**Министерство науки и высшего образования Российской Федерации** федеральное государственное автономное образовательное учреждение высшего образования

# «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

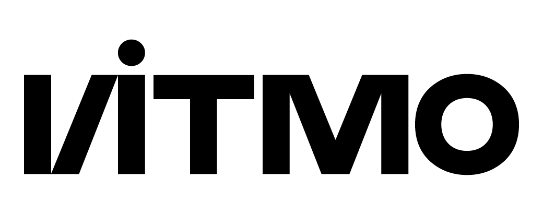
**Отчет**

по Лабораторной Работе № 3 по дисциплине **«Базы Данных**»

Автор: Белов Г.П

Факультет: ИКТ Группа: К3239

Преподаватель: Говорова Марина Михайловна



Санкт-Петербург, 2023

# Содержание работы

**Цель работы:**

Овладеть практическими навыками установки СУБД PostgreSQL и создания базы данных в pgadmin 4, создания таблиц базы данных PostgreSQL 1Х, назначение ограничений на данные, заполнения таблиц рабочими данными, резервного копирования и восстановления БД.

# Практическое задание:

1. Создать базу данных с использованием pgAdmin 4 (согласно индивидуальному заданию).
2. Создать схему в составе базы данных.
3. Создать таблицы базы данных.
4. Установить ограничения на данные: Primary Key, Unique, Check, Foreign Key.
5. Заполнить таблицы БД рабочими данными.
6. Создать резервную копию БД.
7. Восстановить БД.

# Вариант 19. БД «Банк»

Схема логической модели базы данных, сгенерированная в Generate ERD указана на рисунке

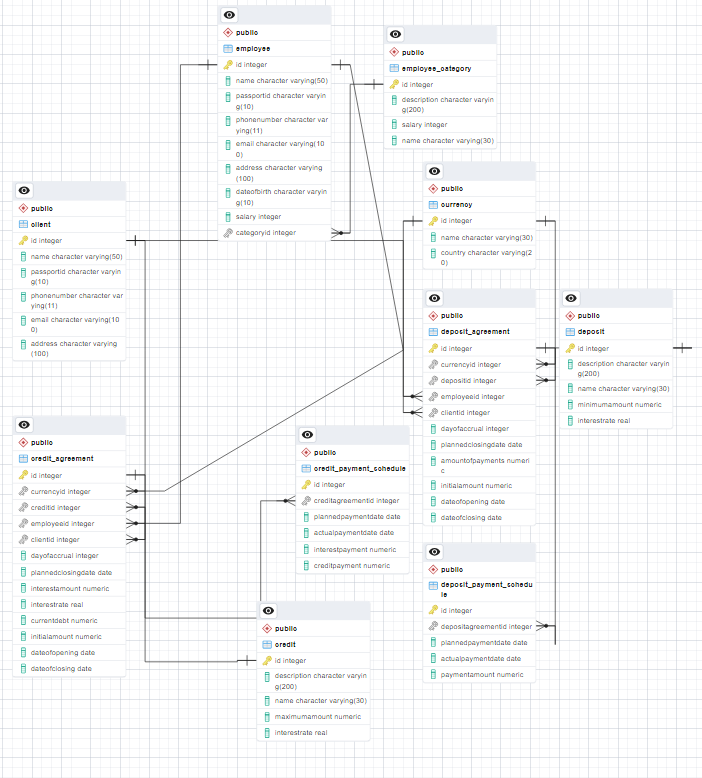


Рисунок 1 – Схема логической модели базы данных.

Листинг – Описание атрибутов сущностей.

