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

**Факультет безопасности информационных технологий**

**Дисциплина:**

«Информационная безопасность баз данных»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

«Реализация БД в рамках СУБД»

**Выполнили:**

Ахраров Али, студент группы N3350

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись)

**Проверил:**

Салихов Максим Русланович

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(отметка о выполнении)

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись)

1. **Цель работы:**

Изучение и овладение практическими навыками работы с современными системами управления базами данных (СУБД), включая проектирование, создание, наполнение и модификацию базы данных, а также реализацию индексации и создание представлений. Основная цель данной работы — не только научиться проектировать и создавать базы данных, но и эффективно управлять ими, реализовывать их оптимизацию и предоставление информации через удобные инструменты, такие как представления и индексы.

1. **Задание**
   1. Выбрать систему управления базами данных (СУБД), которая будет использована для выполнения лабораторной работы, и кратко обосновать свой выбор.
   2. Создать базу данных (БД) в выбранной СУБД на основе итоговой схемы отношений из ЛР 1. Заполнить созданную БД информацией, создав минимум 7-8 записей для каждой из основных таблиц.
   3. Привести SQL-код для выполнения следующих задач:
   4. - создание всех таблиц;
   5. - внесение данных в созданные таблицы;
   6. - выполнение хотя бы одной SQL-команды для изменения структуры таблицы.
   7. Индексировать таблицы. Добавить индексы для атрибутов, по которым происходит объединение таблиц, а также для атрибутов, по которым выполняется поиск/фильтрация данных.
   8. Установить взаимосвязи между таблицами для обеспечения целостности данных.
   9. Выполнить тестовые запросы к БД для проверки её функциональности.
   10. Создать представления, описанные в ЛР 1, для удобного доступа к данным.
2. **СУБД**

Для реализации базы данных была выбрана СУБД PostgreSQL с графическим интерфейсом pgAdmin4. PostgreSQL была выбрана благодаря её популярности, широкому функционалу и возможностям поддержки масштабируемых операций с данными. Кроме того, PostgreSQL является открытым программным обеспечением и имеет обширную документацию, что делает её идеальной для образовательных целей. Выбор также был обусловлен поддержкой различных типов индексов, удобными средствами для создания представлений, а также наличием обширного сообщества разработчиков, что делает её полезной для освоения студентами.

1. **Создание Базы Данных**

Создадим базу данных в pgAdmin4 с использованием следующего SQL-кода:

CREATE DATABASE university\_db

WITH

OWNER = postgres

ENCODING = 'UTF8'

TABLESPACE = pg\_default

CONNECTION LIMIT = -1

IS\_TEMPLATE = False;

База данных под названием `university\_db` создаётся с целью хранения информации о преподавателях, студентах, курсах, группах и других элементах, необходимых для полноценного функционирования учебного процесса в университете.

1. **Создание таблиц**

Создадим таблицы в соответствии с разработанной схемой отношений.

CREATE TABLE IF NOT EXISTS public.teachers

(

teacher\_id integer NOT NULL DEFAULT nextval('teachers\_teacher\_id\_seq'::regclass),

teacher\_firstname character varying(50) COLLATE pg\_catalog."default" NOT NULL,

teacher\_lastname character varying(50) COLLATE pg\_catalog."default" NOT NULL,

teacher\_phone character varying(9) COLLATE pg\_catalog."default" NOT NULL,

teacher\_email character varying(50) COLLATE pg\_catalog."default" NOT NULL,

teacher\_specialization character varying(50) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT teachers\_pkey PRIMARY KEY (teacher\_id),

CONSTRAINT unique\_email UNIQUE (teacher\_email)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.teachers

OWNER to "Али";

Таблица courses:

CREATE TABLE IF NOT EXISTS public.courses

(

course\_id integer NOT NULL DEFAULT nextval('courses\_course\_id\_seq'::regclass),

course\_name character varying(50) COLLATE pg\_catalog."default",

description character varying(255) COLLATE pg\_catalog."default",

duration character varying(10) COLLATE pg\_catalog."default",

cost numeric,

teacher\_id integer,

CONSTRAINT courses\_pkey PRIMARY KEY (course\_id),

CONSTRAINT courses\_teacher\_id\_fkey FOREIGN KEY (teacher\_id)

