

**АВТОНОМНАЯ НЕКОММЕРЧЕСКАЯ ОРГАНИЗАЦИЯ
ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ
МОСКОВСКИЙ МЕЖДУНАРОДНЫЙ КОЛЛЕДЖ ЦИФРОВЫХ
ТЕХНОЛОГИЙ
«АКАДЕМИЯ ТОП»**

ПРОЕКТ

по дисциплине «Технология доступа к базам данных (ADO.NET)»

«Разработка системы обслуживания банкоматов»

Выполнили: Бодров Д.Е., Иванен М.А. (гр. РПО-2)

Преподаватель: Рослова О.А

2025

Содержание

Введение	3
1. Анализ предметной области	4
2. Цели и задачи проекта	5
3. Проектирование базы данных	6
3.1 Описание таблиц	7
3.2 ER-диаграмма и связи	9
3.3 Нормализация до 3НФ	11
3.4 Индексы и представления	12
4. Интерфейс программы	13
5. Тестирование программы	15
6. Выводы	16
Приложение. Программный код и SQL-скрипты	23

Введение

Автоматические банковские терминалы (банкоматы) играют ключевую роль в современной банковской инфраструктуре, обеспечивая клиентам быстрый и удобный доступ к финансовым услугам 24/7. Разработка учебного прототипа системы обслуживания банкоматов позволяет изучить принципы проектирования надежных информационных систем, организацию хранения и обработки финансовых данных, а также реализацию бизнес-логики и интерфейса взаимодействия.

В рамках данного проекта разработана консольная система обслуживания банкоматов, включающая реализацию модели данных на основе реляционной базы данных MS SQL Server, модуль доступа к данным на основе ADO.NET и прикладную логику на языке C# (.NET 8). Проект предназначен для демонстрации основных операций: аутентификация клиента по карте и PIN, проверка баланса, снятие и пополнение средств, журналирование транзакций.

Документ содержит полное описание предметной области, проектирование базы данных, скрипты создания и наполнения БД, описание интерфейса приложения, результаты тестирования и фрагменты ключевого кода.

1. Анализ предметной области

1.1 Описание предметной области

Система обслуживания банкоматов моделирует процессы, характерные для банковских автоматизированных систем самообслуживания. Типичный банкомат обеспечивает идентификацию клиента по номеру карты и PIN, предоставляет операции по проверке баланса, выполнению списаний и зачислений, а также формирует запись о каждой транзакции для последующего аудита.

1.2 Участники и роли системы

- Клиент — владелец банковской карты, который выполняет операции через банкомат.
- Банкомат (терминал) — устройство, выполняющее интерфейсный ввод/вывод и инициирующее транзакции.
- Система учёта (сервер БД) — обеспечивает хранение и целостность данных о клиентах, счетах, картах и транзакциях.

1.3 Нефункциональные требования

- Безопасность: хранение PIN в виде хэша, защита от SQL-инъекций, контроль статуса карты (заблокирована/активна).
- Надёжность: использование транзакций в СУБД для обеспечения атомарности операций снятия/пополнения.
- Масштабируемость: структура БД спроектирована с учётом добавления новых атрибутов и сущностей.

2. Цели и задачи проекта

Цель проекта: Разработать учебный прототип системы обслуживания банкоматов, демонстрирующий основные операции и принципы работы с банковскими данными.

Задачи:

1. Проанализировать предметную область и составить ER-модель.
2. Спроектировать реляционную базу данных и реализовать её в MS SQL Server.
3. Реализовать приложение на C# с использованием ADO.NET для доступа к БД.
4. Наполнить БД тестовыми данными и провести функциональное тестирование.
5. Подготовить отчёт, сопровождающий проект.

Методы: проектирование ER-модели, SQL-скрипты создания и наполнения БД, использование ADO.NET и C# для реализации бизнес-логики.

3. Проектирование базы данных

База данных AtmService реализована по реляционной модели и включает следующие сущности: Clients, Accounts, Cards, Atms, Transactions, ClientAccounts.

Далее представлено подробное описание таблиц с указанием атрибутов, типов данных, ключей и ограничений.

3.1 Таблица `Clients`

Хранение персональных данных клиентов.

