



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени
Н.Э. Баумана
(национальный исследовательский университет)» (МГТУ
им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____ «Фундаментальные науки»

КАФЕДРА _____ «Математическое моделирование»

ОТЧЁТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Студент _____ Конононенко Артём Александрович

фамилия, имя, отчество

Группа ФН12-21М

Тип практики _____ научно-исследовательская работа

Название предприятия _____ МГТУ имени Н. Э. Баумана

Студент

подпись, дата

Конононенко А.А.

фамилия, и.о.

Руководитель практики

подпись, дата

Вишняков И.Э.

фамилия, и.о.

Оценка _____

2024 г.

ЗАДАНИЕ

на прохождение производственной практики

Студент Конононенко Артём Александрович / Конононенко А.А. / ФН12-21М
(фамилия, имя, отчество; инициалы; индекс группы)

1. Построить модель «сущность-связь» для выбранной предметной области.
2. Преобразовать модель «сущность-связь» в реляционную модель.
3. Реализовать БД средствами СУБД SQL Server.

Руководитель практики от кафедры _____ / _____
(подпись, дата)

Студент _____ / _____
(подпись, дата) (Фамилия И.О.)

ОГЛАВЛЕНИЕ

Введение	4
1 Задача	5
2 Практическая реализация	6
2.1 Предметная область и требования к ней	6
2.2 Модель «сущность-связь»	6
3. Постановка задачи	8
4. Практическая реализация	9
4.1 Реляционная модель	9
4.2 Обоснование правил обеспечения ограничений минимальной кардинальности	12
5. Реализация базы данных средствами СУБД SQL Server	14
Заключение	22
ПРИЛОЖЕНИЕ А	23

Введение

База данных и система управления базой данных (СУБД) являются важными компонентами информационных систем компании. Процесс создания базы данных включает несколько шагов: от словесного описания информации до формализации объектов в рамках определенной модели. Этапы проектирования включают анализ и описание предметной области, создание концепции, разработку логической структуры и физическую реализацию базы данных.

Цель данной исследовательской работы - разработка модели "сущность-связь" и ее реализация в SQL Server.

Для этого нужно выполнить ряд задач: выбрать предметную область, определить требования к ней, создать модель "сущность-связь", преобразовать ее в реляционную модель, выбрать типы данных, ключи и правила ограничений, а также реализовать базу данных с использованием SQL Server.

1 Задача

1. Выбрать простейшую предметную область, соответствующую 4-5 сущностям.
2. Сформировать требования к предметной области.
3. Создать модель «сущность-связь» для предметной области с обоснованием выбора кардинальных чисел связей.

2 Практическая реализация

2.1 Предметная область и требования к ней

Для реализации задачи в качестве предметной области была выбран банк. Данная область подразумевает выполнение сделок клиентов в банкоматах, принадлежащих банку. В связи с этим к предметной области были сформулированы следующие требования:

- Узнать баланс карты.
- Выдать деньги (клиенту).
- Переводить деньги на карту (принимать деньги).
- Печать чеков.
- Работа с картами различных банков.

2.2 Модель «сущность-связь»

Для построения предложенной модели «сущность-связь» были выделены четыре сущности:

- Клиент – сущность клиента с идентификатором “# паспорта” и атрибутами: фамилия, имя, отчество, пол.
- Карта – сущность банковской карты клиента с идентификатором “# карты” и атрибутами: название банка, тип карты (кредитка или сберегательная).
- Сделка – сущность сделки между клиентом и банкоматом с идентификатором “# сделки” и атрибутам: счёт отправителя, сумма, код авторизации, счёт получателя, дата и время, адрес банкомата, QR-код сканирования.
- Банкомат – сущность банкомата с идентификатором “# банкомата” и атрибутами: код банкомата, номер отделения.

Между выделенными сущностями были построены связи, отвечающие ранее сформулированным требованиям.

Клиент – Карта: у карты может быть ровно один клиент, а у клиента может быть от нуля, до нескольких карт, клиент связан с заказом связью типа «один-ко-многим», а минимальное кардинальное число у клиента равно 1, а у карты – 0.

Клиент может не иметь карты, но у карты обязан быть владелец, так как при создании карты, она к какому-то клиенту банка прикрепляется.

Карта – Сделка: между этими сущностями возникает связь типа «один-ко-многим» с минимальными кардинальными числами 1 у карты и 0 у сделки, так как с помощью карты можно совершить от 1 до N заказов, а каждый заказ должен быть оплачен картой.

Банкомат – Сделка: связь типа «один-ко-многим», с минимальных кардинальными числами 1 для банкомата и 0 для сделки, так как сделка совершается только с помощью банкомата, но у нового банкомата может сделок и не быть.

ER-модель для работы банкомата представлена на Рис. 1.



Рис. 1. Модель «сущность-связь»

3. Постановка задачи

1. Преобразовать модель «сущность-связь», созданную в лабораторной работе №1, в реляционную модель согласно процедуре преобразования.
2. Обосновать выбор типов данных, ключей, правил обеспечения ограничений минимальной кардинальности.

