кругу ролей решена при помощи добавления в таблицу «Employee» новой колонки «factory role», которая представляет собой роль MANid(менеджер) и EMPid(рядовой сотрудник). Для создание точных доступов к определенной таблице, используются правила «policy». Для того, чтобы ими воспользоваться нужно сначала разрешить работу на уровне строк, при помощи команды, а далее логика команды заключается в проверке того, что пользователь СУБД содержится в таблице «Employee» и является исполнителем (или автором) задания, таким образом команды, что для этого потребовалось выполнить:

Листинг 2.4 Роли и привелегии:

```
CREATE ROLE factory_admin NOSUPERUSER CREATEDB NOINHERIT LOGIN PASSWORD 'root'CONNECTION LIMIT 1;
CREATE GROUP factory_manager NOSUPERUSER NOCREATEDB NOCREATEROLE NOINHERIT
NOLOGIN;
CREATE GROUP employee NOSUPERUSER NOCREATEDB NOCREATEROLE NOINHERIT NOLOGIN ;
```

```
GRANT ALL ON ALL TABLES IN SCHEMA public TO factory_admin;
GRANT SELECT, INSERT, UPDATE ON jobs TO factory_manager;
GRANT SELECT, UPDATE ON jobs TO employee;

ALTER TABLE jobs ENABLE ROW LEVEL SECURITY;

CREATE POLICY empl_select_p ON jobs FOR SELECT TO employee USING (
     (SELECT employee_id
      FROM employee
      WHERE factory_role = current_user
      LIMIT 1) = jobs.executor_empl_id
    );
CREATE POLICY empl_update_p ON jobs FOR UPDATE TO employee USING (
     (SELECT employee_id
      FROM employee
      WHERE factory_role = current_user
      LIMIT 1) = jobs.executor_empl_id
    );

CREATE POLICY man_update_p ON jobs FOR UPDATE TO factory_manager USING(
         (SELECT employee_id
      FROM employee
      WHERE factory_role = current_user
      LIMIT 1) = jobs.creator_empl_id
             OR  (SELECT employee_id
      FROM employee
      WHERE factory_role = current_user
      LIMIT 1) = jobs.executor_empl_id
    );

CREATE ROLE MAN1;
CREATE ROLE EMP1;
CREATE ROLE EMP2;
GRANT factory_manager TO MAN1;
GRANT employee TO EMP1,EMP2;
```

Для того, чтобы по происшествии 12 месяцев после даты завершения задания сведения о нём удалялись из системы, требуется создать триггер, который будет срабатывать когда настоящее время будет равно значению в колонке deadline + 1 год. Триггер создаётся при помощи команды:

Листинг 2.5 Создание триггера:

REATE [**CONSTRAINT**] **TRIGGER** имя { **BEFORE** | **AFTER** | **INSTEAD OF** } { событие [**OR** ...] }

```
    ON имя_таблицы
    [ FROM ссылающаяся_таблица ]
```

```
    [ NOT DEFERRABLE | [ DEFERRABLE ] [ INITIALLY IMMEDIATE | INITIALLY
DEFERRED ] ]
    [ REFERENCING { { OLD | NEW } TABLE [ AS ] имя_переходного_отношения }
[ ... ] ]
    [ FOR [ EACH ] { ROW | STATEMENT } ]
    [ WHEN ( условие ) ]
    EXECUTE { FUNCTION | PROCEDURE } имя_функции ( аргументы )

Здесь допускается событие:

    INSERT
    UPDATE [ OF имя_столбца [, ... ] ]
    DELETE
    TRUNCATE
```

Где требуется функция, которая возвращает TRIGGER. Команда создания функции выглядит так:

Листинг 2.6. Создание функции:

```
CREATE [ OR REPLACE ] FUNCTION

    имя ( [ [ режим_аргумента ] [ имя_аргумента ] тип_аргумента [ { DEFAULT |
= } выражение_по_умолчанию ] [, ...] ] )
    [ RETURNS тип_результата
      | RETURNS TABLE ( имя_столбца тип_столбца [, ...] ) ]
  { LANGUAGE имя_языка
    | TRANSFORM { FOR TYPE имя_типа } [, ... ]
    | WINDOW
    | { IMMUTABLE | STABLE | VOLATILE }
    | [ NOT ] LEAKPROOF
    | { CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT }
    | { [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER }
    | PARALLEL { UNSAFE | RESTRICTED | SAFE }
    | COST стоимость_выполнения
    | ROWS строк_в_результате
    | SUPPORT вспомогательная_функция
    | SET параметр_конфигурации { TO значение | = значение | FROM CURRENT }
    | AS 'определение'
    | AS 'объектный_файл', 'объектный_символ'
  } ...
```

Таким образом конечный набор команд для создания триггера:

Листинг 2.7. Создание триггера для таблицы Jobs:

```
CREATE TRIGGER one_year_ttl AFTER INSERT OR UPDATE
```

```
    ON jobs FOR EACH STATEMENT
    EXECUTE FUNCTION remove_all_jobs_after_year();

CREATE FUNCTION remove_all_jobs_after_year() RETURNS TRIGGER AS $$
    BEGIN
        DELETE FROM jobs WHERE current_timestamp >= jobs.deadline +
make_interval(years := 1);
    END;
    $$ LANGUAGE plpgsql;
```

