Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

«Создание таблиц базы данных PostgreSQL. Заполнение таблиц рабочими данными»

по дисциплине «Проектирование и реализация баз данных»

Обучающийся: Гайдук Алина Сергеевна
Факультет прикладной информатики
Группа K3241
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии 2023
Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2025

Цель работы

Овладеть практическими навыками создания таблиц базы данных PostgreSQL 1X, заполнения их рабочими данными, резервного копирования и восстановления БД.

Программное обеспечение

СУБД PostgreSQL 1X, pgAdmin 4.

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.

Указание:

Создать две резервные копии:

- с расширением CUSTOM для восстановления БД;
- с расширением PLAIN для листинга (в отчете);
- при создании резервных копий БД настроить параметры Dump options для Type of objects и Queries.

7. Восстановить БД.

Модель для создания базы данных

Модель представляет организацию сессии внутри университета, включает в себя информацию о преподавателях и их должностях, студентах, стипендии, дисциплинах и их типах, расписании, аттестационной комиссии и прочем. Подробнее ознакомиться с моделью можно на рисунке 1.

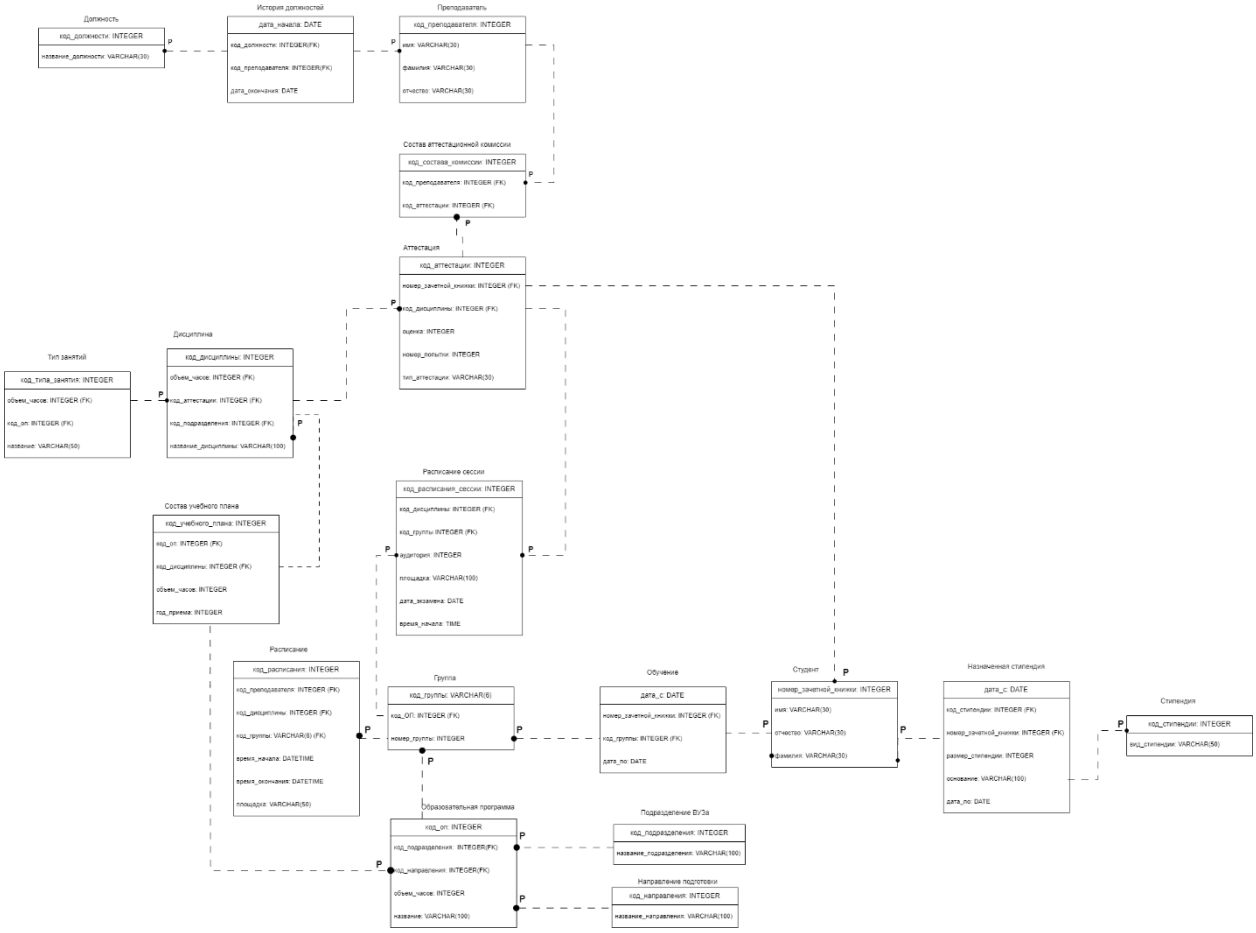

Рисунок 1 – Модель для создания базы данных

Ход работы

Я установила pgAdmin 4 согласно указаниям лабораторной работы, с помощью GUI создала базу данных «lab3». Создание таблиц осуществлялось с помощью написания SQL-запросов, представленных в листинге 1.

```
-- Таблица должностей
CREATE TABLE job_post (
    id SERIAL PRIMARY KEY, -- Уникальный идентификатор должности
    post_name VARCHAR(30) NOT NULL -- Название должности
);
```

```sql
-- Таблица преподавателей
CREATE TABLE teacher (
    id SERIAL PRIMARY KEY, -- Уникальный идентификатор преподавателя
    name VARCHAR(30) NOT NULL, -- Имя
    surname VARCHAR(30) NOT NULL, -- Фамилия
    patronymic VARCHAR(30) -- Отчество (необязательно)
);

-- История назначения преподавателей на должности
CREATE TABLE job_post_history (
    id SERIAL PRIMARY KEY, -- Уникальный идентификатор записи
    start_date DATE NOT NULL, -- Дата начала назначения
    job_post_id INTEGER NOT NULL, -- Ссылка на должность
    teacher_id INTEGER NOT NULL, -- Ссылка на преподавателя
    end_date DATE, -- Дата окончания назначения (необязательно)
    FOREIGN KEY (job_post_id) REFERENCES job_post(id) ON DELETE CASCADE,
