Частное образовательное учреждение высшего образования

«Казанский инновационный университет имени В. Г. Тимирясова (ИЭУП)»

Факультет менеджмента и инженерного бизнеса

Кафедра компьютерного моделирования и техносферной безопасности

**КУРСОВОЙ ПРОЕКТ**

**по дисциплине «Базы данных»**

**на тему: «База данных «****Выдача банком кредитов»»**

Направление подготовки: 09.03.03 «Прикладная информатика»

**Выполнил:**

обучающийся гр. 1021

очной формы обучения

Хисаметдинов Ч.Р.

**Руководитель**:

доцент каф. КМиТБ  
Григорян К.А.

Казань, 2024 г.

**АННОТАЦИЯ**

на курсовой проект

по дисциплине «Базы данных»

|  |
| --- |
| Ф.И.О. студента: Хисаметдинов Чингиз Рафагатевич |
| Student’s Name: Khisametdinov Chingiz Rafagatevich |
| Направление: 09.03.03 «Прикладная информатика» |
| № группы: 1021 |

|  |
| --- |
| Название курсового проекта: База данных «Выдача банком кредитов» |
|  |
| Course project title: «Issuing loans by the bank» Database |
|  |

Аннотация:

|  |
| --- |
| В ходе данной курсовой работы была исследована, спроектирована и |
| разработана предметная область «Выдача банком кредитов». В качестве СУБД |
| был использован PostgreSQL. В процессе разработки базы данных были |
| созданы: ER-модель, запросы, представления, триггеры, хранимые |
| процедуры. |
|  |

Abstract:

|  |  |
| --- | --- |
| In the course of this work, the subject area «Issuing loans by the bank» has been researched, | |
| designed and developed. PostgreSQL has been used as the DBMS. During the | |
| development of the database, the following have been created: ER-model, queries, | |
| views, triggers, stored procedures. | |
|  | |
| Руководитель:  доцент Григорян К.А. | Перевод проверил:  старший преподаватель Курочкин Р.В. |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2024 г. | «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2024 г. |

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 4](#_Toc136383892)

[1. ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc136383893)

[2. КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 7](#_Toc136383894)

[3. ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 10](#_Toc136383895)

[4. ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 12](#_Toc136383896)

[5. ЗАПРОСЫ 17](#_Toc136383897)

[6. ПРЕДСТАВЛЕНИЯ 24](#_Toc136383898)

[7. ФУНКЦИИ 27](#_Toc136383899)

[8. ТРИГГЕРЫ 29](#_Toc136383900)

[ЗАКЛЮЧЕНИЕ 32](#_Toc136383901)

[СПИСОК ЛИТЕРАТУРЫ 33](#_Toc136383902)

# ВВЕДЕНИЕ

В настоящее время базы данных являются неотъемлемой частью информационных систем и используются в различных сферах деятельности. Одной из таких сфер является банковское кредитование, где базы данных позволяют хранить и обрабатывать информацию о клиентах, кредитах, задолженностях и других аспектах кредитной деятельности банка. В данной курсовой работе мы будем рассматривать проектирование и реализацию базы данных для автоматизации процесса выдачи банком кредитов.

Цель данной работы — формирование навыков практической работы студентов, связанной с разработкой и реализацией баз данных, а также закрепление, углубление и обобщение знаний, полученных в ходе изучения предмета. В ходе выполнения курсового проекта студенты будут осваивать общий методологический подход, используемый при проектировании и разработке баз данных, а также практически применять его для создания базы данных для автоматизации процесса выдачи банком кредитов.

Для выполнения работы необходимо проанализировать предметную область, построить концептуальную и логическую модель выбранной предметной области, выполнить построение физической модели базы данных, создать SQL-запросы, представления, хранимые процедуры, триггеры.

В качестве СУБД (системы управления базами данных) будет использован PostgreSQL.

Результатом данной работы будет готовая база данных, которая сможет эффективно хранить и обрабатывать информацию о клиентах, кредитах, задолженностях и других аспектах кредитной деятельности банка.

# 1. ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

В данной главе проводится анализ предметной области "Выдача банком кредитов" и описываются основные бизнес-процессы и сущности, связанные с данной предметной областью. Также рассматриваются требования к базе данных и описываются правила.

1.1 Описание бизнес-процессов и сущностей в предметной области

Банк — это финансовая организация, которая предоставляет услуги кредитования, депозитов, переводов и другие финансовые услуги. Основными бизнес-процессами в банке являются:

Поиск и оформление кредита — клиент выбирает подходящий кредитный продукт, подаёт заявку на кредит и получает одобрение от банка.

Выдача и погашение кредита — клиент получает заёмные средства, использует их по своему усмотрению и возвращает кредит банку с процентами в установленные сроки.

Учёт и контроль кредитных операций — банк ведёт учёт всех выданных и погашенных кредитов, контролирует соблюдение условий кредитных договоров и обеспечивает возврат заёмных средств.

1.2 Анализ требований к базе данных

Для эффективного управления кредитной деятельностью банка необходимо иметь электронную базу данных, которая будет хранить информацию о клиентах, кредитах, задолженностях и других сущностях. Требования к базе данных включают:

* Хранение информации о клиентах, кредитах, задолженностях и других сущностях.
* Учёт наличия и доступности кредитов.
* Оформление кредитов и контроль их выполнения.
* Регистрация клиентов и учёт их кредитной деятельности.
* Обеспечение безопасности данных и контроль доступа к ним.

1.3 Описание правил

Для реализации данных бизнес-процессов и управления сущностями в предметной области необходимо создание базы данных «Банк». Она должна содержать информацию о клиентах, кредитах, задолженностях и других сущностях.