Для решения задачи поиска в базе клиентов и контактных лиц по их атрбиутам можно воспользоваться «Индексами», для их создания потребуется выполнить команду:

Листинг 2.8 команда по созданию индексов:

```
      CREATE [ UNIQUE ] INDEX [ CONCURRENTLY ] [ [ IF NOT EXISTS ] имя ] ON [
ONLY ] имя_таблицы [ USING метод ]

    ( { имя_столбца | ( выражение ) } [ COLLATE правило_сортировки ] [
класс_операторов [ ( параметр_класса_оп = значение [, ... ] ) ] ] [ ASC | DESC
] [ NULLS { FIRST | LAST } ] [, ...] )
    [ INCLUDE ( имя_столбца [, ...] ) ]
    [ WITH ( параметр_хранения [= значение] [, ... ] ) ]
    [ TABLESPACE табл_пространство ]
    [ WHERE предикат ]
```

Листинг 2.9. реальные Индексы:

```
CREATE INDEX client_name_index ON clients(first_name);

CREATE INDEX client_address_index ON clients(address);
CREATE INDEX client_org_name_index ON clients(organisation_name);
```

Для автоматизации потребуется создать некоторые функции, которые позволят получать отчёты, этими функциями являются:

Листинг 2.10 функции автоматизации:

```
-- получение отчета

CREATE FUNCTION get_report(emp_id integer, startDate timestamp with time zone,
endDate timestamp with time zone)
RETURNS TABLE(EmployeeId integer,
                        AllTasks integer,
                        InTimeCompeted integer,
                        NonInTimeCompleted integer,
```

```
                        FailedTasks integer,
                        InWorkTasks integer) AS '
        SELECT emp_id,
                        get_count_tasks(emp_id, startDate, endDate),
                        get_count_completed(emp_id, startDate, endDate),
                        get_count_completed_failDate(emp_id, startDate,
endDate),
                        get_count_noCompleted(emp_id, startDate, endDate),
                        get_count_inWork(emp_id, startDate, endDate)
' LANGUAGE SQL;

-- получение кол-во НАЗНАЧЕННЫХ заданий для заданного работника
CREATE FUNCTION get_count_tasks(emp_id integer, startDate timestamp with time
zone, endDate timestamp with time zone)
RETURNS integer AS '
        SELECT count(*) FROM jobs
        WHERE ((emp_id = executor_empl_id)
                        AND (startDate <= created) AND (endDate >= created));
' LANGUAGE SQL;

-- получение кол-во выполненных (В СРОК) заданий для заданного работника
CREATE FUNCTION get_count_completed(emp_id integer, startDate timestamp with
time zone, endDate timestamp with time zone)
RETURNS integer AS '
        SELECT count(*) FROM jobs
        WHERE ((emp_id = executor_empl_id)
                        AND (startDate <= created) AND (endDate >= created)
                        AND (startDate <= completed) and (endDate >=
completed)
                        AND (completed <= deadline));
' LANGUAGE SQL;

-- получение кол-во выполненных (НЕ В СРОК) заданий для заданного работника
CREATE FUNCTION get_count_completed_failDate(emp_id integer, startDate
timestamp with time zone, endDate timestamp with time zone)
RETURNS integer AS '
        SELECT count(*) FROM jobs
        WHERE ((emp_id = executor_empl_id)
                        AND (startDate <= created) AND (endDate >= created)
                        AND (startDate <= completed) AND (endDate <=
completed)
                        AND (completed > deadline))
' LANGUAGE SQL;

-- получение кол-во невыполненных заданий (СРОК ИСТЕК) для заданного работника
CREATE FUNCTION get_count_noCompleted(emp_id integer, startDate timestamp with
time zone, endDate timestamp with time zone)
RETURNS integer AS '
        SELECT count(*) FROM jobs
        WHERE ((emp_id = executor_empl_id)
                        AND (startDate <= created) AND (endDate >= created)
```

```
                        AND (completed = NULL) AND NOW() > deadline)
' LANGUAGE SQL;

-- получение кол-во невыполненных заданий (СРОК НЕ ИСТЕК) для заданного
работника
CREATE FUNCTION get_count_inWork(emp_id integer, startDate timestamp with time
zone, endDate timestamp with time zone)
RETURNS integer AS '
        SELECT count(*) FROM jobs
        WHERE ((emp_id = executor_empl_id)
                        AND (startDate <= created) AND (endDate >= created)
                        AND (completed = NULL) AND NOW() < deadline)
' LANGUAGE SQL;
```

Причём для полноценного функционирования также потребовалось добавить ещё 1 колонку в таблицу «Jobs» под названием «Completed», которая отражает момент, когда задание выполнилось. В свою очередь для выгрузки этих данных была создана функция:

Листинг 2.11 Выгрузка:

```
create function send_report(integer, timestamp with time zone, timestamp with
time zone) returns void

    language sql
as
$$
CREATE TEMP TABLE table_report ON COMMIT DROP
AS
SELECT * FROM get_report($1, $2, $3);
COPY (SELECT * FROM table_report) TO E'/home/dmitry/Documents/report.csv' CSV
HEADER;
$$;
```