REFERENCES public.teachers (teacher\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT check\_cost CHECK (cost > 0::numeric)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.courses

OWNER to postgres;

Таблица students:

CREATE TABLE IF NOT EXISTS public.students

(

student\_id integer NOT NULL DEFAULT nextval('students\_student\_id\_seq'::regclass),

student\_firstname character varying(50) COLLATE pg\_catalog."default" NOT NULL,

student\_lastname character varying(50) COLLATE pg\_catalog."default" NOT NULL,

student\_birth date NOT NULL,

student\_email character varying(50) COLLATE pg\_catalog."default" NOT NULL,

student\_phone character varying(9) COLLATE pg\_catalog."default" NOT NULL,

student\_registrationdate date NOT NULL,

student\_university character varying(50) COLLATE pg\_catalog."default",

student\_speciality character varying(50) COLLATE pg\_catalog."default",

CONSTRAINT students\_pkey PRIMARY KEY (student\_id),

CONSTRAINT unique\_email\_students UNIQUE (student\_email)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.students

OWNER to "Али";

Таблица groups:

CREATE TABLE IF NOT EXISTS public.groups

(

group\_id integer NOT NULL DEFAULT nextval('groups\_group\_id\_seq'::regclass),

group\_name character varying(50) COLLATE pg\_catalog."default" NOT NULL,

course\_id integer NOT NULL,

group\_start date NOT NULL,

group\_end date NOT NULL,

CONSTRAINT groups\_pkey PRIMARY KEY (group\_id),

CONSTRAINT groups\_course\_id\_fkey FOREIGN KEY (course\_id)

REFERENCES public.courses (course\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.groups

OWNER to "Али"

Таблица assignment:

CREATE TABLE IF NOT EXISTS public.assignment

(

assignment\_id integer NOT NULL,

assignment\_name character varying(255) COLLATE pg\_catalog."default" NOT NULL,

assignment\_date date NOT NULL,

course\_id integer NOT NULL,

CONSTRAINT assignment\_pkey PRIMARY KEY (assignment\_id),

CONSTRAINT fk\_assignment\_course FOREIGN KEY (course\_id)

REFERENCES public.courses (course\_id) MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

TABLESPACE pg\_default;

Таблица assignment\_result:

CREATE TABLE IF NOT EXISTS public.assignment\_result

(

result\_id integer NOT NULL,

student\_id integer NOT NULL,

assignment\_id integer NOT NULL,

score integer,

grade character varying(2) COLLATE pg\_catalog."default",

CONSTRAINT assignment\_result\_pkey PRIMARY KEY (result\_id),

CONSTRAINT fk\_assignment\_result\_student FOREIGN KEY (student\_id)

REFERENCES public.students (student\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE,

CONSTRAINT fk\_assignment\_result\_assignment FOREIGN KEY (assignment\_id)

REFERENCES public.assignment (assignment\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE

)

TABLESPACE pg\_default;

Таблица attendance:

CREATE TABLE IF NOT EXISTS public.attendance

(

attendance\_id integer NOT NULL,

student\_id integer NOT NULL,

course\_id integer NOT NULL,

class\_date date NOT NULL,

attendance\_status character varying(10) COLLATE pg\_catalog."default",

CONSTRAINT attendance\_pkey PRIMARY KEY (attendance\_id),

CONSTRAINT fk\_attendance\_student FOREIGN KEY (student\_id)

REFERENCES public.students (student\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE,

CONSTRAINT fk\_attendance\_course FOREIGN KEY (course\_id)

REFERENCES public.courses (course\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE

)

TABLESPACE pg\_default;

Таблица enrollments:

CREATE TABLE IF NOT EXISTS public.enrollments

(

enrollment\_id integer NOT NULL,

student\_id integer NOT NULL,

course\_id integer NOT NULL,

enrollment\_date date NOT NULL,

CONSTRAINT enrollments\_pkey PRIMARY KEY (enrollment\_id),

CONSTRAINT fk\_enrollments\_student FOREIGN KEY (student\_id)

REFERENCES public.students (student\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE,

CONSTRAINT fk\_enrollments\_course FOREIGN KEY (course\_id)

REFERENCES public.courses (course\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE

)

TABLESPACE pg\_default;

Таблица payment:

CREATE TABLE IF NOT EXISTS public.payment

(

payment\_id integer NOT NULL,

amount numeric NOT NULL,

student\_id integer NOT NULL,

payment\_date date NOT NULL,

payment\_method character varying(50) COLLATE pg\_catalog."default" NOT NULL,

course\_id integer NOT NULL,

CONSTRAINT payment\_pkey PRIMARY KEY (payment\_id),

CONSTRAINT fk\_payment\_student FOREIGN KEY (student\_id)

REFERENCES public.students (student\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE,

CONSTRAINT fk\_payment\_course FOREIGN KEY (course\_id)

REFERENCES public.courses (course\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE

)

TABLESPACE pg\_default;

Таблица schedule:

CREATE TABLE IF NOT EXISTS public.schedule

(

schedule\_id integer NOT NULL,

course\_id integer NOT NULL,

schedule\_date date NOT NULL,

start\_time time without time zone NOT NULL,

end\_time time without time zone NOT NULL,

classroom character varying(50) COLLATE pg\_catalog."default",

CONSTRAINT schedule\_pkey PRIMARY KEY (schedule\_id),

CONSTRAINT fk\_schedule\_course FOREIGN KEY (course\_id)

REFERENCES public.courses (course\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE

)

TABLESPACE pg\_default;

Таблицы были разработаны на основе ER-диаграммы, созданной в ЛР 1. Каждая из таблиц связана между собой через внешние ключи, что обеспечивает целостность данных и позволяет структурировать информацию, сохраняя её связанность.

1. **Внесение данных в таблицы**

Заполним таблицы основными данными для проверки связей и выполнения запросов.

Данные в таблицу teachers:

INSERT INTO teachers (teacher\_firstname, teacher\_lastname, teacher\_phone, teacher\_email, teacher\_specialization) VALUES

('Иван', 'Петров', '123456789', 'ivan.petrov@univ.ru', 'Математика'),

('Ольга', 'Смирнова', '234567890', 'olga.smirnova@univ.ru', 'Физика'),

('Анна', 'Кузнецова', '345678901', 'anna.kuznetsova@univ.ru', 'Химия'),

('Максим', 'Сидоров', '456789012', 'max.sidorov@univ.ru', 'Биология'),

('Дмитрий', 'Иванов', '567890123', 'dmitriy.ivanov@univ.ru', 'Информатика'),

('Алексей', 'Козлов', '678901234', 'alexey.kozlov@univ.ru', 'История'),

('Елена', 'Новикова', '789012345', 'elena.novikova@univ.ru', 'География');

Данные в таблицу courses:

INSERT INTO courses (course\_name, description, duration, cost, teacher\_id) VALUES

('Алгебра', 'Основы алгебры', '3 месяца', 1000, 1),

('Механика', 'Введение в механику', '4 месяца', 1200, 2),

('Органическая химия', 'Изучение органических соединений', '5 месяцев', 1500, 3),

('Биология клетки', 'Изучение строения клетки', '3 месяца', 1300, 4),

('Основы программирования', 'Изучение языков программирования', '6 месяцев', 2000, 5),

('История средних веков', 'Изучение средневековой истории', '4 месяца', 1100, 6),

('Физическая география', 'Изучение земной поверхности', '3 месяца', 1000, 7);

В таблицах `teachers` и `courses` были добавлены данные, чтобы обеспечить основу для проверки всех возможных связей между курсами и их преподавателями.

1. **Модификация структуры таблиц**

Добавим новый столбец в таблицу teachers для демонстрации модификации структуры таблицы:

ALTER TABLE teachers ADD COLUMN teacher\_position VARCHAR(50);

Добавление нового столбца `teacher\_position` позволяет расширить информацию о преподавателях, включая их должности, что делает базу данных более информативной.

1. **Индексация таблиц**

Создадим индексы для повышения скорости выполнения операций объединения и поиска данных по ключевым атрибутам.

CREATE INDEX idx\_teacher\_id ON teachers(teacher\_id);

CREATE INDEX idx\_course\_id ON courses(course\_id);

CREATE INDEX idx\_student\_id ON students(student\_id);

CREATE INDEX idx\_group\_course\_id ON groups(course\_id);

CREATE INDEX idx\_teacher\_email ON teachers(teacher\_email);

CREATE INDEX idx\_student\_email ON students(student\_email);

Индексация позволяет улучшить производительность системы, особенно при выполнении запросов, связанных с фильтрацией и объединением больших объемов данных.