    FOREIGN KEY (teacher_id) REFERENCES teacher(id) ON DELETE CASCADE,
    CHECK (end_date IS NULL OR end_date > start_date) -- Конец должен быть
позже начала, если указан
);

-- Таблица студентов
CREATE TABLE student (
    id SERIAL PRIMARY KEY, -- Уникальный идентификатор студента
    record_book_number INTEGER NOT NULL UNIQUE, -- Номер зачетной книжки
(уникальный и положительный)
    name VARCHAR(30) NOT NULL, -- Имя
    surname VARCHAR(30) NOT NULL, -- Фамилия
    patronymic VARCHAR(30), -- Отчество (необязательно)
    CHECK (record_book_number > 0)
);

-- Таблица типов стипендий
CREATE TABLE scholarship (
    id SERIAL PRIMARY KEY, -- Уникальный идентификатор стипендии
    scholarship_type VARCHAR(50) NOT NULL -- Тип стипендии (например,
академическая, социальная)
);

-- Назначенные стипендии студентам
CREATE TABLE awarded_scholarship (
    id SERIAL PRIMARY KEY, -- Уникальный идентификатор записи
    start_date DATE NOT NULL, -- Дата начала выплаты
    scholarship_id INTEGER NOT NULL, -- Ссылка на тип стипендии
    student_id INTEGER NOT NULL, -- Ссылка на студента
    scholarship_size INTEGER NOT NULL, -- Размер стипендии
    reason VARCHAR(100), -- Причина назначения (необязательно)
    end_date DATE NOT NULL, -- Дата окончания выплаты
    FOREIGN KEY (scholarship_id) REFERENCES scholarship(id) ON DELETE
CASCADE,
    FOREIGN KEY (student_id) REFERENCES student(id) ON DELETE RESTRICT,
    CHECK (scholarship_size >= 0),
    CHECK (end_date > start_date)
);

-- Подразделения (например, кафедры или факультеты)
CREATE TABLE subdivision (
    id SERIAL PRIMARY KEY, -- Уникальный идентификатор
    name VARCHAR(100) NOT NULL UNIQUE -- Название подразделения (уникальное)
);

-- Направления подготовки
CREATE TABLE direction (
```

```
    id SERIAL PRIMARY KEY, -- Уникальный идентификатор
    name VARCHAR(100) NOT NULL UNIQUE -- Название направления (уникальное)
);

-- Образовательные программы
CREATE TABLE educational_program (
    id SERIAL PRIMARY KEY, -- Уникальный идентификатор
    subdivision_id INTEGER NOT NULL, -- Ссылка на подразделение
    direction_id INTEGER NOT NULL, -- Ссылка на направление
    hours_number INTEGER NOT NULL, -- Общее количество часов
    name VARCHAR(100) NOT NULL UNIQUE, -- Название программы (уникальное)
    FOREIGN KEY (subdivision_id) REFERENCES subdivision(id) ON DELETE
CASCADE,
    FOREIGN KEY (direction_id) REFERENCES direction(id) ON DELETE CASCADE,