Атрибут	Тип данных	Ограничения / Примечания
Id	INT IDENTITY(1,1)	PRIMARY KEY
FirstName	NVARCHAR(100)	NOT NULL
LastName	NVARCHAR(100)	NOT NULL
PassportNo	NVARCHAR(50)	NOT NULL UNIQUE
Phone	NVARCHAR(20)	NOT NULL

3.1 Таблица `Accounts`

Управление банковскими счетами и их балансами.

Атрибут	Тип данных	Ограничения / Примечания
Id	INT IDENTITY(1,1)	PRIMARY KEY
Number	NVARCHAR(50)	NOT NULL UNIQUE
Currency	NVARCHAR(3)	NOT NULL DEFAULT 'UZS'
Balance	DECIMAL(18,2)	NOT NULL DEFAULT 0.00

3.1 Таблица `Cards`

Хранение информации о банковских картах и их статусах.

Атрибут	Тип данных	Ограничения / Примечания
Id	INT IDENTITY(1,1)	PRIMARY KEY

AccountId	INT	NOT NULL, FK -> Accounts(Id) ON DELETE CASCADE
CardNumber	NVARCHAR(20)	NOT NULL UNIQUE
Expiry	DATETIME2	NOT NULL
PinHash	NVARCHAR(255)	NOT NULL
Status	INT	NOT NULL DEFAULT 1 -- 1=Active,2=Blocked,3=Expired

3.1 Таблица `Atms`

Справочник банкоматов и их местоположения.

Атрибут	Тип данных	Ограничения / Примечания
Id	INT IDENTITY(1,1)	PRIMARY KEY
Name	NVARCHAR(100)	NOT NULL
Location	NVARCHAR(200)	NOT NULL

3.1 Таблица `Transactions`

Журналирование всех финансовых операций.

Атрибут	Тип данных	Ограничения / Примечания
Id	INT IDENTITY(1,1)	PRIMARY KEY
CardId	INT	NOT NULL, FK -> Cards(Id)
AtmId	INT	NOT NULL, FK -> Atms(Id)
Type	INT	NOT NULL -- 1=Withdraw,2=Deposit,3=Transfer
Amount	DECIMAL(18,2)	NOT NULL
PerformedAt	DATETIME2	NOT NULL DEFAULT GETUTCDATE()