1. **Установка взаимосвязей между таблицами**

Установим ограничения и внешние ключи для поддержания целостности данных и улучшения работы с ними.

ALTER TABLE courses ADD CONSTRAINT fk\_courses\_teachers FOREIGN KEY (teacher\_id)

REFERENCES teachers (teacher\_id) ON DELETE CASCADE;

ALTER TABLE groups ADD CONSTRAINT fk\_groups\_courses FOREIGN KEY (course\_id)

REFERENCES courses (course\_id) ON DELETE CASCADE;

ALTER TABLE assignment ADD CONSTRAINT fk\_assignment\_courses FOREIGN KEY (course\_id)

REFERENCES courses (course\_id) ON DELETE CASCADE;

ALTER TABLE assignment\_result ADD CONSTRAINT fk\_assignment\_result\_assignments FOREIGN KEY (assignment\_id)

REFERENCES assignment (assignment\_id) ON DELETE CASCADE;

ALTER TABLE assignment\_result ADD CONSTRAINT fk\_assignment\_result\_students FOREIGN KEY (student\_id)

REFERENCES students (student\_id) ON DELETE CASCADE;

ALTER TABLE attendance ADD CONSTRAINT fk\_attendance\_students FOREIGN KEY (student\_id)

REFERENCES students (student\_id) ON DELETE CASCADE;

ALTER TABLE attendance ADD CONSTRAINT fk\_attendance\_courses FOREIGN KEY (course\_id)

REFERENCES courses (course\_id) ON DELETE CASCADE;

ALTER TABLE enrollments ADD CONSTRAINT fk\_enrollments\_students FOREIGN KEY (student\_id)

REFERENCES students (student\_id) ON DELETE CASCADE;

ALTER TABLE enrollments ADD CONSTRAINT fk\_enrollments\_courses FOREIGN KEY (course\_id)

REFERENCES courses (course\_id) ON DELETE CASCADE;

ALTER TABLE payment ADD CONSTRAINT fk\_payment\_students FOREIGN KEY (student\_id)

REFERENCES students (student\_id) ON DELETE CASCADE;

ALTER TABLE payment ADD CONSTRAINT fk\_payment\_courses FOREIGN KEY (course\_id)

REFERENCES courses (course\_id) ON DELETE CASCADE;

ALTER TABLE schedule ADD CONSTRAINT fk\_schedule\_courses FOREIGN KEY (course\_id)

REFERENCES courses (course\_id) ON DELETE CASCADE;

Установка внешних ключей помогает гарантировать, что информация в связанных таблицах остаётся целостной. Например, если преподаватель будет удалён, то все связанные курсы также будут автоматически удалены благодаря опции `ON DELETE CASCADE`.

1. **Создание представлений**

Создадим представления для упрощения доступа к информации пользователями.

Представление compact\_schedule:

CREATE OR REPLACE VIEW public.compact\_schedule AS

SELECT c.course\_name AS "Название дисциплины",

t.teacher\_firstname || ' ' || t.teacher\_lastname AS "ФИО преподавателя",

s.schedule\_date AS "Дата занятия",

s.start\_time AS "Время начала",

s.end\_time AS "Время окончания",

s.classroom AS "Аудитория"

FROM schedule s

JOIN courses c ON s.course\_id = c.course\_id

JOIN teachers t ON c.teacher\_id = t.teacher\_id;

Представление consultations:

CREATE OR REPLACE VIEW public.consultations AS

SELECT t.teacher\_firstname || ' ' || t.teacher\_lastname AS "ФИО преподавателя",

t.teacher\_phone AS "Телефон",

t.teacher\_email AS "Электронная почта",

t.teacher\_specialization AS "Специализация",

t.teacher\_position AS "Должность"

FROM teachers t;

Эти представления помогают пользователям получать сводную информацию, такую как расписание занятий или контакты преподавателей, в удобном формате.

1. **Заключение**

В ходе выполнения лабораторной работы была создана база данных на основе СУБД PostgreSQL. База данных содержит несколько таблиц, связанных между собой через внешние ключи. Были реализованы индексация и представления для оптимизации работы с данными и удобного доступа пользователей к информации.