    CHECK (hours_number > 0)
);

-- Учебные группы
CREATE TABLE student_group (
    id SERIAL PRIMARY KEY, -- Уникальный идентификатор группы
    educational_program_id INTEGER NOT NULL, -- Ссылка на образовательную
программу
    group_number INTEGER NOT NULL UNIQUE, -- Номер группы (уникальный и
положительный)
    FOREIGN KEY (educational_program_id) REFERENCES educational_program(id)
ON DELETE RESTRICT,
    CHECK (group_number > 0)
);

-- Обучение студента в конкретной группе
CREATE TABLE education (
    id SERIAL PRIMARY KEY, -- Уникальный идентификатор
    start_date DATE NOT NULL, -- Дата начала обучения
    student_group_id INTEGER NOT NULL, -- Ссылка на группу
    student_id INTEGER NOT NULL, -- Ссылка на студента
    end_date DATE NOT NULL, -- Дата окончания обучения
    FOREIGN KEY (student_group_id) REFERENCES student_group(id) ON DELETE
CASCADE,
    FOREIGN KEY (student_id) REFERENCES student(id) ON DELETE RESTRICT,
    CHECK (end_date > start_date)
);

-- Учебные предметы
CREATE TABLE subject (
    id SERIAL PRIMARY KEY, -- Уникальный идентификатор
    hours_number INTEGER NOT NULL, -- Количество учебных часов
    name VARCHAR(100) UNIQUE NOT NULL, -- Название предмета (уникальное)
    CHECK (hours_number > 0)
);

-- Аттестации студентов (экзамены, зачеты и т.п.)
CREATE TABLE attestation (
    id SERIAL PRIMARY KEY, -- Уникальный идентификатор
    student_id INTEGER NOT NULL, -- Ссылка на студента
    subject_id INTEGER NOT NULL, -- Ссылка на предмет
    mark INTEGER NOT NULL, -- Оценка
    attempt_number INTEGER NOT NULL, -- Попытка сдачи
    attestation_type VARCHAR(30) NOT NULL, -- Тип аттестации
    FOREIGN KEY (student_id) REFERENCES student(id) ON DELETE RESTRICT,
    FOREIGN KEY (subject_id) REFERENCES subject(id) ON DELETE RESTRICT,
    CHECK (mark BETWEEN 2 AND 5),
    CHECK (attempt_number BETWEEN 1 AND 3)
);
```

```sql
-- Комиссия, принимавшая участие в аттестации
CREATE TABLE attestation_committee (
    id SERIAL PRIMARY KEY, -- Уникальный идентификатор
    teacher_id INTEGER NOT NULL, -- Преподаватель — член комиссии
    attestation_id INTEGER NOT NULL, -- Ссылка на аттестацию
    FOREIGN KEY (attestation_id) REFERENCES attestation(id) ON DELETE
CASCADE,
    FOREIGN KEY (teacher_id) REFERENCES teacher(id) ON DELETE CASCADE
);

-- Расписание сессий
CREATE TABLE session_schedule (
    id SERIAL PRIMARY KEY, -- Уникальный идентификатор
