

Министерство образования Республики Беларусь

Учреждение образования  
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа №2-3  
на тему  
СОЗДАНИЕ РЕЛЯЦИОННОЙ СХЕМЫ ДАННЫХ

Выполнил:

Кравченко М.Д.

Проверила:

Силич С.С.

МИНСК 2025

## СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ.....</b>	<b>3</b>
<b>1 ОПИСАНИЕ РЕЛЯЦИОННОЙ МОДЕЛИ «БАНК» .....</b>	<b>4</b>
<b>1.1 Описание таблицы и сущностей .....</b>	<b>4</b>
<b>1.2 Типы связей.....</b>	<b>5</b>
<b><u>2</u> ПРЕОБРАЗОВАНИЕ В «БУМАЖНУЮ» РЕЛЯЦИОННУЮ МОДЕЛЬ .....</b>	<b>6</b>
<b><u>3</u> «АВТОМАТИЗИРОВАННОЕ» ПРЕОБРАЗОВАНИЕ .....</b>	<b>9</b>
<b><u>ЗАКЛЮЧЕНИЕ</u> .....</b>	<b>12</b>

## ВВЕДЕНИЕ

В банковском деле эффективное управление данными является важным аспектом для обеспечения качественного обслуживания клиентов и повышения эффективности безопасности операций. Информационные системы, построенные на основе реляционных баз данных, позволяют структурировать, хранить и обрабатывать большие объемы данных, что особенно актуально для банков. PostgreSQL, как одна из наиболее популярных и надежных систем управления базами данных, предоставляет широкие возможности для реализации таких систем.

Целью данной лабораторной работы является преобразование ER-модели «Банк», разработанной ранее, в реляционную схему данных с использованием PostgreSQL. В процессе работы будут выполнены следующие задачи:

- Преобразование сущностей и атрибутов ER-диаграммы в таблицы и столбцы;
- Определение первичных и внешних ключей;
- Реализация связей между таблицами, включая связи "многие ко многим";
- Нормализация данных для устранения избыточности и обеспечения целостности.

Результатом работы станет реляционная схема данных, которая может быть использована для создания базы данных «Банк» в PostgreSQL. Эта схема включает таблицы для пользователей, счетов, транзакций, паспортных данных, адресов, должностей, сотрудников, отделений банка и стран. Для реализации связей "многие ко многим" добавлены промежуточные таблицы, такие как `employee_position` для управления назначениями сотрудников на должности и `employee_branch` для учета работы сотрудников в отделениях банка.

# 1 ОПИСАНИЕ РЕЛЯЦИОННОЙ МОДЕЛИ «БАНК»

Исходное задание: создание реляционной схемы данных, преобразовать ER-диаграмму в реляционную модель данных, нормализовать реляционные отношения.

## 1.1 Описание таблицы и сущностей

В рамках реляционной модели «Банк» предусмотрены следующие таблицы и соответствующие им сущности.

Таблица User соответствует сущности клиента банка и включает поля: id клиента (уникальный идентификатор клиента), адрес электронной почты, номер телефона, тип клиента (частное лицо, бизнес-клиент и т. д.), дата регистрации и passport\_id (ссылка на паспорт клиента, как внешний ключ).

Таблица Passport отражает сущность паспортных данных и содержит поля: id паспорта (уникальный идентификатор паспорта), номер паспорта, ФИО, дату выдачи, кем выдан, address\_id (ссылка на адрес клиента, как внешний ключ) и country\_id (ссылка на страну, как внешний ключ).

Таблица Address соответствует сущности адреса и включает поля: id адреса (уникальный идентификатор адреса), город, улица, номер дома и почтовый индекс.

Таблица Bill отражает сущность банковского счёта и содержит поля: id счёта (уникальный идентификатор счёта), номер счёта, баланс, валюта счёта, дата открытия счёта и user\_id (ссылка на клиента, как внешний ключ).

Таблица Transaction соответствует сущности транзакции и включает поля: id транзакции (уникальный идентификатор транзакции), сумма транзакции, валюта транзакции, дата и время проведения транзакции, статус транзакции, bill\_id (ссылка на счёт, как внешний ключ) и user\_id (ссылка на клиента, как внешний ключ).

Таблица Bank\_Branch отражает сущность отделения банка и содержит поля: id отделения (уникальный идентификатор отделения), название отделения, контактный номер, часы работы, данные управляющего и address\_id (ссылка на адрес отделения, как внешний ключ).