4. Практическая реализация

4.1 Реляционная модель

На основании модели «сущность-связь», изображённой на рисунке 1 была получена реляционная модель, изображённая на рисунке 2.

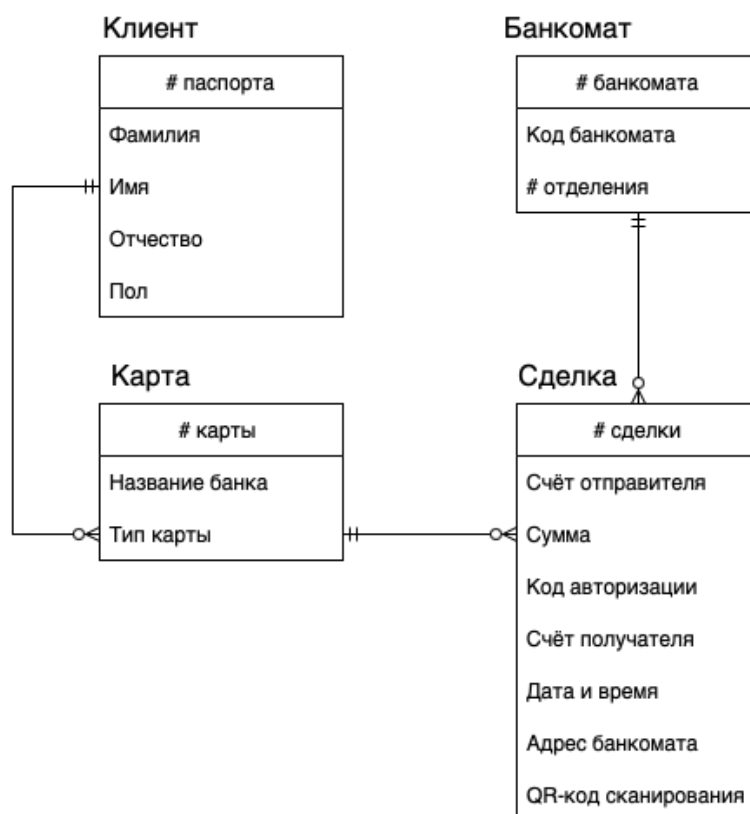


Рисунок 1 – модель «сущность-связь»

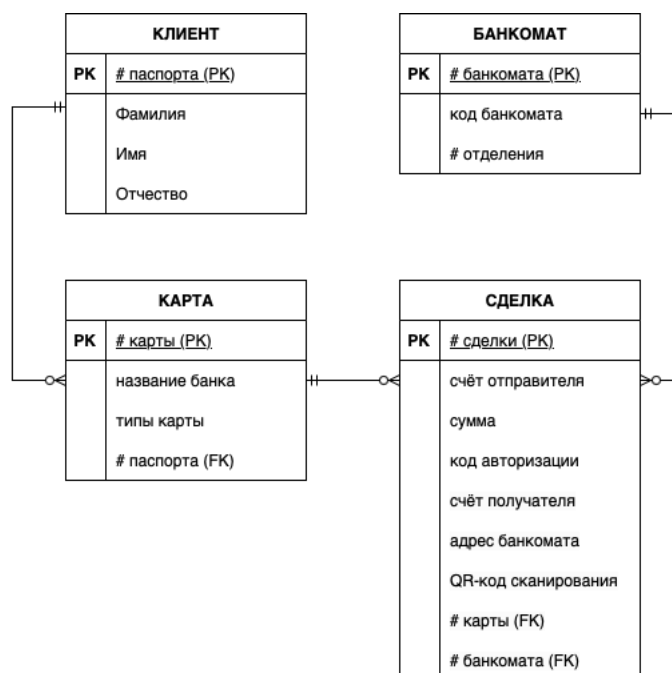


Рисунок 2 – реляционная модель

А также реализованы таблицы для каждой сущности. В таблице 2.1.1 представлены типы данных и их значения по умолчанию для сущности КЛИЕНТ.

Таблица 2.1.1 – CLIENT

Column Name	Type	Key	NULL Status	Remarks
# паспорта	Int	Primary	NOT NULL	
Имя	Nvarchar (30)	No	NOT NULL	AK 1.1
Фамилия	Nvarchar (30)	No	NOT NULL	AK 1.2
Отчество	Nvarchar (30)	No	NOT NULL	AK 1.3
Email	Int	No	NOT NULL	AK 1.4

В таблице 2.1.2 представлены типы данных и их значения по умолчанию для сущности Карта.

Таблица 2.1.2 – Карта

Column Name	Type	Key	NULL Status	Remarks
-------------	------	-----	-------------	---------

# карты	Int	Primary	NOT NULL	
Название банка	Nvarchar(30)	No	NOT NULL	
Тип карты	Nvarchar(30)	No	NOT NULL	
# паспорта	Int	Foreign	NOT NULL	

В таблице 2.1.3 представлены типы данных и их значения по умолчанию для сущности Сделка.