    subject_id INTEGER NOT NULL, -- Ссылка на предмет
    student_group_id INTEGER NOT NULL, -- Ссылка на группу
    classroom INTEGER, -- Номер аудитории
    place VARCHAR(100), -- Место проведения (здание и пр.)
    exam_date DATE NOT NULL, -- Дата экзамена
    exam_time TIME NOT NULL, -- Время экзамена
    FOREIGN KEY (subject_id) REFERENCES subject(id) ON DELETE CASCADE,
    FOREIGN KEY (student_group_id) REFERENCES student_group(id) ON DELETE
CASCADE,
    CHECK (classroom > 0)
);

-- Таблица расписания занятий
CREATE TABLE schedule (
    id SERIAL PRIMARY KEY, -- Уникальный идентификатор записи расписания
    teacher_id INTEGER NOT NULL, -- Преподаватель, проводящий занятие
    subject_id INTEGER NOT NULL, -- Преподаваемый предмет
    student_group_id INTEGER NOT NULL, -- Группа студентов, посещающая
занятие
    start_time TIME NOT NULL, -- Время начала занятия
    end_time TIME NOT NULL, -- Время окончания занятия
    place VARCHAR(50), -- Место проведения (аудитория, корпус и т.п.)
    FOREIGN KEY (subject_id) REFERENCES subject(id) ON DELETE RESTRICT,
    FOREIGN KEY (teacher_id) REFERENCES teacher(id) ON DELETE RESTRICT,
    FOREIGN KEY (student_group_id) REFERENCES student_group(id) ON DELETE
CASCADE,
    CHECK(end_time > start_time) -- Занятие должно заканчиваться позже, чем
начинается
);

-- Таблица состава учебного плана
CREATE TABLE curriculum_composition (
    id SERIAL PRIMARY KEY, -- Уникальный идентификатор записи
    educational_program_id INTEGER NOT NULL, -- Образовательная программа
    subject_id INTEGER NOT NULL, -- Предмет, входящий в программу
    hours_number INTEGER NOT NULL, -- Количество часов по предмету
    start_year INTEGER NOT NULL, -- Год начала действия данной части
учебного плана
    FOREIGN KEY (educational_program_id) REFERENCES educational_program(id)
ON DELETE CASCADE,
    FOREIGN KEY (subject_id) REFERENCES subject(id) ON DELETE CASCADE,
    CHECK (hours_number > 0),
    CHECK (start_year BETWEEN 2000 AND 2050) -- Ограничение по допустимому
диапазону годов
);

-- Таблица типов занятий по предмету
CREATE TABLE subject_type (
    id SERIAL PRIMARY KEY, -- Уникальный идентификатор типа занятия
```

```
    hours_number INTEGER NOT NULL, -- Количество часов
    subject_id INTEGER NOT NULL, -- Предмет
    name VARCHAR(50) NOT NULL, -- Название
    FOREIGN KEY (subject_id) REFERENCES subject(id) ON DELETE CASCADE,
    CHECK (hours_number > 0)
);
```