Таблица Position соответствует сущности должности и включает поля: id должности (уникальный идентификатор должности), название должности, обязанности, требуемый опыт и максимальная зарплата.

Таблица Bank\_Employee отражает сущность сотрудника банка и содержит поля: id сотрудника (уникальный идентификатор сотрудника), зарплата, рабочий график, дата оформления на работу, дата начала работы, дата окончания работы, отдел, passport\_id (ссылка на паспорт сотрудника, как внешний ключ) и bank\_branch\_id (ссылка на отделение банка, как внешний ключ).

Таблица Country отражает сущность страны и содержит поля: id страны (уникальный идентификатор сотрудника), название, регион, код страны, язык

страны.

Для реализации связи многие-ко-многим предусмотрена промежуточная таблица Employee\_Position. Она содержит поля: employee\_id (ссылка на сотрудника, как внешний ключ), position\_id (ссылка на должность, как внешний ключ), дата начала и дата окончания работы на указанной должности.

Для реализации связи многие-ко-многим предусмотрена промежуточная таблица Employee\_Branch. Она содержит поля: employee\_id (ссылка на сотрудника, как внешний ключ), bank\_branch\_id (ссылка на отделение банка, как внешний ключ), дата начала и дата окончания работы в указанном отделении.

## 1.2 Типы связей

Для описания взаимосвязей между объектами были выделены следующие связи:

1. Связь «User – Passport» (один-к-одному). Каждый клиент имеет один паспорт, и каждый паспорт принадлежит одному клиенту.

2. Связь «Bank\_Employee – Passport» (один-к-одному). Каждый сотрудник банка имеет один паспорт, и каждый паспорт может принадлежать только одному сотруднику.

3. Связь «Passport – Country» (многие-к-одному). Один паспорт принадлежит одной стране, но одна страна может включать множество паспортов

4. Связь «Passport – Address» (многие-к-одному). Один паспорт связан с одним адресом, где проживает его владелец. Однако один адрес может быть связан с несколькими паспортами.

5. Связь «Bank\_Employee – Position» (многие-ко-многим). Один сотрудник может занимать несколько должностей (карьерный рост), и одна должность может быть занята несколькими сотрудниками. Для этой связи используется промежуточная таблица Employee\_Position.

6. Связь «Bank\_Employee – Bank\_Branch» (многие-ко-многим). Один сотрудник может работать в нескольких отделениях банка (с течением времени), и одно отделение может включать нескольких сотрудников. Для этой связи используется промежуточная таблица Employee\_Branch.

7. Связь «Bank\_Branch – Address» (один-к-одному). Каждое отделение банка имеет один адрес, и каждый адрес принадлежит одному отделению.

8. Связь «User – Bill» (один-ко-многим). Один клиент может иметь несколько счетов. Но каждый счёт принадлежит только одному клиенту.

9. Связь «Bill – Transaction» (один-ко-многим). Один счёт может участвовать в нескольких транзакциях. Но каждая транзакция связана только с одним счётом.

10. Связь «User – Transaction» (один-ко-многим). Один клиент может иметь несколько транзакций. Но каждая транзакция принадлежит только одному клиенту.

## 2 ПРЕОБРАЗОВАНИЕ В «БУМАЖНУЮ» РЕЛЯЦИОННУЮ МОДЕЛЬ

Схема ER-диаграммы представлена на рисунке 2.1.