**SET** statement\_timeout = 0; **SET** lock\_timeout = 0;

**SET** idle\_in\_transaction\_session\_timeout = 0; **SET** client\_encoding = 'UTF8';

**SET** standard\_conforming\_strings = **ON**;

**SELECT** pg\_catalog.set\_config('search\_path', '', **FALSE**); **SET** check\_function\_bodies = **FALSE**;

**SET** xmloption = content;

**SET** client\_min\_messages = warning; **SET** row\_security = off;

**CREATE** EXTENSION **IF** **NOT** **EXISTS** adminpack **WITH** SCHEMA pg\_catalog;

COMMENT **ON** EXTENSION adminpack **IS** 'administrative functions for PostgreSQL';

**SET** default\_tablespace = '';

**SET** default\_table\_access\_method = heap;

**CREATE** **TABLE** public.client ( id **INTEGER** **NOT** **NULL**,

name **CHARACTER** **VARYING**(50) **NOT** **NULL**, passportid **CHARACTER** **VARYING**(10) **NOT** **NULL**, phonenumber **CHARACTER** **VARYING**(11) **NOT** **NULL**, email **CHARACTER** **VARYING**(100) **NOT** **NULL**, address **CHARACTER** **VARYING**(100) **NOT** **NULL**,

**CONSTRAINT** client\_email\_check **CHECK** (((email)::text ~ '^[a-zA-Z0-9.!#$%&''\*+/=?^\_`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?(?:**\.**[a-zA-Z 0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?)\*$'::text)),

**CONSTRAINT** client\_passportid\_check **CHECK** (((passportid)::text ~ '^[0-9]{10}$'::text)), **CONSTRAINT** client\_phonenumber\_check **CHECK** (((phonenumber)::text ~ '^[0-9]{11}$'::text))

);

**ALTER** **TABLE** public.client OWNER **TO** postgres;

**CREATE** **SEQUENCE** public.client\_id\_seq **AS** **INTEGER**

**START** **WITH** 1

**INCREMENT** **BY** 1 NO MINVALUE NO MAXVALUE CACHE 1;

**ALTER** **SEQUENCE** public.client\_id\_seq OWNER **TO** postgres;

**ALTER** **SEQUENCE** public.client\_id\_seq OWNED **BY** public.client.id;

**CREATE** **TABLE** public.credit ( id **INTEGER** **NOT** **NULL**,

description **CHARACTER** **VARYING**(200) **NOT** **NULL**, name **CHARACTER** **VARYING**(30) **NOT** **NULL**, maximumamount **NUMERIC** **NOT** **NULL**, interestrate **REAL** **NOT** **NULL**,

**CONSTRAINT** credit\_interestrate\_check **CHECK** ((interestrate < (40)::**DOUBLE** **PRECISION**)), **CONSTRAINT** credit\_maximumamount\_check **CHECK** ((maximumamount > (0)::**NUMERIC**))

);

**ALTER** **TABLE** public.credit OWNER **TO** postgres;

**CREATE** **TABLE** public.credit\_agreement ( id **INTEGER** **NOT** **NULL**,

currencyid **INTEGER** **NOT** **NULL**,

creditid **INTEGER** **NOT** **NULL**, employeeid **INTEGER** **NOT** **NULL**, clientid **INTEGER** **NOT** **NULL**, dayofaccrual **INTEGER** **NOT** **NULL**, plannedclosingdate **DATE** **NOT** **NULL**, interestamount **NUMERIC** **DEFAULT** 0, interestrate **REAL** **NOT** **NULL**, currentdebt **NUMERIC** **DEFAULT** 0, initialamount **NUMERIC** **NOT** **NULL**, dateofopening **DATE** **NOT** **NULL**, dateofclosing **DATE**,

**CONSTRAINT** credit\_agreement\_check **CHECK** ((plannedclosingdate > dateofopening)),

**CONSTRAINT** credit\_agreement\_dayofaccrual\_check **CHECK** (((dayofaccrual > 0) **AND** (dayofaccrual < 29))),

**CONSTRAINT** credit\_agreement\_initialamount\_check **CHECK** ((initialamount > (0)::**NUMERIC**)),

**CONSTRAINT** credit\_agreement\_interestrate\_check **CHECK** ((interestrate < (40)::**DOUBLE** **PRECISION**))

);