Таблица 2.1.3 – Сделка

Column Name	Type	Key	NULL Status	Remarks
# сделки	Bigint	Primary	NOT NULL	
Счёт отправителя	Int	No	NOT NULL	
Сумма	float	No	NULL	
Код авторизации	Int	No	NOT NULL	
Счёт получателя	Int	No	NOT NULL	
Дата и время	Datetime	No	NOT NULL	
Адрес банкомата	Nvarchar(50)	No	NOT NULL	
QR-код сканирования	Int	No	NOT NULL	
# карты	Int	Foreign	NOT NULL	
# банкомата	Int	Foreign	NOT NULL	

В таблице 2.1.4 представлены типы данных и их значения по умолчанию для сущности Банкомат.

Таблица 2.1.4 – Банкомат

Column Name	Type	Key	NULL Status	Remarks
# Сделки	Int	Primary	NOT NULL	
Код банкомата	Int	No	NOT NULL	
Номер отделения	Int	No	NOT NULL	

4.2 Обоснование правил обеспечения ограничений минимальной кардинальности

Обоснование правил обеспечения ограничений минимальной кардинальности приведено на следующих таблицах:

1) Клиент к Карта идентифицирующая связь М-О 1:N;

Таблица 2.2.1 Клиент к Карта

Клиент Обязательный родитель	Действия для Клиент (родитель)	Действия для Клиент Карта (ребенок)
Вставка	Без ограничений	Указать родителя.
Изменение первичного или внешнего ключа	Каскадное обновление	Запрещено – клиент не может меняться.
Удаление	Каскадное удаление ребенка	Без ограничений

2) Карта к Сделка идентифицирующая связь М-О 1:N;

Таблица 2.2.3 Карта к Сделка

Карта Обязательный родитель	Действия для Карта (родитель)	Действия для Сделка Карта (ребенок)
Вставка	Без ограничений	Указать родителя.
Изменение первичного или внешнего ключа	запрет	Запрещено – карта не может меняться.
Удаление	Каскадное удаление ребенка.	Без ограничений

3) Банкомат к Сделка идентифицирующая связь М-О 1:N;

Таблица 2.2.3 Банкомат к Сделка

Банкомат Обязательный родитель	Действия для Банкомат (родитель)	Действия для Сделка (ребенок)
Вставка	Без ограничений	Указать родителя.

Изменение первичного или внешнего ключа	Каскадное обновление	Запрет
Удаление	Каскадное удаление ребенка.	Без ограничений

5. Реализация базы данных средствами СУБД SQL Server

Создание базы данных, представленной на рисунке 3, было осуществлено с применением языка запросов Transact-SQL.

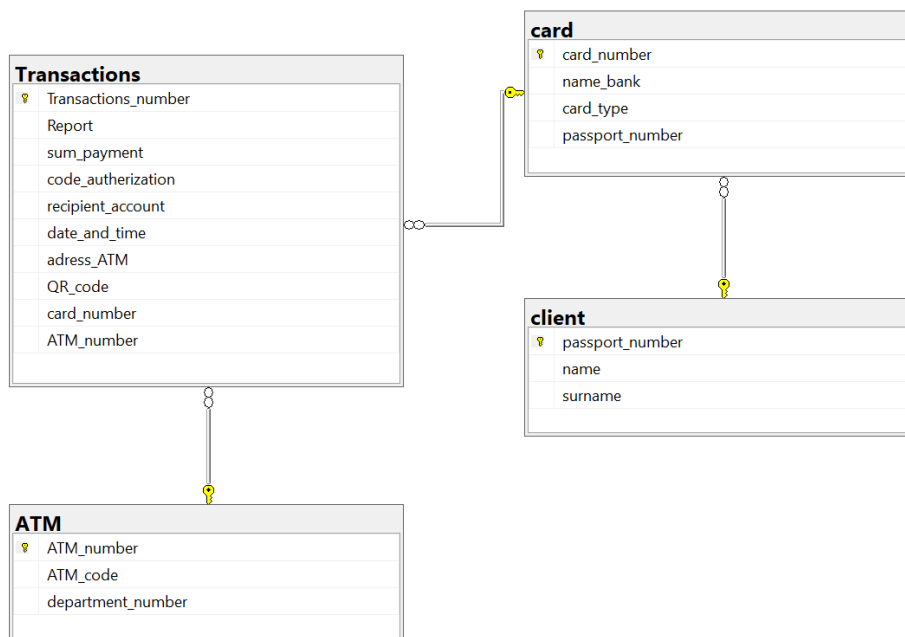


Рисунок 3 – Схема базы данных

В этой базе данных были сформированы следующие запросы для манипуляции данными (DML):