Листинг 1 – Создание таблиц с помощью SQL запросов

Данные запросы включают в себя ограничения, такие как Primary Key, Unique, Check, Foreign Key.

Далее нажатием правой кнопкой мыши на созданную мной базу данных в дереве управления pgAdmin 4, открылось контекстное меню, в котором я выбрала «ERD for Database». После этого появилась схема логической модели базы данных, представленная на рисунке 2. Данная схема составлена в соответствии с исходной моделью.

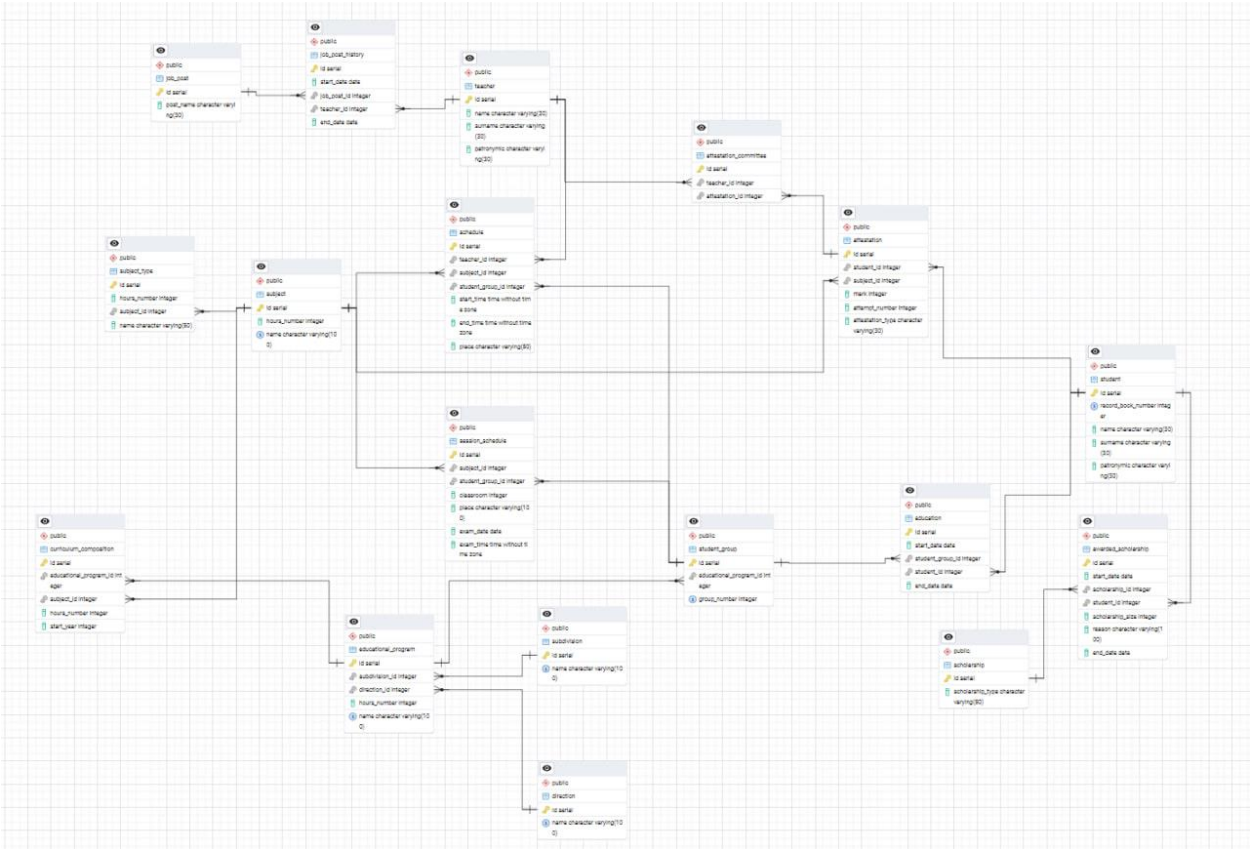

Рисунок 2 – Схема логической модели БД

Следующим шагом было необходимо вставить рабочие данные в таблицы базы данных. Вставка происходила с помощью INSERT. На листинге 2 представлен скрипт вставки данных.

```
-- Добавление должностей
INSERT INTO job_post (post_name) VALUES
('Преподаватель'),
('Доцент'),
('Профессор');

-- Добавление преподавателей
INSERT INTO teacher (name, surname, patronymic) VALUES
('Иван', 'Иванов', 'Иванович'),
('Пётр', 'Петров', 'Петрович'),
('Анна', 'Смирнова', 'Сергеевна');

-- Назначение преподавателей на должности (история)
INSERT INTO job_post_history (start_date, job_post_id, teacher_id, end_date)
VALUES
('2020-09-01', 1, 1, '2022-08-31'), -- Иванов был преподавателем до 2022
('2022-09-01', 2, 1, NULL), -- стал доцентом с 2022
('2021-01-01', 1, 2, NULL); -- Петров преподаватель с 2021

-- Добавление студентов
INSERT INTO student (record_book_number, name, surname, patronymic) VALUES
(1001, 'Алексей', 'Сидоров', 'Алексеевич'),
(1002, 'Мария', 'Кузнецова', 'Игоревна'),
(1003, 'Олег', 'Новиков', 'Дмитриевич');

-- Виды стипендий
INSERT INTO scholarship (scholarship_type) VALUES
('Академическая'),
('Социальная'),
('Именная');

-- Назначение стипендий студентам
INSERT INTO awarded_scholarship (start_date, scholarship_id, student_id,
scholarship_size, reason, end_date) VALUES
('2024-01-01', 1, 1, 5000, 'Высокая успеваемость', '2024-06-30'),
('2024-01-01', 2, 2, 3000, 'Социальная поддержка', '2024-06-30');

-- Подразделения
INSERT INTO subdivision (name) VALUES
('Факультет информатики'),
('Факультет математики');

-- Направления подготовки
INSERT INTO direction (name) VALUES
('Информатика и вычислительная техника'),
('Прикладная математика');

-- Образовательные программы
INSERT INTO educational_program (subdivision_id, direction_id, hours_number,
name) VALUES
(1, 1, 2400, 'Бакалавриат ИВТ'),
