

МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»
(РУТ(МИИТ))

Институт управления и цифровых технологий

Кафедра «Вычислительные системы, сети и информационная безопасность»

КУРСОВАЯ РАБОТА
По дисциплине «Технологии хранения данных»
На тему: «Банкоматы»

Группа: УИС-211
Студент: Чибает А. Т.,
Преподаватели: Новиков А. И.,
Сосновская А.В.

Москва
2024 г.

СОДЕРЖАНИЕ

1	ЗАДАНИЕ ПО КУРСОВОЙ РАБОТЕ ДЛЯ ДИСЦИПЛИНЫ «ТЕХНОЛОГИИ ХРАНЕНИЯ ДАННЫХ».....	4
2	ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	6
3	СУЩНОСТИ И АТТРИБУТЫ БАЗЫ ДАННЫХ.....	7
4	РАЗРАБОТКА ER-МОДЕЛИ БАЗЫ ДАННЫХ В НОТАЦИИ ЧЕНА	8
5	РАЗРАБОТКА ER-МОДЕЛИ БАЗЫ ДАННЫХ В НОТАЦИИ IDEFIX	13
5.1	Erwin Data Modeler.....	13
5.2	Draw.io	16
6	РАЗРАБОТКА РЕЛЯЦИОННОЙ МОДЕЛИ БАЗЫ ДАННЫХ И ЗАПРОСОВ К БАЗЕ ДАННЫХ.....	20
7	РАЗРАБОТКА РЕЛЯЦИОННОЙ МОДЕЛИ БАЗЫ ДАННЫХ И ЗАПРОСОВ К БАЗЕ ДАННЫХ.....	24
7.1	Таблица «Банк»	24
7.2	Таблица «Клиент».....	25
7.3	Таблица «Банкомат»	27
7.4	Таблица «Операция»	29
7.5	Таблица «Выдача_наличных».....	31
7.6	Таблица «Валюта»	32
7.7	Таблица «Операция_валюта».....	33
7.8	ER-диаграмма	35
8	ЗАПРОСЫ НА ЯЗЫКЕ SQL.....	36
9	РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.....	42
9.1	Введение	42
9.1.1	Область применения	42
9.1.2	Краткое описание возможностей	42
9.1.3	Уровень подготовки пользователя	42
9.2	Назначения и условия применения.....	42
9.2.1	Виды деятельности, функции, для автоматизации которых предназначено данное средство автоматизации	42
9.3	Подготовка к работе	42
9.3.1	Состав и содержание дистрибутивного носителя данных.....	42

9.4	Порядок загрузки данных и программ.....	42
9.5	Уровень подготовки пользователя	43
9.6	Описание операций.....	43
9.6.7	Выполняемые функции	43
9.7	Описание операций технологического процесса обработки данных, необходимых для выполнения задач.....	43
	Операция 1. Авторизация.	43
	Операция 2. Удаление и добавление строк	45
	Операция 3. Ввод SQL запросов.....	47
ПРИЛОЖЕНИЕ А	49
А.1	Файл bank.ru, содержащий класс базы данных и контроллеры....	49

1 ЗАДАНИЕ ПО КУРСОВОЙ РАБОТЕ ДЛЯ ДИСЦИПЛИНЫ «ТЕХНОЛОГИИ ХРАНЕНИЯ ДАННЫХ»

Разработать приложение на одном из языков высокого уровня (C++, C#, Java, Python и др), которое позволяет работать с базой данных.

Приложение должно иметь визуальный интерфейс, позволяющий вводить, корректировать и удалять информацию из таблиц базы данных.

Приложение может быть, как локальным, так и веб-приложением.

Приложение должно выполнять поисковые запросы, которые были реализованы в лабораторных работах. Приложение должно позволять вводить параметры запроса, а не использовать конкретные константы запросов, которые были в запросах, приведенных в лабораторных работах.

Возможна реализация интерфейса, который позволит вводить произвольный запрос на языке SQL и показывать результат выполнения запроса в виде таблицы.

Предметная область для базы данных выбирается в соответствии с заданием на выполнение лабораторных работ.

Схема базы данных должна быть спроектирована по алгоритмам проектирования реляционной модели базы данных (алгоритмы: Фэджина, Делобеля-Кейси, Бернштейна).

Для реализации проекта выбирается схема реляционной базы данных, полученная по одному из алгоритмов проектирования.

При выполнении курсовой работы используется такая же система управления базами данных как при выполнении лабораторных работ.

Содержание отчета по курсовой работе

1. Титульный лист.
2. Содержание.
3. Постановка задачи.
4. Описание предметной области.
5. ER-модель базы данных в нотации IDEF1X.
6. Проектирование реляционной модели базы данных по алгоритмам проектирования.

7. Описание схемы базы данных на языке SQL.
8. Описание используемых запросов на языке SQL.
9. Руководство пользователя по работе с приложением.
10. Исходный код программы.

2 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

В варианте предметной области «Банкоматы» указано, что в базе данных должна храниться информация:

о БАНКАХ: код банка, название банка, юридический адрес; БАНКОМАТАХ: номер банкомата, адрес банкомата, код банка (обслуживающего банкомат); КЛИЕНТАХ: номер карточки клиента, Ф.И.О. клиента, адрес клиента, код банка (обслуживающего клиента); ОПЕРАЦИЯХ выдачи наличных денег клиентам: номер карточки клиента, номер банкомата, дата, время, комиссия (Да/Нет), сумма выдачи (руб.).

При проектировании БД необходимо учитывать следующее:

банк обслуживает несколько банкоматов. Банкомат обслуживается одним банком;

банк обслуживает несколько клиентов. Клиент обслуживается одним банком; банкомат обслуживает несколько клиентов. Клиент обслуживается несколькими банкоматами;

банкомат осуществляет несколько операций обналичивания денег. Операция обналичивания денег связана с одним банкоматом;

клиент осуществляет несколько операций обналичивания денег. Операция обналичивания денег связана с одним банкоматом.

Кроме того, следует учесть:

каждый банк обязательно имеет в обслуживании банкоматы. Каждый банкомат обязательно обслуживается банком;

каждый банк обязательно имеет клиентов. Каждый клиент обязательно обслуживается банком;

каждый банкомат обязательно обслуживает клиентов. Каждый клиент обязательно обслуживается банкоматами; банкомат не обязательно осуществляет постоянно операции выдачи наличных денег. Каждая операция выдачи наличных денег обязательно связана с банкоматом;

клиент не обязательно осуществляет операции обналичивания денег. Каждая операция обналичивания денег обязательно связана с клиентом.

3 СУЩНОСТИ И АТТРИБУТЫ БАЗЫ ДАННЫХ

1. Объект «Банк» (атттрибуты: bank_id; название_банка; юр_адрес)
Ключ - bank_id
2. Объект «Клиент» (атттрибуты: bank_id; серия_номер_паспорта; фиио_клиента; адрес_клиента; пинкод)
Ключ – серия_номер_паспорта
Внешний ключ - bank_id
3. Объект «Операция» (атттрибуты: operation_id; серия_номер_паспорта; АТМ_id; дата; время; комиссия; сумма_выдачи; тип_операции; статус_операции; баланс)
Ключ - operation_id
Внешний ключ – серия_номер_паспорта; АТМ_id
4. Объект «Банкомат» (атттрибуты: АТМ_id; bank_id; адрес_банкомата)
Ключ - АТМ_id
Внешний ключ - bank_id
5. Объект «Выдача_наличных» (атттрибуты: серия_номер_паспорта; АТМ_id; сумма_выдачи)
Внешний ключ - серия_номер_паспорта; АТМ_id
6. Объект «Валюта» (атттрибуты: код_валюты, название_валюты, сокращение; страна; текущий_курс)
Ключ - код_валюты
7. Объект «Операция_валюта» (атттрибуты: operation_id; код_валюты; сумма_в_валюте; выдача_в_валюте)
Внешний ключ - operation_id; код_валюты

4 РАЗРАБОТКА ER-МОДЕЛИ БАЗЫ ДАННЫХ В НОТАЦИИ ЧЕНА

Разработать ER-модель базы данных в нотации Чена для одной из предметных областей. Предметная область выбирается в соответствии с порядковым номером студента в журнале. Допускается выбор предметной области, не указанной в списке, по согласованию с преподавателем. В приведенном описании предметных областей кратко перечислены сущности предметной области. Перечень атрибутов сущностей определяется студентом самостоятельно и согласовывается с преподавателем. Для разработки модели используется инструмент ERDPlus или другой по согласованию с преподавателем.

ERDPlus — это веб-инструмент для моделирования баз данных, который позволяет быстро и легко создавать диаграммы отношений сущностей (ERDs).

Он работает с большинством современных инструментов RDBMS, включая Oracle, MySQL, Microsoft SQL Server, PostgreSQL, Teradata, IBM DB2, Microsoft Access и другие.

Инструмент поддерживает рисование регулярных и слабых сущностей, различных типов атрибутов (регулярных, уникальных, многозначных, производных, составных и необязательных) и всех возможных ограничений количества отношений (обязательных-многих, необязательных-многих, обязательных-одного и необязательных-одного).

Он также поддерживает быстрое создание внешних ключей и линий ссылочной целостности с помощью простых действий «клик-точка-соединение».

Описание работы с ERDPlus:

Откройте браузер, перейдите на главную страницу ERDPlus и нажмите «TRY IT!» (рисунок 4.1). Редактор будет запущен. ERDPlus предоставляет интуитивно понятный интерфейс для создания и редактирования ER-диаграмм, диаграмм сущность-связь (Entity-Relationship), а также других диаграмм баз данных.

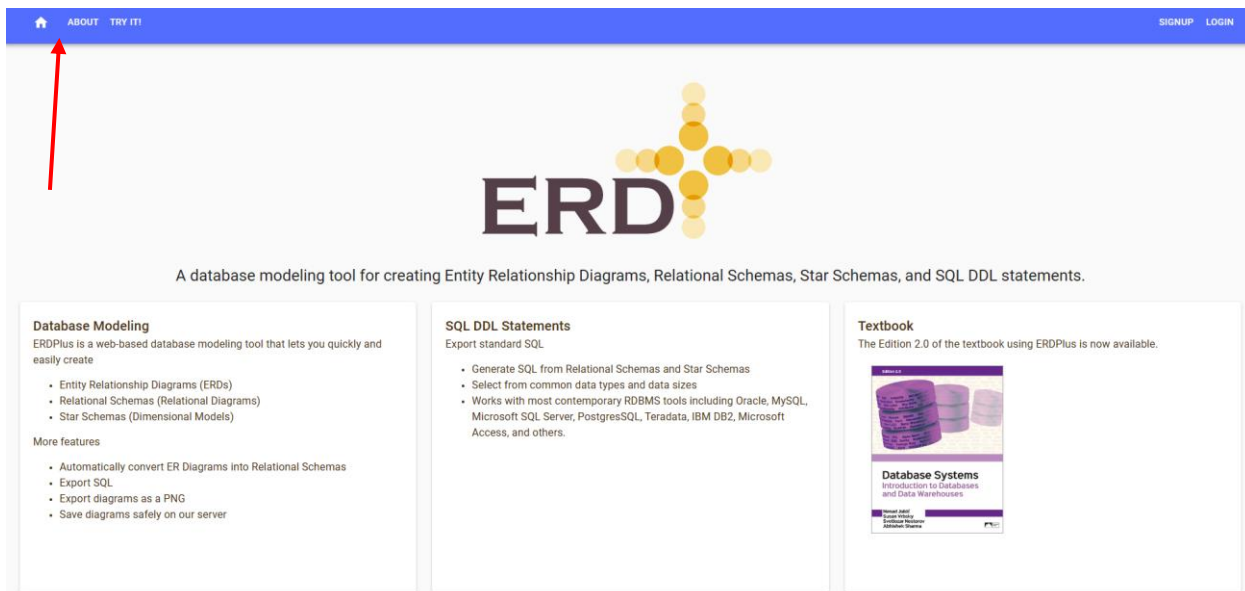


Рисунок 4.1 – Начало работы с ERDPlus

Вот подробное описание работы в ERDPlus:

1. Начало работы:

Создание нового проекта: При запуске редактора ERDPlus, вам будет предложено создать новый проект. Это может быть новая ER-диаграмма, диаграмма сущность-связь, или диаграмма классов.

Выбор типа диаграммы: в верхнем меню вы можете выбрать тип диаграммы, которую хотите создать. ERDPlus поддерживает ER-диаграммы, UML-диаграммы, и диаграммы классов.

2. Интерфейс и основные инструменты:

Меню и панель инструментов: в верхней части окна редактора расположена панель инструментов, содержащая основные функции для работы с диаграммами: “CONNECT”, “ENTITY”, “ATTRIBUTE”, “RELATIONSHIP”, и другие представлены на рисунке 4.2.



Рисунок 4.2 – Основные инструменты

Рабочее поле: Центральное рабочее поле предназначено для размещения элементов диаграммы. Здесь вы можете создавать и редактировать элементы, а также настраивать их взаимосвязи.

Контекстное меню и свойства элемента: Щелкнув правой кнопкой мыши на элементе диаграммы, вы откроете контекстное меню с дополнительными опциями. В правой части экрана отображаются свойства выбранного элемента, такие как имя, тип, атрибуты и связи.

3. Создание и редактирование элементов:

Добавление сущностей: Для добавления новой сущности выберите “ENTITY” на панели инструментов и щелкните на рабочем поле. В появившемся окне введите имя сущности и её атрибуты. Атрибуты могут быть простыми или составными, а также обязательными или необязательными.

Добавление атрибутов: Выберите “ATTRIBUTE” и щелкните на сущности, чтобы добавить атрибут. Вы можете задать тип данных, ключевые характеристики (например, первичный ключ, уникальный ключ), а также другие свойства.

Создание связей: Для создания связей между сущностями выберите “RELATIONSHIP” и соедините две сущности. Укажите тип связи (один к одному, один ко многим, многие ко многим) и при необходимости добавьте атрибуты связи.

4. Настройка и оптимизация диаграммы:

Организация элементов: Перетаскивайте элементы по рабочему полю для лучшей организации диаграммы. Используйте сетку и направляющие для выравнивания элементов.

Редактирование свойств: Щелкнув на элемент, вы можете изменить его свойства в правой части экрана. Это позволяет быстро вносить изменения и оптимизировать диаграмму.

5. Сохранение и экспорт:

Сохранение проекта: ERDPlus автоматически сохраняет ваш проект, но также вы можете вручную сохранить его в облако или локально на компьютер.

Экспорт диаграммы: Для экспорта диаграммы используйте опцию “Export” в

меню. Доступны различные форматы, включая PDF, PNG, и специализированные форматы для дальнейшего импорта в другие приложения.