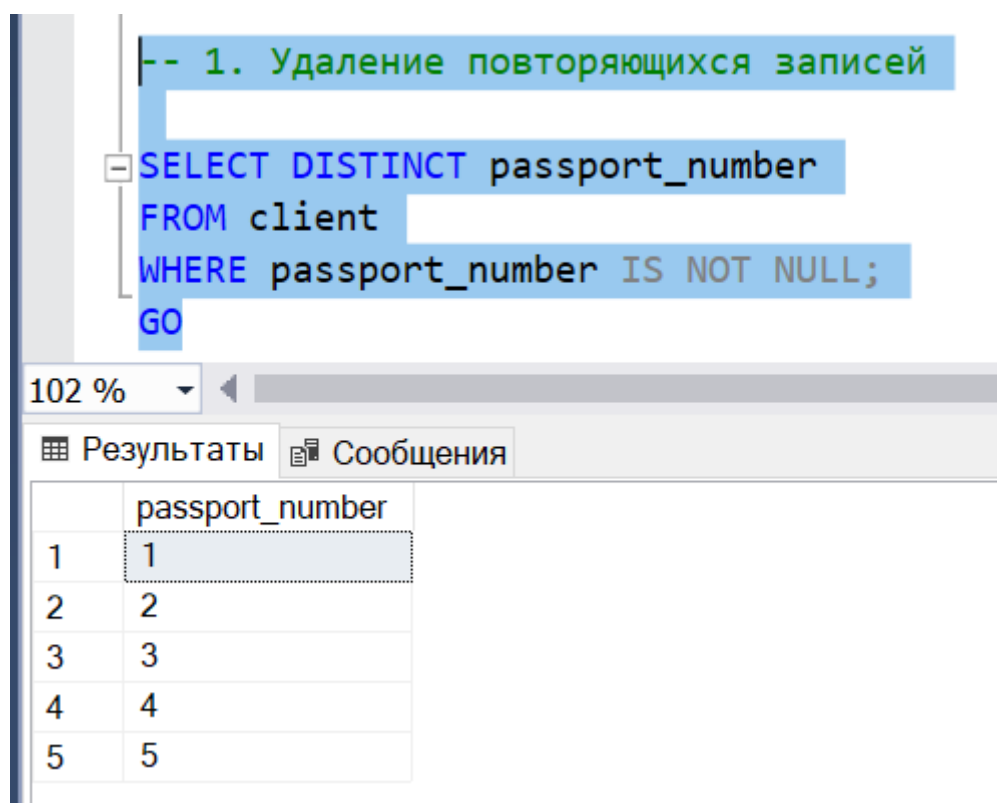
- выборка записей (SELECT),
- добавление записей (INSERT), включая варианты с использованием SELECT,
- изменение записей (UPDATE) и удаление записей (DELETE).

Были продемонстрированы такие возможности языка, как удаление дубликатов (DISTINCT), выбор, сортировка и переименование полей,

соединение таблиц (INNER JOIN, LEFT JOIN), установка условий для выбора записей, включая условия LIKE, BETWEEN, IN, сортировка записей (ORDER BY), группировка записей (GROUP BY + HAVING), использование функций агрегирования (COUNT, AVG, SUM, MIN, MAX), объединение результатов нескольких запросов и вложенные запросы.

Листинг кода представлен в приложении.

Для демонстрации выполним следующие задачи:



```
-- 1. Удаление повторяющихся записей
SELECT DISTINCT passport_number
FROM client
WHERE passport_number IS NOT NULL;
GO
```

102 %

Результаты Сообщения

	passport_number
1	1
2	2
3	3
4	4
5	5

-- 2. Выбор, упорядочивание и именование полей

```
SELECT CONCAT_WS('.', LEFT(name, 1), LEFT(surname, 1)) as FI  
FROM client  
ORDER BY FI;  
GO
```

```
SELECT CONCAT_WS('.', LEFT(name, 1), LEFT(surname, 1)) as FI  
FROM client  
ORDER BY FI DESC;  
GO
```

102 %

Результаты Сообщения

	FI
1	A.B
2	E.G
3	K.R
4	M.T
5	N.K

	FI
1	N.K
2	M.T
3	K.R
4	E.G
5	A.B


```
-- 3. Соединение таблиц
--3.1. INNER JOIN

SELECT *
FROM client INNER JOIN card ON client.passport_number = card.passport_number;
GO

-- 3.2. LEFT JOIN

SELECT *
FROM client LEFT OUTER JOIN card ON client.passport_number = card.passport_number;
GO

-- 3.3. RIGHT JOIN

SELECT *
FROM client RIGHT OUTER JOIN card ON client.passport_number = card.passport_number;
GO

-- 3.4. CROSS JOIN

SELECT *
FROM client CROSS JOIN card;
GO

SELECT 'FULL OUTER JOIN';
GO

-- 3.5. FULL OUTER JOIN

SELECT *
FROM card FULL OUTER JOIN client ON client.passport_number = card.passport_number;
GO
```

