МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**
(БГТУ им. В.Г. Шухова)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

Лабораторная работа №2

по дисциплине: Базы данных
тема: «Создание объектов базы данных в СУБД»

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверили:
ст. пр. Панченко Максим Владимирович

Белгород 2024 г.

Лабораторная работа №2
Создание объектов базы данных в СУБД
Вариант 8

Цель работы: изучить основные возможности языка SQL для создания структуры базы данных. Научиться создавать базы данных, таблицы, связи, ограничения, а также создавать, изменять и удалять данные.

1. Составить SQL-запросы для создания структуры базы данных, полученной в результате лабораторной работы №1. Указать используемые типы данных, ограничения значений полей; для связей: действия с записями подчинённой таблицы при удалении и изменении соответствующей записи главной таблицы.

   Запустим базу данных в докере, создадим пользователя postgres с паролем postgres и пробросим порт 5432, по которому СУБД будет принимать запросы

```
docker run -e POSTGRES_PASSWORD=postgres -p 5432:5432 postgres
```

   Создадим рабочую БД

```
create database db_course;
```

   Создадим схему для второй лабораторной работы

```
create schema lab_2;
```

   Создадим таблицу home

```
create table home(
    address text unique primary key not null,
    commisioning date null,
    floors int not null constraint floors_check check (floors > 0),
    index int not null constraint index_check check (index >= 100000 and index <= 999999)
);
```

   Адрес хранится в строке, дата введения в эксплуатацию может быть null, если дом ещё строится. Проверим на этажи - не может же быть -1 этаж в доме, а также индекс. Он может быть в диапазоне 100000-999999.
   Создадим таблицу contract

```
create table contract(
    id serial not null primary key,
    transaction_date date not null,
    until_date date not null,
    check (until_date > transaction_date),
    payment int not null constraint check_contract_payment check (payment >= 0),
```

```
    home text not null references home (address) on delete restrict
);
```

id - просто номер договора, имеет тип serial - автоинкрементирующееся число. transaction_date и until_date - обязательные даты начала и конца договора. Проверяем, что конечная дата должна быть больше начальной. payment - ежемесячный платёж по договору, больше нуля. home - внешний ключ к дому, при удалении дома мы должны отклонить запрос, так как у нас всё ещё есть договоры с жильцами.

Создадим таблицу resident

```
create table resident(
    passport_data text not null primary key unique,
    snp text not null,
    email text null,
    phone text null
);
```

passport_data - текст, должна быть уникальной, первичный ключ, уникальное. snp - текст, ФИО обязательное поле, в мире много тёзок, поэтому неуникальное. email - текст, необязательное, может быть null. phone - текст, необязательное, может быть null.

Создадим таблицу residents_contracts

```
create table residents_contracts(
        resident_passport_data text not null references resident (passport_data) on delete cascade,
        contract_id int not null references contract (id) on delete cascade
);
```

При помощи этой базы данных организуется связь многое ко многому. resident_passport_ contract_id - внешние ключи для жильца и контракта соответственно. Если контракт или жилец ”пропадают связь тоже можно удалить, поэтому on delete cascade.

Создадим таблицу payment

```
create table payment(
        id text not null unique primary key,
        paid_date date null,
        until_date date not null,
        contract_id int null references contract (id) on delete set null
);
```

id - является строкой, так как УИП содержит из 25 символов. paid_date может быть null, в отличие от until_date, дедлайн должен быть. contract_id внешний ключ к контракту, может быть полезно хранить чеки вне контрактов, поэтому может быть null а при удалении контракта устанавливается поле на null.

Создадим таблицу task

```
create table task(
    id serial not null primary key,
    payment int not null check (payment >= 0),
    completed_date date null,
    until_date date not null,
    home text null references home (address) on delete set null
);
```

id - автоинкрементирующееся число, первичный ключ. payment - оплата за задачу, число, не может быть < 0, обязательно. completed_date - дата окончания работы, может быть null (если работа ещё выполняется). until_date - дата дедлайна, обязательно. home - внешний ключ к дому, может быть null, например дом могут снести. Однако историю выполнения заказов всё ещё может быть полезно хранить для анализа компетенции рабочих (например, если домов снесли много, значит рабочие выполняют работу плохо), on delete set null.