(2, 2, 2400, 'Бакалавриат ПМ');

-- Группы студентов
INSERT INTO student_group (educational_program_id, group_number) VALUES
(1, 101),
(2, 102);
```

```sql
-- Зачисление студентов в группы
INSERT INTO education (start_date, student_group_id, student_id, end_date)
VALUES
('2021-09-01', 1, 1, '2025-06-30'),
('2021-09-01', 1, 2, '2025-06-30'),
('2021-09-01', 2, 3, '2025-06-30');

-- Предметы
INSERT INTO subject (hours_number, name) VALUES
(72, 'Программирование'),
(108, 'Математический анализ');

-- Аттестации студентов по предметам
INSERT INTO attestation (student_id, subject_id, mark, attempt_number,
attestation_type) VALUES
(1, 1, 5, 1, 'Экзамен'),
(2, 1, 4, 1, 'Экзамен'),
(3, 2, 3, 2, 'Зачёт');

-- Члены комиссии для аттестации
INSERT INTO attestation_committee (teacher_id, attestation_id) VALUES
(1, 1),
(2, 2),
(3, 3);

-- Расписание экзаменов (сессия)
INSERT INTO session_schedule (subject_id, student_group_id, classroom,
place, exam_date, exam_time) VALUES
(1, 1, 101, 'Корпус А, ауд. 101', '2025-01-15', '10:00'),
(2, 2, 202, 'Корпус В, ауд. 202', '2025-01-16', '13:00');

-- Учебный план по программам
INSERT INTO curriculum_composition (educational_program_id, subject_id,
hours_number, start_year) VALUES
(1, 1, 72, 2021),
(1, 2, 108, 2021),
(2, 2, 108, 2021);

-- Расписание занятий
INSERT INTO schedule (teacher_id, subject_id, student_group_id, start_time,
end_time, place) VALUES
(1, 1, 1, '09:00', '10:30', 'Аудитория 101'),
(2, 2, 2, '11:00', '12:30', 'Аудитория 202');

-- Типы занятий по предметам
INSERT INTO subject_type (hours_number, educational_program_id, name) VALUES
(72, 1, 'Лекция'),
(108, 1, 'Практика'),
(108, 2, 'Лабораторная');
```

Листинг 2 – Вставка данных в таблицы

Далее с помощью запросов SELECT * FROM получилось вывести таблицы с новыми данными. Примеры заполненных таблиц представлены на рисунках 3–7.

	id [PK] integer	hours_number integer	name character varying (100)
1	1	72	Программирование
2	2	108	Математический анализ

Рисунок 3 – Таблица subject

	id [PK] integer	name character varying (30)	surname character varying (30)	patronymic character varying (30)
1	1	Иван	Иванов	Иванович
2	2	Пётр	Петров	Петрович
3	3	Анна	Смирнова	Сергеевна

Рисунок 4 – Таблица teacher

	id [PK] integer	post_name character varying (30)
1	1	Преподаватель
2	2	Доцент
3	3	Профессор