102 %

Результаты Сообщения

	passport_number	name	surname	card_number	name_bank	card_type	passport_number
1	2	Ksenia	Rumyantseva	1	Sber	credit_card	2
2	1	Egor	Golovin	2	Sber	credit_card	1
3	2	Ksenia	Rumyantseva	3	Uralsib	cumulative_card	2

	passport_number	name	surname	card_number	name_bank	card_type	passport_number
1	1	Egor	Golovin	2	Sber	credit_card	1
2	2	Ksenia	Rumyantseva	1	Sber	credit_card	2
3	2	Ksenia	Rumyantseva	3	Uralsib	cumulative_card	2
4	3	Nikita	Kochnov	NULL	NULL	NULL	NULL
5	4	Artem	Burin	NULL	NULL	NULL	NULL
6	5	Maxim	Tyatin	NULL	NULL	NULL	NULL

	passport_number	name	surname	card_number	name_bank	card_type	passport_number
1	2	Ksenia	Rumyantseva	1	Sber	credit_card	2
2	1	Egor	Golovin	2	Sber	credit_card	1
3	2	Ksenia	Rumyantseva	3	Uralsib	cumulative_card	2

	passport_number	name	surname	card_number	name_bank	card_type	passport_number
1	1	Egor	Golovin	1	Sber	credit_card	2
2	2	Ksenia	Rumyantseva	1	Sber	credit_card	2
3	3	Nikita	Kochnov	1	Sber	credit_card	2
4	4	Artem	Burin	1	Sber	credit_card	2
5	5	Maxim	Tyatin	1	Sber	credit_card	2
6	1	Egor	Golovin	2	Sber	credit_card	1
7	2	Ksenia	Rumyantseva	2	Sber	credit_card	1
8	3	Nikita	Kochnov	2	Sber	credit_card	1

(Отсутствует имя столбца)

	card_number	name_bank	card_type	passport_number	passport_number	name	surname
1	1	Sber	credit_card	2	2	Ksenia	Rumyantseva
2	2	Sber	credit_card	1	1	Egor	Golovin
3	3	Uralsib	cumulativ	2	2	Ksenia	Rumyantseva

Запрос успешно выполнен. HOMEWORKPC\MSSQL22EXP (16.0... HOMEWORKPC\Rust_su (63) Bank_company_11 00:00:00 34 строки

```
-- 4. Условия выбора записей (в том числе, условия / LIKE / BETWEEN / IN / EXISTS)
-- 4.1. BETWEEN
SELECT *
FROM Transactions
WHERE sum_payment BETWEEN 1500 AND 3000;
GO

-- 4.2. LIKE (=)
SELECT *
FROM Transactions
WHERE sum_payment LIKE 1000;
GO

SELECT *
FROM Transactions
WHERE sum_payment = 1000;
GO

-- 4.3. IN
SELECT *
FROM Transactions
WHERE sum_payment IN (1000, 2500) ;
GO

-- 4.4. EXISTS +-
SELECT *
FROM card
WHERE EXISTS (SELECT *
FROM client
WHERE client.passport_number = card.passport_number);SSS
GO
```

	Transactions_number	Report	sum_payment	code_authorization	recipient_account	date_and_time	adress_ATM	QR_code	card_number	ATM_number
1	2	91598721	2500	78235	29384736	2022-05-01 08:45:00.000	89654374	7654	2	2
2	4	91862511	2000	46582	62542190	2022-05-01 09:00:00.000	89654372	7483	2	2
3	5	91962511	3000	46582	62542190	2022-05-01 08:50:00.000	89654372	7483	1	1

	Transactions_number	Report	sum_payment	code_authorization	recipient_account	date_and_time	adress_ATM	QR_code	card_number	ATM_number
1	1	91655511	1000	34582	66342190	2022-05-01 08:30:00.000	89654372	2183	1	1
2	3	91762511	1000	46582	62542190	2022-05-01 08:40:00.000	89654372	7483	1	1

	Transactions_number	Report	sum_payment	code_authorization	recipient_account	date_and_time	adress_ATM	QR_code	card_number	ATM_number
1	1	91655511	1000	34582	66342190	2022-05-01 08:30:00.000	89654372	2183	1	1
2	3	91762511	1000	46582	62542190	2022-05-01 08:40:00.000	89654372	7483	1	1

	Transactions_number	Report	sum_payment	code_authorization	recipient_account	date_and_time	adress_ATM	QR_code	card_number	ATM_number
1	1	91655511	1000	34582	66342190	2022-05-01 08:30:00.000	89654372	2183	1	1
2	2	91598721	2500	78235	29384736	2022-05-01 08:45:00.000	89654374	7654	2	2
3	3	91762511	1000	46582	62542190	2022-05-01 08:40:00.000	89654372	7483	1	1

	card_number	name_bank	card_type	passport_number
1	1	Sber	credit_card	2
2	2	Sber	credit_card	1
3	3	Uralsib	cumulative_card	2

Запрос успешно выполнен. HOMEWORKPC\MSSQL22EXP (16.0... HOMEWORKPC\Rust-su (63) Bank_company_11 00:00:00 3 строки

```
-- 5. Группировка записей (GROUP BY + HAVING, использование функций агрегирования (COUNT / AVG / SUM / MIN / MAX))

-- 5.1. GROUP BY + COUNT
SELECT sum_payment, COUNT(*) AS COUNT
FROM Transactions
GROUP BY sum_payment;
GO

-- 5.2. GROUP BY + AVG
SELECT card_number, AVG(sum_payment) AS 'AVG(RAM)'
FROM Transactions
GROUP BY card_number;
GO

-- 5.3. GROUP BY + SUM
SELECT card_number, SUM(sum_payment) AS 'SUM(RAM)'
FROM Transactions
GROUP BY card_number;
GO

-- 5.4. GROUP BY + MIN
SELECT card_number, MIN(sum_payment) AS 'MIN(RAM)'
FROM Transactions
GROUP BY card_number;
GO

-- 5.5. GROUP BY + MAX +-
SELECT card_number, MAX(sum_payment) AS 'MAX(RAM)'
FROM Transactions
GROUP BY card_number
--HAVING MAX(sum_payment) = 1000;
GO
```