6. Импорт и работа с существующими диаграммами:

Импорт диаграммы: Если у вас уже есть диаграммы, созданные в других инструментах или ранее сохраненные проекты ERDPlus, вы можете импортировать их для дальнейшей работы. Используйте опцию “Import” в меню и выберите нужный файл.

7. Дополнительные функции и расширенные возможности (рисунок 4.3:

Диаграммы классов UML: ERDPlus также поддерживает создание UML-диаграмм классов, что полезно для моделирования объектно-ориентированных систем.

Диаграммы многозначных связей: Вы можете создавать диаграммы многозначных связей, которые позволяют моделировать более сложные отношения между сущностями.

Совместная работа: ERDPlus поддерживает функции совместной работы, что позволяет нескольким пользователям редактировать одну диаграмму в реальном времени.

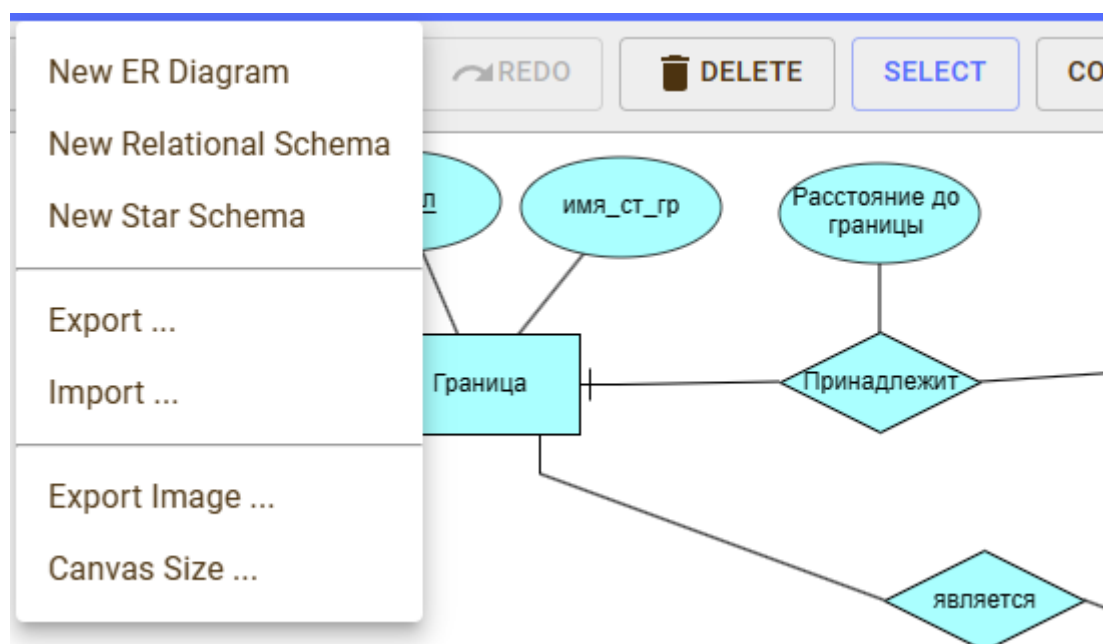


Рисунок 4.3 – Дополнительные параметры

ERDPlus является мощным инструментом для создания и управления диаграммами баз данных. Благодаря интуитивно понятному интерфейсу и широкому набору функций, он подходит как для новичков, так и для опытных пользователей, позволяя эффективно моделировать и визуализировать структуры данных.

На рисунке 4.4 показана разработанная ER-модель базы данных в нотации Чена в предметной области «Банкоматы».

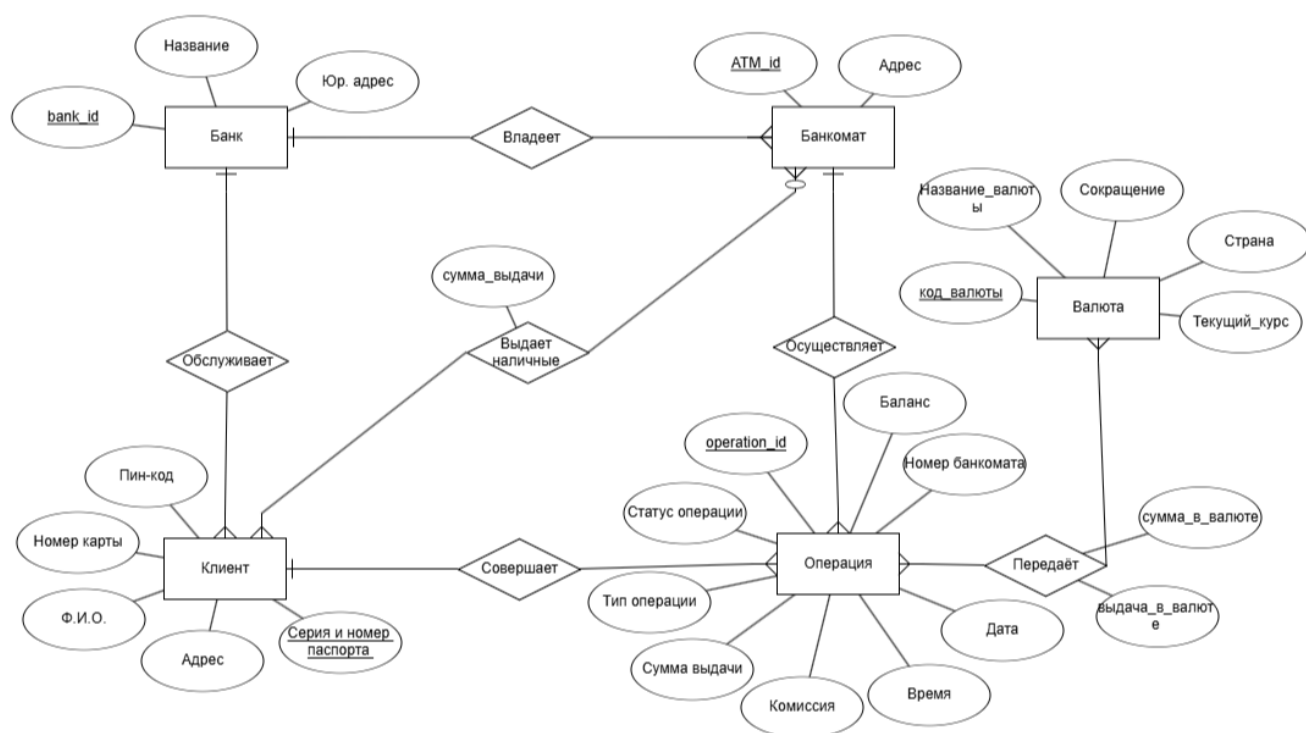


Рисунок 4.4 - ER-модель базы данных в нотации Чена

5 РАЗРАБОТКА ER-МОДЕЛИ БАЗЫ ДАННЫХ В НОТАЦИИ IDEFIX

5.1 Erwin Data Modeler

ER-модель базы данных в нотации IDEFIX была разработана в двух программах: Erwin и Draw.io.

Работа с Erwin Data Modeler включает несколько ключевых шагов, необходимых для создания и редактирования концептуальной модели данных. Вот более подробное описание процесса работы с этим инструментом:

1. Запуск программы:

Перейдите в меню "Пуск" -> "Программы" -> "CA" -> "AllFusion" -> "ERwin Data Modeler r7" -> "ERwin Data Modeler r7".

При появлении диалогового окна с подсказками "AllFusion ERwin Data Modeler Tips" нажмите на кнопку "Close", чтобы закрыть его.

2. Создание новой модели:

Начните с создания новой модели данных. Это можно сделать через меню "File" -> "New" или используя соответствующую иконку на панели инструментов.

3. Добавление сущностей:

Выберите инструмент для создания сущностей. В Erwin Data Modeler это вторая иконка на панели инструментов (обычно изображена в виде прямоугольника или другой фигуры, обозначающей сущность).

Щелкните в рабочей области модели, чтобы добавить сущность. Вы можете перемещать и изменять размер сущности по мере необходимости (рисунок 5.1.1).



Рисунок 5.1.1 – Создание сущности

4. Настройка сущностей:

Дважды щелкните на созданной сущности, чтобы открыть окно свойств.

Введите название сущности на русском языке, чтобы оно было понятно специалистам предметной области.

На этом этапе атрибуты сущностей на концептуальном уровне не добавляются,

но можно указать общие свойства, такие как описание и бизнес-правила.

5. Создание связей между сущностями:

Для добавления связи выберите один из инструментов для создания связей на панели инструментов (обычно это одна из трех последних иконок, например, линия или стрелка).

Щелкните на сущности, с которой хотите начать связь, и протяните линию до другой сущности, чтобы создать связь.

Дважды щелкните на созданной связи, чтобы открыть её свойства. Здесь можно указать тип связи (например, один к одному, один ко многим), а также дополнительные характеристики, такие как обязательность и кардинальность.

6. Добавление атрибутов:

Хотя атрибуты обычно не рассматриваются на концептуальном уровне, их можно добавить на этапе детального проектирования.

Дважды щелкните на сущности, чтобы открыть её свойства, и перейдите на вкладку "Attributes".

Добавьте новый атрибут, указав его название и тип данных. Например, можно выбрать тип данных из предустановленного списка или задать пользовательский тип данных, как показано на рисунке 5.1.2.

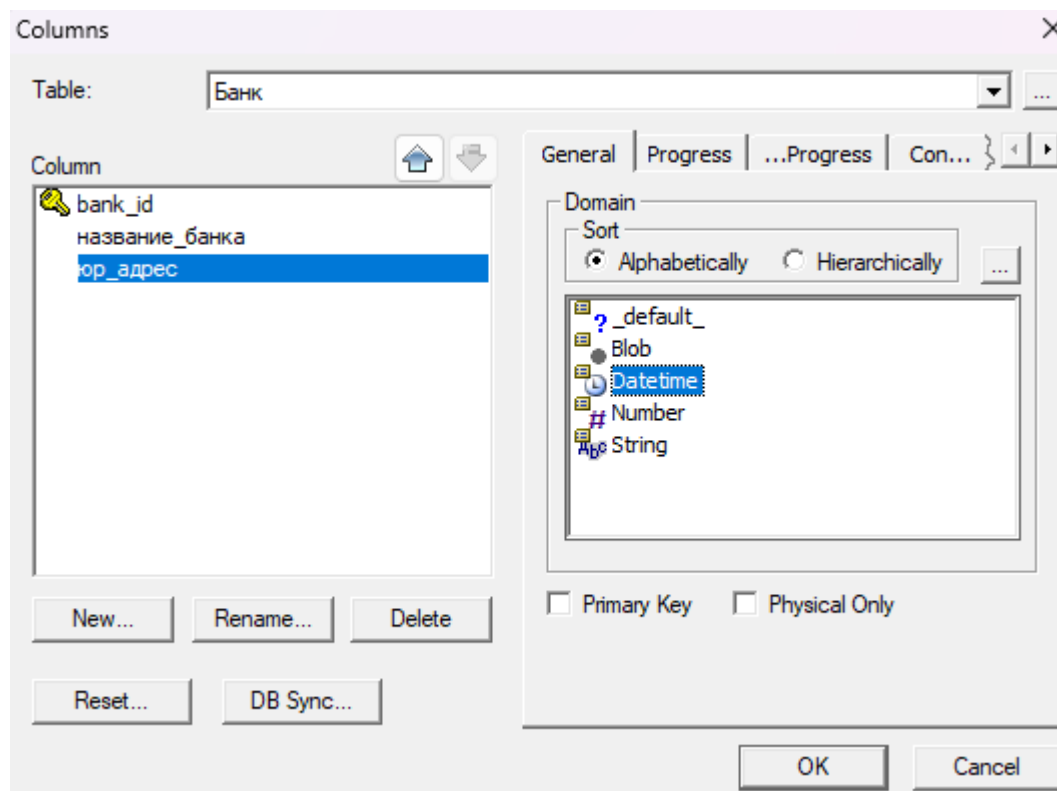


Рисунок 5.1.2 – Добавление атрибутов

7. Документирование модели:

Используйте встроенные возможности Erwin Data Modeler для документирования вашей модели. Можно добавить комментарии, описания, бизнес-правила и другие метаданные, чтобы сделать модель понятной для всех участников проекта.

8. Сохранение и экспорт модели:

Сохраните модель через меню "File" -> "Save" или "Save As".

Erwin Data Modeler позволяет экспортировать модель в различные форматы для интеграции с другими инструментами или для создания документации.

Советы и рекомендации:

Именованное: Убедитесь, что используете понятные и описательные названия для всех сущностей и связей, чтобы облегчить понимание модели всеми участниками проекта.

Консультации: Регулярно консультируйтесь с предметными специалистами, чтобы убедиться, что модель точно отражает требования бизнеса и специфику

предметной области.

Версионность: Ведите учет версий вашей модели, чтобы отслеживать изменения и при необходимости возвращаться к предыдущим версиям.

Работа с Erwin Data Modeler требует внимательности и тщательного планирования, чтобы создать эффективную и точную модель данных, которая будет служить основой для дальнейшего проектирования и разработки информационных систем.

На рисунке 5.1.3 показана разработанная ER-модель базы данных в нотации IDEFIX в предметной области «Банкоматы», созданная при помощи Erwin.