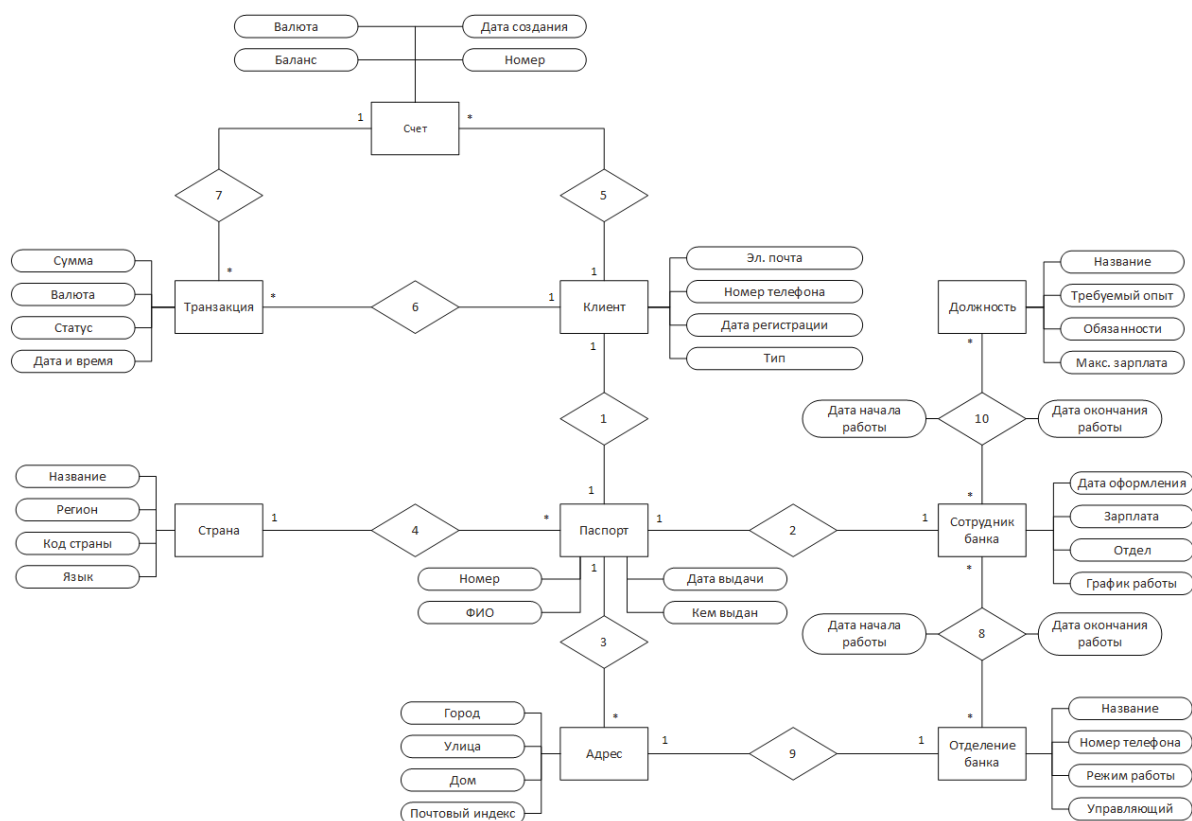


Рисунок 2.1 – ER-диаграмма

Порядок перевода ER-модели в реляционную модель выполняется с помощью алгоритма, состоящего из пяти шагов:

1. Каждый объект на ER-диаграмме превращается в реляционное отношение (далее для краткости – таблицу), имя объекта становится именем таблицы. Можно выделить девять таблиц со следующими именами: «Паспорт», «Клиент», «Счет», «Транзакция», «Страна», «Адрес», «Отделение банка», «Сотрудник банка», «Должность».

2. Каждый атрибут объекта становится столбцом с тем же именем.

3. Уникальные атрибуты объекта превращаются в первичный ключ таблицы. Таким образом были добавлены следующие первичные ключи: id паспорта, id клиента, id счета, id транзакции, id страны, id адреса, id отделения банка, id сотрудника банка, id должности. Сопоставление объектов URD и

UML представлено на рисунке 2.2.