**ALTER** **TABLE** public.credit\_agreement OWNER **TO** postgres;

**CREATE** **SEQUENCE** public.credit\_agreement\_id\_seq **AS** **INTEGER**

**START** **WITH** 1

**INCREMENT** **BY** 1 NO MINVALUE NO MAXVALUE CACHE 1;

**ALTER** **SEQUENCE** public.credit\_agreement\_id\_seq OWNER **TO** postgres;

**ALTER** **SEQUENCE** public.credit\_agreement\_id\_seq OWNED **BY** public.credit\_agreement.id;

**CREATE** **SEQUENCE** public.credit\_id\_seq **AS** **INTEGER**

**START** **WITH** 1

**INCREMENT** **BY** 1 NO MINVALUE NO MAXVALUE CACHE 1;

**ALTER** **SEQUENCE** public.credit\_id\_seq OWNER **TO** postgres;

**ALTER** **SEQUENCE** public.credit\_id\_seq OWNED **BY** public.credit.id;

**CREATE** **TABLE** public.credit\_payment\_schedule ( id **INTEGER** **NOT** **NULL**,

creditagreementid **INTEGER** **NOT** **NULL**, plannedpaymentdate **DATE** **NOT** **NULL**, actualpaymentdate **DATE**, interestpayment **NUMERIC** **NOT** **NULL**, creditpayment **NUMERIC** **NOT** **NULL**

);

**ALTER** **TABLE** public.credit\_payment\_schedule OWNER **TO** postgres;

**CREATE** **SEQUENCE** public.credit\_payment\_schedule\_id\_seq **AS** **INTEGER**

**START** **WITH** 1

**INCREMENT** **BY** 1 NO MINVALUE NO MAXVALUE CACHE 1;

**ALTER** **SEQUENCE** public.credit\_payment\_schedule\_id\_seq OWNER **TO** postgres;

**ALTER** **SEQUENCE** public.credit\_payment\_schedule\_id\_seq OWNED **BY** public.credit\_payment\_schedule.id;

**CREATE** **TABLE** public.currency ( id **INTEGER** **NOT** **NULL**,

name **CHARACTER** **VARYING**(30) **NOT** **NULL**, country **CHARACTER** **VARYING**(20) **NOT** **NULL**

);

**ALTER** **TABLE** public.currency OWNER **TO** postgres;

**CREATE** **SEQUENCE** public.currency\_id\_seq **AS** **INTEGER**

**START** **WITH** 1

**INCREMENT** **BY** 1 NO MINVALUE NO MAXVALUE CACHE 1;

**ALTER** **SEQUENCE** public.currency\_id\_seq OWNER **TO** postgres;

**ALTER** **SEQUENCE** public.currency\_id\_seq OWNED **BY** public.currency.id;

**CREATE** **TABLE** public.deposit ( id **INTEGER** **NOT** **NULL**,

description **CHARACTER** **VARYING**(200) **NOT** **NULL**,

name **CHARACTER** **VARYING**(30) **NOT** **NULL**, minimumamount **NUMERIC** **NOT** **NULL**, interestrate **REAL** **NOT** **NULL**,

**CONSTRAINT** deposit\_interestrate\_check **CHECK** ((interestrate < (20)::**DOUBLE** **PRECISION**)), **CONSTRAINT** deposit\_minimumamount\_check **CHECK** ((minimumamount > (0)::**NUMERIC**))

);

**ALTER** **TABLE** public.deposit OWNER **TO** postgres;

**CREATE** **TABLE** public.deposit\_agreement ( id **INTEGER** **NOT** **NULL**,

currencyid **INTEGER** **NOT** **NULL**, depositid **INTEGER** **NOT** **NULL**, employeeid **INTEGER** **NOT** **NULL**, clientid **INTEGER** **NOT** **NULL**, dayofaccrual **INTEGER** **NOT** **NULL**, plannedclosingdate **DATE** **NOT** **NULL**,

amountofpayments **NUMERIC** **DEFAULT** 0, initialamount **NUMERIC** **NOT** **NULL**, dateofopening **DATE** **NOT** **NULL**, dateofclosing **DATE**,

**CONSTRAINT** deposit\_agreement\_check **CHECK** ((plannedclosingdate > dateofopening)),

**CONSTRAINT** deposit\_agreement\_dayofaccrual\_check **CHECK** (((dayofaccrual > 0) **AND** (dayofaccrual < 29))),

**CONSTRAINT** deposit\_agreement\_initialamount\_check **CHECK** ((initialamount > (0)::**NUMERIC**))

);