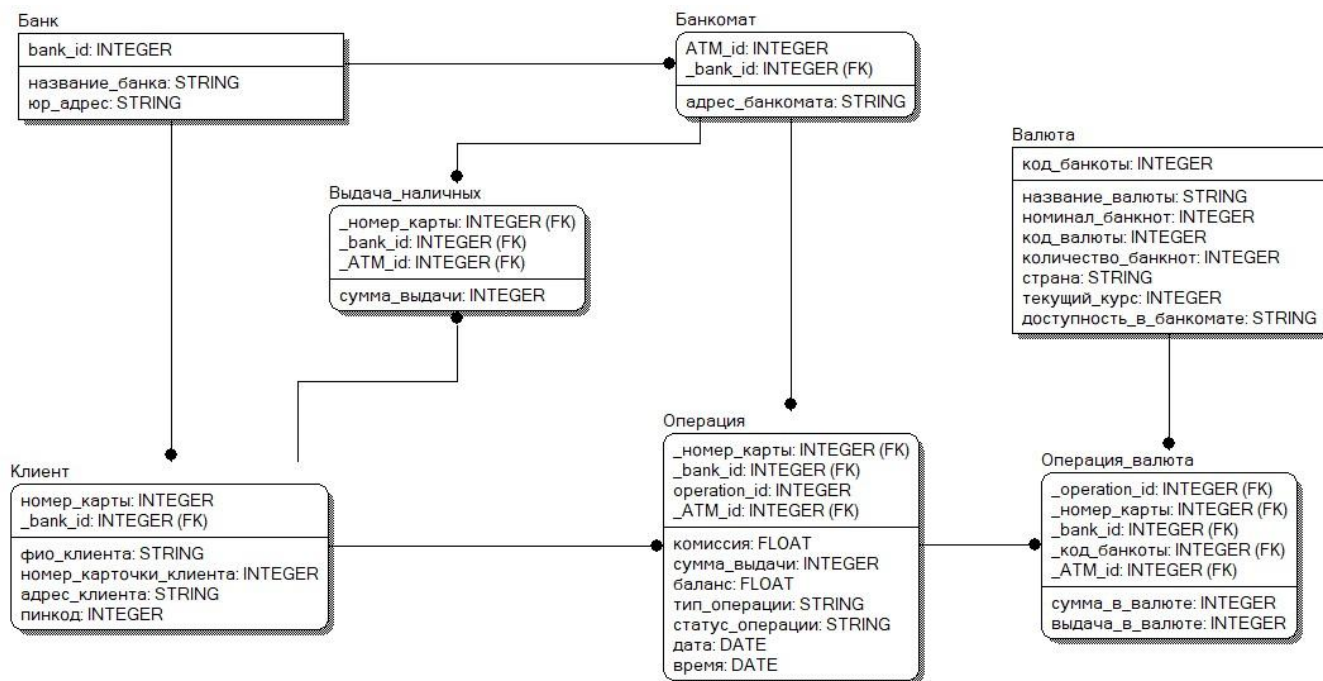


Рисунок 5.1.3 - ER-модель базы данных в нотации IDEFIX, созданная при помощи Erwin

5.2 Draw.io

Создание схемы в нотации IDEFIX в draw.io включает несколько шагов, от запуска программы до создания и настройки сущностей и связей. Вот подробное описание процесса:

1. Запуск программы:

Откройте браузер и перейдите на сайт draw.io.

Выберите опцию "Create New Diagram" (Создать новую диаграмму) или загрузите существующую диаграмму, если уже работали с проектом ранее.

Выберите "Blank Diagram" (Пустая диаграмма) и задайте имя файла. Нажмите "Create" (Создать).

2. Выбор шаблона или библиотеки:

Для работы с нотацией IDEFIX, вам понадобится соответствующая библиотека фигур. В левом меню нажмите на "More Shapes" (Больше фигур).

В появившемся окне найдите и выберите библиотеку фигур, соответствующую IDEFIX, если она доступна, или используйте основные фигуры для создания необходимых элементов вручную. Нажмите "Apply" (Применить).

3. Добавление сущностей:

Выберите прямоугольник или другую подходящую фигуру из библиотеки и перетащите её на рабочую область.

Дважды щелкните по фигуре, чтобы ввести название сущности. Для лучшего понимания, используйте русскоязычные названия, понятные специалистам предметной области.

4. Настройка сущностей:

После добавления фигуры, вы можете настроить её внешний вид, изменив размеры, цвет и другие параметры. Используйте правую панель для настройки стиля.

5. Добавление связей:

Выберите линию или стрелку из библиотеки фигур и перетащите её на рабочую область.

Соедините две сущности, щелкнув на одну и перетащив линию к другой.

Дважды щелкните на линии, чтобы добавить текст и описать тип связи (например, "один к одному", "один ко многим").

6. Добавление атрибутов:

Для добавления атрибутов к сущностям, вы можете использовать текстовые блоки или аннотации.

Дважды щелкните на сущности, чтобы добавить атрибуты, такие как название атрибута и его тип данных. Атрибуты могут быть перечислены внутри или рядом с

сущностью, в зависимости от вашего предпочтения.

7. Документирование схемы:

Добавляйте текстовые блоки и аннотации для описания сущностей, связей и бизнес-правил. Это поможет сделать вашу схему понятной для всех участников проекта.

Используйте различные цвета и стили для выделения важных элементов и улучшения читабельности схемы.

8. Сохранение и экспорт схемы:

Сохраните схему через меню "File" -> "Save" или "Save As". Вы можете сохранить файл на компьютер, в облако или экспортировать в различные форматы, такие как PNG, PDF или SVG.

Для совместной работы можно сохранить схему в Google Drive или других облачных сервисах, поддерживаемых draw.io.

Именованное: Убедитесь, что используете понятные и описательные названия для всех сущностей и связей, чтобы облегчить понимание схемы всеми участниками проекта.

Консультации: Регулярно консультируйтесь с предметными специалистами, чтобы убедиться, что схема точно отражает требования бизнеса и специфику предметной области.

Версионность: Ведите учет версий вашей схемы, чтобы отслеживать изменения и при необходимости возвращаться к предыдущим версиям.

Работа с draw.io предоставляет гибкость и простоту использования, что делает его отличным инструментом для создания схем в нотации IDEFIX, позволяя эффективно визуализировать структуру данных и их взаимосвязи.

На рисунке 5.2.1 показана разработанная ER-модель базы данных в нотации IDEFIX в предметной области «Банкоматы», созданная при помощи Draw.io.

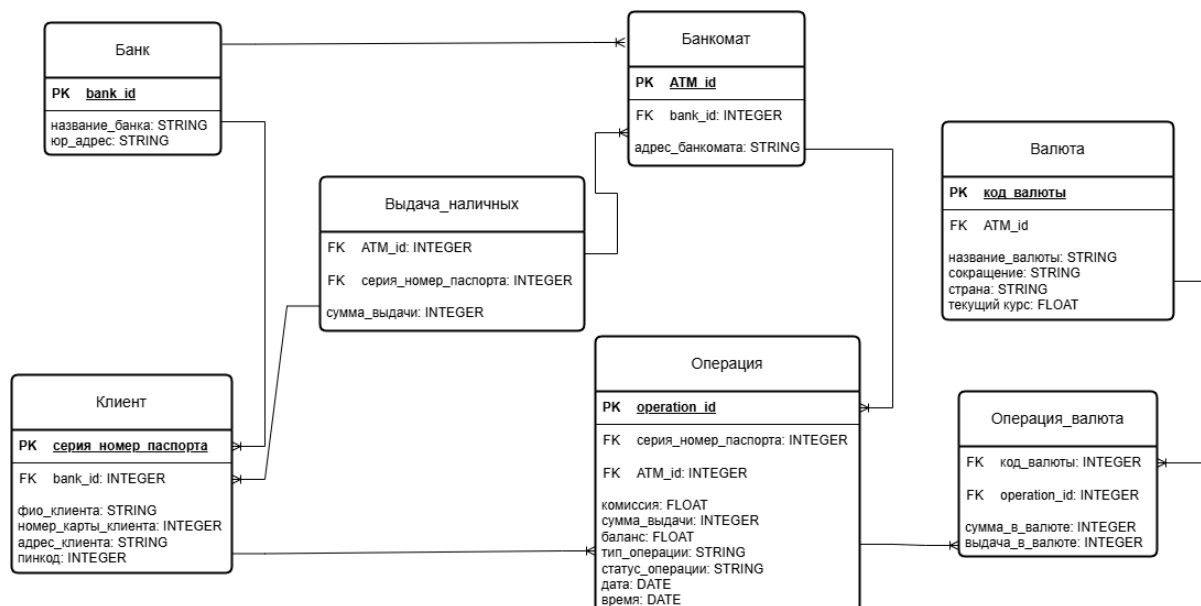


Рисунок 5.2.1 - ER-модель базы данных в нотации IDEFIX, созданная при помощи Draw.io

6 РАЗРАБОТКА РЕЛЯЦИОННОЙ МОДЕЛИ БАЗЫ ДАННЫХ И ЗАПРОСОВ К БАЗЕ ДАННЫХ

Требуется написать 5 запросов на поиск информации из базы данных на языке реляционной алгебры и языке реляционного исчисления:

Запрос на поиск информации из таблицы по некоторому условию с выдачей некоторого подмножества атрибутов таблицы (1 запрос);

Запрос, выполняющий поиск в двух таблицах, по некоторым условиям с выдачей некоторого подмножества атрибутов (2 запроса);

Запрос, выполняющий поиск в трех таблицах, по некоторым условиям с выдачей некоторого подмножества атрибутов (2 запроса);

Написать по одному запросу на добавление, удаление и корректировку информации в таблице на языке реляционной алгебры и языке реляционного исчисления.

Запросы:

1. Запрос на поиск информации из таблицы по некоторому условию с выдачей некоторого подмножества атрибутов таблицы (1 запрос):

Запрос – Вывести курс валюты США и Евро.

Язык реляционной алгебры:

$W = \pi_{\text{(текущий_курс)}}(\sigma_{\text{(название_валюты='Доллар США' } \vee \text{ название_валюты='Евро')}}(\text{Валюта}))$

Язык реляционного исчисления:

GET W (Валюта): (название_валюты = 'Доллар США' \vee название_валюты = 'Евро')

2. Запрос, выполняющий поиск в двух таблицах, по некоторым условиям с выдачей некоторого подмножества атрибутов (2 запроса):

2.1 Запрос – Вывести баланс клиента с ФИО «Сомойлин Ян Максимович».

Язык реляционной алгебры:

$W = \pi_{\text{(баланс)}}(\sigma_{\text{(Клиент.фio_клиента='Сомойлин Ян Максимович')}}(\text{Клиент} \bowtie \text{Операция}))$

Язык реляционного исчисления:

RANGE Операция ОПХ

RANGE Клиент ККХ

GET W (ОПХ.БАЛАНС): ЭККХ (ККХ.ФИО_КЛИЕНТА = "Сомойлин Ян Максимович" & ОПХ.СЕРИЯ_НОМЕР_ПАСПОРТА = ККХ.СЕРИЯ_НОМЕР_ПАСПОРТА)

2.2 Запрос – Вывести все адреса банкоматов, которые привязаны к банку «Сбербанк».

Язык реляционной алгебры:

$W = \pi_{\text{адрес_банкомата}} (\sigma_{\text{название_банка} = 'Сбербанк'} (\text{Банк} \bowtie \text{Банкомат}))$

Язык реляционного исчисления:

RANGE Банкомат БМХ

RANGE Банк БНХ

GET W (БМХ.адрес_банкомата): ЭБНХ (БНХ.название_банка = "Сбербанк" & БНХ.bank_id = БМХ.bank_id)

3. Запрос, выполняющий поиск в трех таблицах, по некоторым условиям с выдачей некоторого подмножества атрибутов (2 запроса):

3.1 Запрос – Вывести сумму снятия и адрес банкомата, в котором Крючкова Елизавета Андреевна снимала наличные.

Язык реляционной алгебры:

$W = \pi_{\text{сумма_выдачи, адрес_банкомата}} (\sigma_{\text{фio_клиента} = 'Крючкова Елизавета Андреевна' \& \text{тип_операции} = 'Выдача наличных'} (\text{Операция} \bowtie \text{Клиент} \bowtie \text{Банкомат}))$

Язык реляционного исчисления:

RANGE Операция ОПХ

RANGE Клиент КЛХ

RANGE Банкомат БМХ

GET W (ОПХ.сумма_выдачи, БМХ.адрес_банкомата): ЭКЛХ
(КЛХ.фio_клиента='Крючкова Елизавета Андреевна' &
ОПХ.тип_операции='Выдача наличных' &
ОПХ.серия_номер_паспорта=КЛХ.серия_номер_паспорта &

ОПХ.АТМ_id=БМХ.АТМ_id)

3.2. Запрос – Вывести всех клиентов из «Тинькофф», которые проводили операции на сумму более 10000.

Язык реляционной алгебры:

$W = \pi_{(Клиент.фio_клиента)} (\sigma_{(Банк.название_банка = 'Тинькофф' \& Операция.сумма_выдачи > 10000)} (Клиент \bowtie Банк \bowtie Операция))$

Язык реляционного исчисления:

RANGE Клиент КЛХ

RANGE Операция ОПХ

RANGE Банк БНХ

GET W (КЛХ.фio_клиента): ЭОПХ ЭБНХ (БНХ.название_банка = "Тинькофф" & КЛХ.bank_id = БНХ.bank_id & КЛХ.серия_номер_паспорта = ОПХ.серия_номер_паспорта & ОПХ.сумма_выдачи > 10000)

4. Запрос на добавление информации в таблицу (1 запрос):

Запрос - Добавить новый вид валюты с кодом валюты 1131, названием «Грузинский Лари», сокращением «GEL», страной «Грузия» и текущим курсом «31.24».

Язык реляционной алгебры:

$W = ((1131, "Грузинский Лари", "GEL", "Грузия", 31.24))$

Валюта = Валюта \cup W

Язык реляционного исчисления:

GET W (ВДХ):

W.КОД_ВАЛЮТЫ = 1131

W.НАЗВАНИЕ_ВАЛЮТЫ = "Грузинский Лари"

W.СОКРАЩЕНИЕ = "GEL"

W.СТРАНА = "Грузия"

W.ТЕКУЩИЙ_КУРС = 31.24

PUT W (АХ)

5. Запрос на удаление информации в таблице (1 запрос):

Запрос - Удалить банк «МТС-Банк» из списка всех банков.

Язык реляционной алгебры:

$\text{Банк} = \text{Банк} - \sigma_{(\text{название_банка} = \text{«МТС-Банк»})}(\text{Банк})$

Язык реляционного исчисления:

RANGE Банк ВХ

HOLD W (ВХ): $(\text{ВХ.название_банка} = \text{"МТС-Банк"})$

DELETE W

6. Запрос на корректировку информации в таблице (1 запрос):

Запрос - Изменить название банка «Тинькофф» на «Т-Банк».