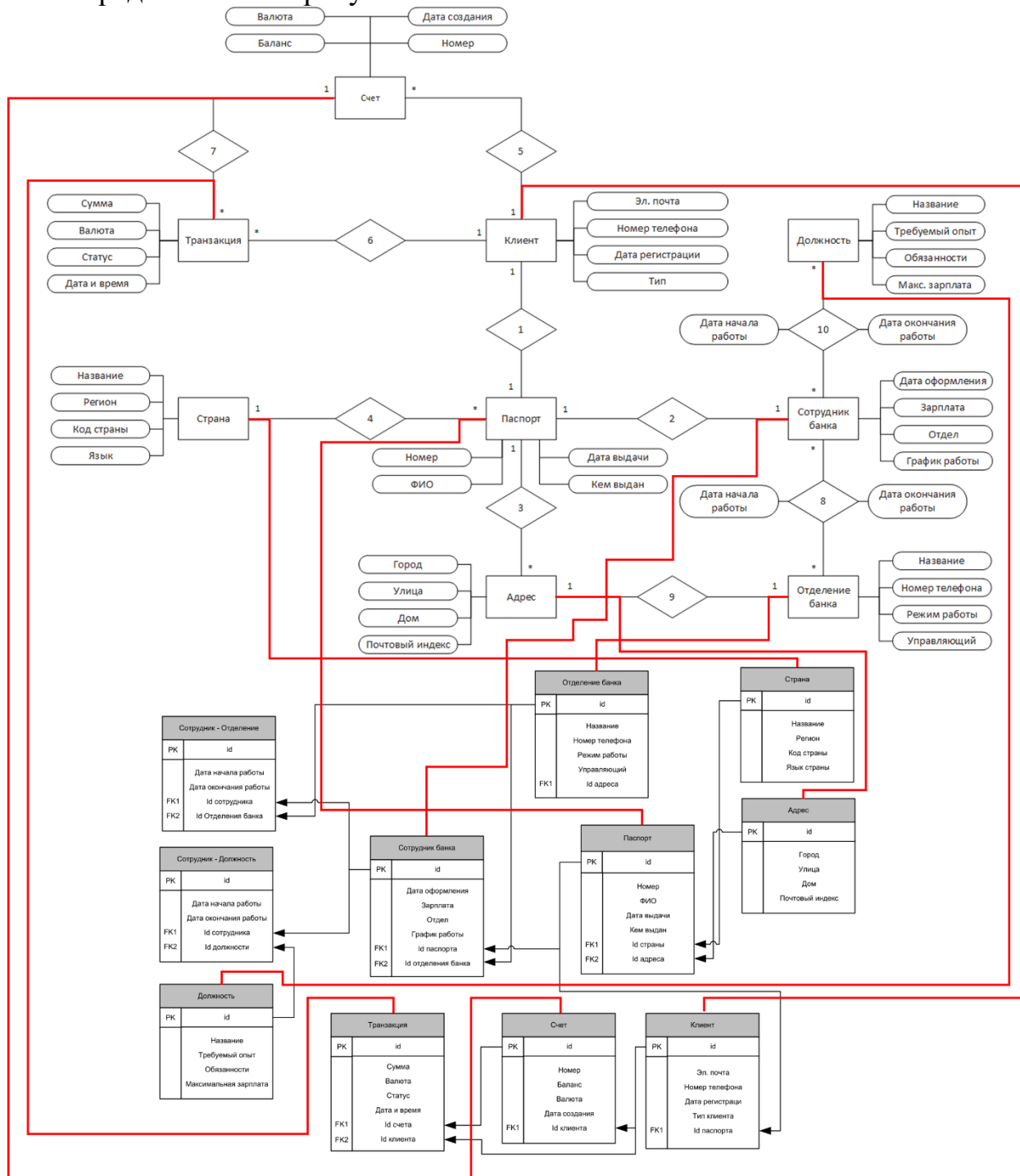


Рисунок 2.2 – Сопоставление объектов URD и UML

4. Связи «один-ко-многим» становятся ссылками в уже существующих таблицах, при этом внешний ключ добавляется в виде столбца в таблицу, соответствующую объекту со стороны «многие» связи. Внешние ключи ссылаются на первичные ключи целевых таблиц. Пример связи «один-ко-многим» представлен на рисунке 2.3.

