Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОННИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа №1
Создание ER-диаграммы. Описание технических требований к приложению

Студент:

Преподаватель:

МИНСК 2024

ВВЕДЕНИЕ

В лабораторной работе выполняется концептуального проектирование БД с использованием UML-модели представления данных (модели «сущность-связь»). Требуется разработать UML-модель данных с учетом семантических ограничений заданной предметной области и представить модель в виде UML- диаграммы.

Темой данной лабораторной работы является разработка UML-диаграммы сущностей и связей в организации «Магазин продуктов».

«Магазин продуктов» - это стандартная модель, работающая по принципу клиент, товар, продавец.

1 СОЗДАНИЕ UML-ДИАГРАММЫ

Исходное задание: Создать концептуальную модель организации «Магазин продуктов» и представить сущности и связи в виде ER-диаграммы.

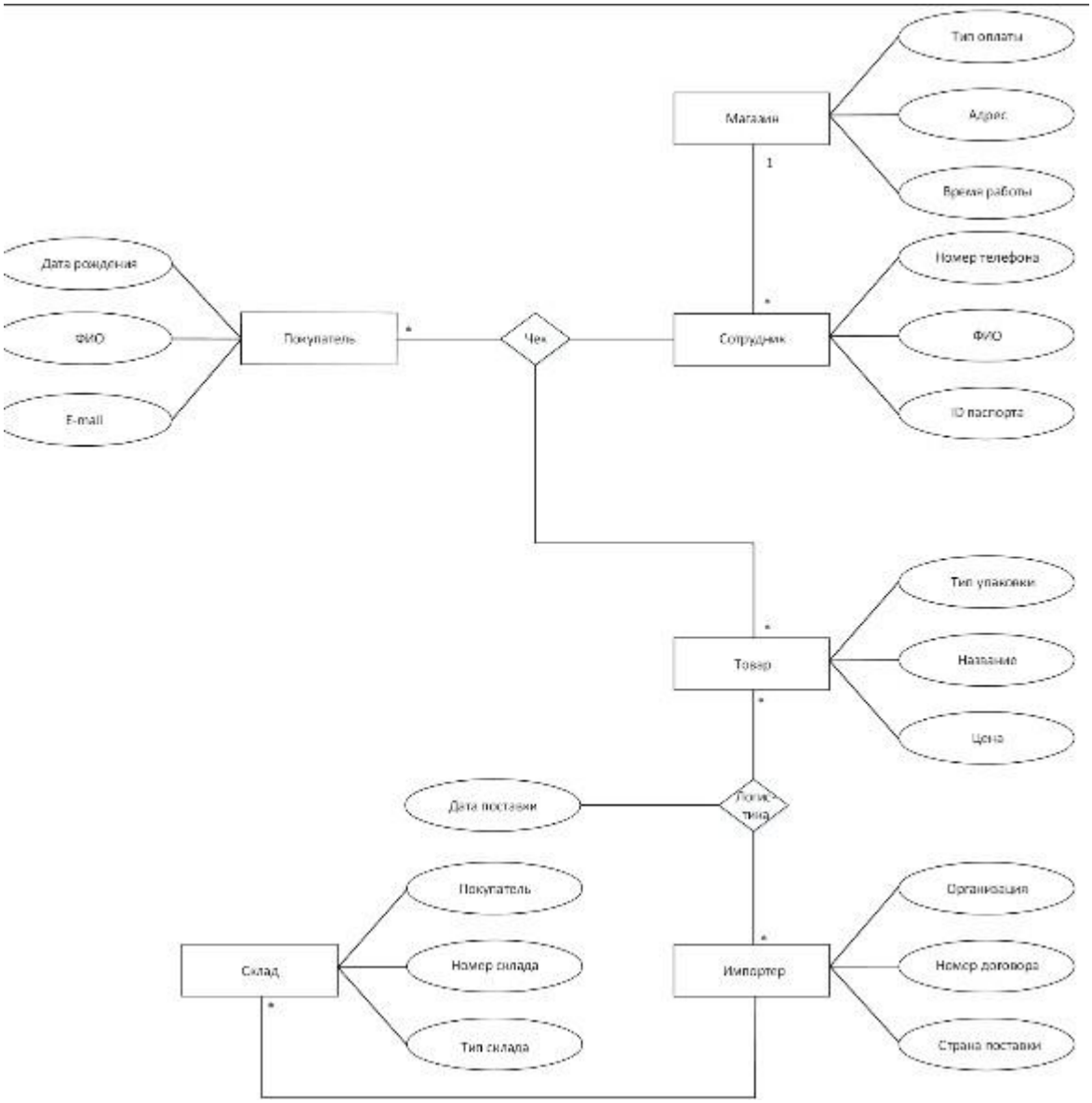

Рисунок 1.1 – ER-диаграмма

Концептуальная UML-диаграмма представлена на рисунке 1.2.