Язык реляционной алгебры:

$W1 = \sigma_{(\text{название_банка} = \text{'Тинькофф'})} \text{Банк}$

$W2 = \pi_{(\text{bank_id}, \text{'Т-Банк'} \text{ as } \text{название_банка}, \text{юр_адрес})} W1$

$\text{Банк} = (\text{Банк} - W1) \cup W2$

Язык реляционного исчисления:

RANGE Банк ВЛХ

HOLD W (ВЛХ): $\text{ВЛХ.название_банка} = \text{'Тинькофф'}$

$\text{W.название_банка} = \text{'Т-Банк'}$

UPDATE W (ВЛХ)

7 РАЗРАБОТКА РЕЛЯЦИОННОЙ МОДЕЛИ БАЗЫ ДАННЫХ И ЗАПРОСОВ К БАЗЕ ДАННЫХ

Требуется разработать реляционную модель базы данных в соответствии ER-моделью базы данных. Работа выполнялась в программе pgAdmin 4. Версия PostgreSQL 16

7.1 Таблица «Банк»

CREATE запрос:

```
CREATE TABLE Банк (  
    bank_id INT PRIMARY KEY,  
    название_банка VARCHAR(255) NOT NULL,  
    юр_адрес VARCHAR(255) NOT NULL  
);
```

INSERT запрос:

```
INSERT INTO Банк (bank_id, название_банка, юр_адрес)  
VALUES  
(1, 'Сбербанк', 'ул. Вавилова, д.19, г. Москва'),  
(2, 'Тинькофф', '2-я Хуторская, д.38А, стр. 26, г. Москва'),  
(3, 'Альфа-Банк', 'ул. Каланчевская, д.27, г. Москва'),  
(4, 'ВТБ', 'ул. Воронцовская, д.43, стр.1, г. Москва'),  
(5, 'Райффайзенбанк', 'Смоленская-Сенная площадь, д.28, г. Москва'),  
(6, 'Россельхозбанк', 'Тагаринский пер., д.3, г. Москва'),  
(7, 'Почта-Банк', 'ул. Преображенская Пл., д. 8, г. Москва'),  
(8, 'Газпромбанк', 'ул. Наметкина, д.16, корпус 1, г. Москва'),  
(9, 'Банк Зенит', 'ул. Одесская, д.2, г. Москва'),  
(10, 'Промсвязьбанк', 'ул. Смирновская, д.10, стр.22, г. Москва'),  
(11, 'Совкомбанк', 'пр. Текстильщиков, д. 46, г. Кострома'),  
(12, 'Точка', '3-й Крутицкий пер., д.11, г. Москва'),  
(13, 'Уралсиб', 'ул. Ефремова, д.8, г. Москва'),  
(14, 'МТС Банк', 'Проспект Андропова, д. 18, корп.1, г. Москва'),  
(15, 'Хоум Банк', 'ул. Правды, д.8, кор.1, г. Москва');
```

Наполнение таблицы «Банк» изображено на рисунке 7.1.1.

Query

Query History

1

SELECT * FROM public."Банк"

2

ORDER BY bank_id ASC

Data Output

Messages

Notifications

≡

📄

▼

📋

▼

🗑

🗄

⬇

📈

	bank_id [PK] integer	название_банка character varying (255)	юр_адрес character varying (255)
1	1	Сбербанк	ул. Вавилова, д.19, г. Москва
2	2	Тинькофф	2-я Хуторская, д.38А, стр. 26, г. Москва
3	3	Альфа-Банк	ул. Каланчевская, д.27, г. Москва
4	4	ВТБ	ул. Воронцовская, д.43, стр.1, г. Москва
5	5	Райффайзенбанк	Смоленская-Сенная площадь, д.28, г. Моск...
6	6	Россельхозбанк	Гагаринский пер., д.3, г. Москва
7	7	Почта-Банк	ул. Преображенская Пл., д. 8, г. Москва
8	8	Газпромбанк	ул. Наметкина, д.16, корпус 1, г. Москва
9	9	Банк Зенит	ул. Одесская, д.2, г. Москва
10	10	Промсвязьбанк	ул. Смирновская, д.10, стр.22, г. Москва
11	11	Совкомбанк	пр. Текстильщиков, д. 46, г. Кострома
12	12	Точка	3-й Крутицкий пер., д.11, г. Москва
13	13	Уралсиб	ул. Ефремова, д.8, г. Москва
14	14	МТС Банк	Проспект Андропова, д. 18, корп.1, г. Москва
15	15	Хоум Банк	ул. Правды, д.8, кор.1, г. Москва

Рисунок 7.1.1 – Таблица «Банк»

7.2 Таблица «Клиент»

CREATE запрос:

CREATE TABLE Клиент (

серия_номер_паспорта VARCHAR(20) PRIMARY KEY,

фio_клиента VARCHAR(255) NOT NULL,

адрес_клиента VARCHAR(255) NOT NULL,

пинкод VARCHAR(4) NOT NULL,

bank_id INT,

FOREIGN KEY (bank_id) REFERENCES Банк(bank_id)

);

INSERT запрос:

```

INSERT INTO Клиент (серия_номер_паспорта, фио_клиента, адрес_клиента,
пинкод, bank_id)
VALUES
('1234567890', 'Макарова Элина Ивановна', 'г. Калининград, ул. Ленина, д.10', '9999',
1),
('5672759105', 'Иванова Дарина Станиславовна', 'г. Москва, ул. Петра Романова,
д.13', '5469', 4),
('4518736666', 'Сомойлин Ян Максимович', 'г. Москва, ул. 8 марта, д.44', '6401', 3),
('4285492056', 'Степанов Константин Михайлович', 'г. Санкт-Петербург, ул. Германа
Титова, д.20', '7373', 2),
('1046802548', 'Калмыкова Эмилия Романовна', 'г. Нижний Новгород, пер. Ладыгина,
42', '9508', 7),
('9347823910', 'Спиридонова Злата Константиновна', 'г. Ростов, ул. Ломоносова, 86',
'3609', 13),
('1933479240', 'Астахова Полина Владимировна', 'г. Тула, пер. Бухарестская, 27',
'7498', 1),
('8234842101', 'Кузнецов Кирилл Максимович', 'г. Киров, пл. Бухарестская, 71',
'3456', 1),
('4859910202', 'Михайлов Денис Артёмович', 'г. Псков, шоссе Домодедовская, 36',
'5891', 12),
('7728193485', 'Морозов Александр Иванович', 'г. Москва, Холмовская Улица, д. 31',
'9268', 13),
('8346290157', 'Борисов Михаил Константинович', 'г. Брянск, наб. Ломоносова, 41',
'6789', 1),
('9283663777', 'Медведев Михаил Никитич', 'г. Подольск, Озерная ул., д. 13', '2191',
13),
('6274194738', 'Потапов Андрей Михайлович', 'г. Чебоксары, Социалистическая ул.,
д. 4', '6821', 12),
('9927126483', 'Шувалова Екатерина Петровна', 'г. Мурманск, Заслонова ул., д. 21',
'8413', 10),
('7341927622', 'Тарасов Эмир Владимирович', 'г. Тула, Советский пер., д. 5', '3615', 6),
('3296578947', 'Трофимова Полина Кирилловна', 'г. Рязань, Шоссейная ул., д. 6',
'1921', 11),
('6627478924', 'Крючкова Елизавета Андреевна', 'г. Томск, Цветочная ул., д. 23',
'4189', 9),
('5466494853', 'Чернышева Ксения Савельевна', 'г. Самара, Рабочая ул., д. 17', '5477',
4),
('3784492056', 'Тарасова Василиса Андреевна', 'г. Чита, Пролетарская ул., д. 9', '5924',
4),
('5536384656', 'Моисеев Михаил Артёмович', 'г. Рыбинск, Полевая ул., д. 5', '2581',
15);

```

Наполнение таблицы «Клиент» изображено на рисунке 7.2.1.

Query

Query History

Scratch F

1 SELECT * FROM public."Клиент"

2 ORDER BY "серия_номер_паспорта" ASC

Data Output

Messages

Notifications

	серия_номер_паспорта [PK] character varying (20)	фio_клиента character varying (255)	адрес_клиента character varying (255)	пинкод character varying (4)	bank_id integer
1	1046802548	Калмыкова Эмилия Романовна	г. Нижний Новгород, пер. Ладыгина, 42	9508	7
2	1234567890	Макарова Элина Ивановна	г. Калининград, ул. Ленина, д.10	9999	1
3	1933479240	Астахова Полина Владимировна	г. Тула, пер. Бухарестская, 27	7498	1
4	3296578947	Трофимова Полина Кирилловна	г. Рязань, Шоссейная ул., д. 6	1921	11
5	3784492056	Тарасова Василиса Андреевна	г. Чита, Пролетарская ул., д. 9	5924	4
6	4285492056	Степанов Константин Михайлович	г. Санкт-Петербург, ул. Германа Титова, д.20	7373	2
7	4518736666	Сомойлин Ян Максимович	г. Москва, ул. 8 марта, д.44	6401	3
8	4859910202	Михайлов Денис Артёмович	г. Псков, шоссе Домодедовская, 36	5891	12
9	5466494853	Чернышева Ксения Савельевна	г. Самара, Рабочая ул., д. 17	5477	4
10	5536384656	Моисеев Михаил Артёмович	г. Рыбинск, Полевая ул., д. 5	2581	15
11	5672759105	Иванова Дарина Станиславовна	г. Москва, ул. Петра Романова, д.13	5469	4
12	6274194738	Потапов Андрей Михайлович	г. Чебоксары, Социалистическая ул., д. 4	6821	12
13	6627478924	Крюкова Елизавета Андреевна	г. Томск, Цветочная ул., д. 23	4189	9
14	7341927622	Тарасов Эмир Владимирович	г. Тула, Советский пер., д. 5	3615	6
15	7728193485	Морозов Александр Иванович	г. Москва, Холмовская Улица, д. 31	9268	13
16	8234842101	Кузнецов Кирилл Максимович	г. Киров, пл. Бухарестская, 71	3456	1
17	8346290157	Борисов Михаил Константинович	г. Брянск, наб. Ломоносова, 41	6789	1
18	9283663777	Медведев Михаил Никитич	г. Подольск, Озерная ул., д. 13		

Successfully run. Total query runtime: 1

✓ Successfully run. Total query runtime: 1

Рисунок 7.2.1 – Таблица «Клиент»

7.3 Таблица «Банкомат»

CREATE запрос:

```
CREATE TABLE Банкомат (
    ATM_id INT PRIMARY KEY,
    адрес_банкомата VARCHAR(255) NOT NULL,
    bank_id INT,
    FOREIGN KEY (bank_id) REFERENCES Банк(bank_id)
);
```

INSERT запрос:

```
INSERT INTO Банкомат (ATM_id, адрес_банкомата, bank_id)
VALUES
(1001, 'г. Москва, ул. Образцова, д.9', 1),
(2002, 'г. Казань, ул. Заря, д.20', 2),
(3003, 'г. Пермь, ул. 2-й Звонкий Переулок, д.21', 3),
(4004, 'г. Казань, ул. 22 Партсъезд, д.18', 4),
(9005, 'г. Ярославль, Березовая ул., д. 10', 9),
(1006, 'г. Севастополь, Строителей ул., д. 18', 1),
(4007, 'г. Грозный, Новый пер., д. 1', 4),
(5008, 'г. Ставрополь, Первомайская ул., д. 14', 5),
```

(1209, 'г. Якутск, Максима Горького ул., д. 5', 12),
 (6010, 'г. Химки, Полевой пер., д. 11', 6),
 (1011, 'г. Екатеринбург, Западная ул., д. 17', 1),
 (1512, 'г. Рязань, Максима Горького ул., д. 3', 15),
 (1013, 'г. Пенза, Калинина ул., д. 8', 10),
 (6014, 'г. Пермь, Заречный пер., д. 15', 6),
 (8015, 'г. Астрахань, Березовая ул., д. 19', 8),
 (7016, 'г. Якутск, Красноармейская ул., д. 5', 7),
 (3017, 'г. Самара, Озерный пер., д. 5', 3),
 (1318, 'г. Саранск, Дачная ул., д. 20', 13),
 (5019, 'г. Одинцово, Школьная ул., д. 22', 5),
 (8020, 'г. Москва, Дорожная ул., д. 25', 8);

Наполнение таблицы «Банкомат» изображено на рисунке 7.3.1.

Query Query History

```
1 SELECT * FROM public."Банкомат"
2 ORDER BY atm_id ASC
```

Data Output Messages Notifications

	atm_id [PK] integer	адрес_банкомата character varying (255)	bank_id integer
1	1001	г. Москва, ул. Образцова, д.9	1
2	1006	г. Севастополь, Строителей ул., д. 18	1
3	1011	г. Екатеринбург, Западная ул., д. 17	1
4	1013	г. Пенза, Калинина ул., д. 8	10
5	1209	г. Якутск, Максима Горького ул., д. 5	12
6	1318	г. Саранск, Дачная ул., д. 20	13
7	1512	г. Рязань, Максима Горького ул., д. 3	15
8	2002	г. Казань, ул. Заря, д.20	2
9	3003	г. Пермь, ул. 2-й Звонкий Переулок, д.21	3
10	3017	г. Самара, Озерный пер., д. 5	3
11	4004	г. Казань, ул. 22 Партсъезд, д.18	4
12	4007	г. Грозный, Новый пер., д. 1	4
13	5008	г. Ставрополь, Первомайская ул., д. 14	5
14	5019	г. Одинцово, Школьная ул., д. 22	5
15	6010	г. Химки, Полевой пер., д. 11	6
16	6014	г. Пермь, Заречный пер., д. 15	6
17	7016	г. Якутск, Красноармейская ул., д. 5	7
18	8015	г. Астрахань, Березовая ул., д. 19	8

Рисунок 7.3.1 – Таблица «Банкомат»

7.4 Таблица «Операция»

CREATE запрос:

```
CREATE TABLE Операция (  
    operation_id INT PRIMARY KEY,  
    дата DATE NOT NULL,  
    время TIME NOT NULL,  
    комиссия DECIMAL(10, 2) NOT NULL,  
    сумма_выдачи DECIMAL(10, 2) NOT NULL,  
    тип_операции VARCHAR(50) NOT NULL,  
    статус_операции VARCHAR(50) NOT NULL,  
    баланс DECIMAL(10, 2) NOT NULL,  
    серия_номер_паспорта VARCHAR(20),  
    ATM_id INT,  
    FOREIGN KEY (серия_номер_паспорта) REFERENCES  
        Клиент(серия_номер_паспорта),  
    FOREIGN KEY (ATM_id) REFERENCES Банкомат(ATM_id)  
);
```

INSERT запрос:

```
INSERT INTO Операция (operation_id, дата, время, комиссия, сумма_выдачи,  
тип_операции, статус_операции, баланс, серия_номер_паспорта, ATM_id)  
VALUES  
(1, '2024-05-20', '17:30:18', 3.00, 3000.00, 'Выдача наличных', 'Успешно', 31856.29,  
'1234567890', 1001),  
(2, '2024-05-21', '12:35:33', 5.50, 5500.00, 'Выдача наличных', 'Успешно', 146500.11,  
'5672759105', 4004),  
(3, '2024-05-22', '18:01:53', 10.00, 10000.00, 'Выдача наличных', 'Успешно', 302.73,  
'4518736666', 3003),  
(4, '2024-05-22', '21:13:26', 24.65, 24650.00, 'Выдача наличных', 'Успешно', 55600.11,  
'4285492056', 2002),  
(5, '2024-05-23', '14:35:48', 30.00, 30000.00, 'Выдача наличных', 'Успешно', 31856.29,  
'1046802548', 7016),  
(6, '2024-05-24', '07:12:55', 2.20, 2200.00, 'Выдача наличных', 'Успешно', 146500.11,  
'9347823910', 1318),  
(7, '2024-05-25', '22:13:14', 16.00, 16000.00, 'Выдача наличных', 'Успешно', 302.73,  
'1933479240', 1011),
```