BalanceAfter	DECIMAL(18,2)	NOT NULL
--------------	---------------	----------

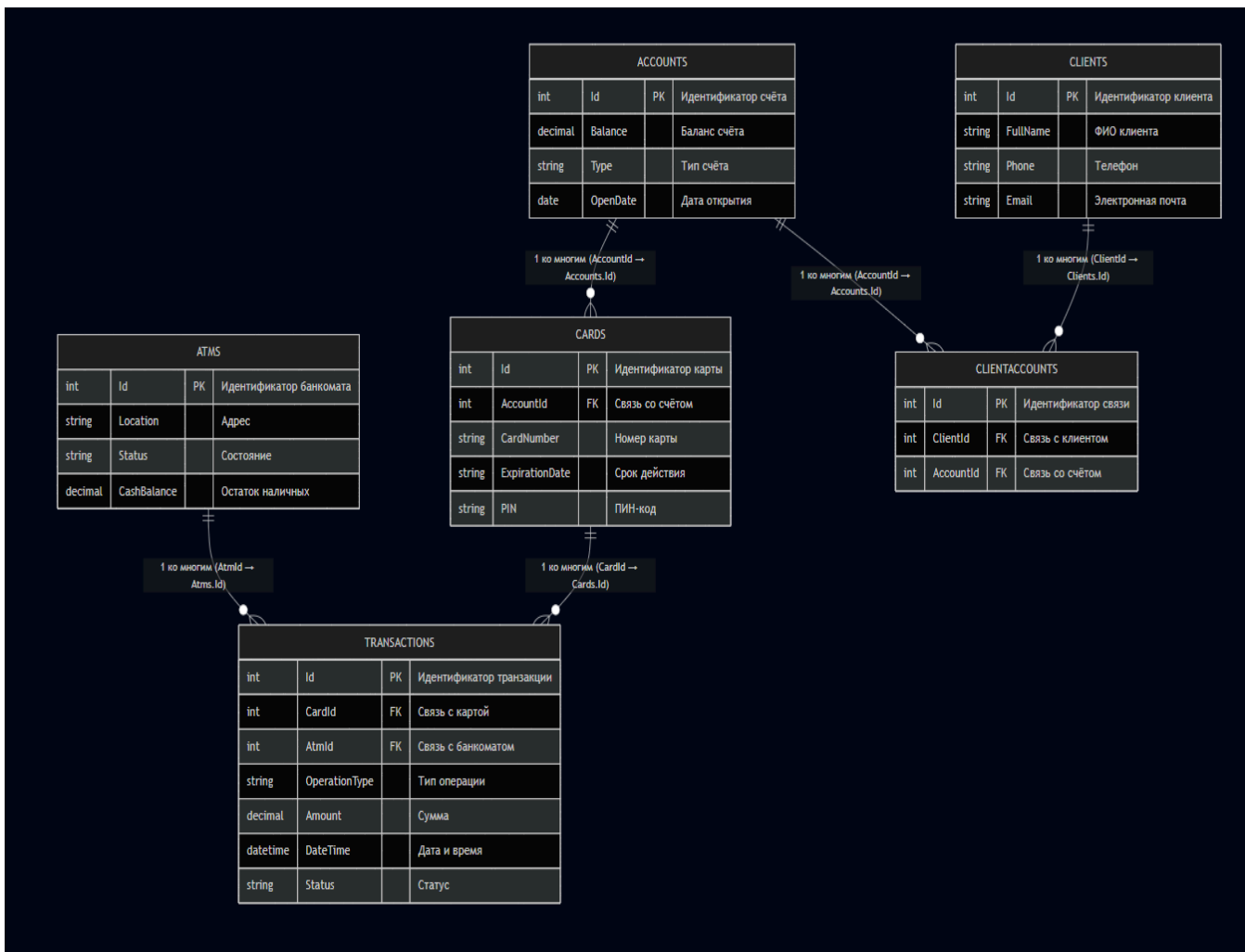
3.1 Таблица `ClientAccounts`

Связь многие-ко-многим между клиентами и счетами.

Атрибут	Тип данных	Ограничения / Примечания
ClientId	INT	NOT NULL, FK -> Clients(Id) ON DELETE CASCADE
AccountId	INT	NOT NULL, FK -> Accounts(Id) ON DELETE CASCADE
Role	NVARCHAR(20)	NOT NULL DEFAULT 'Owner'