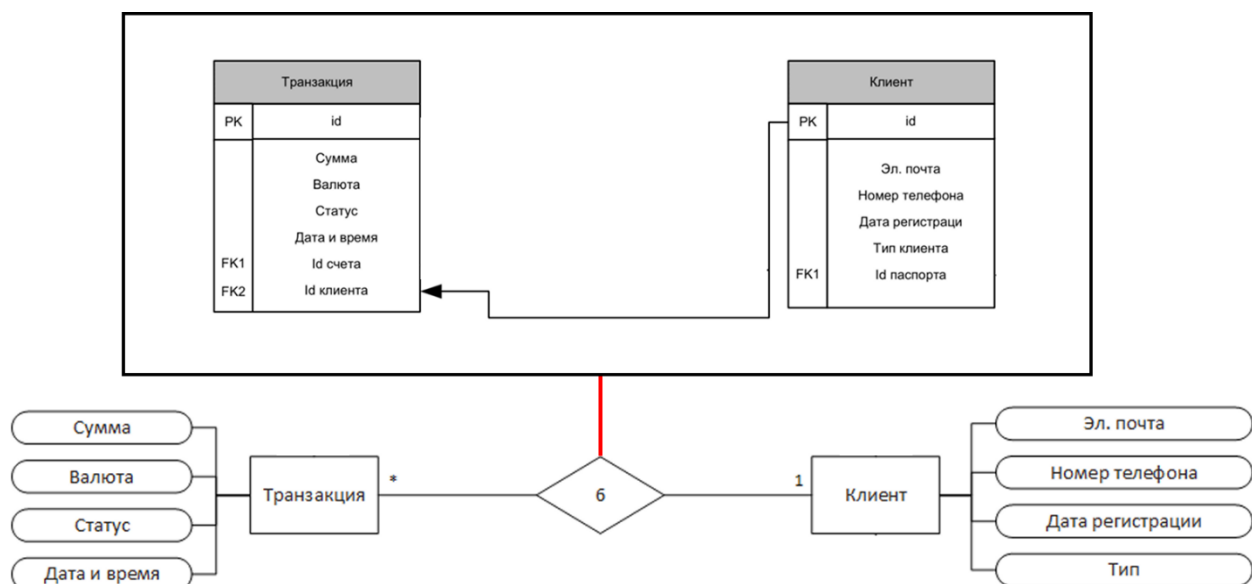


Рисунок 2.3 – Связь «один-ко-многим»

5. Связи «многие-ко-многим» реализуются через отдельную таблицу. Была создана таблица «Диагноз-медикаменты», в которой находятся два поля внешних ключей: «id диагноза» и «id медикамента». Пример связи «многие-ко-многим» представлен на рисунке 2.4.

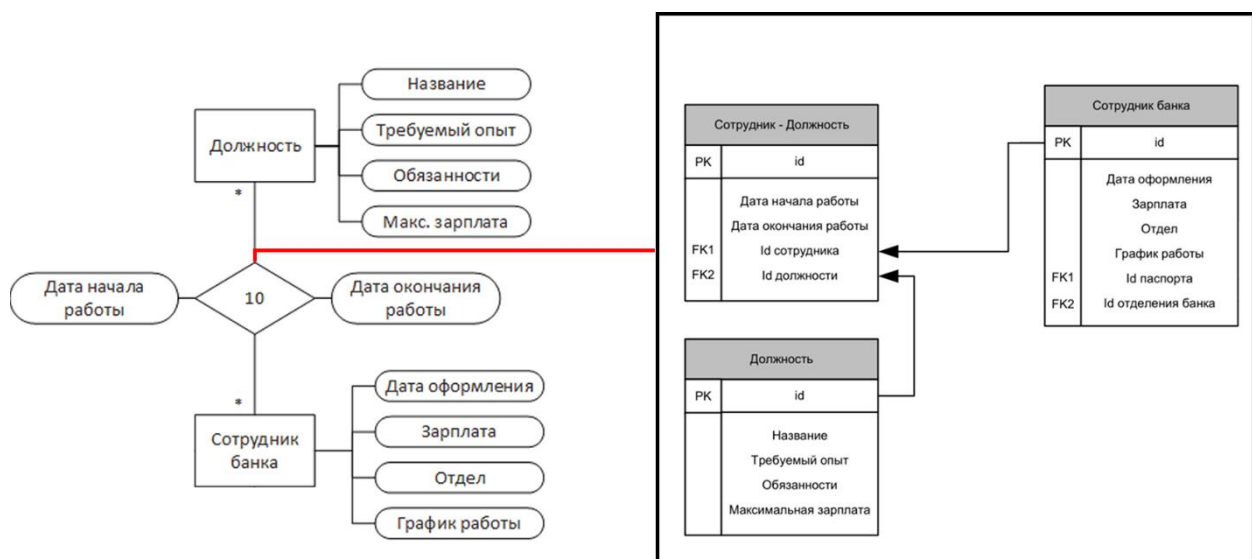


Рисунок 2.4 – Связь «многие-ко-многим»

UML-диаграмма реляционной схемы данных «бумажного» преобразования представлена на рисунке 2.5