Были выявлены следующие правила:

* Один клиент может получить несколько кредитов.
* Один кредит может быть выдан нескольким клиентам.
* Нельзя выдать кредит, если у клиента есть непогашенная задолженность.
* Нельзя выдать больше кредитов, чем доступно в банке.
* Кредит можно выдать на определённый срок, который должен быть указан при оформлении. Если кредит не погашен в срок, клиент обязан выплатить штраф.

Для учета всех этих данных необходима база данных, которая будет представлять все сущности и их связи. В следующих разделах будет описан процесс построения концептуальной и логической моделей базы данных.

# 2. КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

Сущности:

1. Клиент (client)

2. Cотрудник (employee)

3. Кредит (loan)

4. Залог (collateral)

5. Типы документов (types\_of\_documents)

6. Требуемые документы (required\_documents)

7. Договор о кредите (loan\_agreement)

8. Список документов (package\_of\_documents)

Таблица "Клиент" (client):

* client\_id (идентификатор клиента, целое число, первичный ключ)
* full\_name (ФИО клиента, не может быть NULL)
* email (электронная почта)
* phone\_num (номер телефона, целое число, не может быть NULL)

Таблица "Cотрудник" (employee):

* employee\_id (идентификатор сотрудника, целое число, первичный ключ)
* full\_name (ФИО сотрудника, строка, не может быть NULL)
* email (электронная почта, строка, не может быть NULL)
* snils (номер СНИЛС, строка, не может быть NULL)

Таблица "Кредит" (loan):

* loan\_id (идентификатор кредита, целое число, первичный ключ)
* loan\_name (наименование кредита, строка, не может быть NULL)
* min\_sum (минимальная сумма кредита, тип данных деньги, не может быть NULL)
* max\_sum (максимальная сумма кредита, дата, не может быть NULL)
* min\_month (минимальный период кредита в месяцах, целое число, не может быть NULL)
* max\_month (максимальный период кредита в месяцах, целое число не может быть NULL)
* min\_percent (минимальная ставка, фиксированная запятая, не может быть NULL)
* max\_percent (максимальная ставка, фиксированная запятая, не может быть NULL)

Таблица "Залог" (collateral):

* collateral\_id (идентификатор залога, целое число, первичный ключ)
* collateral\_type (вид залога, строка, не может быть NULL)
* decription (описание, тип данных текст)

Таблица "Типы документов" (types\_of\_documents):

* types\_of\_documents\_id (идентификатор вида документа, целое число, первичный ключ)
* document\_name (наименование документа, строка, не может быть NULL)

Таблица "Требуемые документы" (required\_documents):

* required\_documents\_id (идентификатор залога, целое число, первичный ключ)
* types\_of\_document\_id (идентификатор вида документа, целое число, внешний ключ, не может быть NULL)
* loan\_id (идентификатор кредита, целое число, внешний ключ, не может быть NULL)
* copy\_or\_original (метка – документ оригинал или копия, тип булев)

Таблица "Договор о кредите" (loan\_agreement):

* loan\_agreement\_id (идентификатор договора, целое число, первичный ключ)
* loan\_id (идентификатор кредита, целое число, внешний ключ, не может быть NULL)
* employee\_id (идентификатор сотрудника, целое число, внешний ключ, не может быть NULL)
* client\_id (идентификатор клиента, целое число, внешний ключ, не может быть NULL)
* date\_of\_signing (дата подписания, временная метка с часовым поясом, не может быть NULL)
* interest\_rate (ставка, фиксированная запятая, не может быть NULL)
* term\_in\_months (срок кредита, целое число, не может быть NULL)
* guarantor\_id (идентификатор поручителя, целое число, внешний ключ, не может быть NULL)
* collateral\_id (идентификатор залога, целое число, внешний ключ, не может быть NULL)
* date\_1st\_payment (дата первого платежа, временная метка с часовым поясом, не может быть NULL)
* sum\_of\_loan (сумма кредита, тип деньги, не может быть NULL)
* interval\_of\_payments (период оплаты, интервал, не может быть NULL)

Таблица "Список документов" (package\_of\_documents):

* package\_of\_documents\_id (идентификатор списка документов, целое число, первичный ключ)
* loan\_agreement\_id (идентификатор договора, целое число, внешний ключ, не может быть NULL)
* required\_documents\_id (идентификатор требуемых докуметов, целое число, внешний ключ, не может быть NULL)
* decription (описание, тип данных текст)

Отношения между сущностями:

- Отношение между таблицами client и loan\_agreement является один к одному (one-to-one), так как один договор относиться только к одному клиенту, так и один клиент относится к одному договору так как может оформить только один.

- Отношение между таблицами collateral и loan\_agreement также является один к одному (one-to-many), так как один залог может соотвествовать только одному договору, и наоборот.

- Отношение между таблицами employee и loan\_agreement является один ко многим (one-to-many), так как один сотрудник может составить множество договоров.

- Отношение между таблицами loan и loan\_agreement является один ко многим (one-to-many), так как один вид кредита может присутствовать во множестве договоров.

- Отношение между таблицами package\_of\_documents и loan\_agreement является один ко многим (one-to-many), так как один договор может присутствовать во множестве список документов.

- Отношение между таблицами loan и required\_documents является один ко многим (one-to-many), так как один вид кредита может присутствовать во множестве требуемых документов.

- Отношение между таблицами required\_documents и package\_of\_documents является многие ко многим (many-to-many), так как множество записей требуемого документа может присутствовать во множестве записей списков документов.

# 3. ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

Логическая модель базы данных представляет собой более детализированную версию концептуальной модели, описывая каждую таблицу и ее поля.

Описание связей между таблицами

Связи между таблицами в базе данных определяют отношения между данными и позволяют извлекать информацию из нескольких таблиц одновременно. В нашей базе данных существуют следующие связи:

1)Связь между таблицами "Договор о кредите" и "Клиент":

Поле "client\_id" в таблице "Договор о кредите" является внешним ключом, связанным с полем "client\_id" в таблице "Клиент".