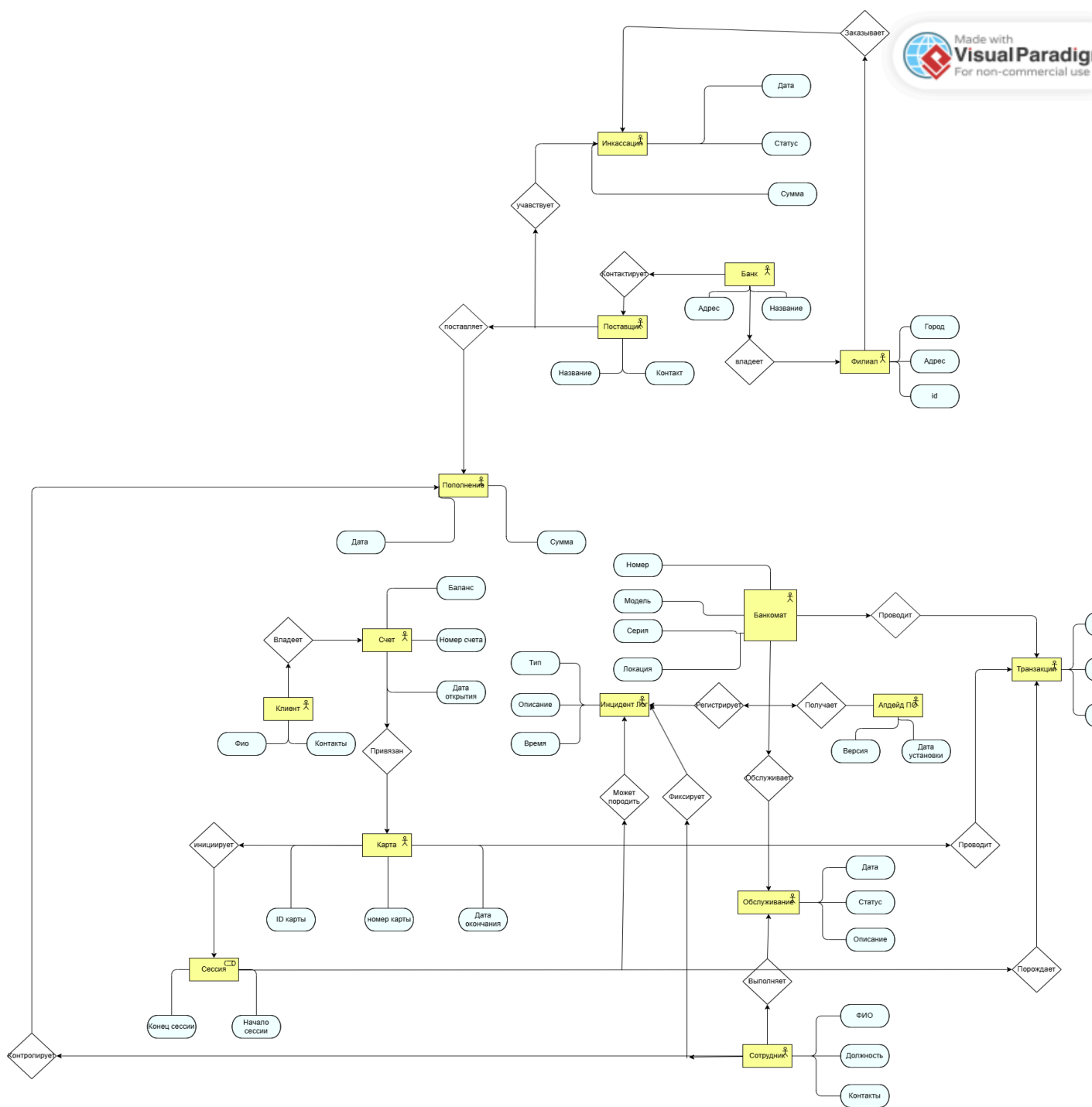
3.2 ER-диаграмма и связи

Ниже следует ER-диаграмма, отражающая связи между сущностями.



Ключевые внешние связи:

- Cards.AccountId → Accounts.Id
- Transactions.CardId → Cards.Id
- Transactions.AtmId → Atms.Id
- ClientAccounts.ClientId → Clients.Id
- ClientAccounts.AccountId → Accounts.Id



3.3 Нормализация до 3НФ

База данных проверена на соответствие нормальным формам.

1НФ: Каждое поле атомарно, отсутствуют повторяющиеся группы.

2НФ: Все таблицы имеют простые первичные ключи или корректно выделенные составные ключи; все неключевые атрибуты полностью зависят от первичного ключа.

3НФ: Устранены транзитивные зависимости. Пример: информация о клиентах вынесена в таблицу Clients, информация о счетах — в таблицу Accounts; связь многие-ко-многим реализована через таблицу ClientAccounts.

Таким образом, проект соответствует требованиям 3НФ, что уменьшает избыточность и упрощает обновление данных.

3.4 Индексы и представления

Для повышения производительности в системе созданы индексы для часто используемых полей и представления для формирования отчетов.

Примеры индексов:

- IX_Cards_AccountId ON Cards(AccountId)
- IX_Transactions_PerformedAt ON Transactions(PerformedAt)

Примеры представлений:

- vw_ClientCards — объединяет информацию о клиентах, счетах и картах.
- vw_TransactionHistory — предоставляет развернутую историю транзакций с данными о клиенте и банкомате.

4. Интерфейс программы

Приложение реализовано в виде консольного интерфейса на C# (.NET 8). При запуске приложение предлагает ввести номер карты и PIN. После успешной аутентификации пользователь видит главное меню с выбором операций. Интерфейс содержит сообщение об ошибках и подсказки.