**ALTER** **TABLE** public.deposit\_agreement OWNER **TO** postgres;

**CREATE** **SEQUENCE** public.deposit\_agreement\_id\_seq **AS** **INTEGER**

**START** **WITH** 1

**INCREMENT** **BY** 1 NO MINVALUE NO MAXVALUE CACHE 1;

**ALTER** **SEQUENCE** public.deposit\_agreement\_id\_seq OWNER **TO** postgres;

**ALTER** **SEQUENCE** public.deposit\_agreement\_id\_seq OWNED **BY** public.deposit\_agreement.id;

**CREATE** **SEQUENCE** public.deposit\_id\_seq **AS** **INTEGER**

**START** **WITH** 1

**INCREMENT** **BY** 1 NO MINVALUE NO MAXVALUE CACHE 1;

**ALTER** **SEQUENCE** public.deposit\_id\_seq OWNER **TO** postgres;

**ALTER** **SEQUENCE** public.deposit\_id\_seq OWNED **BY** public.deposit.id;

**CREATE** **TABLE** public.deposit\_payment\_schedule ( id **INTEGER** **NOT** **NULL**,

depositagreementid **INTEGER** **NOT** **NULL**, plannedpaymentdate **DATE** **NOT** **NULL**, actualpaymentdate **DATE**, paymentamount **NUMERIC** **NOT** **NULL**

);

**ALTER** **TABLE** public.deposit\_payment\_schedule OWNER **TO** postgres;

**CREATE** **SEQUENCE** public.deposit\_payment\_schedule\_id\_seq **AS** **INTEGER**

**START** **WITH** 1

**INCREMENT** **BY** 1 NO MINVALUE NO MAXVALUE CACHE 1;

**ALTER** **SEQUENCE** public.deposit\_payment\_schedule\_id\_seq OWNER **TO** postgres;

**ALTER** **SEQUENCE** public.deposit\_payment\_schedule\_id\_seq OWNED **BY** public.deposit\_payment\_schedule.id;

**CREATE** **TABLE** public.employee ( id **INTEGER** **NOT** **NULL**,

name **CHARACTER** **VARYING**(50) **NOT** **NULL**, passportid **CHARACTER** **VARYING**(10) **NOT** **NULL**, phonenumber **CHARACTER** **VARYING**(11) **NOT** **NULL**, email **CHARACTER** **VARYING**(100) **NOT** **NULL**, address **CHARACTER** **VARYING**(100) **NOT** **NULL**, dateofbirth **CHARACTER** **VARYING**(10) **NOT** **NULL**, salary **INTEGER** **NOT** **NULL**,

categoryid **INTEGER** **NOT** **NULL**,

**CONSTRAINT** employee\_email\_check **CHECK** (((email)::text ~ '^[a-zA-Z0-9.!#$%&''\*+/=?^\_`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?(?:**\.**[a-zA-Z 0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?)\*$'::text)),

**CONSTRAINT** employee\_passportid\_check **CHECK** (((passportid)::text ~ '^[0-9]{10}$'::text)),

**CONSTRAINT** employee\_phonenumber\_check **CHECK** (((phonenumber)::text ~ '^[0-9]{11}$'::text)),

**CONSTRAINT** employee\_salary\_check **CHECK** ((salary > 0))

);

**ALTER** **TABLE** public.employee OWNER **TO** postgres;

**CREATE** **TABLE** public.employee\_category ( id **INTEGER** **NOT** **NULL**,

description **CHARACTER** **VARYING**(200) **NOT** **NULL**,

salary **INTEGER** **NOT** **NULL**,

name **CHARACTER** **VARYING**(30) **NOT** **NULL**,

**CONSTRAINT** employee\_category\_salary\_check **CHECK** ((salary > 0))

);