Поле "guarantor\_id" в таблице "Договор о кредите" является внешним ключом, связанным с полем "client\_id" в таблице "Клиент".

Первая связь показывает, какой клиент оформил договор.

Вторая связь показывает, кто был назначен поручителем.

2)Связь между таблицами "Договор о кредите" и "Сотрудник":

Поле "employee\_id" в таблице "Договор о кредите" является внешним ключом, связанным с полем "employee\_id" в таблице "Сотрудник".

Эта связь показывает, какой сотрудник составлял данный договор.

3)Связь между таблицами "Договор о кредите" и "Залог":

Поле "collateral\_id" в таблице "Договор о кредите" является внешним ключом, связанным с полем "collateral\_id" в таблице "Выдача книг".

Эта связь показывает, какой залог был предоставлен для данного займа.

4)Связь между таблицами "Договор о кредите" и "Кредит":

Поле "loan\_id" в таблице "Кредит" является внешним ключом, связанным с полем "loan\_id" в таблице "Договор о кредите".

Эта связь показывает, какой кредит был предоставлен для данного займа.

5)Связь между таблицами "Пакет документов" и "Договор о кредите":

Поле "loan\_agreement\_id" в таблице "Договор о кредите" является внешним ключом, связанным с полем "loan\_agreement\_id" в таблице "Пакет документов".

Эта связь показывает, к пакету каких требуемых документов относится данный договор.

6)Связь между таблицами "Пакет документов" и "Требуемые документы":

Поле "required\_documents\_id" в таблице "Пакет документов" является внешним ключом, связанным с полем "required\_documents\_id" в таблице "Требуемые документы".

Эта связь показывает, к пакету каких договоров относятся данные документы.

6)Связь между таблицами "Кредит" и "Требуемые документы":

Поле "loan\_id" в таблице "Требуемые документы" является внешним ключом, связанным с полем "loan\_id" в таблице "Кредит".

Эта связь показывает, какому кредиту соответствует набор требуемых докуменов.

7)Связь между таблицами "Требуемые документы" и "Виды документов":

Поле "types\_of\_documents\_id" в таблице "Требуемые документы" является внешним ключом, связанным с полем "types\_of\_documents\_id" в таблице "Виды документов".

Эта связь показывает, набор требуемых документов.

Все эти связи помогают оптимизировать работу с базой данных и позволяют получать более полную информацию.

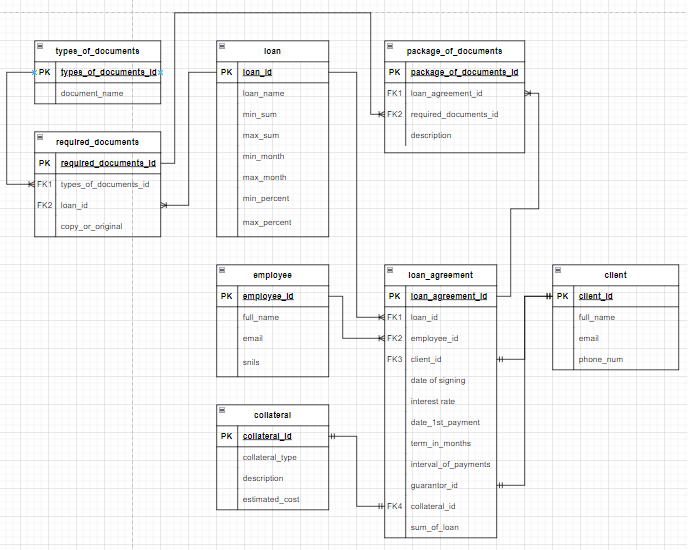


Рис. 1 – ER-модель базы данных «Выдача банком кредитов».

# 4. ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

4.1.Создание таблиц в СУБД

Код на SQL для создания таблицы «Кредит»:  
CREATE TABLE public.loan

(  
loan\_id SERIAL PRIMARY KEY,  
loan\_name VARCHAR(255),  
min\_sum MONEY NOT NULL,  
max\_sum MONEY NOT NULL,  
min\_month SMALLINT NOT NULL,  
max\_month SMALLINT NOT NULL,  
min\_percent DECIMAL(5, 2) NOT NULL,  
max\_percent DECIMAL(5, 2) NOT NULL);

Код на SQL для создания таблицы «Договор о кредите»:  
CREATE TABLE public.loan\_agreement

(  
loan\_agreement\_id SERIAL PRIMARY KEY,  
loan\_id INT REFERENCES public.loan(loan\_id),  
employee\_id INT NOT NULL,  
client\_id INT NOT NULL,  
date\_of\_signing INT NOT NULL,  
interest\_rate INT NOT NULL,  
date\_1st\_payment INT NOT NULL,  
term\_in\_months INT NOT NULL,  
interval\_of\_payments INT NOT NULL,  
guarantor\_id INT REFERENCES public.client(client\_id),  
collateral\_id INT REFERENCES public.collateral(collateral\_id)  
sum\_of\_loan MONEY NOT NUL);

Код на SQL для создания таблицы «Клиент»:  
CREATE TABLE public.client

(  
client\_id SERIAL PRIMARY KEY,  
full\_name VARCHAR(255),  
email VARCHAR(255),  
phone\_num varchar(15) NOT NULL);

Код на SQL для создания таблицы «Сотрудник»:  
CREATE TABLE public.employee

(  
employee\_id SERIAL PRIMARY KEY,  
full\_name VARCHAR(255) NOT NULL,  
email VARCHAR(255) NOT NULL,  
snils varchar(15) NOT NULL);