(8, '2024-05-26', '17:53:16', 15.00, 15000.00, 'Выдача наличных', 'Успешно', 55600.11, '8234842101', 1001),
 (9, '2024-05-27', '11:40:03', 9.00, 9000.00, 'Выдача наличных', 'Успешно', 31856.29, '4859910202', 1209),
 (10, '2024-05-28', '16:14:19', 49.00, 49000.00, 'Выдача наличных', 'Успешно', 146500.11, '7728193485', 1318),
 (11, '2024-05-29', '06:01:59', 2.00, 2000.00, 'Выдача наличных', 'Успешно', 302.73, '8346290157', 1011),
 (12, '2024-05-30', '20:19:11', 21.00, 21000.00, 'Выдача наличных', 'Успешно', 55600.11, '9283663777', 1318),
 (13, '2024-05-31', '13:32:15', 3.60, 3600.00, 'Выдача наличных', 'Успешно', 31856.29, '6274194738', 1209),
 (14, '2024-05-31', '13:31:33', 42.00, 42000.00, 'Выдача наличных', 'Успешно', 146500.11, '9927126483', 1013),
 (15, '2024-06-01', '22:22:22', 7.00, 7000.00, 'Выдача наличных', 'Успешно', 302.73, '7341927622', 6010),
 (16, '2024-06-02', '10:03:46', 44.00, 44000.00, 'Выдача наличных', 'Успешно', 55600.11, '3296578947', 8020),
 (17, '2024-06-03', '09:21:19', 2.50, 2500.00, 'Выдача наличных', 'Успешно', 31856.29, '6627478924', 9005),
 (18, '2024-06-04', '14:38:23', 23.00, 23000.00, 'Выдача наличных', 'Успешно', 146500.11, '5466494853', 4007),
 (19, '2024-06-07', '20:10:40', 2.00, 2000.00, 'Выдача наличных', 'Успешно', 302.73, '3784492056', 4004),
 (20, '2024-06-12', '19:53:00', 38.00, 38000.00, 'Выдача наличных', 'Успешно', 55600.11, '5536384656', 1512);

Наполнение таблицы «Операция» изображено на рисунке 7.4.1.

Query

Query History

1

SELECT * FROM public."Операция"

2

ORDER BY operation_id ASC

Scratch Pad

X

Data Output

Messages

Notifications

	operation_id [PK] integer	data date	время time without time zone	комиссия numeric (10,2)	сумма_выдачи numeric (10,2)	тип_операции character varying (50)	статус_операции character varying (50)	баланс numeric (10,2)	серия_номер_паспорта character varying (20)
1	1	2024-05-20	17:30:18	3.00	3000.00	Выдача наличных	Успешно	31856.29	1234567890
2	2	2024-05-21	12:35:33	5.50	5500.00	Выдача наличных	Успешно	146500.11	5672759105
3	3	2024-05-22	18:01:53	10.00	10000.00	Выдача наличных	Успешно	302.73	4518736666
4	4	2024-05-22	21:13:26	24.65	24650.00	Выдача наличных	Успешно	55600.11	4285492056
5	5	2024-05-23	14:35:48	30.00	30000.00	Выдача наличных	Успешно	31856.29	1046802548
6	6	2024-05-24	07:12:55	2.20	2200.00	Выдача наличных	Успешно	146500.11	9347823910
7	7	2024-05-25	22:13:14	16.00	16000.00	Выдача наличных	Успешно	302.73	1933479240
8	8	2024-05-26	17:53:16	15.00	15000.00	Выдача наличных	Успешно	55600.11	8234842101
9	9	2024-05-27	11:40:03	9.00	9000.00	Выдача наличных	Успешно	31856.29	4859910202
10	10	2024-05-28	16:14:19	49.00	49000.00	Выдача наличных	Успешно	146500.11	7728193485
11	11	2024-05-29	06:01:59	2.00	2000.00	Выдача наличных	Успешно	302.73	8346290157
12	12	2024-05-30	20:19:11	21.00	21000.00	Выдача наличных	Успешно	55600.11	9283663777
13	13	2024-05-31	13:32:15	3.60	3600.00	Выдача наличных	Успешно	31856.29	6274194738
14	14	2024-05-31	13:31:33	42.00	42000.00	Выдача наличных	Успешно	146500.11	9927126483
15	15	2024-06-01	22:22:22	7.00	7000.00	Выдача наличных	Успешно	302.73	7341927622
16	16	2024-06-02	10:03:46	44.00	44000.00	Выдача наличных	Успешно	55600.11	3296578947
17	17	2024-06-03	09:21:19	2.50	2500.00	Выдача наличных	Успешно	31856.29	6627478924
18	18	2024-06-04	14:38:23	23.00	23000.00	Выдача наличных	Успешно	146500.11	5466494853

Рисунок 7.4.1 – Таблица «Операция»

7.5 Таблица «Выдача_наличных»

CREATE запрос:

```
CREATE TABLE Выдача_наличных (  
    сумма_выдачи DECIMAL(10, 2) NOT NULL,  
    серия_номер_паспорта VARCHAR(20),  
    ATM_id INT,  
    FOREIGN KEY (серия_номер_паспорта) REFERENCES  
Клиент(серия_номер_паспорта),  
    FOREIGN KEY (ATM_id) REFERENCES Банкомат(ATM_id)  
);
```

INSERT запрос:

```
INSERT INTO Выдача_наличных (сумма_выдачи, серия_номер_паспорта, ATM_id)  
VALUES  
(3000.00, '1234567890', 1001),  
(5500.00, '5672759105', 4004),  
(10000.00, '4518736666', 3003),  
(24650.00, '4285492056', 2002),  
(30000.00, '1046802548', 7016),  
(2200.00, '9347823910', 1318),  
(16000.00, '1933479240', 1011),  
(15000.00, '8234842101', 1001),  
(9000.00, '4859910202', 1209),  
(49000.00, '7728193485', 1318),  
(2000.00, '8346290157', 1011),  
(21000.00, '9283663777', 1318),  
(3600.00, '6274194738', 1209),  
(42000.00, '9927126483', 1013),  
(7000.00, '7341927622', 6010),  
(44000.00, '3296578947', 8020),  
(2500.00, '6627478924', 9005),  
(23000.00, '5466494853', 4007),  
(2000.00, '3784492056', 4004),  
(38000.00, '5536384656', 1512);
```

Наполнение таблицы «Выдача_наличных» изображено на рисунке 7.5.1.

Query

Query History

1

SELECT * FROM public."Выдача_наличных"

Data Output

Messages

Notifications

сумма_выдачи

numeric (10,2)

серия_номер_паспорта

character varying (20)

atm_id

integer

1

3000.00

1234567890

1001

2

5500.00

5672759105

4004

3

10000.00

4518736666

3003

4

24650.00

4285492056

2002

5

30000.00

1046802548

7016

6

2200.00

9347823910

1318

7

16000.00

1933479240

1011

8

15000.00

8234842101

1001

9

9000.00

4859910202

1209

10

49000.00

7728193485

1318

11

2000.00

8346290157

1011

12

21000.00

9283663777

1318

13

3600.00

6274194738

1209

14

42000.00

9927126483

1013

15

7000.00

7341927622

6010

16

44000.00

3296578947

8020

17

2500.00

6627478924

9005

18

23000.00

5466494853

4007

19

2000.00

3784492056

4004

Рисунок 7.5.1 – Таблица «Выдача_Наличных»

7.6 Таблица «Валюта»

CREATE запрос:

```
CREATE TABLE Валюта (
    код_валюты INT PRIMARY KEY,
    название_валюты VARCHAR(50) NOT NULL,
    сокращение VARCHAR(10) NOT NULL,
    страна VARCHAR(50) NOT NULL,
    текущий_курс DECIMAL(10, 2) NOT NULL
);
```

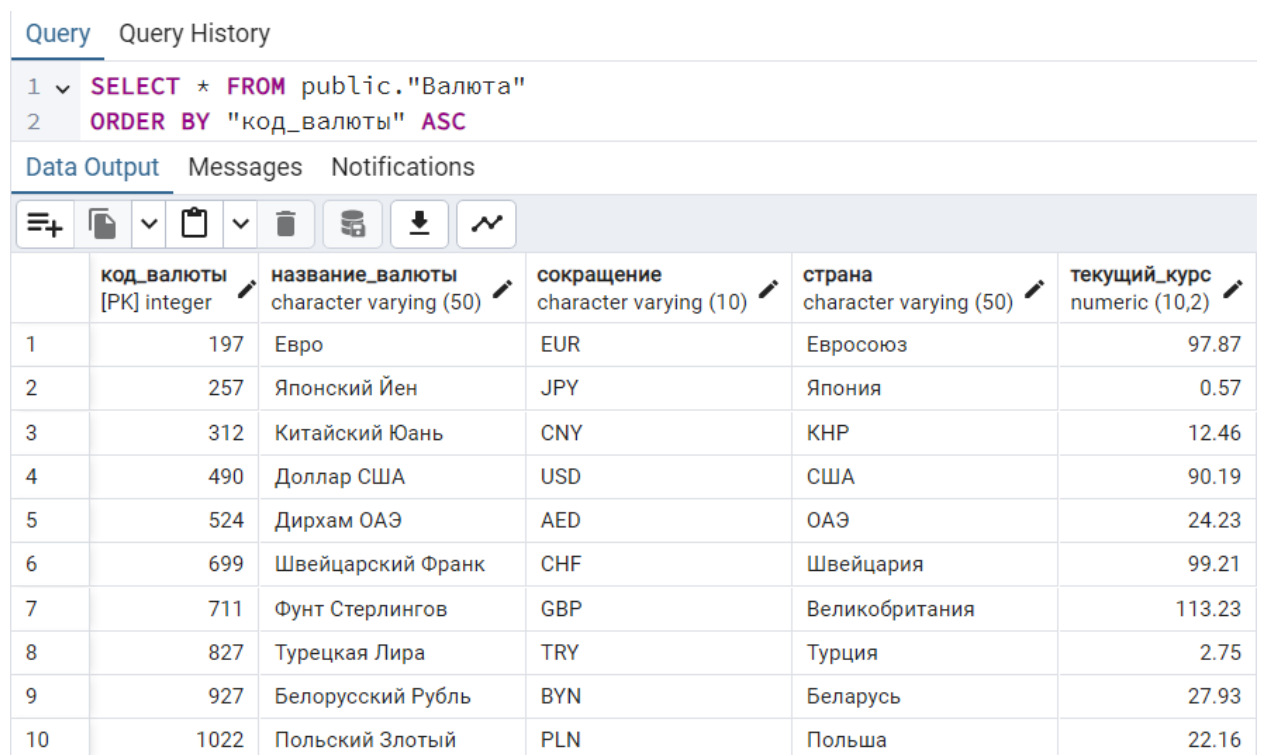

INSERT запрос:

INSERT INTO Валюта (код_валюты, название_валюты, сокращение, страна, текущий_курс)

VALUES

(197, 'Евро', 'EUR', 'Евросоюз', 97.87),
(257, 'Японский Йен', 'JPY', 'Япония', 0.57),
(312, 'Китайский Юань', 'CNY', 'КНР', 12.46),
(490, 'Доллар США', 'USD', 'США', 90.19),
(524, 'Дирхам ОАЭ', 'AED', 'ОАЭ', 24.23),
(699, 'Швейцарский Франк', 'CHF', 'Швейцария', 99.21),
(711, 'Фунт Стерлингов', 'GBP', 'Великобритания', 113.23),
(827, 'Турецкая Лира', 'TRY', 'Турция', 2.75),
(927, 'Белорусский Рубль', 'BYN', 'Беларусь', 27.93),
(1022, 'Польский Злотый', 'PLN', 'Польша', 22.16);

Наполнение таблицы «Валюта» изображено на рисунке 7.6.1.



Query		Query History			
1	SELECT	*	FROM	public."Валюта"	
2	ORDER BY	"код_валюты"	ASC		
Data Output		Messages			
	код_валюты [PK] integer	название_валюты character varying (50)	сокращение character varying (10)	страна character varying (50)	текущий_курс numeric (10,2)
1	197	Евро	EUR	Евросоюз	97.87
2	257	Японский Йен	JPY	Япония	0.57
3	312	Китайский Юань	CNY	КНР	12.46
4	490	Доллар США	USD	США	90.19
5	524	Дирхам ОАЭ	AED	ОАЭ	24.23
6	699	Швейцарский Франк	CHF	Швейцария	99.21
7	711	Фунт Стерлингов	GBP	Великобритания	113.23
8	827	Турецкая Лира	TRY	Турция	2.75
9	927	Белорусский Рубль	BYN	Беларусь	27.93
10	1022	Польский Злотый	PLN	Польша	22.16

Рисунок 7.6.1 – Таблица «Валюта»

7.7 Таблица «Операция_валюта»

CREATE запрос:

CREATE TABLE Операция_валюта (

operation_id INT,

код_валюты INT,

сумма_в_валюте DECIMAL(10, 2) NOT NULL,
 выдача_в_валюте BOOLEAN NOT NULL,
 FOREIGN KEY (operation_id) REFERENCES Операция(operation_id),
 FOREIGN KEY (код_валюты) REFERENCES Валюта(код_валюты),
 PRIMARY KEY (operation_id, код_валюты)
);

INSERT запрос:

```
INSERT INTO Операция_валюта (operation_id, код_валюты, сумма_в_валюте,
выдача_в_валюте)
VALUES
(1, 312, 3000.00, TRUE),
(6, 490, 2200.00, TRUE),
(11, 197, 2000.00, TRUE),
(13, 257, 3600.00, TRUE);
```

Наполнение таблицы «Операция_валюта» изображено на рисунке 7.7.1.

Query Query History

1 SELECT * FROM public."Операция_валюта"
 2 ORDER BY operation_id ASC, "код_валюты" ASC

Data Output Messages Notifications

	operation_id [PK] integer	код_валюты [PK] integer	сумма_в_валюте numeric (10,2)	выдача_в_валюте boolean
1	1	312	3000.00	true
2	6	490	2200.00	true
3	11	197	2000.00	true
4	13	257	3600.00	true

Рисунок 7.7.1 – Таблица «Операция_валюта»

7.8 ER-диаграмма

На рисунке 7.8.1 показана ER-диаграмма, сгенерированная через pgAdmin.