**ALTER** **TABLE** public.employee\_category OWNER **TO** postgres;

**CREATE** **SEQUENCE** public.employee\_category\_id\_seq **AS** **INTEGER**

**START** **WITH** 1

**INCREMENT** **BY** 1 NO MINVALUE NO MAXVALUE CACHE 1;

**ALTER** **SEQUENCE** public.employee\_category\_id\_seq OWNER **TO** postgres;

**ALTER** **SEQUENCE** public.employee\_category\_id\_seq OWNED **BY** public.employee\_category.id;

**CREATE** **SEQUENCE** public.employee\_id\_seq **AS** **INTEGER**

**START** **WITH** 1

**INCREMENT** **BY** 1 NO MINVALUE NO MAXVALUE CACHE 1;

**ALTER** **SEQUENCE** public.employee\_id\_seq OWNER **TO** postgres;

**ALTER** **SEQUENCE** public.employee\_id\_seq OWNED **BY** public.employee.id;

**ALTER** **TABLE** **ONLY** public.client **ALTER** **COLUMN** id **SET** **DEFAULT**

**NEXTVAL**('public.client\_id\_seq'::regclass);

**ALTER** **TABLE** **ONLY** public.credit **ALTER** **COLUMN** id **SET** **DEFAULT**

**NEXTVAL**('public.credit\_id\_seq'::regclass);

**ALTER** **TABLE** **ONLY** public.credit\_agreement **ALTER** **COLUMN** id **SET** **DEFAULT** **NEXTVAL**('public.credit\_agreement\_id\_seq'::regclass);

**ALTER** **TABLE** **ONLY** public.credit\_payment\_schedule **ALTER** **COLUMN** id **SET** **DEFAULT** **NEXTVAL**('public.credit\_payment\_schedule\_id\_seq'::regclass);

**ALTER** **TABLE** **ONLY** public.currency **ALTER** **COLUMN** id **SET** **DEFAULT** **NEXTVAL**('public.currency\_id\_seq'::regclass);

**ALTER** **TABLE** **ONLY** public.deposit **ALTER** **COLUMN** id **SET** **DEFAULT**

**NEXTVAL**('public.deposit\_id\_seq'::regclass);

**ALTER** **TABLE** **ONLY** public.deposit\_agreement **ALTER** **COLUMN** id **SET** **DEFAULT** **NEXTVAL**('public.deposit\_agreement\_id\_seq'::regclass);

**ALTER** **TABLE** **ONLY** public.deposit\_payment\_schedule **ALTER** **COLUMN** id **SET** **DEFAULT** **NEXTVAL**('public.deposit\_payment\_schedule\_id\_seq'::regclass);

**ALTER** **TABLE** **ONLY** public.employee **ALTER** **COLUMN** id **SET** **DEFAULT** **NEXTVAL**('public.employee\_id\_seq'::regclass);

**ALTER** **TABLE** **ONLY** public.employee\_category **ALTER** **COLUMN** id **SET** **DEFAULT** **NEXTVAL**('public.employee\_category\_id\_seq'::regclass);

COPY public.client (id, name, passportid, phonenumber, email, address) **FROM** stdin;

3 Михаил Дмитриевич Прохоров 1914323849 72298165946 prohorovm79@mail.ru Сочи, Параллельная ул., 9/2

4 Леонид Арнольдович Федун 2809346819 78843693164 fedunleonid@mail.ru Выборг, Ленинградское ш., 45Б

\.

COPY public.credit (id, description, name, maximumamount, interestrate) **FROM** stdin;

2 От 21 года на дату получения кредита Автокредит наличными Стандартный 5000000 7.7

1 От 21 года на дату получения кредита, общий стаж работы не менее 3 лет Под залог\nнедвижимости 5000000 6.8

\.

COPY public.credit\_agreement (id, currencyid, creditid, employeeid, clientid, dayofaccrual, plannedclosingdate, interestamount, interestrate, currentdebt, initialamount, dateofopening, dateofclosing) **FROM** stdin;

1 2 1 7 3 25 2024-10-24 0 8.7 56000 600000

2023-10-24 \N

2 2 2 7 3 12 2023-12-12 5000 8.6 20000 550000

2023-03-12 \N

3 2 2 7 4 13 2024-11-13 0 9.3 0 500000

2023-11-13 \N

\.