Код на SQL для создания таблицы «Залог»:  
CREATE TABLE public.collateral

(  
collateral\_id SERIAL PRIMARY KEY,  
type VARCHAR(255) NOT NULL,  
description text,  
estimated\_cost money NOT NULL);

Код на SQL для создания таблицы «Виды документов»:  
CREATE TABLE public.types\_of\_documents

(  
types\_of\_documents\_id SERIAL PRIMARY KEY,  
document\_name VARCHAR(255) NOT NULL);

Код на SQL для создания таблицы «Требуемые документы»:

CREATE TABLE public.required\_documents

(  
required\_documents\_id SERIAL PRIMARY KEY,  
types\_of\_documents\_id int REFERENCES public.client(client\_id),  
loan\_id int REFERENCES public.loan(loan\_id),  
copy\_or\_original boolean);

Код на SQL для создания таблицы «Пакет документов»:

CREATE TABLE public.package\_of\_documents

(  
package\_of\_documents\_id SERIAL PRIMARY KEY,  
loan\_agreement\_id int REFERENCES public.loan\_agreement(loan\_agreement\_id),  
required\_documents\_id int REFERENCES public.required\_documents(required\_documents\_id),  
description text);

4.2.Заполнение таблиц:

1) client

INSERT INTO public.client (full\_name, email, phone\_num)

VALUES

('Иван Иванов', 'ivan@mail.ru', '+79999999999'),

('Петр Петров', 'petr@mail.ru', '+78888888888'),

('Мария Маринина', 'mariya@mail.ru', '+77777777777'),

('Сергей Сергеев', 'sergey@mail.ru', '+76666666666'),

('Анна Антонова', 'anna@mail.ru', '+75555555555'),

('Дмитрий Дмитриев', 'dmitriy@mail.ru', '+74444444444'),

('Елена Еленина', 'elena@mail.ru', '+73333333333'),

('Алексей Алексеев', 'aleksey@mail.ru', '+72222222222'),

('Ольга Ольгина', 'olga@mail.ru', '+71111111111'),

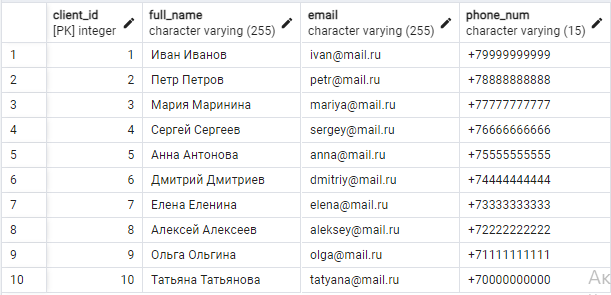
('Татьяна Татьянова', 'tatyana@mail.ru', '+70000000000')  


Рис. 2 – заполненное отношение «Клиент».

2) employee

INSERT INTO public.employee (full\_name, email, snils)

VALUES

('Иванов Иван', 'ivanov@bank.com', '123-45-6789'),

('Петров Пётр', 'petrov@bank.com', '234-56-7890'),

('Сидоров Сидор', 'sidorov@bank.com', '345-67-8901'),

('Васильев Василий', 'vasiliev@bank.com', '456-78-9012'),

('Смирнов Сергей', 'smirnov@bank.com', '567-89-0123'),

('Николаев Николай', 'nikolaev@bank.com', '678-90-1234'),

('Алексеев Алексей', 'alekseev@bank.com', '789-01-2345'),

('Михайлов Михаил', 'mikhailov@bank.com', '890-12-3456'),

('Фёдоров Фёдор', 'fedorov@bank.com', '901-23-4567'),

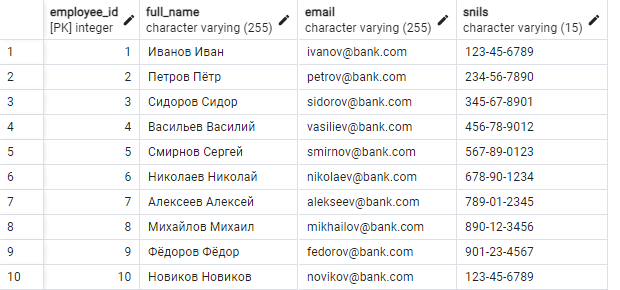
('Новиков Новиков', 'novikov@bank.com', '123-45-6789');  


Рис. 3 – заполненное отношение «Сотрудник».

3) loan

INSERT INTO public.loan (loan\_name, min\_sum, max\_sum, min\_month, max\_month, min\_percent, max\_percent)

VALUES

('Кредит под залог', 1000000, 7000000, 6, 80, 0.10, 0.20);

('Потребительский кредит',10000,1000000,6,60,0.11,0.25),

('Ипотека', 500000, 10000000, 12, 360, 0.09, 0.12),

('Автокредит',100000, 5000000, 12, 60, 0.04, 0.15),

('Микрозайм', 1000, 50000, 1, 30, 0.3, 7);

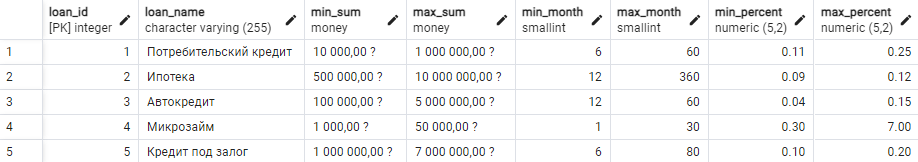


Рис. 4 – заполненное отношение «Кредит».