Рисунок 5 – Таблица job_post

	id [PK] integer	start_date date	job_post_id integer	teacher_id integer	end_date date
1	1	2020-09-01	1	1	2022-08-31
2	2	2022-09-01	2	1	[null]
3	3	2021-01-01	1	2	[null]

Рисунок 6 – Таблица job_post_history

	id [PK] integer	record_book_number integer	name character varying (30)	surname character varying (30)	patronymic character varying (30)
1	1	1001	Алексей	Сидоров	Алексеевич
2	2	1002	Мария	Кузнецова	Игоревна
3	3	1003	Олег	Новиков	Дмитриевич

Рисунок 7 – Таблица student

Далее я создала две резервные копии базы данных – Custom для восстановления и Plain для листинга в отчете (см. рис. 8)

Рисунок 8 – Создание резервных копий базы данных

В результате создания Custom копии появилась копия БД с расширением .backup (см. рис. 9).

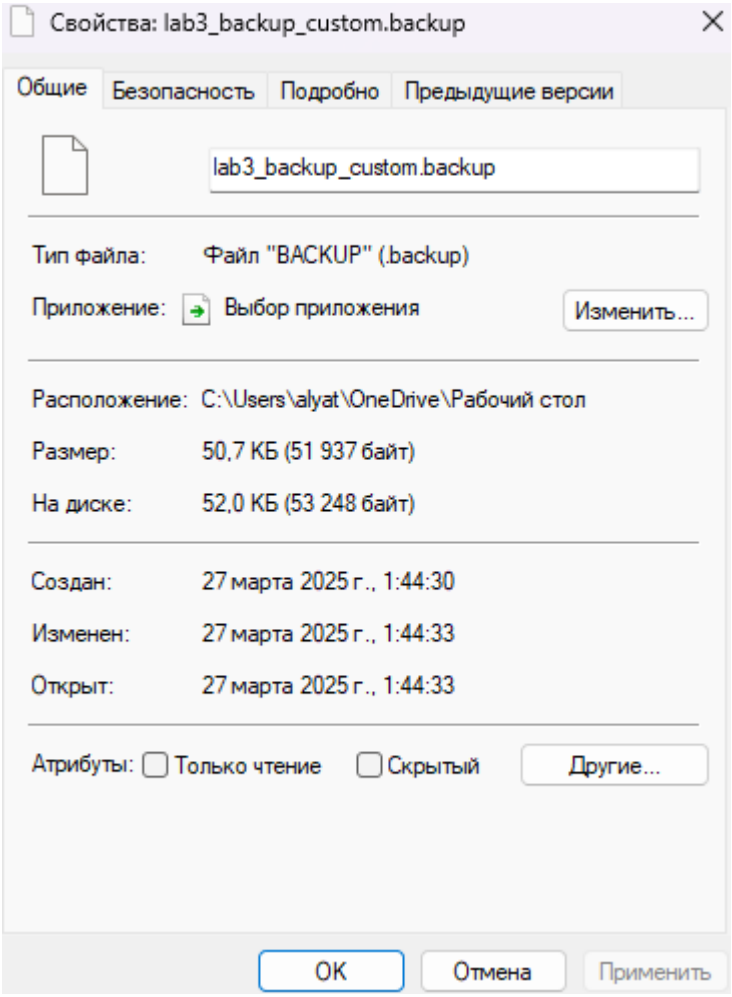

Рисунок 9 – Файл backup

Затем я создала пустую базу данных, кликнув по ней ПКМ нажала «Restore» и восстановила исходную базу данных с помощью .backup файла (см. рис. 10).

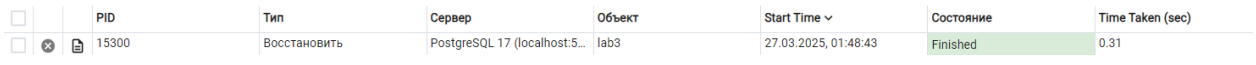

Рисунок 10 – Восстановление БД

Вывод

В ходе выполнения данной лабораторной работы я научилась работать с pgAdmin 4, закрепила знания по написанию SQL-запросов, узнала о механизме заполнения таблиц рабочими данными, научилась создавать резервные копии баз данных и восстанавливать их.