COPY public.credit\_payment\_schedule (id, creditagreementid, plannedpaymentdate, actualpaymentdate, interestpayment, creditpayment) **FROM** stdin;

1 2 2023-04-12 2023-04-12 2000 16000

2 2 2023-05-12 2023-05-12 2000 16000

3 2 2023-06-12 2023-06-12 2000 16000

4 2 2023-07-12 2023-07-12 2000 16000

5 2 2023-08-12 2023-08-12 2000 16000

6 2 2023-09-12 2023-09-12 2000 16000

7 2 2023-10-12 \N 2000 16000

8 2 2023-11-12 \N 2000 16000

\.

COPY public.currency (id, name, country) **FROM** stdin;

2 rub Russia

\.

COPY public.deposit (id, description, name, minimumamount, interestrate) **FROM** stdin;

1 Вклад на данных условиях доступен для клиентов, у которых в течение последних 30 дней на дату обращения отсутствуют счета в Банке, открытые на их имя Надежный вклад

100000 6

2 Вклад на данных условиях доступен для клиентов, у которых в течение последних 30 дней на дату обращения отсутствуют счета в Банке, открытые на их имя Прибыльный вклад

250000 9.6

\.

COPY public.deposit\_agreement (id, currencyid, depositid, employeeid, clientid, dayofaccrual, plannedclosingdate, amountofpayments, initialamount, dateofopening, dateofclosing) **FROM** stdin;

2 2 2 7 4 13 2023-12-13 15000 300000 2023-04-12 \N

1 2 1 7 4 12 2024-12-12 0 500000 2023-11-12 \N

\.

COPY public.deposit\_payment\_schedule (id, depositagreementid, plannedpaymentdate, actualpaymentdate, paymentamount) **FROM** stdin;

\.

COPY public.employee (id, name, passportid, phonenumber, email, address, dateofbirth, salary, categoryid) **FROM** stdin;

7 Игорь Викторович Макаров 1919385795 78843693165 markovbankir@bank.ru

Воронеж, ул. Карла Маркса, 67/2 12.05.1989 66500 2

\.

COPY public.employee\_category (id, description, salary, name) **FROM** stdin;

2 Менеджер, руководящий банком и проводящий им финансовые операции 60000 Банкир

\.

