НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування та спеціалізованих комп'ютерних систем

Лабораторна робота №1

з дисципліни **Бази даних і засоби управління**
на тему: "Проектування бази даних та ознайомлення з базовими операціями СУБД PostgreSQL"

Виконала:
студентка III курсу
групи КВ-02
Майстренко Ольга Олексіївна
Перевірив: Павловський Володимир Ілліч

Київ – 2022

Метою роботи є здобуття практичних навичок створення реляційних баз даних за допомогою PostgreSQL.

Завдання роботи полягає у наступному:

1. Розробити модель «сутність-зв'язок» предметної галузі, обраної студентом самостійно, відповідно до пункту «Вимоги до ER-моделі».

2. Перетворити розроблену модель у схему бази даних (таблиці) PostgreSQL.

3. Виконати нормалізацію схеми бази даних до третьої нормальної форми (3НФ).

4. Ознайомитись із інструментарієм PostgreSQL та pgAdmin 4 та внести декілька рядків даних у кожну з таблиць засобами pgAdmin 4.

Зміст звіту

1. Опис проблемного середовища;
2. Концептуальна модель предметної області;
3. Логічна модель (схема) даних БД;
4. Склад СУБД PostgreSQL;
5. Список обмежень цілісності в термінах СУБД PostgreSQL;
6. Фізична модель (схема) даних БД в pgAdmin III;
7. Приклад вмісту БД.

Опис предметної області «Університет»

Обрана предметна область передбачає збереження інформацію про будівлі університету, кабінети, що належать будівлям та про студентів, що там навчаються. Університет має кілька будівель. В одній будівлі розміщено кілька кабінетів. У різних кабінетах навчаються різні студенти.

Опис сутностей предметної області

Для побудови бази даних для обраної області було виділено сутності, зображені на рисунку 1:

1. Авдиторія (Classroom), з атрибутами: код авдиторії, номер корпусу та номер авдиторії. Призначена для збереження інформації про навчальні авдиторії;
2. Група (Group), з атрибутами: код групи, ім’я групи та код студента. Призначена для зберігання інформації про навчальні групи;
3. Студент (Student), з атрибутами: код студента, ім’я та прізвище. Призначена для збереження особистої інформації про студента;
4. Предмет (Subject), з атрибутами: код предмету і назва. Призначена для збереження різноманіття предметів, які вивчаються студентами;
5. Лектор (Lector), з атрибутами: код лектора, прізвище та ім’я. Призначена для зберігання інформаці про лекторів університету;
6. (Асоціативна) Студентська база (Student_Base), з атрибутами: код зв’язку, код авдиторії, код групи, код лектора та код предмета.

Опис зв’язків між сутностями предметної області

Сутність «Авдиторія» має зв’язок N:M по відношенню до сутності «Група», сутності «Предмет», сутності «Викладач» та сутності «Студент», тому що одній і тій самій авдиторії може бути пара у різних груп, у різних стулентів, там можуть відбуватися пари з різних предметів, що проводять різні викладачі. Так само і навпаки групи та викладачі можуть використовувати різні авдиторії для різних предметів.

Сутність «Група» має зв’язок N:M по відношенню до сутностей, «Предмет», «Викладач» та «Авдиторія» адже одна і та ж група може відвідувати різні предмети з різними викладачами та в різних авдиторіях. Аналогічно різні предмети можуть проводити різні викладачі у різних групах.

Також ця сутність має зв’язок 1:N по відношенню до сутності «Студент», бо багато студентів можуть належати 1-й групі, але 1 студент може належати лише 1-й групі.

Сутність «Викладач» має зв’язок N:M по відношенню до сутності «Предмет», «Авдиторія» та «Група» адже один викладач може читати різні предмети в різних авдиторіях різним групам і так само один предмет може мати кілька різних викладачів та проводитися в інших авдиторіях, а в інших груп можуть бути інші викладачі.

Асоціативна сутність «Студентська база» – додаткова сутність, що допомагає реалізувати зв’язки N:M між усіма сутностями, окрім зв’язку «Група» — «Студент» .

Концептуальна модель предметної області «Університет»

Концептуальна модель наведена на рисунку 1.