102 %

Результаты Сообщения

	sum_payment	COUNT
1	1000	2
2	2000	1
3	2500	1
4	3000	1

	card_number	AVG(RAM)
1	1	1666,66666666667
2	2	2250

	card_number	SUM(RAM)
1	1	5000
2	2	4500

	card_number	MIN(RAM)
1	1	1000
2	2	2000

	card_number	MAX(RAM)
1	1	3000
2	2	2500

```

-- 6. Объединение результатов нескольких запросов (UNION / UNION ALL / EXCEPT / INTERSECT)

-- 6.1. UNION
SELECT *
FROM Transactions
WHERE sum_payment = 1000

UNION

SELECT *
FROM Transactions
WHERE sum_payment = 2500;
GO

-- 6.1.2. UNION2
SELECT *
FROM Transactions
WHERE sum_payment = 1000

UNION

SELECT *
FROM Transactions
WHERE sum_payment = 1000
GO

```

102 %

Результаты Сообщения

	Transactions_number	Report	sum_payment	code_authorization	recipient_account	date_and_time	adress_ATM	QR_code	card_number	ATM_number
1	1	91655511	1000	34582	66342190	2022-05-01 08:30:00.000	89654372	2183	1	1
2	3	91762511	1000	46582	62542190	2022-05-01 08:40:00.000	89654372	7483	1	1
3	2	91598721	2500	78235	29384736	2022-05-01 08:45:00.000	89654374	7654	2	2

	Transactions_number	Report	sum_payment	code_authorization	recipient_account	date_and_time	adress_ATM	QR_code	card_number	ATM_number
1	1	91655511	1000	34582	66342190	2022-05-01 08:30:00.000	89654372	2183	1	1
2	3	91762511	1000	46582	62542190	2022-05-01 08:40:00.000	89654372	7483	1	1

```
-- 6.2. UNION ALL
SELECT *
FROM Transactions
WHERE sum_payment = 1000

UNION ALL

SELECT *
FROM Transactions
WHERE sum_payment = 1000
GO
```

```
-- 6.3. EXCEPT
SELECT *
FROM Transactions
WHERE sum_payment IN (1000, 2000, 2500)

EXCEPT

SELECT *
FROM Transactions
WHERE sum_payment = 1000;
GO
```

```
-- 6.4. INTERSECT
SELECT *
FROM Transactions
WHERE sum_payment IN (1000, 2000, 2500)

INTERSECT

SELECT *
FROM Transactions
WHERE sum_payment = 1000;
GO
```

102 %

Результаты Сообщения

	Transactions_number	Report	sum_payment	code_authorization	recipient_account	date_and_time	adress_ATM	QR_code	card_number	ATM_number
1	1	91655511	1000	34582	66342190	2022-05-01 08:30:00.000	89654372	2183	1	1
2	3	91762511	1000	46582	62542190	2022-05-01 08:40:00.000	89654372	7483	1	1
3	1	91655511	1000	34582	66342190	2022-05-01 08:30:00.000	89654372	2183	1	1
4	3	91762511	1000	46582	62542190	2022-05-01 08:40:00.000	89654372	7483	1	1

	Transactions_number	Report	sum_payment	code_authorization	recipient_account	date_and_time	adress_ATM	QR_code	card_number	ATM_number
1	2	91598721	2500	78235	29384736	2022-05-01 08:45:00.000	89654374	7654	2	2
2	4	91862511	2000	46582	62542190	2022-05-01 09:00:00.000	89654372	7483	2	2

	Transactions_number	Report	sum_payment	code_authorization	recipient_account	date_and_time	adress_ATM	QR_code	card_number	ATM_number
1	1	91655511	1000	34582	66342190	2022-05-01 08:30:00.000	89654372	2183	1	1
2	3	91762511	1000	46582	62542190	2022-05-01 08:40:00.000	89654372	7483	1	1

```
-- 7.1. Вложенные запросы

SELECT *
FROM Transactions
WHERE ATM_number IN (SELECT T.ATM_number
                     FROM Transactions AS T
                     WHERE T.sum_payment IN (2000, 2500));
GO
```

102 %

Результаты Сообщения

	Transactions_number	Report	sum_payment	code_authorization	recipient_account	date_and_time	adress_ATM	QR_code	card_number	ATM_number
1	2	91598721	2500	78235	29384736	2022-05-01 08:45:00.000	89654374	7654	2	2
2	4	91862511	2000	46582	62542190	2022-05-01 09:00:00.000	89654372	7483	2	2

Заключение

В данной работе была определена предметная область для создания базы данных и установлены требования к ней. Далее была разработана ER-модель базы данных "Банк" и обоснован выбор кардинальных чисел связи. После этого было произведено преобразование спроектированной ER-модели в реляционную модель с обоснованием типов данных, ключей, правил обеспечения минимальной кардинальности. Изучены основные возможности языка запросов Transact-SQL. База данных реализована с использованием СУБД SQL Server. В базе данных реализованы все запросы, определенные на этапе проектирования, а также описаны механизмы обеспечения целостности данных в выбранной системе управления базами данных.