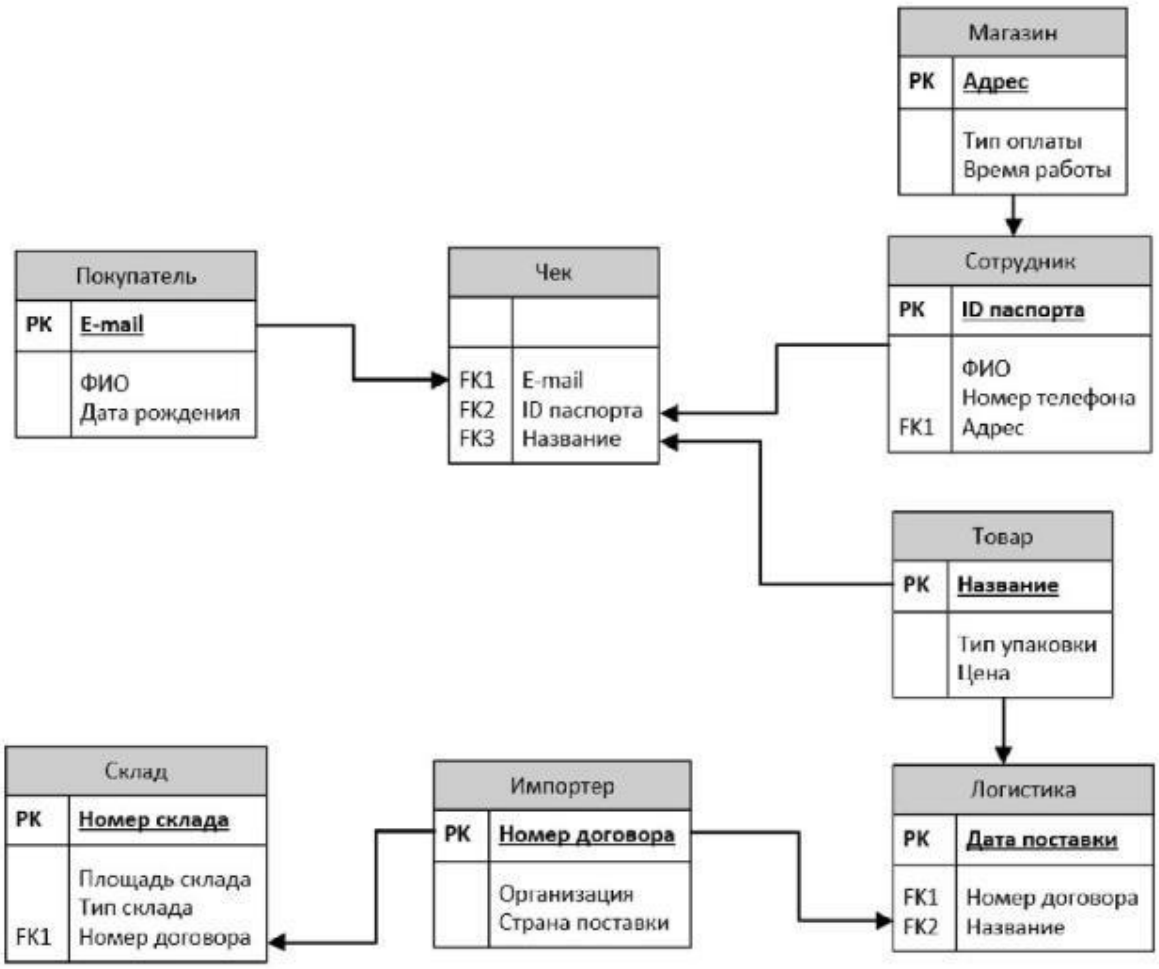

Рисунок 1.2 – UML-диаграмма

1.1 Предметная область

Предметная область – «Магазин продуктов». Модели по типу «сотрудник-покупатель». Предоставляемая услуга – продажа товаров из магазина.

1.2 Типы объектов

Для модели «Магазин продуктов» было выделено 6 типов объектов

1) «Магазин» – физическое пространство, в котором будет происходить продажи товаров.

2) «Покупатель» – представляет клиента магазина.

3) «Сотрудник» – представляет работника магазина.

4) «Товар» – представляет справочную информацию о внутреннем составляющем продуктов.

5) «Импортер» - представляет поставщика магазина.

6) «Склад» - физическое пространство, в котором хранятся товары, до их поставки в магазин.

Остальные объекты являются промежуточными.

1.3 Атрибуты объектов

Атрибуты объекта «Импортер» – «Номер договора», «Организация» и «Страна поставки».

Для объекта «Склад» – «Номер склада», «Площадь склада», «Тип склада» и «Номер договора».

Для объекта «Товар» – «Название», «Тип упаковки», «Цена».

Атрибуты объекта «Магазин» – «Адрес», «Тип оплаты» и «Время работы».

Для объекта «Сотрудник» – «ID-паспорта», «ФИО», «Адрес», «Номер телефона».

Для объекта «Покупатель» – «E-mail», «Дата рождения», «ФИО».

1.4 Типы связей

Самым главным объектом в данной организации являются «Покупатель», «Сотрудник» и «Товар». Этот объект имеет прямые связи с другими объектами, так объединяет всю информацию об используемых данных покупателя, сотрудника, который будет осуществлять продажу, магазине, в котором будет проходить акт покупки-продажи.