4.1 Пошаговые сценарии взаимодействия

Сценарий 1 — Авторизация:

1. Ввод номера карты.
2. Ввод PIN.
3. Проверка статуса карты и срока действия.
4. В случае успеха — переход в меню операций.

```
=== БАНКОМАТ ===  
Введите номер карты: 8600123456781234  
Введите PIN: 0000  
Авторизация успешна!
```

```
Клиент: Иван Иванов  
Счет: 8600123456789012  
Баланс: 1 500 000,00 UZS  
  
=== МЕНЮ ===  
1. Проверить баланс  
2. Снять наличные  
3. Пополнить счет  
4. История операций  
5. Выход  
Выберите действие:
```

Сценарий 2 — Снятие наличных:

1. Выбор опции 'Снять наличные'.
2. Ввод суммы.
3. Проверка доступного баланса и наличия средств в банкомате.
4. Выполнение транзакции в рамках транзакции БД, уменьшение баланса и запись в Transactions.

```
Введите сумму для снятия: 200000  
Операция успешна!
```


1	Авторизация по карте	Card=8600123456781234, PIN=0000	Ввод номера и PIN, проверка статуса и срока действия	Авторизация успешна, переход в меню	Успех
2	Проверка баланса	Card valid	Запрос баланса по счету	Показать корректный баланс	Успех
3	Снятие средств	Сумма 200000, баланс 1500000	Проверка баланса, создание транзакции, списание	Баланс уменьшается на 200000, запись в Transactions	Успех
4	Пополнение	Сумма 500000	Увеличение баланса и запись в Transactions	Баланс увеличен на 500000	Успех
5	Просмотр истории	CardId	Вытягивание записей из vw_TransactionHistory	Вывод списка операций, отсортированных по дате	Успех

Методика тестирования: ручное функциональное тестирование, модульные тесты для сервисов аутентификации и транзакций.

6. Выводы

В результате выполнения курсовой работы реализована учебная система обслуживания банкоматов, включающая:

- Проект реляционной базы данных, соответствующий 3НФ;
- Консольное приложение на C# с подключением к MS SQL Server через ADO.NET;
- Набор SQL-скриптов для создания структуры БД и наполнения тестовыми данными;
- Набор тест-кейсов, подтверждающих корректность ключевых операций.

В ходе работы возникли и были решены следующие проблемы: корректная настройка внешних ключей и каскадного удаления, реализация корректного хеширования PIN, обеспечение атомарности транзакций при снятии/пополнении средств. Для дальнейшего развития проекта можно рекомендовать добавление GUI, интеграцию с внешней системой аутентификации и расширение набора отчётов для аналитики.

Приложение. Программный код и SQL-скрипты

Ниже приведены ключевые фрагменты кода и SQL-скрипты. Полный код приложения находится в проектном архиве.

SQL: Скрипт создания таблиц (фрагмент)

```
-- Таблица Clients
CREATE TABLE Clients (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    FirstName NVARCHAR(100) NOT NULL,
    LastName NVARCHAR(100) NOT NULL,
    PassportNo NVARCHAR(50) NOT NULL UNIQUE,
    Phone NVARCHAR(20) NOT NULL
);

-- Таблица Accounts
CREATE TABLE Accounts (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    Number NVARCHAR(50) NOT NULL UNIQUE,
    Currency NVARCHAR(3) NOT NULL DEFAULT 'UZS',
    Balance DECIMAL(18,2) NOT NULL DEFAULT 0.00
);
```

C#: Фрагменты реализации сервисов

```
public bool Authenticate(string cardNumber, string pin)
{
    var card = db.Cards.FirstOrDefault(c => c.CardNumber == cardNumber);
    if (card == null || card.Status != 1) return false;
    return card.PinHash == Hash(pin);
}

public bool Withdraw(string cardNumber, decimal amount)
{
    using var transaction = db.Database.BeginTransaction();
    var card = db.Cards.Include(c => c.Account).FirstOrDefault(c => c.CardNumber == cardNumber);
    if (card == null || card.Account.Balance < amount) return false;
    card.Account.Balance -= amount;
    db.Transactions.Add(new Transaction { CardId = card.Id, Amount = amount, Type = 1, BalanceAfter = card.Account.Balance });
    db.SaveChanges();
    transaction.Commit();
    return true;
}
```