ПРИЛОЖЕНИЕ А

-- Создание базы данных

```
USE master;  
GO
```

```
DECLARE @SQL nvarchar(1000);  
IF EXISTS (SELECT 1 FROM sys.databases WHERE [name] = N'Bank_company_11')  
BEGIN  
SET @SQL = N'USE [Bank_company_11];
```

```
ALTER DATABASE Bank_company_11 SET SINGLE_USER WITH ROLLBACK IMMEDIATE;  
USE [tempdb];
```

```
DROP DATABASE Bank_company_11;';  
EXEC (@SQL);  
END;  
GO
```

```
DROP DATABASE IF EXISTS Bank_company_11;  
GO
```

```
CREATE DATABASE Bank_company_11  
ON (NAME = Research_DB_lab_7_dat, FILENAME = 'C:\NIR\Bank_company_11.mdf',  
SIZE = 10, MAXSIZE = UNLIMITED, FILEGROWTH = 5% )  
LOG ON ( NAME = Research_DB_lab_7_log, FILENAME = 'C:\NIR\Bank_company_11.ldf',  
SIZE = 5MB, MAXSIZE = 25MB, FILEGROWTH = 5MB );  
GO
```

-- Создание таблицы "Клиент"

```
USE Bank_company_11;  
GO
```

-- Создание и заполнение таблицы "Клиент"

```
CREATE TABLE client (passport_number int not null primary key,  
name varchar(30) not null,  
surname varchar(30) not null)
```

```
insert into client(passport_number,name,surname)  
values  
(1,'Egor','Golovin'),  
(2,'Ksenia','Rumyantseva'),  
(3,'Nikita','Kochnov'),  
(4,'Artem','Burin'),  
(5,'Maxim','Tyatin');  
GO
```

-- Создание и заполнение таблицы "Карта (банковская)"

```
CREATE TABLE card  
(card_number int primary key,  
name_bank varchar(30) not null,  
card_type Nvarchar(30) not null,  
passport_number int not null,  
FOREIGN KEY (passport_number) REFERENCES client(passport_number)  
ON DELETE CASCADE ON UPDATE CASCADE);  
GO
```

```
insert into card  
values (1,'Sber','credit_card', 2),  
(2,'Sber','credit_card', 1),  
(3,'Uralsib','cumulative_card', 2);  
GO
```

```

SELECT *
FROM card;
GO
--GO
--CREATE TABLE trade (number_trade NVARCHAR(30) not null primary key,
--sender_number varchar(30) not null,
--sum varchar(30) not null,

--) --ON MyFileGroup

--insert into trade(passport,name,surname) values (1234876511,'Egor','Golovin'),
--(4455332271,'Ksenia','Rumyantseva');

-- Создание таблицы "Банкомат"

CREATE TABLE ATM (ATM_number int not null primary key,
ATM_code int not null,
department_number int not null)

insert into ATM(ATM_number,ATM_code,department_number) values (1,9822,1),
(2,9723,2);

GO

-- Создание таблицы "Сделки"

create table Transactions
(
    Transactions_number int not null primary key,
    Report int not null,
    sum_payment float,
    code_authorization int not null,
    recipient_account int not null, --счёт получателя
    date_and_time Datetime not null,
    adress_ATM Nvarchar(50) not null,
    QR_code int not null,
    card_number int not null,
    FOREIGN KEY (card_number) REFERENCES card(card_number) ON -- внешний ключ ссылается
на таблицу card
    DELETE CASCADE ON UPDATE CASCADE ,
    ATM_number int not null,
    FOREIGN KEY (ATM_number) REFERENCES ATM(ATM_number) ON -- внешний ключ ссылается на
таблицу ATM
    DELETE CASCADE ON UPDATE CASCADE
);

insert into
Transactions(Transactions_number,Report,sum_payment,code_authorization,recipient_account,dat
e_and_time,adress_ATM, QR_code,card_number,ATM_number)
values (1, 91655511 , 1000, 34582, 66342190, '2022-01-05 08:30:00', 89654372, 2183, 1, 1 ),
(2, 91598721, 2500, 78235, 29384736, '2022-01-05 08:45:00', 89654374, 7654, 2, 2),
(3, 91762511, 1000, 46582, 62542190, '2022-01-05 08:40:00', 89654372, 7483, 1, 1 ),
(4, 91862511, 2000, 46582, 62542190, '2022-01-05 09:00:00', 89654372, 7483, 2, 2 ),
(5, 91962511, 3000, 46582, 62542190, '2022-01-05 08:50:00', 89654372, 7483, 1, 1 );
GO

SELECT * FROM client;
SELECT * FROM card;