**SELECT** pg\_catalog.setval('public.client\_id\_seq', 4, **TRUE**);

**SELECT** pg\_catalog.setval('public.credit\_agreement\_id\_seq', 1, **FALSE**);

**SELECT** pg\_catalog.setval('public.credit\_id\_seq', 1, **FALSE**);

**SELECT** pg\_catalog.setval('public.credit\_payment\_schedule\_id\_seq', 8, **TRUE**);

**SELECT** pg\_catalog.setval('public.currency\_id\_seq', 2, **TRUE**);

**SELECT** pg\_catalog.setval('public.deposit\_agreement\_id\_seq', 1, **FALSE**);

**SELECT** pg\_catalog.setval('public.deposit\_id\_seq', 1, **FALSE**);

**SELECT** pg\_catalog.setval('public.deposit\_payment\_schedule\_id\_seq', 1, **FALSE**);

**SELECT** pg\_catalog.setval('public.employee\_category\_id\_seq', 2, **TRUE**);

**SELECT** pg\_catalog.setval('public.employee\_id\_seq', 7, **TRUE**);

**ALTER** **TABLE** public.credit\_agreement

**ADD** **CONSTRAINT** ch1 **CHECK** ((dateofopening > dateofclosing)) **NOT** VALID;

**ALTER** **TABLE** public.deposit\_agreement

**ADD** **CONSTRAINT** ch1 **CHECK** ((dateofopening < dateofclosing)) **NOT** VALID;

**ALTER** **TABLE** **ONLY** public.client

**ADD** **CONSTRAINT** client\_pkey **PRIMARY** **KEY** (id);

**ALTER** **TABLE** **ONLY** public.credit\_agreement

**ADD** **CONSTRAINT** credit\_agreement\_pkey **PRIMARY** **KEY** (id);

**ALTER** **TABLE** **ONLY** public.credit\_payment\_schedule

**ADD** **CONSTRAINT** credit\_payment\_schedule\_pkey **PRIMARY** **KEY** (id);

**ALTER** **TABLE** **ONLY** public.credit

**ADD** **CONSTRAINT** credit\_pkey **PRIMARY** **KEY** (id);

**ALTER** **TABLE** **ONLY** public.currency

**ADD** **CONSTRAINT** currency\_pkey **PRIMARY** **KEY** (id);

**ALTER** **TABLE** **ONLY** public.deposit\_agreement

**ADD** **CONSTRAINT** deposit\_agreement\_pkey **PRIMARY** **KEY** (id);

**ALTER** **TABLE** **ONLY** public.deposit\_payment\_schedule

**ADD** **CONSTRAINT** deposit\_payment\_schedule\_pkey **PRIMARY** **KEY** (id);

**ALTER** **TABLE** **ONLY** public.deposit

**ADD** **CONSTRAINT** deposit\_pkey **PRIMARY** **KEY** (id);

**ALTER** **TABLE** **ONLY** public.employee\_category

**ADD** **CONSTRAINT** employee\_category\_pkey **PRIMARY** **KEY** (id);

**ALTER** **TABLE** **ONLY** public.employee

**ADD** **CONSTRAINT** employee\_pkey **PRIMARY** **KEY** (id);

**ALTER** **TABLE** **ONLY** public.credit\_agreement

**ADD** **CONSTRAINT** credit\_agreement\_clientid\_fkey **FOREIGN** **KEY** (clientid) **REFERENCES** public.client(id);

**ALTER** **TABLE** **ONLY** public.credit\_agreement

**ADD** **CONSTRAINT** credit\_agreement\_creditid\_fkey **FOREIGN** **KEY** (creditid) **REFERENCES** public.credit(id);

**ALTER** **TABLE** **ONLY** public.credit\_agreement

**ADD** **CONSTRAINT** credit\_agreement\_currencyid\_fkey **FOREIGN** **KEY** (currencyid) **REFERENCES** public.currency(id);

**ALTER** **TABLE** **ONLY** public.credit\_agreement

**ADD** **CONSTRAINT** credit\_agreement\_employeeid\_fkey **FOREIGN** **KEY** (employeeid) **REFERENCES** public.employee(id);

**ALTER** **TABLE** **ONLY** public.credit\_payment\_schedule

**ADD** **CONSTRAINT** credit\_payment\_schedule\_creditagreementid\_fkey **FOREIGN** **KEY** (creditagreementid) **REFERENCES** public.credit\_agreement(id);

**ALTER** **TABLE** **ONLY** public.deposit\_agreement

**ADD** **CONSTRAINT** deposit\_agreement\_clientid\_fkey **FOREIGN** **KEY** (clientid) **REFERENCES** public.client(id);

**ALTER** **TABLE** **ONLY** public.deposit\_agreement

**ADD** **CONSTRAINT** deposit\_agreement\_currencyid\_fkey **FOREIGN** **KEY** (currencyid) **REFERENCES** public.currency(id);

**ALTER** **TABLE** **ONLY** public.deposit\_agreement

**ADD** **CONSTRAINT** deposit\_agreement\_depositid\_fkey **FOREIGN** **KEY** (depositid) **REFERENCES** public.deposit(id);

**ALTER** **TABLE** **ONLY** public.deposit\_agreement

**ADD** **CONSTRAINT** deposit\_agreement\_employeeid\_fkey **FOREIGN** **KEY** (employeeid) **REFERENCES** public.employee(id);

**ALTER** **TABLE** **ONLY** public.deposit\_payment\_schedule

**ADD** **CONSTRAINT** deposit\_payment\_schedule\_depositagreementid\_fkey **FOREIGN** **KEY** (depositagreementid) **REFERENCES** public.deposit\_agreement(id);

**ALTER** **TABLE** **ONLY** public.employee

**ADD** **CONSTRAINT** employee\_categoryid\_fkey **FOREIGN** **KEY** (categoryid) **REFERENCES** public.employee\_category(id);

# Вывод

# В результате выполнения лабораторной работы я освоил навыки работы с базами данных PostgreSQL, используя программу pgAdmin4. Я создал базу данных с необходимыми таблицами в соответствии с поставленной задачей, установил необходимые связи и ограничения, а также заполнил таблицы данными. В процессе работы я также создал две резервные копии базы данных, которые позволят быстро восстановить ее в случае потерь, а также просмотреть код.