Остальные объекты независимы, что в дальнейшем поможет разработать модульное ПО для данного типа организации.

2 ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ

Технические требования содержат принципы построения взаимодействия клиент-серверного приложения в рамках работы с базой данных, но оторвано от конкретной реализации будь то Postgres или BearkleyDB.

Технические требования подразделяются на требования для серверного приложения и требования для интерфейса клиентского приложения.

2.1 Серверное приложение

1) Серверное приложение для реализации соединения с базой данный Postgres будет написано на языке NodeJS.

2) Должны быть предусмотрены CRUD операции для всех таблиц из ER-диаграммы представленной на рисунке 1.

3) Серверное приложением должно представлять из себя REST API сервер.

4) Серверные операции должны быть описаны обще, для дальнейшего масштабирования и наследования.

5) В серверном приложении должны быть описаны все используемые сущности базы данных.

6) Приложение должно быть оптимизированным.

2.2 Клиентское приложение. Интерфейс

1) Клиенсткое приложение должно быть написано в SPA, для обеспечения быстродействия и реактивности. Использовать один из популярных фреймворков.

2) Интерфейс прилжения должен отвечать принципам UI/UX. Дизайн должен быть удобен, понятен и однозначен.

3) Взаимодействие с серверным приложением должно просиходить через REST API.

4) Приложение должно иметь минималистичный дизайн.

5) Приложение должно быть оптимизированным.

2.3 Код

```
CREATE TABLE IF NOT EXISTS public.Магазин(
    ID_магазина INT NOT NULL PRIMARY KEY,
    Адрес VARCHAR(100) NOT NULL,
    Время_работы VARCHAR(100) NOT NULL,
    Тип_оплаты VARCHAR(100) NOT NULL
);
```

Рисунок 2.1 – Таблица сущности Магазин

```
CREATE TABLE IF NOT EXISTS public.Сотрудник(
    ID_сотрудника INT NOT NULL PRIMARY KEY,
    ФИО VARCHAR(100) NOT NULL,
    Магазин_ID INT,
    Номер_телефона VARCHAR(13) NOT NULL,
    ID_паспорта VARCHAR(14) NOT NULL,
    FOREIGN KEY (Магазин_ID) REFERENCES public.Магазин(ID_магазина)
);
```

Рисунок 2.2 – Таблица сущности Сотрудник

```
CREATE TABLE IF NOT EXISTS public.Покупатель(
    ID_покупателя INT NOT NULL PRIMARY KEY,
    ФИО VARCHAR(100) NOT NULL,
    Email VARCHAR(100) NOT NULL,
    Дата_рождения VARCHAR(10) NOT NULL
);
```

Рисунок 2.3 – Таблица сущности Покупатель

```
CREATE TABLE IF NOT EXISTS public.Импортер(
    ID_импортера INT NOT NULL PRIMARY KEY,
    Организация VARCHAR(100) NOT NULL,
    Номер_договора INT NOT NULL,
    Страна_поставки VARCHAR(100) NOT NULL
);
```

Рисунок 2.4 – Таблица сущности Импортер

```
CREATE TABLE IF NOT EXISTS public.Склад(
    ID_склада INT NOT NULL PRIMARY KEY,
    Площадь_склада INT NOT NULL,
    Тип_склада VARCHAR(100) NOT NULL,
    Номер_склада VARCHAR(100) NOT NULL,
    Импортер_ID INT,
    FOREIGN KEY (Импортер_ID) REFERENCES public.Импортер(ID_импортера)
);
```

Рисунок 2.5 – Таблица сущности Склад

```
CREATE TABLE IF NOT EXISTS public.Чек(
    Товар_ID INT NOT NULL,
    Сотрудник_ID INT NOT NULL,
    Покупатель_ID INT NOT NULL,
    FOREIGN KEY (Покупатель_ID) REFERENCES public.Покупатель(ID_покупателя),
    FOREIGN KEY (Сотрудник_ID) REFERENCES public.Сотрудник(ID_сотрудника),
    FOREIGN KEY (Товар_ID) REFERENCES public.Товар(ID_товара)
);
```

Рисунок 2.6 – Таблица сущности Чек

```
CREATE TABLE IF NOT EXISTS public.Логистика(
    Дата_поставки VARCHAR(10) NOT NULL PRIMARY KEY,
    Товар_логистики_ID INT NOT NULL,
    Импортер_логистики_ID INT NOT NULL,
    FOREIGN KEY (Товар_логистики_ID) REFERENCES public.Товар(ID_товара),
    FOREIGN KEY (Импортер_логистики_ID) REFERENCES public.Импортер(ID_импортера)
);
```

Рисунок 2.7 – Таблица сущности Магазин

ЗАКЛЮЧЕНИЕ

В результате работы над лабораторной работой была построена ER-диаграмма организации «Магазин продуктов». Были выделены основные объекты и представлены связи между ними.

Были описаны технические требования для серверного и клиентского приложения с учетом специфики разработки на языках высокого уровня.

Программа для работы с базами данных PostgreSQL была успешно установлена на ПК.