```



```
SELECT * FROM ATM;
SELECT * FROM Transactions;
```

-- 1. Удаление повторяющихся записей

```
SELECT DISTINCT passport_number
FROM client
WHERE passport_number IS NOT NULL;
GO
```

-- 2. Выбор, упорядочивание и именование полей

```
SELECT CONCAT_WS('.', LEFT(name, 1), LEFT(surname, 1)) as FI
FROM client
ORDER BY FI;
GO
```

```
SELECT CONCAT_WS('.', LEFT(name, 1), LEFT(surname, 1)) as FI
FROM client
ORDER BY FI DESC;
GO
```

-- 3. Соединение таблиц

--3.1. INNER JOIN

```
SELECT *
FROM client INNER JOIN card ON client.passport_number = card.passport_number;
GO
```

-- 3.2. LEFT JOIN

```
SELECT *
FROM client LEFT OUTER JOIN card ON client.passport_number = card.passport_number;
GO
```

-- 3.3. RIGHT JOIN

```
SELECT *
FROM client RIGHT OUTER JOIN card ON client.passport_number = card.passport_number;
GO
```

-- 3.4. CROSS JOIN

```
SELECT *
FROM client CROSS JOIN card;
GO
```

```
SELECT 'FULL OUTER JOIN';
GO
```

-- 3.5. FULL OUTER JOIN

```
SELECT *
FROM card FULL OUTER JOIN client ON client.passport_number = card.passport_number;
GO
```

-- 4. Условия выбора записей (в том числе, условия / LIKE / BETWEEN / IN / EXISTS)

-- 4.1. BETWEEN

```
SELECT *
FROM Transactions
WHERE sum_payment BETWEEN 1500 AND 3000;
GO
```

-- 4.2. LIKE (=)

```
SELECT *
FROM Transactions
WHERE sum_payment LIKE 1000;
GO
```

```
SELECT *
FROM Transactions
WHERE sum_payment = 1000;
GO
```

-- 4.3. IN

```
SELECT *
FROM Transactions
WHERE sum_payment IN (1000, 2500) ;
GO
```

-- 4.4. EXISTS +-

```
SELECT *
FROM card
WHERE EXISTS (SELECT *
              FROM client
              WHERE client.passport_number = card.passport_number);
GO
```

-- 5. Группировка записей (GROUP BY + HAVING, использование функций агрегирования (COUNT / AVG / SUM / MIN / MAX)

-- 5.1. GROUP BY + COUNT

```
SELECT sum_payment, COUNT(*) AS COUNT
FROM Transactions
GROUP BY sum_payment;
GO
```

-- 5.2. GROUP BY + AVG

```
SELECT card_number, AVG(sum_payment) AS 'AVG(RAM)'
FROM Transactions
GROUP BY card_number;
GO
```

-- 5.3. GROUP BY + SUM

```
SELECT card_number, SUM(sum_payment) AS 'SUM(RAM)'
FROM Transactions
GROUP BY card_number;
GO
```

-- 5.4. GROUP BY + MIN

```
SELECT card_number, MIN(sum_payment) AS 'MIN(RAM)'
```

```

FROM Transactions
GROUP BY card_number;
GO

-- 5.5. GROUP BY + MAX +-

SELECT card_number, MAX(sum_payment) AS 'MAX(RAM)'
FROM Transactions
GROUP BY card_number
--HAVING MAX(sum_payment) = 1000;
GO

-- 6. Объединение результатов нескольких запросов (UNION / UNION ALL / EXCEPT / INTERSECT)

-- 6.1. UNION

SELECT *
FROM Transactions
WHERE sum_payment = 1000

UNION

SELECT *
FROM Transactions
WHERE sum_payment = 2500;
GO

-- 6.1.2. UNION2

SELECT *
FROM Transactions
WHERE sum_payment = 1000

UNION

SELECT *
FROM Transactions
WHERE sum_payment = 1000
GO

-- 6.2. UNION ALL

SELECT *
FROM Transactions
WHERE sum_payment = 1000

UNION ALL

SELECT *
FROM Transactions
WHERE sum_payment = 1000
GO

-- 6.3. EXCEPT

SELECT *
FROM Transactions
WHERE sum_payment IN (1000, 2000, 2500)

EXCEPT

SELECT *
FROM Transactions
WHERE sum_payment = 1000;

```

GO

-- 6.4. INTERSECT

```
SELECT *  
FROM Transactions  
WHERE sum_payment IN (1000, 2000, 2500)
```

INTERSECT

```
SELECT *  
FROM Transactions  
WHERE sum_payment = 1000;  
GO
```

-- 7.1. Вложенные запросы

```
SELECT *  
FROM Transactions  
WHERE ATM_number IN (SELECT T.ATM_number  
                      FROM Transactions AS T  
                      WHERE T.sum_payment IN (2000, 2500));  
GO
```

-- Удаление всех таблиц

```
/*  
Drop table IF EXISTS Transactions;  
  
DROP TABLE IF EXISTS name_bank;  
  
DROP TABLE IF EXISTS card;  
  
DROP TABLE IF EXISTS ATM;  
  
DROP TABLE IF EXISTS client;  
*/
```