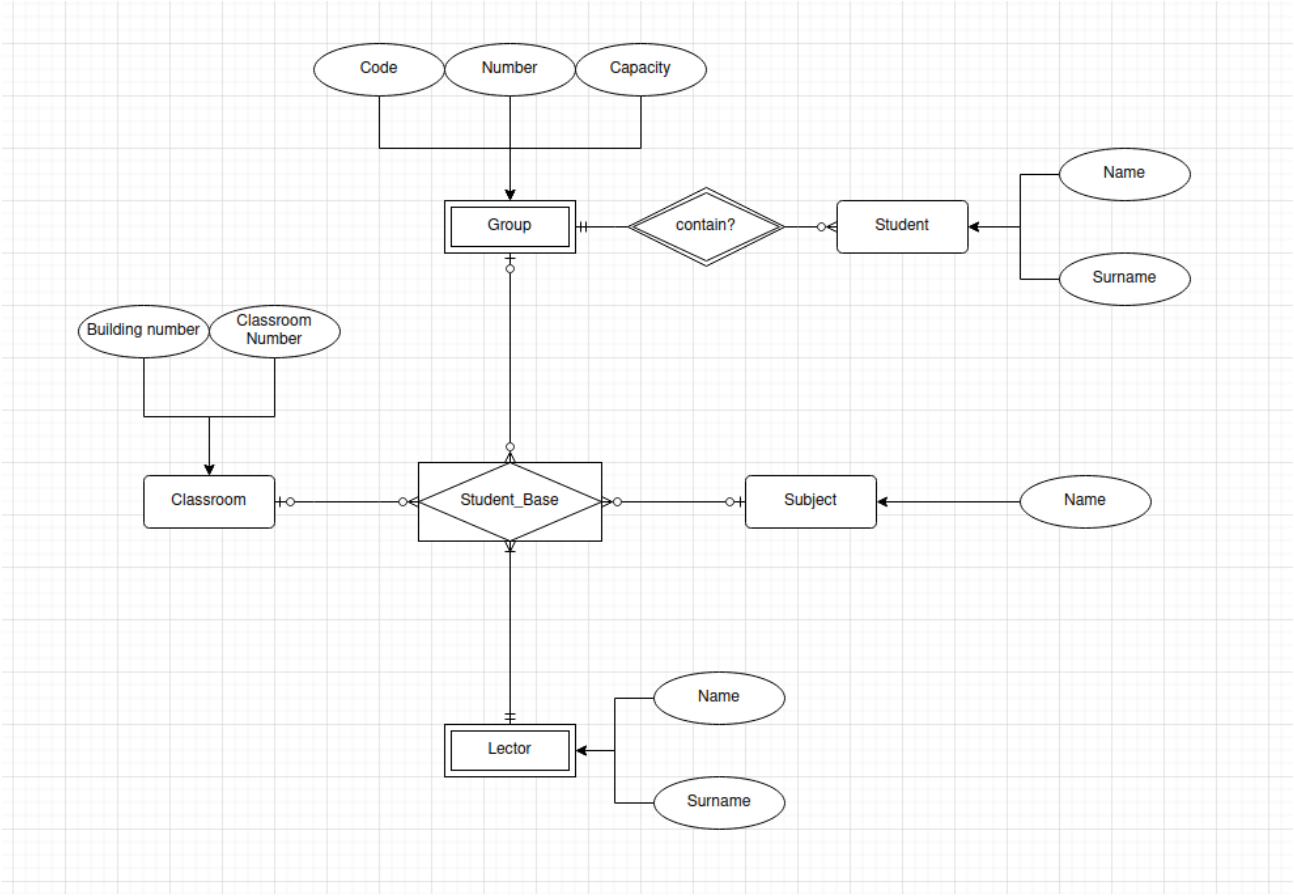


Рисунок 1: Концептуальна модель предметної області «Університет» (інструмент: draw.io)

Перетворення концептуальної моделі у логічну схему бази даних

1. Зв’язок багато-до-багатьох між усіма сутностями, окрім зв’язку «Група» — «Студент» зумовив появу асоціативної сутності «Студентська база» яку перетворено у таблицю «Student_Base» із зовнішніми ключами lector_id, group_id, classroom_id та subject_id;
2. Сутність «Авдиторія» перетворена в таблицю «Classroom»;
3. Сутність «Група» перетворена в таблицю «Group»;
4. Сутність «Предмет» перетворена в таблицю «Subject»;
5. Сутність «Викладач» перетворена в таблицю «Lector»;
6. Сутність «Студент» перетворена в таблицю «Student».

Логічна модель (схема) БД «Університет»

Логічну модель (схему бази даних наведено на рисунку 2.)