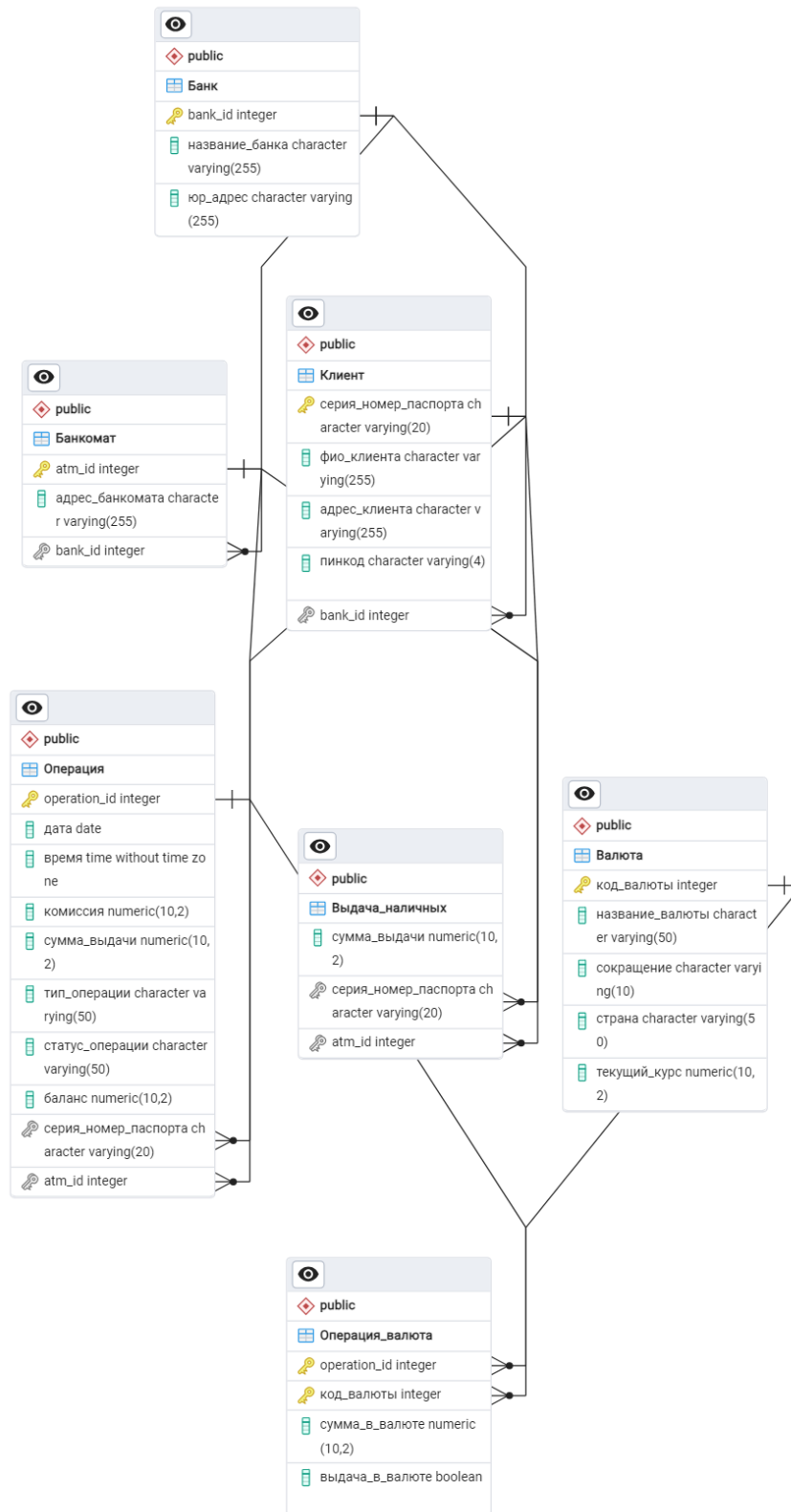


Рисунок 7.8.1 – ER-модель базы данных в pgAdmin

8 ЗАПРОСЫ НА ЯЗЫКЕ SQL

Написать запросы, описанные в пункте 6. При написании запросов на поиск должны быть реализованы следующие конструкции запросов:

- использование операторов WHERE и ORDER BY,
- выборка данных из нескольких таблиц с указанием условия соединения таблиц в операторе WHERE;
- выборка данных из нескольких таблиц с помощью соединения таблиц операторами JOIN;
- использование подзапросов с оператором IN;
- использование подзапросов с оператором EXISTS;
- использование функций агрегирования;
- использование оператора GROUP BY.

После выполнения запросов на удаление и корректировку убедиться, что запрос отработал правильно с помощью выполнения поискового запроса.

Запросы:

1. Запрос – Вывести курс валюты США и Евро.

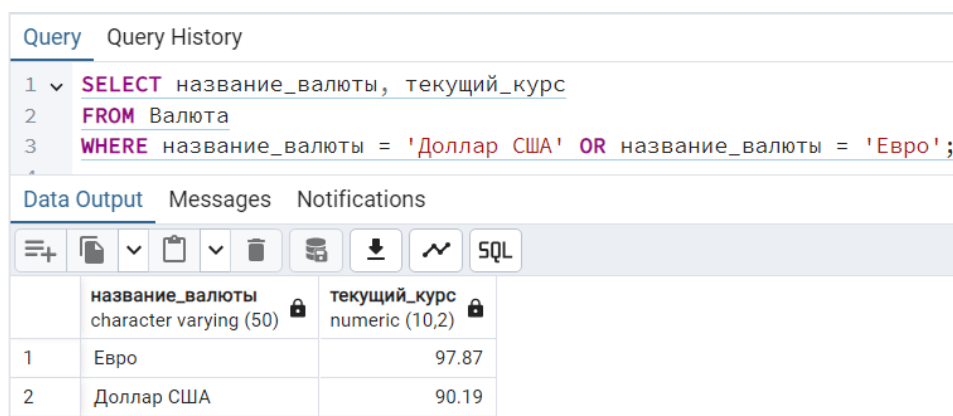
SQL запрос:

SELECT название_валюты, текущий_курс

FROM Валюта

WHERE название_валюты = 'Доллар США' OR название_валюты = 'Евро';

На рисунке 8.1 показан ответ на запрос.



The screenshot shows a database query interface. The 'Query' tab is active, displaying the following SQL query:

```
1 SELECT название_валюты, текущий_курс
2 FROM Валюта
3 WHERE название_валюты = 'Доллар США' OR название_валюты = 'Евро';
```

Below the query, the 'Data Output' tab is active, showing the results of the query in a table format. The table has two columns: 'название_валюты' (character varying (50)) and 'текущий_курс' (numeric (10,2)). The results are as follows:

	название_валюты character varying (50)	текущий_курс numeric (10,2)
1	Евро	97.87
2	Доллар США	90.19

Рисунок 8.1 – Ответ на первый запрос

2.1 Запрос – Вывести баланс клиента с ФИО «Сомойлин Ян Максимович».

SQL запрос:

```
SELECT фιο_клиента,Операция.баланс
```

```
FROM Клиент
```

```
JOIN      Операция      ON      Клиент.серия_номер_паспорта      =
```

```
Операция.серия_номер_паспорта
```

```
WHERE Клиент.фιο_клиента = 'Сомойлин Ян Максимович';
```

На рисунке 8.2 показан ответ на запрос.

The screenshot shows a database query interface with two tabs: 'Query' and 'Query History'. The 'Query' tab is active, displaying the following SQL query:

```
1 SELECT фιο_клиента,Операция.баланс
2 FROM Клиент
3 JOIN Операция ON Клиент.серия_номер_паспорта = Операция.серия_номер_паспорта
4 WHERE Клиент.фιο_клиента = 'Сомойлин Ян Максимович';
```

Below the query, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with two columns: 'фιο_клиента' (character varying (255)) and 'баланс' (numeric (10,2)). The table contains one row of data:

	фιο_клиента character varying (255)	баланс numeric (10,2)
1	Сомойлин Ян Максимович	302.73

Рисунок 8.2 – Ответ на второй запрос

2.2 Запрос – Вывести все адреса банкоматов, которые привязаны к банку «Сбербанк».

```
SELECT название_банка, Банкомат.адрес_банкомата
```

```
FROM Банк
```

```
JOIN Банкомат ON Банк.bank_id = Банкомат.bank_id
```

```
WHERE Банк.название_банка = 'Сбербанк';
```

На рисунке 8.3 показан ответ на запрос.

Query		Query History
1	SELECT	название_банка, Банкомат.адрес_банкомата
2	FROM	Банк
3	JOIN	Банкомат ON Банк.bank_id = Банкомат.bank_id
4	WHERE	Банк.название_банка = 'Сбербанк';
Data Output		Messages
		Notifications
<div> <div>≡</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑</div> <div>🗄</div> <div>⬇</div> <div>📈</div> <div>SQL</div> </div>		
	название_банка character varying (255) 🔒	адрес_банкомата character varying (255) 🔒
1	Сбербанк	г. Москва, ул. Образцова, д.9
2	Сбербанк	г. Севастополь, Строителей ул., д. 18
3	Сбербанк	г. Екатеринбург, Западная ул., д. 17

Рисунок 8.3 – Ответ на третий запрос

3.1 Запрос – Вывести сумму снятия и адрес банкомата, в котором Крючкова Елизавета Андреевна снимала наличные.

SQL запрос:

```

SELECT          Клиент.фио_клиента,          Операция.сумма_выдачи,
Банкомат.адрес_банкомата
FROM Операция
JOIN      Клиент      ON      Операция.серия_номер_паспорта      =
Клиент.серия_номер_паспорта
JOIN Банкомат ON Операция.ATM_id = Банкомат.ATM_id
WHERE Клиент.фио_клиента = 'Крючкова Елизавета Андреевна'
AND Операция.тип_операции = 'Выдача наличных';

```

На рисунке 8.4 показан ответ на запрос.

Query Query History

1

2

3

4

5

6

SELECT

Клиент.фio_клиента, Операция.сумма_выдачи, Банкомат.адрес_банкомата

FROM

Операция

JOIN

Клиент

ON

Операция.серия_номер_паспорта = Клиент.серия_номер_паспорта

JOIN

Банкомат

ON

Операция.ATM_id = Банкомат.ATM_id

WHERE

Клиент.фio_клиента = 'Крюкова Елизавета Андреевна'

AND

Операция.тип_операции = 'Выдача наличных';

Data Output Messages Notifications

≡+

📄

▼

📋

▼

🗑

🗄

⬇

📈

SQL

	фio_клиента character varying (255)	сумма_выдачи numeric (10,2)	адрес_банкомата character varying (255)
1	Крюкова Елизавета Андреевна	2500.00	г. Ярославль, Березовая ул., д. 10

Рисунок 8.4 – Ответ на четвертый запрос

3.2 Вывести всех клиентов из «Тинькофф», которые проводили операции на сумму более 10000.

SQL запрос:

SELECT Клиент.фio_клиента

FROM Клиент

JOIN Операция ON Клиент.серия_номер_паспорта =

Операция.серия_номер_паспорта

JOIN Банк ON Клиент.bank_id = Банк.bank_id

WHERE Банк.название_банка = 'Тинькофф'

AND Операция.сумма_выдачи > 10000;

На рисунке 8.5 показан ответ на запрос.

Query

Query History

1

SELECT

Клиент.фio_клиента

2

FROM

Клиент

3

JOIN

Операция

ON

Клиент.серия_номер_паспорта = Операция.серия_номер_паспорта

4

JOIN

Банк

ON

Клиент.bank_id = Банк.bank_id

5

WHERE

Банк.название_банка = 'Тинькофф'

6

AND


Операция.сумма_выдачи > 10000;

Data Output


Messages

Notifications


≡+





▼




▼










SQL

фio_клиента

character varying (255)



1

Степанов Константин Михайлович

Рисунок 8.5 – Ответ на пятый запрос

4 Запрос - Добавить новый вид валюты с кодом валюты 1131, названием «Грузинский Лари», сокращением «GEL», страной «Грузия» и текущим курсом «31.24».

SQL запрос:

SELECT фиио клиента,Операция.баланс

FROM Клиент

JOIN	Операция	ON	Клиент.серия_номер_паспорта	=
------	----------	----	-----------------------------	---

Операция.серия номер паспорта

WHERE Клиент.фio клиента = 'Сомойлин Ян Максимович';

На рисунке 8.6 показан ответ на запрос.

```
Query Query History
1  INSERT INTO Валюта (код_валюты, название_валюты, сокращение, страна, текущий_курс)
2  VALUES (1131, 'Грузинский Лари', 'GEL', 'Грузия', 31.24);
Data Output Messages Notifications
INSERT 0 1
Query returned successfully in 641 msec.
```

Рисунок 8.6 – Ответ на шестой запрос

5 Запрос - Удалить банк «МТС-Банк» из списка всех банков.

SQL запрос:

DELETE FROM Банк

WHERE название_банка = 'МТС-Банк';

На рисунке 8.7 показан ответ на запрос.

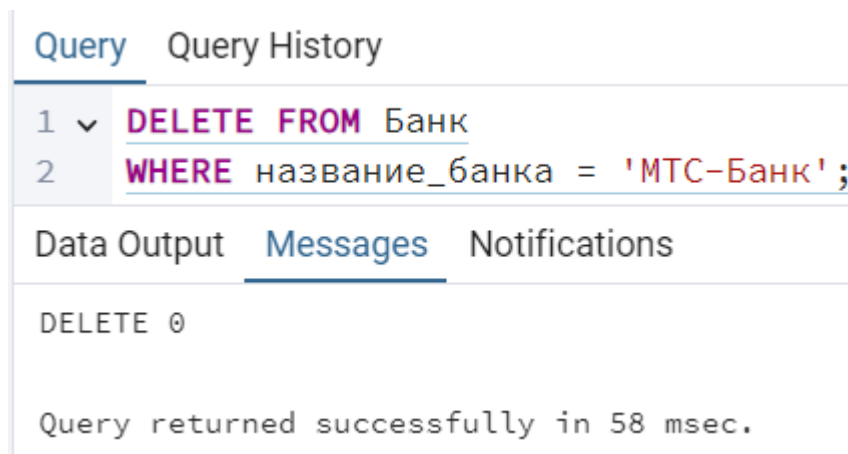


Рисунок 8.7 – Ответ на седьмой запрос

6. Запрос - Изменить название банка «Тинькофф» на «Т-Банк».