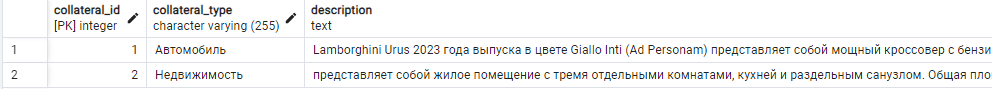
4) collateral

INSERT INTO public.collateral (collateral\_type, description, estimated\_cost)

VALUES

('Недвижимость', 'представляет собой жилое помещение с тремя отдельными комнатами, кухней и раздельным санузлом. Общая площадь около 84 квадратных метра.', 31000000),

('Автомобиль', 'Lamborghini Urus 2023 года выпуска в цвете Giallo Inti (Ad Personam) представляет собой мощный кроссовер с бензиновым двигателем объемом 3996 см³ и мощностью 666 л.с.', 20000000);

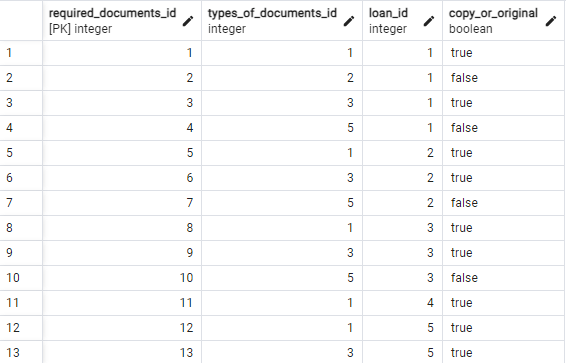


5) required\_documents

INSERT INTO public.required\_documents (types\_of\_documents\_id, loan\_id, copy\_or\_original)

VALUES

(1, 1, TRUE), (2, 1, FALSE), (3, 1, TRUE), (5, 1, FALSE), (1, 2, TRUE), (3, 2, TRUE), (5, 2, FALSE), (1, 3, TRUE), (3, 3, TRUE), (5, 3, FALSE), (1, 4, TRUE), (1, 5, TRUE), (3, 5 ,TRUE);

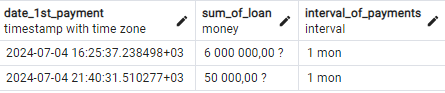


6) loan\_agreement

INSERT INTO public.loan\_agreement (loan\_id, client\_id, employee\_id, interest\_rate, term\_in\_months, collateral\_id, sum\_of\_loan)

VALUES

(5, 3, 3, 0.1, 24, 1, 6000000);

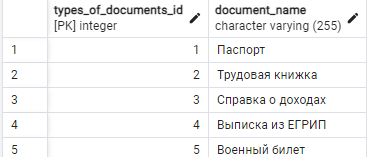
  


7) types\_of\_documents

INSERT INTO public.types\_of\_documents (document\_name)

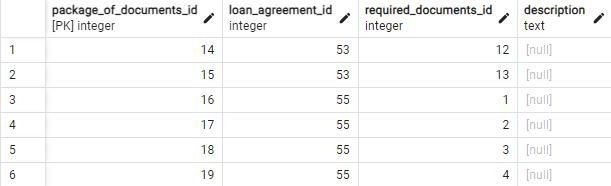
values

('Паспорт'), ('Трудовая книжка'), ('Справка о доходах'), ('Выписка из ЕГРИП'), ('Военный билет');



8) package\_of\_documents

Данная таблица заполняется автоматически при срабатывании триггера, расмотренного на следующих страницах.



4.3.Описание индексов и ограничений

Для улучшения производительности и обеспечения целостности данных в базе данных были добавлены индексы и ограничения.

Для таблицы "loan\_agreement" были созданы внешние ключи на поля "client\_id", "employee\_id", "loan\_id", "guarantor\_id" и "collateral\_id", чтобы связать таблицу "loan\_agreement" с таблицами "client", "employee", "loan" и "collateral".

Для таблицы "package\_of\_documents" были созданы внешние ключи на поля "loan\_agreement\_id" и "required\_documents\_id", чтобы связать таблицу "package\_of\_documents" с таблицами "loan\_agreement" и "required\_documents".

Для таблицы "required\_documents" были созданы внешние ключи на поля "loan\_id" и "types\_of\_documents\_id", чтобы связать таблицу "required\_documents" с таблицами "loan" и "types\_of\_documents".

Ограничения были реализованы в триггерах, смотрите на следующих страницах.

Таким образом, индексы и ограничения улучшают производительность и обеспечивают целостность данных в базе данных.

# 5. ЗАПРОСЫ

1. Выбрать всех клиентов из таблицы client:

SELECT \* FROM client;

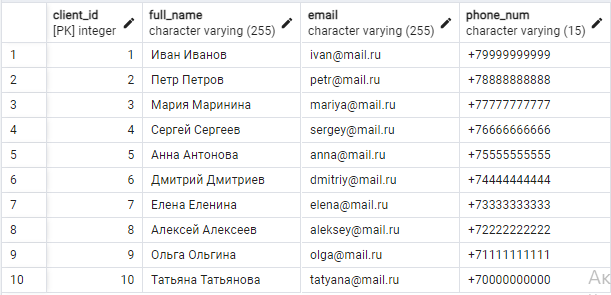


Рис. 5 – результат запроса №1

1. Выбрать всех сотрудников из таблицы employee:

SELECT \* FROM public.employee;

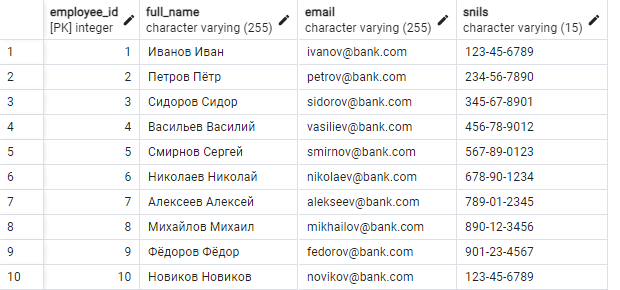


Рис. 6 – результат запроса №2

1. Выбрать все договора из таблицы loan\_agreement:

SELECT \* FROM loan\_agreement;

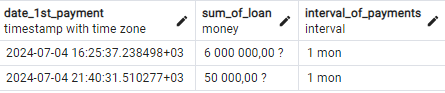
  


Рис. 7 – результат запроса №3

1. Выбрать все loan\_id, где требуется военный билет и только копия:

SELECT loan\_id FROM public.required\_documents

WHERE copy\_or\_original = False and types\_of\_documents\_id = 5;

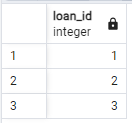


Рис. 8 – результат запроса №4

1. Выбрать ФИО всех потенциальный клиентов:

SELECT full\_name, email FROM public.client

UNION SELECT full\_name, email FROM public.employee;

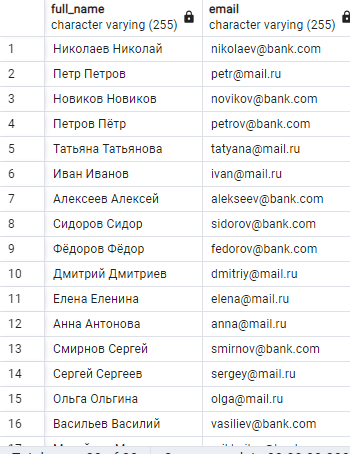
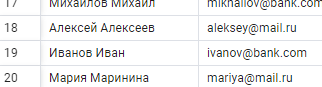
  


Рис. 9 – результат запроса №5

1. Выбрать кредит по сгруппированному полю document\_name со значением ’Паспорт’:

SELECT rd.loan\_id, td.document\_name, rd.copy\_or\_original FROM public.required\_documents AS rd

JOIN public.types\_of\_documents AS td on rd.types\_of\_documents\_id = td.types\_of\_documents\_id

GROUP BY loan\_id, document\_name, copy\_or\_original HAVING document\_name = 'Паспорт';

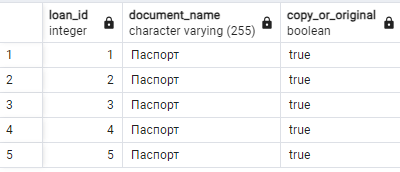


Рис. 10 – результат запроса №6

1. Выбрать договор с количеством получившихся документов:

SELECT loan\_agreement\_id, count(\*) as count\_of\_documents FROM public.package\_of\_documents

GROUP BY loan\_agreement\_id;

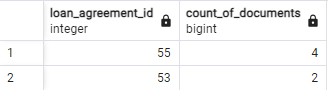


Рис. 11 – результат запроса №7

1. Выбрать все пакеты документов:

SELECT \* FROM public.package\_of\_documents;

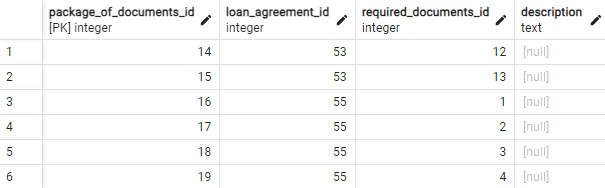


Рис. 12 – результат запроса №8

1. В данном запросе выводятся большой набор данных по клиеннтам:

SELECT \* FROM public.package\_of\_documents as pd

JOIN public.loan\_agreement as la on pd.loan\_agreement\_id = la.loan\_agreement\_id

RIGHT JOIN public.client as cl on la.client\_id = cl.client\_id;

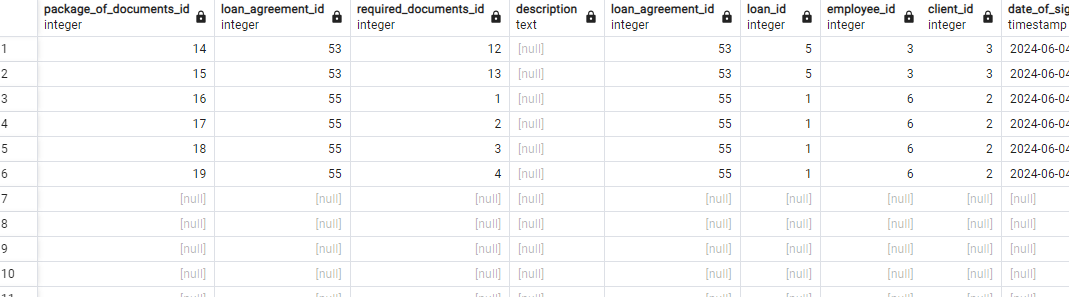


Рис. 13 – результат запроса №9

1. Выбрать все колонки из таблицы залог:

SELECT \* FROM public.collateral;

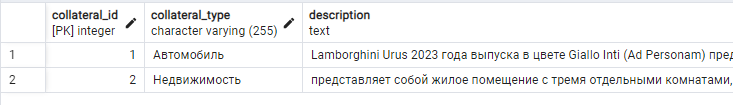


Рис. 14 – результат запроса №10

1. Выбрать все виды документов:

SELECT \* FROM public.types\_of\_documents;

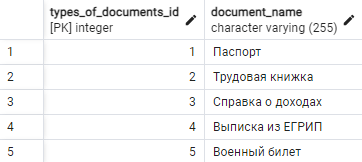


Рис. 15 – результат запроса №11

1. Запрос на все требуемые документы:

SELECT \* FROM public.required\_documents

Рис. 16 – результат запроса №12

1. Запрос на выборку названий книг и ФИО авторов, отсортированных по названию книги в алфавитном порядке:

SELECT title, author FROM Books ORDER BY title ASC;