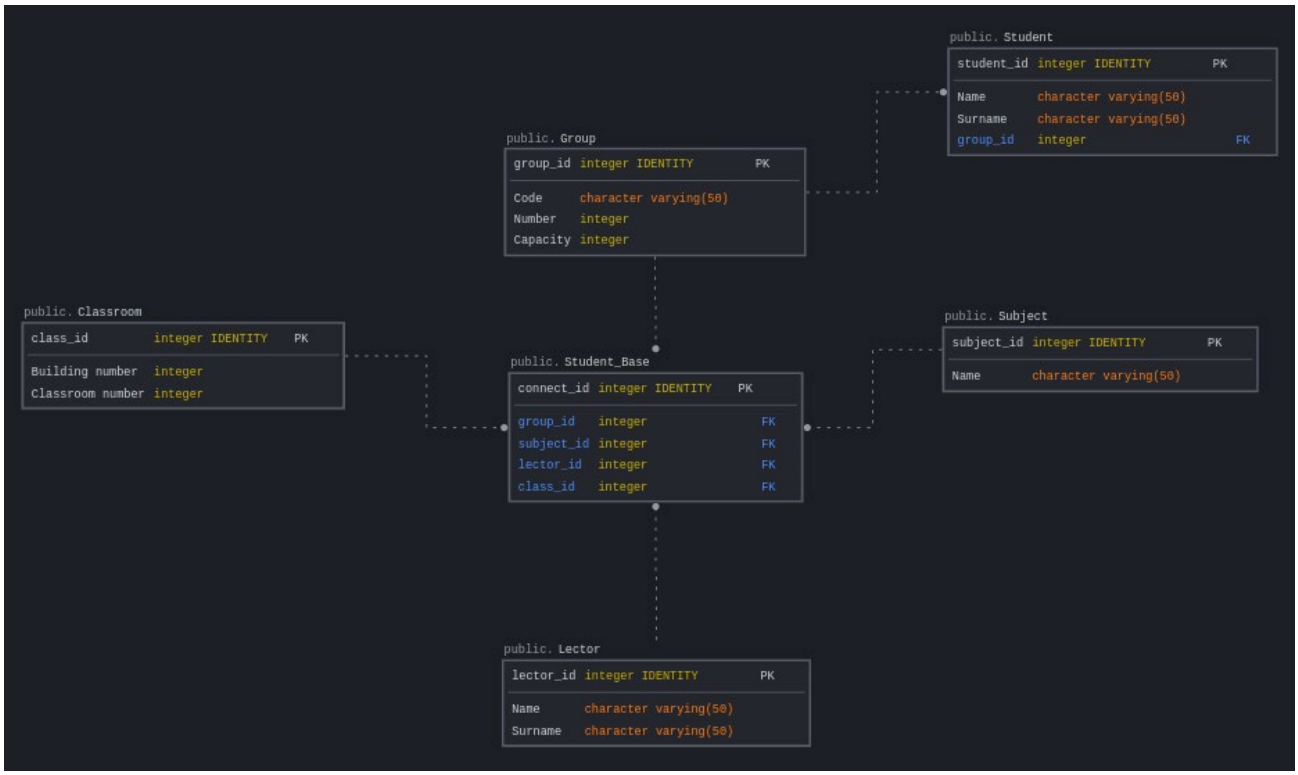


Рисунок 2 - Схема бази даних (інструмент: sqldbm.com)

Опис об’єктів бази даних

Таблиця 1. Опис структури БД

Сутність	Атрибут	Тип атрибуту
Classroom – *містить інформацію про авдиторії.*	**class_id** – *унікальний ідентифікатор авдиторії. Не допускає NULL.* **number** – *номер авдиторії. Не допускає NULL.* **building number** – *номер будівлі. Не допускає NULL.*	**integer** *(числовий)* **integer** *(числовий)* **integer** *(числовий)*
Student – *містить інформацію про студентів.*	**student_id** – *унікальний ідентифікатор студента. Не допускає NULL.* **name** – *ім’я. Не допускає NULL.* **surname** – *прізвище. Не допускає NULL.* **group_id** — *унікальний ідентифікатор групи. Не допускає NULL.*	**integer** *(числовий)* **varchar(50)** *(символьний)* **varchar(50)** *(символьний)* ***integer*** *(числовий)*
Subject – *містить інформацію про предмети.*	**subject_id** – *унікальний ідентифікатор предмета. Не допускає NULL.* **name** – *назва предмету. Не допускає NULL.*	**integer** *(числовий)* **varchar(50)** *(символьний)*
Group – *містить інформацію про групи.*	**group_id** — *унікальний ідентифікатор групи. Не допускає NULL.* **code —** *код групи. Не допускає NULL.* **number —** *номер групи. Не допускає NULL.* **capacity —** *місткість групи. Не допускає NULL.*	**integer** *(числовий)* ***varchar(50)*** *(символьний)* ***integer*** *(числовий)* ***integer*** *(числовий)*
Lector – *містить інформацію про викладачів.*	**lector_id** – *унікальний ідентифікатор студента. Не допускає NULL* **name –** *ім’я. Не допускає NULL* **surname –** *прізвище. Не допускає NULL*	**integer** *(числовий)* **varchar(50)** *(символьний)* **varchar(50)** *(символьний)*
Student_Base –	**connect_id** – *унікальний*	**integer** *(числовий)*