Создадим таблицу worker

```
create table worker(
    inn text unique not null primary key,
    email text null,
    phone text null
);
```

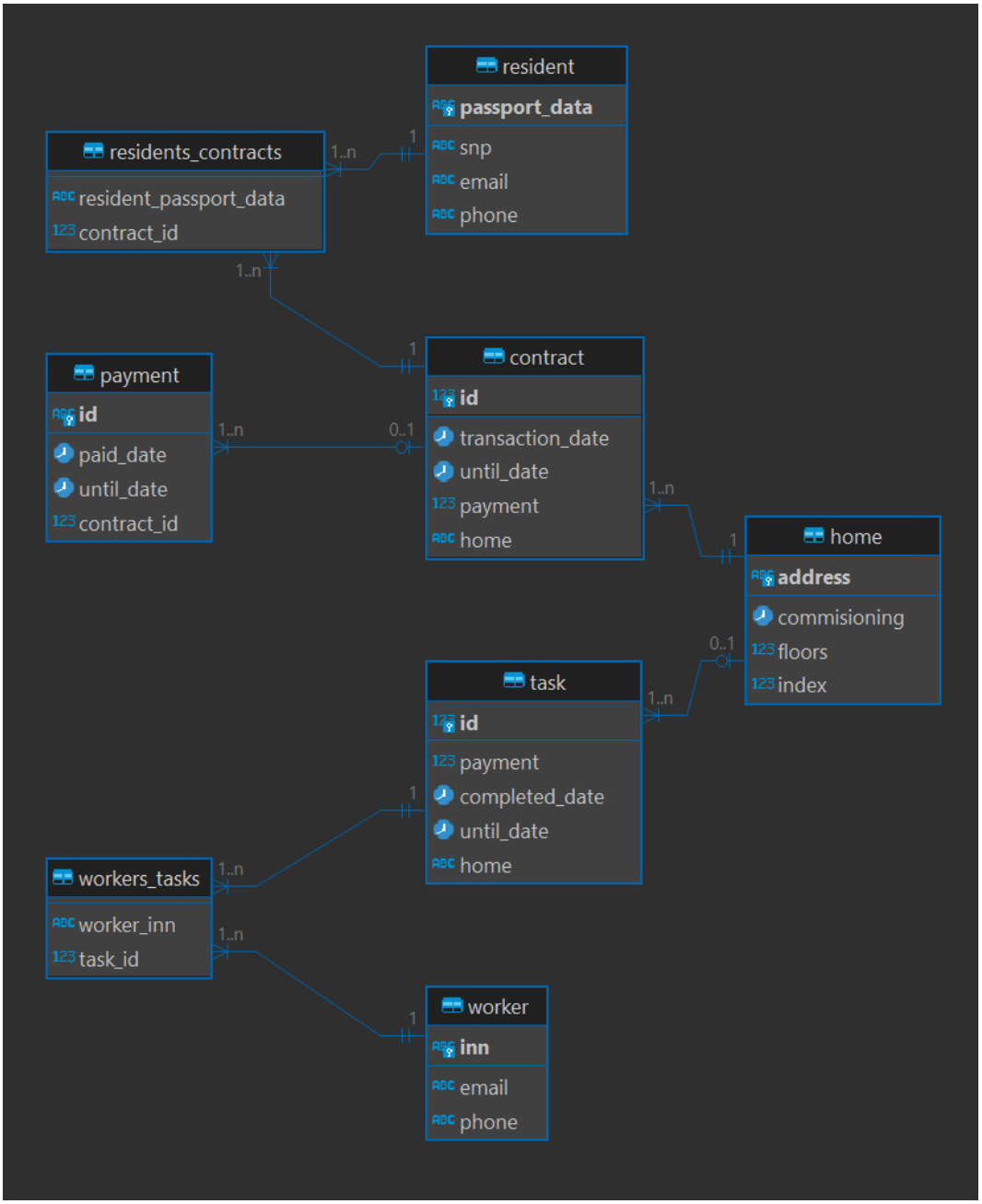
inn - инн, текст, уникальное поле, первичный ключ, обязательно. email - текст, необязательно. phone - текст, необязательно.

Создадим таблицу workers_tasks

```
create table workers_tasks(
    worker_inn text not null references worker (inn) on delete cascade,
    task_id int not null references task (id) on delete cascade
);
```

По аналогии с residents_contracts workers_tasks задаёт связь многое ко многому. Содержит два внешних ключа к рабочему и задаче. При удалении рабочего или задачи связь разрывается, поэтому on delete cascade в обоих внешних ключа.

Итоговая схема:



Вывод: в ходе лабораторной работы изучили основные возможности языка SQL для создания структуры базы данных. Научиться создавать базы данных, таблицы, связи, ограничения, а также создавать, изменять и удалять данные.