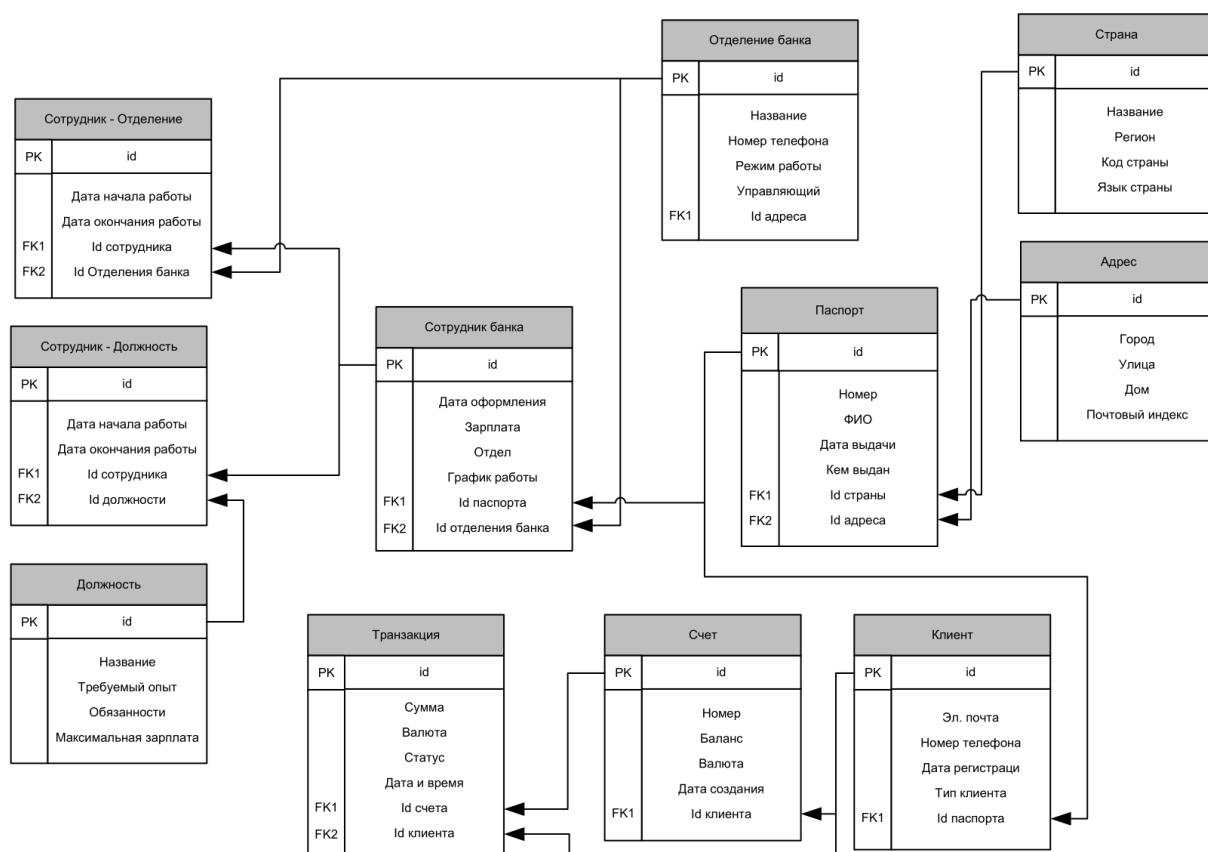


Рисунок 2.5 – UML-диаграмма

### 3 «АВТОМАТИЗИРОВАННОЕ» ПРЕОБРАЗОВАНИЕ

Для перевода ER-диаграммы в реляционную диаграмму используется графический инструмента администрирования и проектирования баз данных – pgAdmin 4. Для проведения операций были выполнены следующие шаги:

1. Открыть программу pgAdmin.
2. Открыть вкладку Tools, а далее ERD Tool (рис. 3.1).

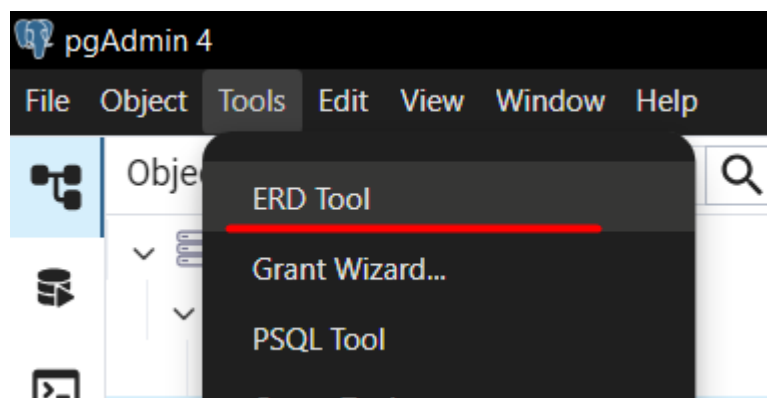


Рисунок 3.1 – Инструмент ERD Tool

3. В открывшейся зоне создать таблицу. На рисунке 3.2 изображена созданная таблица.