(додаткова сутність) містить ідентифіка-тори авдиторії, викладача, предмета, групи.	*ідентифікатор зв'язку. Не допускає NULL* **class_id** – *унікальний ідентифікатор авдиторії. Не допускає NULL.* **lector_id** – *унікальний ідентифікатор студента. Не допускає NULL* **subject_id** – *унікальний ідентифікатор предмета. Не допускає NULL* **group_id** — *унікальний ідентифікатор групи. Не допускає NULL.*	 **integer** *(числовий)* **integer** *(числовий)* ***integer*** *(числовий)* ***integer*** *(числовий)*

Функціональні залежності для кожної таблиці

CLASSROOM:

class_id – number, building number

class_id – number

class_id – building number

GROUP

group_id — code, number, capacity

group_id — code

group_id — number

group_id — capacity

SUBJECT:

subject_id – name

LECTOR:

lector_id – name, surname

lector_id – name

lector_id – surname

STUDENT:

student_id – name, surname, group_id

student_id – name

student_id – surname

student_id — group_id

STUDENT_BASE

connect_id — group_id, subject_id, lector_id, class_id.

connect_id — group_id

connect_id — subject_id

connect_id — lector_id

connect_id — class_id

Відповідність схеми бази даних до третьої нормальної форми

Схема відповідає 1НФ, тому що в таблиці немає дубльованих рядків; у кожній комірці зберігається атомарне (скалярне) значення; у кожному стовпчику зберігаються дані одного типу.

Схема бази даних відповідає 2НФ тому що відповідає 1НФ, а також всі атрибути залежать від усього первинного ключа цілком, а не від якоїсь його частини.

Схема відповідає 3НФ, тому що вона відповідає 2НФ і кожний неключовий атрибут таблиці залежить тільки від первинного ключа, а не має деяку транзитивну залежність, не залежить від інших неключових атрибутів