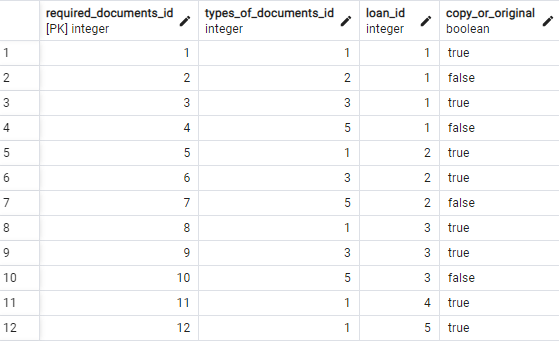


Рис. 17 – результат запроса №13

1. Краткая информация по клиентам:

SELECT cl.\*, lo.loan\_name, la.sum\_of\_loan, la.interest\_rate FROM public.client AS cl

LEFT JOIN public.loan\_agreement AS la ON cl.client\_id = la.client\_id

LEFT JOIN public.loan AS lo ON lo.loan\_id = la.loan\_id

ORDER BY client\_id;

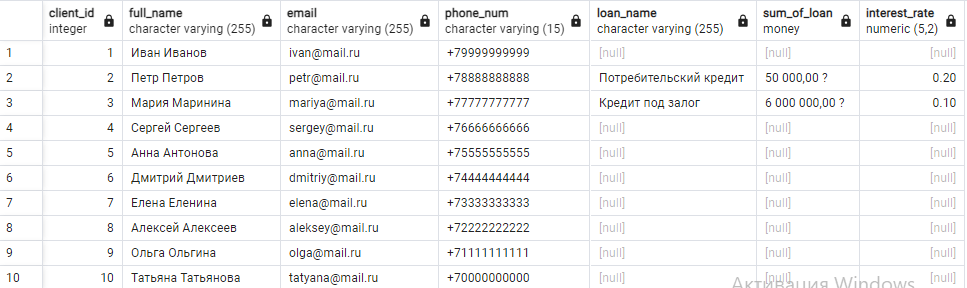


Рис. 18 – результат запроса №14

1. Запрос всех колонок по типам кредита:

SELECT \* FROM public.loan;

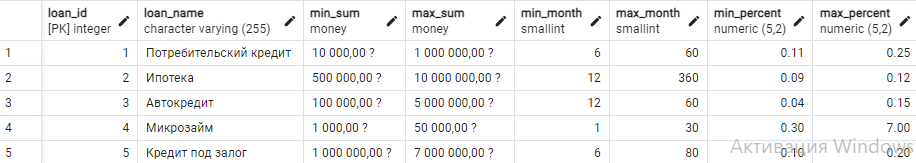


Рис. 19 – результат запроса №15

# 6. ПРЕДСТАВЛЕНИЯ

Представления (Views) - это виртуальные таблицы, создаваемые на основе запросов к реальным таблицам. Представления позволяют сократить длинные и сложные запросы, упростить доступ к данным и повысить безопасность.

1. Представление "Просмотр клиентов на которых оформлен кредит"

CREATE VIEW view\_client\_credit AS

SELECT la.client\_id, c.full\_name AS client\_name, la.guarantor\_id AS guarantor\_name, e.full\_name AS employee\_name, l.loan\_name, la.sum\_of\_loan, la.date\_of\_signing, la.interest\_rate, la.date\_1st\_payment, la.term\_in\_months, la.interval\_of\_payments

FROM public.loan\_agreement as la

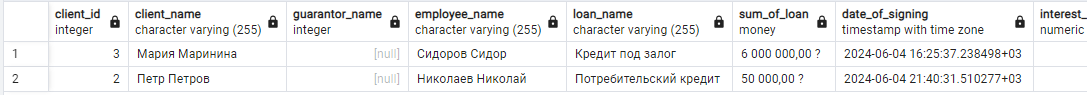
JOIN public.client AS c ON c.client\_id = la.client\_id

JOIN public.employee AS e ON e.employee\_id = la.employee\_id

JOIN public.loan AS l ON l.loan\_id = la.loan\_id

LEFT JOIN public.client AS g ON g.client\_id = la.guarantor\_id

SELECT \* FROM view\_client\_credit;



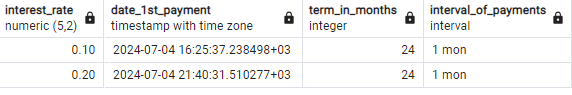


Рис. 20 – результат представления №1

1. Представление "Сумма выплат кредита"

CREATE VIEW count\_payment AS

SELECT la.client\_id, cl.full\_name, count\_payment(la.client\_id) as payment, la.interval\_of\_payments FROM public.loan\_agreement as la

JOIN public.client AS cl on cl.client\_id = la.client\_id;

SELECT \* FROM count\_payment;

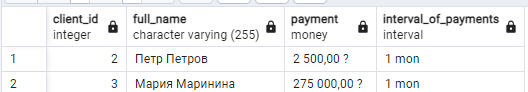


Рис. 21 – результат представления №2

# 7. ФУНКЦИИ

Функции являются блоками кода, которые могут быть вызываться для выполнения определенных действий.

1. Функция для вывода сообщения исключения:

CREATE OR REPLACE FUNCTION raiseErr(errm varchar) returns void

AS

$$

BEGIN

RAISE '%', $1;

END;

$$

LANGUAGE plpgsql;

1. Функция для подсчёта суммы выплачиваемого долга:

CREATE FUNCTION count\_payment(client\_id int)

RETURNS money

AS

$$

DECLARE

payment money;

BEGIN

SELECT (sum\_of\_loan + (sum\_of\_loan \* interest\_rate)) / term\_in\_months INTO payment

FROM public.loan\_agreement AS la

WHERE la.client\_id = $1;

RETURN payment;

END;

$$

LANGUAGE plpgsql;

# 8. ТРИГГЕРЫ

Вот несколько примеров триггеров для базы данных:

* 1. Триггер, автоматически заполняет таблицу package\_of\_documents, после исполнения INSERT для таблицы loan\_agreement:

CREATE OR REPLACE FUNCTION public.make\_package\_of\_documents()

RETURNS TRIGGER

AS

$$

DECLARE

rdi int;

count\_rdi int;

BEGIN

SELECT count(required\_documents\_id) into count\_rdi FROM public.required\_documents

where loan\_id = new.loan\_id;

IF count\_rdi != 1 THEN

FOR i IN 1..count\_rdi LOOP

INSERT INTO public.package\_of\_documents (loan\_agreement\_id, required\_documents\_id)

VALUES

(new.loan\_agreement\_id, (SELECT a FROM (SELECT required\_documents\_id AS a, row\_number() OVER (order by required\_documents\_id ) AS b

FROM public.required\_documents where loan\_id = new.loan\_id) WHERE b = i));

END LOOP;

ELSE

INSERT INTO public.package\_of\_documents (loan\_agreement\_id, required\_documents\_id)

VALUES

(new.loan\_agreement\_id, (SELECT required\_documents\_id

FROM public.required\_documents where loan\_id = new.loan\_id));

END IF;

RETURN new;

END;

$$

LANGUAGE plpgsql;

CREATE TRIGGER make\_package\_of\_documents AFTER INSERT

ON public.loan\_agreement

FOR EACH ROW

EXECUTE FUNCTION public.make\_package\_of\_documents()

* 1. Триггер, выполняет проверку и автоматически заполняет поля:

CREATE OR REPLACE FUNCTION public.check\_loan\_agreement()

RETURNS TRIGGER

AS

$$

BEGIN

PERFORM min\_sum, max\_sum, new.sum\_of\_loan,

CASE

WHEN new.sum\_of\_loan < min\_sum

THEN raiseErr('Сумма кредита меньше, минимально разрешенного значения.')

WHEN new.sum\_of\_loan > max\_sum

THEN raiseErr('Сумма кредита больше, максимально разрешенного значения.')

END

FROM public.loan

WHERE new.loan\_id = loan\_id;

PERFORM min\_month, max\_month, new.term\_in\_months,

CASE

WHEN new.term\_in\_months < min\_month

THEN raiseErr('Количество месяцев меньше, чем минимально разрешенное значение.')

WHEN new.term\_in\_months > max\_month

THEN raiseErr('Количество месяцев больше, чем максимально разрешенное значение.')

END

FROM public.loan

WHERE new.loan\_id = loan\_id;

PERFORM min\_percent, max\_percent, new.interest\_rate,

CASE

WHEN new.interest\_rate < min\_percent

THEN raiseErr('Процентная ставка меньше, чем минимально разрешенное значение.')

WHEN new.interest\_rate > max\_percent

THEN raiseErr('Процентная ставка больше, чем максимально разрешенное значение.')

END

FROM public.loan

WHERE new.loan\_id = loan\_id;

SELECT NOW() INTO new.date\_of\_signing;

SELECT new.date\_of\_signing + interval '1 month' INTO new.date\_1st\_payment;

SELECT interval '1 month' into new.interval\_of\_payments;

return new;

END;

$$

LANGUAGE plpgsql;

CREATE TRIGGER check\_loan\_agreement BEFORE INSERT

ON public.loan\_agreement

FOR EACH ROW

EXECUTE FUNCTION public.check\_loan\_agreement()

Этот триггер срабатывает перед внесением данных в таблицу **loan\_agreement**, для каждой строки. Он проверяет, соотвествует ли введённое значение ставки минимальному и максимальному значению ставки кредита по выбранному кредиту, если нет, то выводит исключение по функции raiseErr(). Тоже самое выполняется и для проверки минимальной и максимальной суммы кредита, минимального и максимального периода. Так же заполняется поле date\_1st\_payment(к значению из поля date\_of\_signing добавляется 1 месяц).

# ЗАКЛЮЧЕНИЕ

В процессе выполнения курсовой работы была создана база данных для банка, включающая таблицы для хранения информации о клиентах, кредитах, процентных ставках и платежах. Были разработаны SQL-запросы для выборки данных из таблиц, использования агрегатных функций и различных операторов (ORDER BY, GROUP BY, JOIN, LEFT JOIN, RIGHT JOIN, UNION, HAVING).

Также были созданы представления для удобного доступа к информации о текущих кредитах, просроченных платежах и общей сумме задолженности каждого клиента. Кроме того, были разработаны функции для добавления, удаления и обновления данных в таблицах.

Для автоматизации процесса начисления процентов и штрафов за просрочку платежей был создан триггер, который автоматически рассчитывает сумму процентов и штрафов при внесении платежа клиентом.

Использование базы данных позволяет банку эффективно управлять своими клиентами, контролировать выдачу кредитов и своевременно отслеживать платежи клиентов. Операции с данными (выборка, добавление, удаление, обновление) могут быть выполнены с помощью SQL-запросов и хранимых процедур, что позволяет автоматизировать многие рутинные задачи и сократить время, затрачиваемое на обслуживание клиентов.

# СПИСОК ЛИТЕРАТУРЫ

1. Документация к PostgreSQL 16.3:  
https://postgrespro.ru/docs/postgresql/16/index  
2. Рогов Е. В. PostgreSQL 16 изнутри. — М.: ДМК Пресс, 2024. — 664 с.;

3. Дворецкая А. Е. Деньги, кредит, банки: учебник для вузов. — 3-е изд., перераб, и доп. — М.: Изд. Юрайт, 2021. — 551 с.

3. Основы проектирования баз данных : учебное пособие / О.Л. Голицына, Т.Л. Партыка, И.И. Попов. — 2-е изд., перераб. и доп. — Москва : ФО-РУМ : ИНФРА-М, 2021. — 416 с. : ил. — (Cреднее профессиональное об-разование);