SQL запрос:

UPDATE Банк

SET название_банка = 'Т-Банк'

WHERE название_банка = 'Тинькофф';

На рисунке 8.8 показан ответ на запрос.

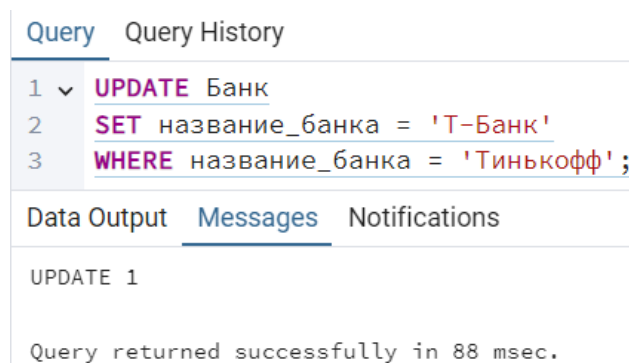


Рисунок 8.8 – Ответ на восьмой запрос

9 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

9.1 Введение

9.1.1 Область применения

Работа с базой данных

9.1.2 Краткое описание возможностей

Система предназначена для отправки запросов, получения ответов, просмотра записей и количества таблиц, изменений записей и структур таблиц базы данных.

9.1.3 Уровень подготовки пользователя

Пользователь системы должен иметь опыт работы с СУБД и ОС Windows (7/8/10/11).

9.1.4 Перечень эксплуатационной документации, с которой необходимо ознакомиться пользователю

Руководство пользователя ОС Windows, руководство по управлению реляционной СУБД

9.2 Назначения и условия применения

9.2.1 Виды деятельности, функции, для автоматизации которых предназначено данное средство автоматизации

Система предназначена для автоматизированной работы с базами данных. Визуальный интерфейс упрощает понимание структуры базы данных и позволяет корректировать таблицы без знаний SQL.

9.2.2 Условия, при соблюдении которых обеспечивается применение средства автоматизации в соответствии с назначением

Работа с сайтом возможна всегда, когда пользователь знает данные для входа на сервер базы данных.

9.3 Подготовка к работе

9.3.1 Состав и содержание дистрибутивного носителя данных

ОС Windows (7/8/10/11).

9.3.2 Порядок загрузки данных и программ

1. Установить сервер PostgreSQL;

2. Скачать файлы программы;
3. Запустить программу.

9.3.3 Уровень подготовки пользователя

1. Запустить программу;
2. Открыть сайт;
3. Пройти авторизацию и выбрать подходящую базу данных;

В случае если авторизация не прошла успешно, следует обратить внимание на ошибку, выведенную на экран (Неверное значение порта/Отсутствует база данных/Неверный пароль).

9.4 Описание операций

9.4.1 Выполняемые функции

Функции: Просмотр таблиц, вывод строк, добавление/удаление таблицы, строки или столбца, фильтрация, корректировка столбца/строки, SQL запросы через web-консоль.

9.4.2 Описание операций технологического процесса обработки данных, необходимых для выполнения задач

Операция 1. Авторизация.

Условия, при которых возможны выполнения операций:

1. Компьютер пользователя подключен к сети;
2. Сайт работает корректно;
3. Сервер PostgreSQL установлен.

Подготовительные действия:

1. На компьютере пользователя необходимо выполнить дополнительные настройки, приведенные в п. 4.3 настоящего документа.

Основные действия в требуемой последовательности:

1. Введите хост/порт/имя пользователя/пароль/имя базы данных и нажмите кнопку «Вход».

На рисунке 9.4.2.1 показан вход в систему.

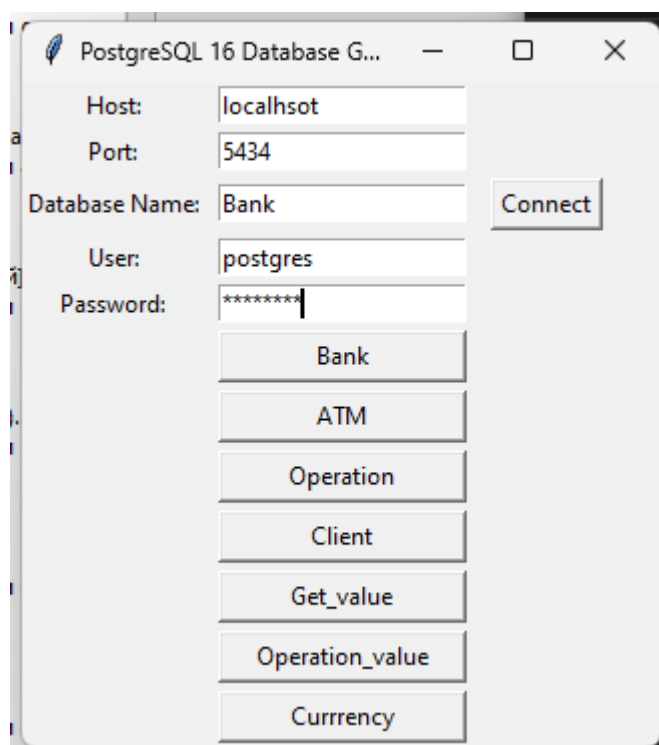


Рисунок 9.4.2.1 – Вход в систему

На рисунке 9.4.2.2 показана главная страница, выводимая после успешной авторизации.

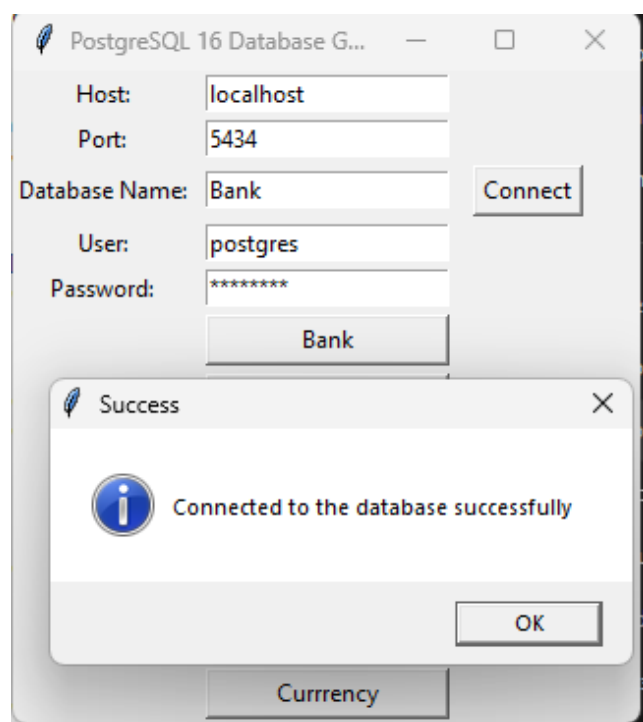


Рисунок 9.4.2.2 – Главная страница

На рисунке 9.4.2.3 показан ответ при вводе некорректных данных.

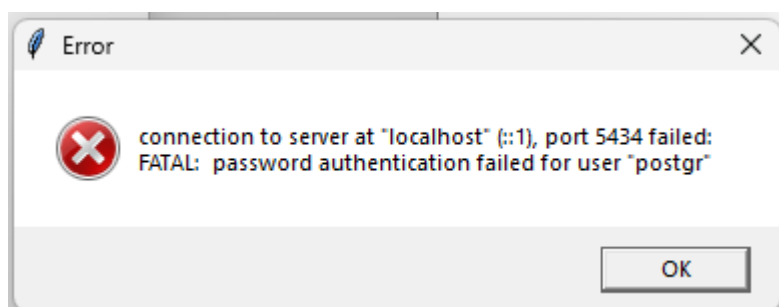


Рисунок 9.4.2.3 – Ошибка авторизации

Операция 2. Удаление и добавление строк

Условия, при которых возможны выполнения операций:

1. Наличие данных для входа в систему.

Подготовительные действия не требуются.

Основные действия в требуемой последовательности:

1. Войти в систему.
2. Перейдите в таблицу, как показано на рисунке 9.4.2.4.

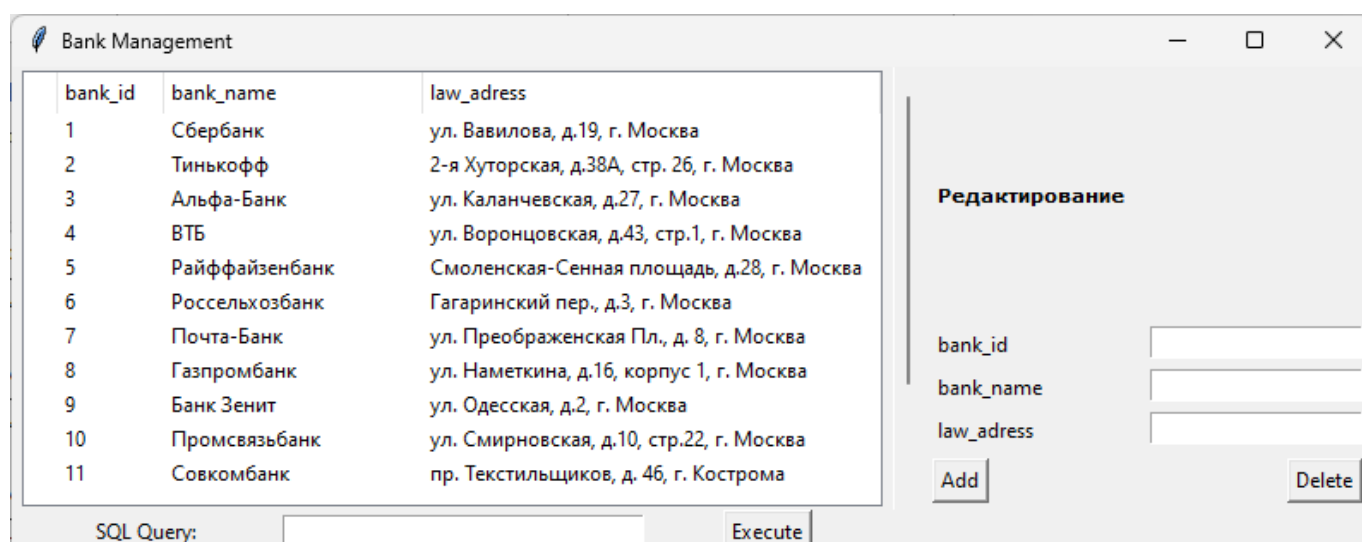
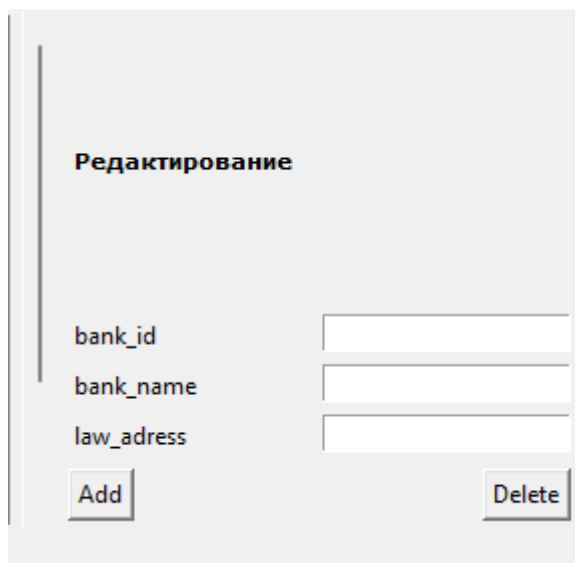


Рисунок 9.4.2.4 – Окно таблицы

3. Справа, в месте для редактирования появятся поля с атрибутами выбранной сущности 9.4.2.5.



Редактирование

bank_id

bank_name

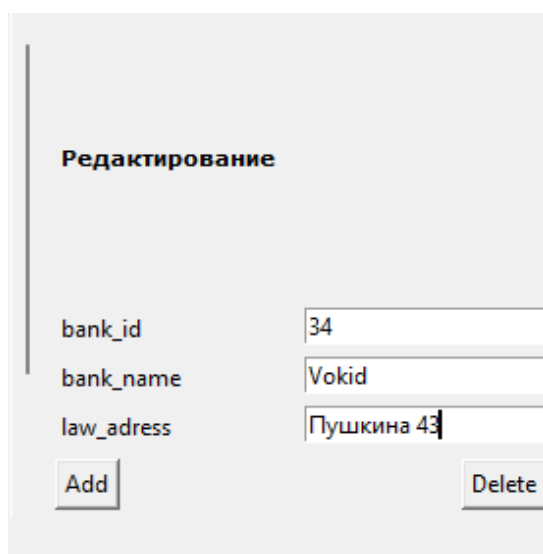
law_adress

Add Delete

Рисунок 9.4.2.5 – Список всех строк

4. В данной области показаны все строки, находящиеся в выбранной таблице.

5. Для добавления новой записи нажмите кнопку «Add», предварительно заполнив нужный. После нажатия откроется форма, содержащая все столбцы для заполнения новой строки. Форма показана на рисунке 9.4.2.6.



Редактирование

bank_id

bank_name

law_adress

Add Delete

Рисунок 9.4.2.6 – Форма для добавления новой строки

6. Если никаких ошибок не будет, то строка успешно появится в таблице, как показано на рисунке 9.4.2.7.

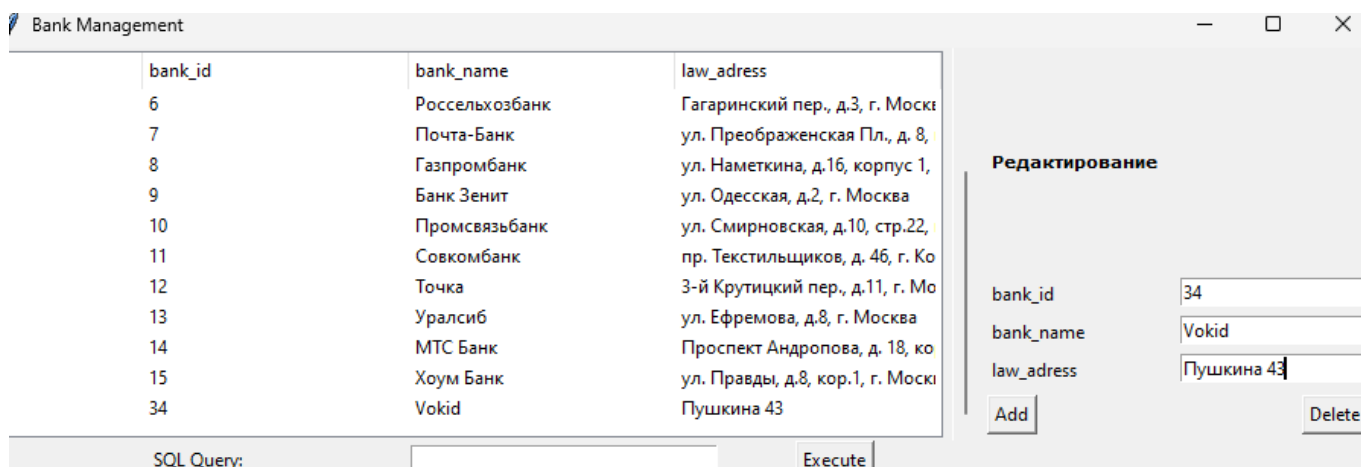


Рисунок 9.4.2.7 – Добавленная строка

7. Если в форме будут допущены какие-то ошибки (например, с уникальностью первичного ключа или с передачей внешних ключа), то строка не будет добавлена, а на странице будет выведена SQL ошибка. Пример ошибки показан на рисунке 9.4.2.8.