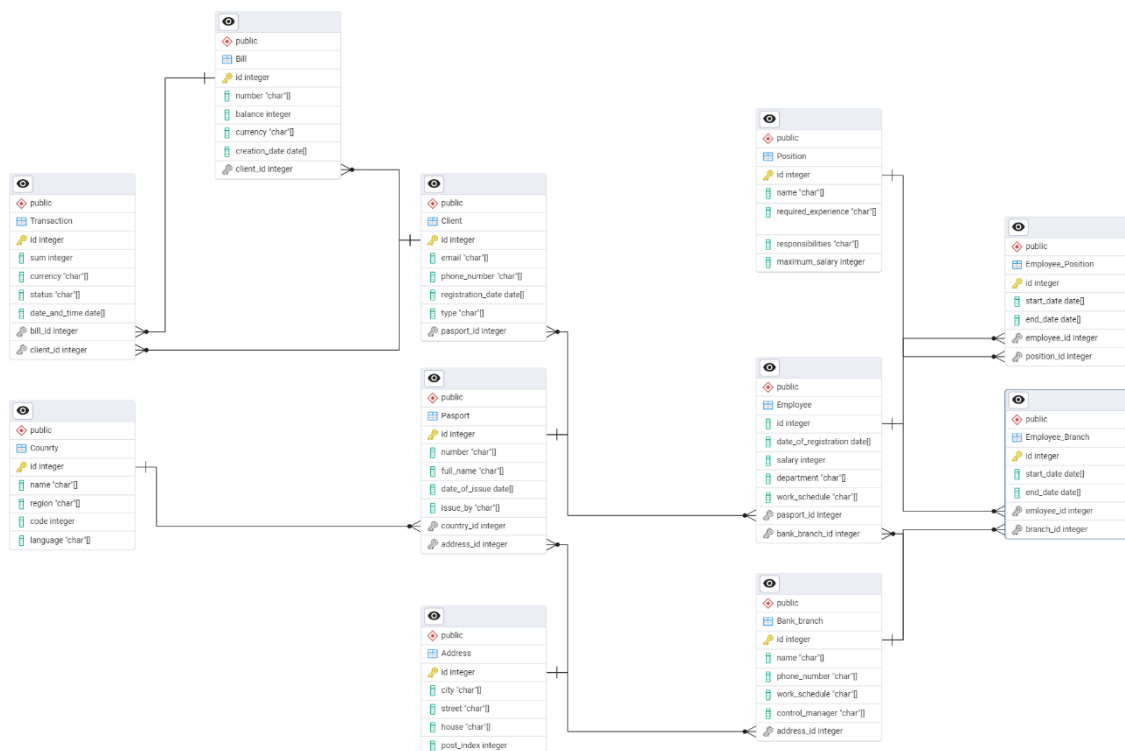


Рисунок 3.2 – Созданная таблица

4. Нажать на кнопку Generate SQL и выполнить SQL-код. На рисунке 3.3 изображена генерация SQL кода

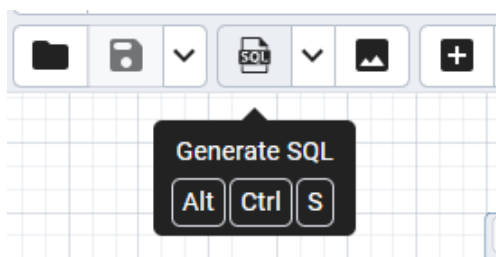
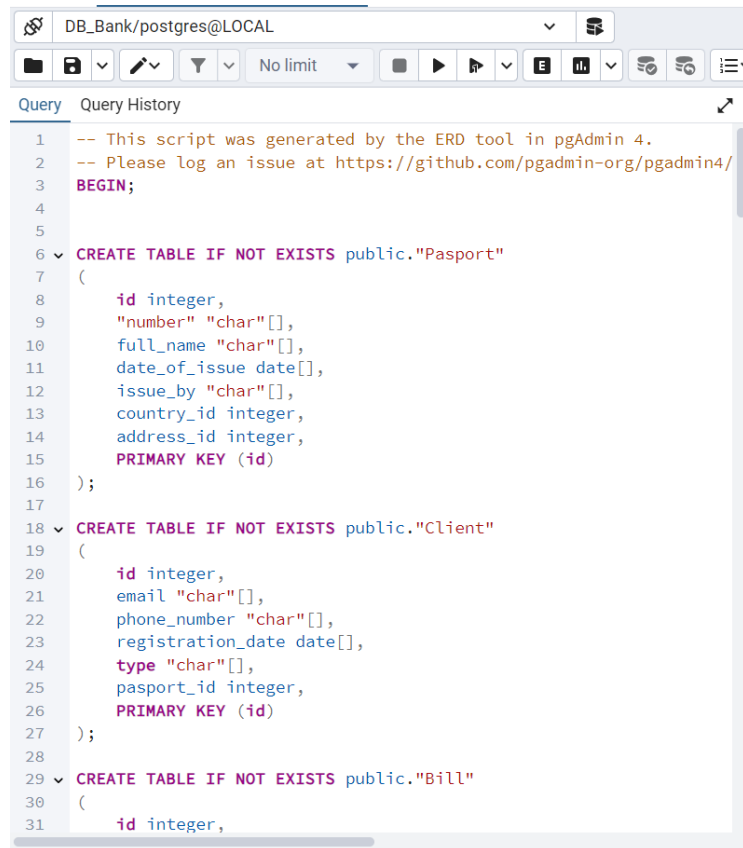