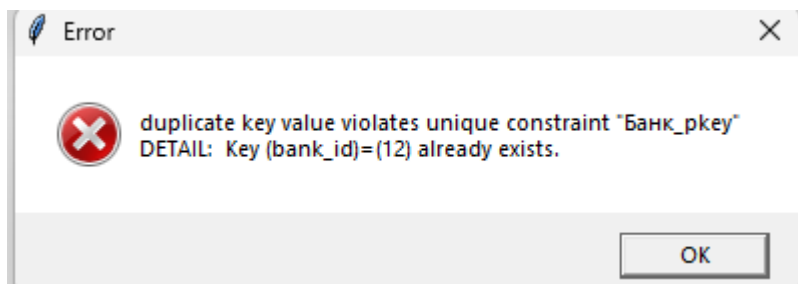


Рисунок 9.4.2.8 – Ошибка при добавлении строки

8. Для удаления записи требуется ввести названия нужных атрибутов и нажать кнопку «Delete».

Операция 3. Ввод SQL запросов.

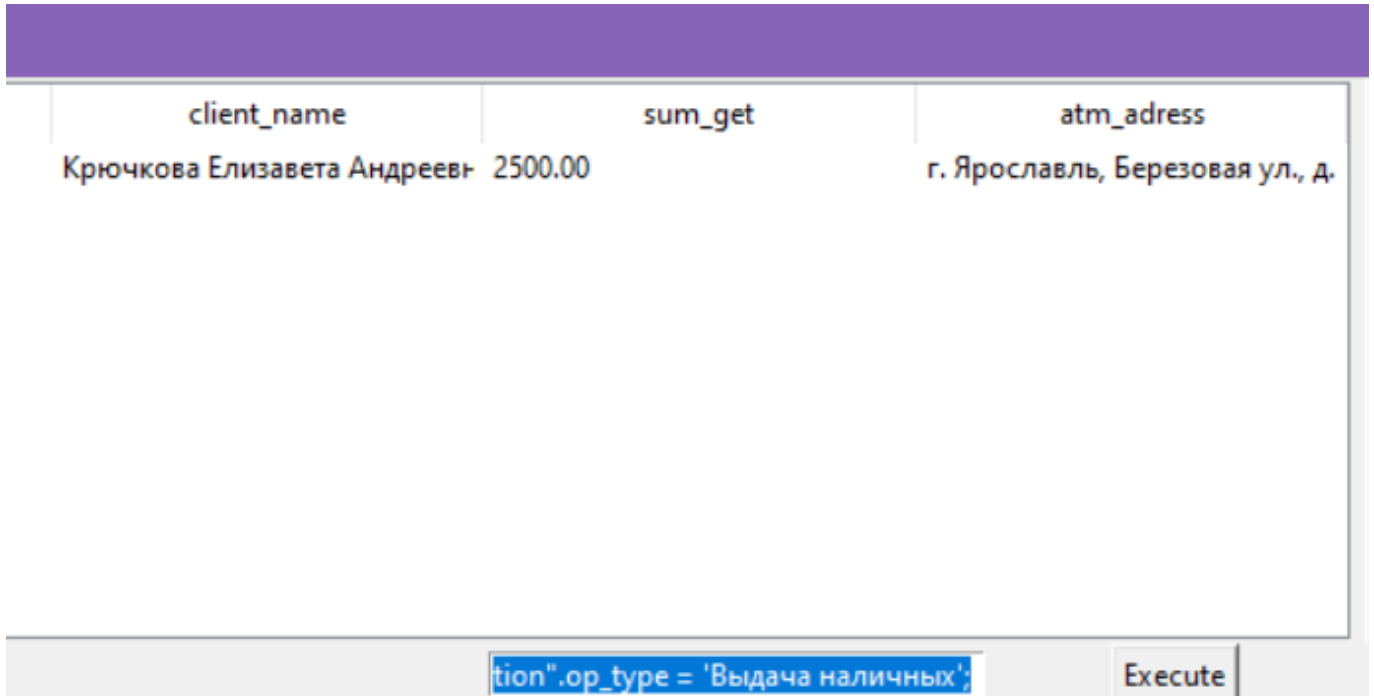
Условия, при которых возможны выполнения операций:

1. Наличие данных для входа в систему.

Подготовительные действия не требуются.

Основные действия в требуемой последовательности:

1. Войти в систему.
2. Перейдите в таблицу, как показано на рисунке 9.4.2.9.



The screenshot shows a database table window with a purple header bar. The table has three columns: 'client_name', 'sum_get', and 'atm_adress'. The first row contains the data: 'Крючкова Елизавета Андреевн', '2500.00', and 'г. Ярославль, Березовая ул., д.'. Below the table, there is a text input field containing the SQL query 'tion".op_type = 'Выдача наличных';' and an 'Execute' button.

client_name	sum_get	atm_adress
Крючкова Елизавета Андреевн	2500.00	г. Ярославль, Березовая ул., д.

Рисунок 9.4.2.9 – Окно таблицы

3. В верхней панели сайта выберите вкладку «Запрос» и перейдите в неё, как показано на рисунке 9.4.2.10.

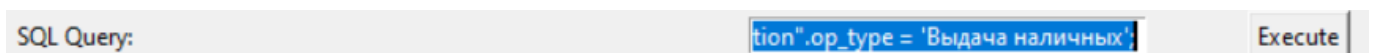


Рисунок 9.4.2.10 – Вкладка «Запрос»

ПРИЛОЖЕНИЕ А

А.1 Файл bank.py, содержащий класс базы данных и контроллеры

```
import tkinter as tk
from tkinter import messagebox, Toplevel, font
from tkinter import ttk
import psycopg2

class DatabaseApp:
    def __init__(self, root):
        self.root = root
        self.root.title("PostgreSQL 16 Database GUI")

        # Создание полей для ввода параметров подключения к базе данных
        self.host_entry = self.create_entry("Host:", 0)
        self.port_entry = self.create_entry("Port:", 1)
        self.dbname_entry = self.create_entry("Database Name:", 2)
        self.user_entry = self.create_entry("User:", 3)
        self.password_entry = self.create_entry("Password:", 4, show="*")

        # Кнопка для подключения к базе данных
        self.connect_button = tk.Button(root, text="Connect",
command=self.connect_db)
        self.connect_button.grid(row=2, rowspan=1, column=2, sticky='nsew',
padx=6, pady=2)

        # Словарь с информацией о сущностях базы данных
        self.entities = {
            'Bank': {'query': 'SELECT * FROM "Bank"', 'attributes':
['bank_id', 'bank_name', 'law_adress']},
            'ATM': {'query': 'SELECT * FROM "ATM"', 'attributes': ['atm_id',
'atm_adress', 'bank_id']},
            'Operation': {'query': 'SELECT * FROM "Operation"', 'attributes':
['operation_id', 'date', 'time', 'commission', 'sum_get', 'op_type',
'status_op', 'balance', 'passport_num_ser', 'atm_id']},
            'Client': {'query': 'SELECT * FROM "Client"', 'attributes':
['pasport_num_ser', 'client_name', 'client_adress', 'pin', 'bank_id']},
            'Get_value': {'query': 'SELECT * FROM "Get_value"', 'attributes':
['sum_get', 'pasport_num_ser', 'atm_id']},
            'Operation_value': {'query': 'SELECT * FROM "Operation_value"',
'attributes': ['operation_id', 'id_currency', 'cum_in_curr', 'get_in_cur']},
            'Currrency': {'query': 'SELECT * FROM "Currrency"', 'attributes':
['id_currency', 'currency_name', 'short_name', 'country', 'curr_value']}
        }

        # Создание кнопок для каждой сущности
        self.entity_buttons = []
        for i, (entity, info) in enumerate(self.entities.items()):
            button = tk.Button(root, text=entity, command=lambda e=entity,
q=info['query'], a=info['attributes']: self.open_entity_window(e, q, a))
```

```

        button.grid(row=6 + i, column=1, sticky='nsew', padx=6, pady=2)
        self.entity_buttons.append(button)

def create_entry(self, label_text, row, show=None):
    """Создает поле ввода с меткой."""
    label = tk.Label(self.root, text=label_text)
    label.grid(row=row, column=0)
    entry = tk.Entry(self.root, show=show)
    entry.grid(row=row, column=1, padx=6, pady=2)
    return entry

def connect_db(self):
    """Подключается к базе данных PostgreSQL с использованием введенных
данных."""
    try:
        self.connection = psycopg2.connect(
            host=self.host_entry.get(),
            port=self.port_entry.get(),
            dbname=self.dbname_entry.get(),
            user=self.user_entry.get(),
            password=self.password_entry.get()
        )
        messagebox.showinfo("Success", "Connected to the database
successfully")
    except Exception as e:
        messagebox.showerror("Error", str(e))

def open_entity_window(self, entity_name, query, attributes):
    """Открывает новое окно для работы с выбранной сущностью."""
    window = Toplevel(self.root)
    window.title(f"{entity_name} Management")

    # Создание таблицы для отображения результатов запросов
    result_tree = ttk.Treeview(window)
    result_tree.grid(row=0, column=0, rowspan=6, columnspan=4,
sticky='nsew', padx=6, pady=2)

    # Добавление вертикальной прокрутки
    vsb = ttk.Scrollbar(window, orient="vertical",
command=result_tree.yview)
    vsb.grid(row=0, column=4, rowspan=6, sticky='ns')
    result_tree.configure(yscrollcommand=vsb.set)

    window.grid_rowconfigure(0, weight=1)
    window.grid_columnconfigure(0, weight=1)

    self.execute_prepared_query(query, result_tree)

    # Поле для ввода произвольного SQL-запроса
    query_label = tk.Label(window, text="SQL Query:")
    query_label.grid(row=6, column=0)

```

```

query_entry = tk.Entry(window, width=50)
query_entry.grid(row=6, column=1, columnspan=2)

# Кнопка для выполнения введенного SQL-запроса
execute_button = tk.Button(window, text="Execute", command=lambda:
self.execute_query(query_entry, result_tree))
execute_button.grid(row=6, column=3)

# Настройка шрифта для метки "Редактирование"
font1 = font.Font(family="Verdana", size=8, weight="bold",
slant="roman")
label = tk.Label(window, text="Редактирование", font=font1)
label.grid(row=0, column=5, rowspan=1, sticky='news', padx=6, pady=2)

# Создание полей для ввода атрибутов сущности
attribute_entries = {}
for i, attribute in enumerate(attributes):
    print(i, attribute)
    label = tk.Label(window, text=attribute)
    label.grid(row=i+1, column=5, rowspan=1, sticky='nw', padx=6,
pady=2)

    entry = tk.Entry(window, width=20)
    entry.grid(row=i+1, column=6, rowspan=1, sticky='nw', padx=6,
pady=2)

    attribute_entries[attribute] = entry

# Кнопка для добавления новой записи
add_button = tk.Button(window, text="Add", command=lambda:
self.add_record(entity_name, attribute_entries, result_tree))
add_button.grid(row=len(attributes)+2, sticky='sw', column=5, padx=6,
pady=4)

# Кнопка для удаления записи
delete_button = tk.Button(window, text="Delete", command=lambda:
self.delete_record(entity_name, attribute_entries, result_tree))
delete_button.grid(row=len(attributes)+2, sticky='se', column=6,
padx=6, pady=4)

def execute_prepared_query(self, query, result_tree):
    """Выполняет подготовленный SQL-запрос и отображает результаты в
таблице."""
    try:
        cursor = self.connection.cursor()
        cursor.execute(query)
        result = cursor.fetchall()

        # Очистка предыдущих результатов
        for i in result_tree.get_children():
            result_tree.delete(i)

        columns = [desc[0] for desc in cursor.description]

```

```

print(columns)
result_tree['columns'] = columns
for col in columns:
    result_tree.heading(col, text=col, anchor='w')
    result_tree.column(col, width=100)

for row in result:
    result_tree.insert('', 'end', values=row)

    cursor.close()
except Exception as e:
    messagebox.showerror("Error", str(e))

def execute_query(self, query_entry, result_tree):
    """Выполняет произвольный SQL-запрос, введенный пользователем, и
отображает результаты в таблице."""
    try:
        cursor = self.connection.cursor()
        cursor.execute(query_entry.get())
        result = cursor.fetchall()

        # Очистка предыдущих результатов
        for i in result_tree.get_children():
            result_tree.delete(i)

        columns = [desc[0] for desc in cursor.description]
        result_tree['columns'] = columns
        for col in columns:
            result_tree.heading(col, text=col)
            result_tree.column(col, width=100)

        for row in result:
            result_tree.insert('', 'end', values=row)

        cursor.close()
    except Exception as e:
        messagebox.showerror("Error", str(e))

def add_record(self, entity_name, attribute_entries, result_tree):
    """Добавляет новую запись в базу данных."""
    try:
        cursor = self.connection.cursor()
        table_name = entity_name
        columns = ', '.join(attribute_entries.keys())
        values = ', '.join([f"'{entry.get()}'" for entry in
attribute_entries.values()])
        query = f'INSERT INTO "{table_name}" ({columns}) VALUES
({values})'
        cursor.execute(query)
        self.connection.commit()

```

```

        self.execute_prepared_query(self.entities[entity_name]['query'],
result_tree)
        cursor.close()
    except Exception as e:
        messagebox.showerror("Error", str(e))

def delete_record(self, entity_name, attribute_entries, result_tree):
    """Удаляет запись из базы данных."""
    try:
        cursor = self.connection.cursor()
        table_name = entity_name
        conditions = 'AND '.join([f"{attr}='{entry.get()}'" for attr,
entry in attribute_entries.items()])
        query = f'DELETE FROM "{table_name}" WHERE {conditions}'
        cursor

```