Рисунок 3.3 – Генерация SQL кода

На рисунке 3.4 показан SQL код для создания реляционной модели.



The screenshot shows the pgAdmin 4 Query Editor interface. The title bar indicates the connection is 'DB\_Bank/postgres@LOCAL'. The 'Query' tab is active, displaying a SQL script. The script starts with a comment: '-- This script was generated by the ERD tool in pgAdmin 4. -- Please log an issue at https://github.com/pgadmin-org/pgadmin4/'. It begins with 'BEGIN;' and contains three 'CREATE TABLE IF NOT EXISTS' statements. The first table is 'public.Pasport' with columns: 'id' (integer, primary key), 'number' (char), 'full\_name' (char), 'date\_of\_issue' (date), 'issue\_by' (char), 'country\_id' (integer), and 'address\_id' (integer). The second table is 'public.Client' with columns: 'id' (integer, primary key), 'email' (char), 'phone\_number' (char), 'registration\_date' (date), 'type' (char), and 'passport\_id' (integer). The third table is 'public.Bill' with column: 'id' (integer). The script is partially visible, ending at line 31.

```
1  -- This script was generated by the ERD tool in pgAdmin 4.
2  -- Please log an issue at https://github.com/pgadmin-org/pgadmin4/
3  BEGIN;
4
5
6  CREATE TABLE IF NOT EXISTS public."Pasport"
7  (
8      id integer,
9      "number" "char"[],
10     full_name "char"[],
11     date_of_issue date[],
12     issue_by "char"[],
13     country_id integer,
14     address_id integer,
15     PRIMARY KEY (id)
16 );
17
18 CREATE TABLE IF NOT EXISTS public."Client"
19 (
20     id integer,
21     email "char"[],
22     phone_number "char"[],
23     registration_date date[],
24     type "char"[],
25     passport_id integer,
26     PRIMARY KEY (id)
27 );
28
29 CREATE TABLE IF NOT EXISTS public."Bill"
30 (
31     id integer,
```

Рисунок 3.4 – SQL код

5. После запуска скрипта в панели Tables должны отображаться созданные таблицы и ключи, что представлено на рисунке 3.5:

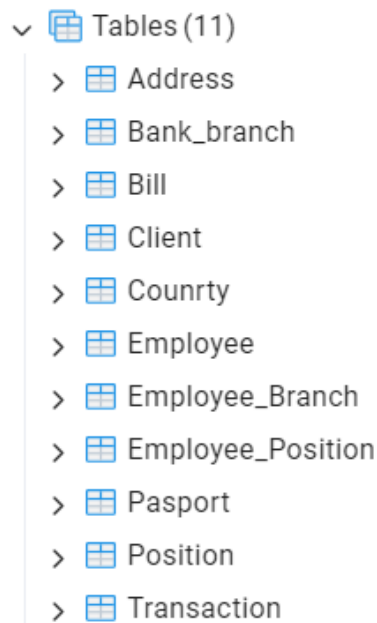


Рисунок 3.5 – Сгенерированные таблицы

## **ЗАКЛЮЧЕНИЕ**

В результате выполнения лабораторной работы была успешно преобразована ER-модель «Банк» в реляционную схему данных с использованием PostgreSQL. Были созданы таблицы, соответствующие сущностям ER-диаграммы, определены первичные и внешние ключи, а также реализованы связи между таблицами. В процессе работы были рассмотрены два варианта преобразования: «бумажный» и «автоматизированный», что позволило выявить и устранить возможные несоответствия.

Разработанная реляционная схема данных в дальнейшем будет использована для создания базы данных «Банк» в PostgreSQL. Эта база данных обеспечит эффективное управление информацией о клиентах, сотрудниках, счетах, и транзакциях, что в свою очередь позволит улучшить качество обслуживания в банке.