Таблиці бази даних у pgAdmin4

Рисунок 3.1 - Схема бази даних у pgAdmin 4

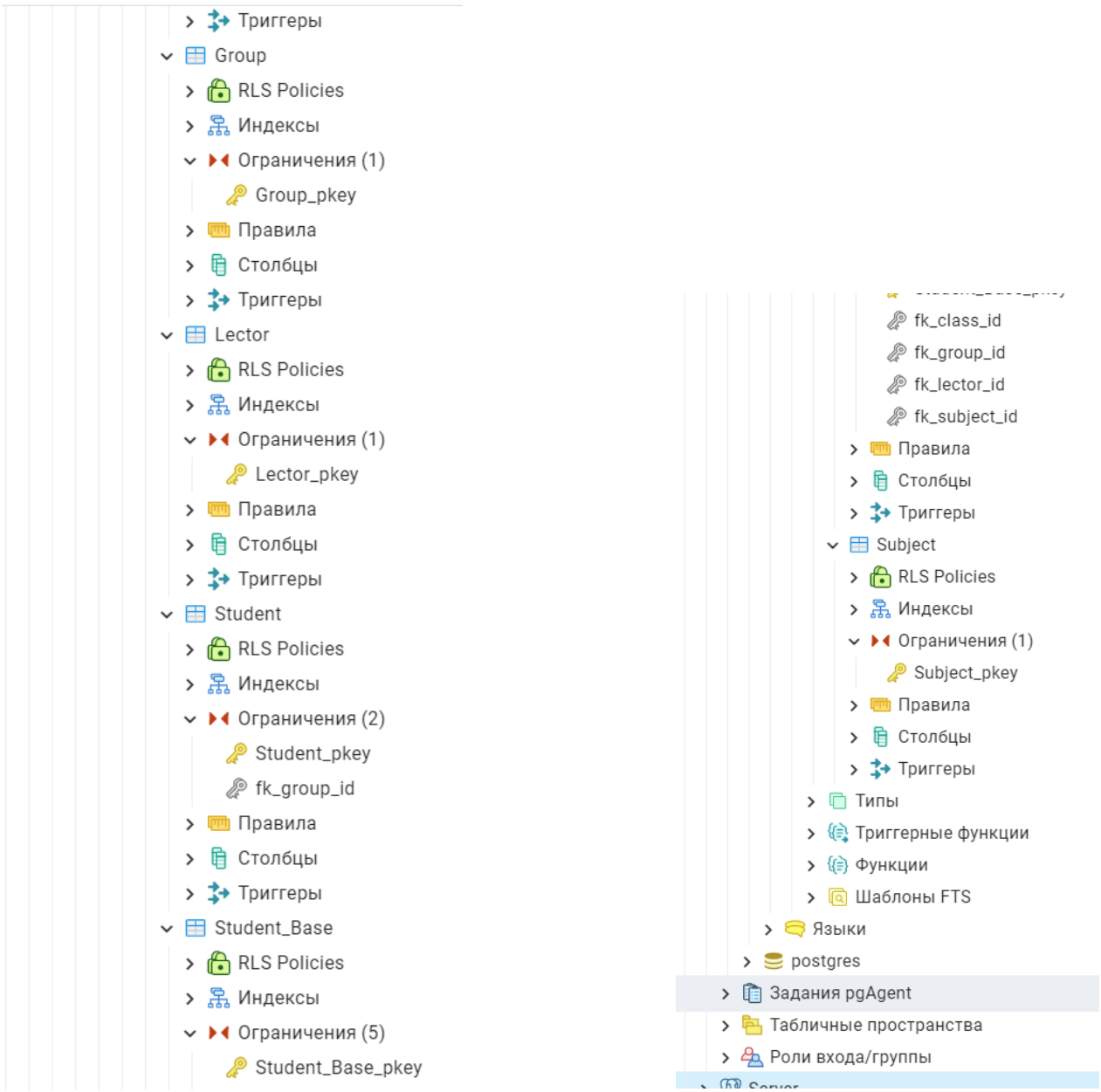
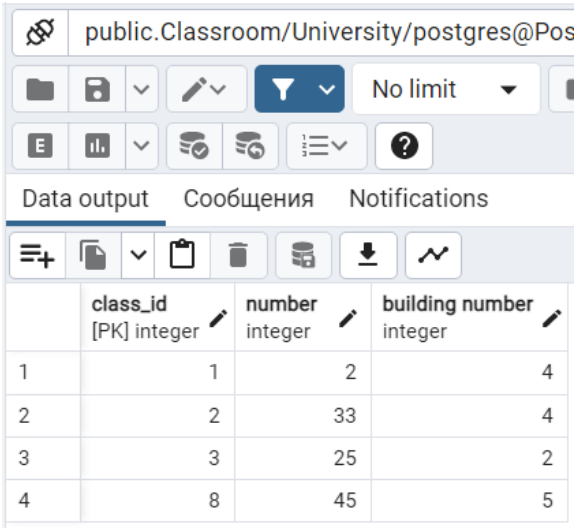


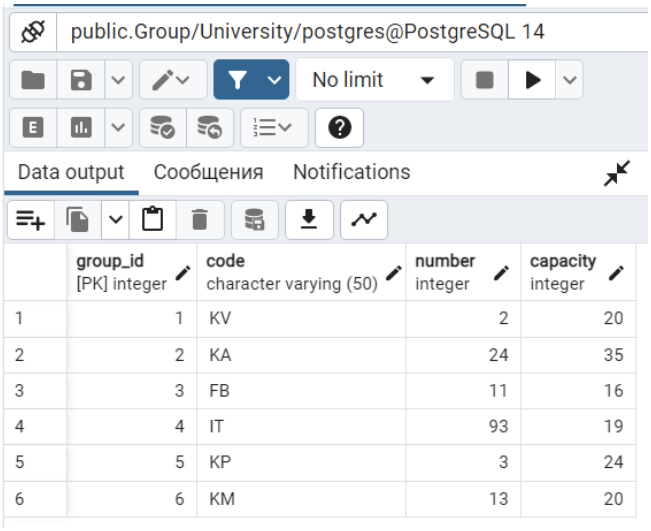
Рисунок 3.2 - Схема бази даних у pgAdmin 4

Фотографії вмісту таблиць



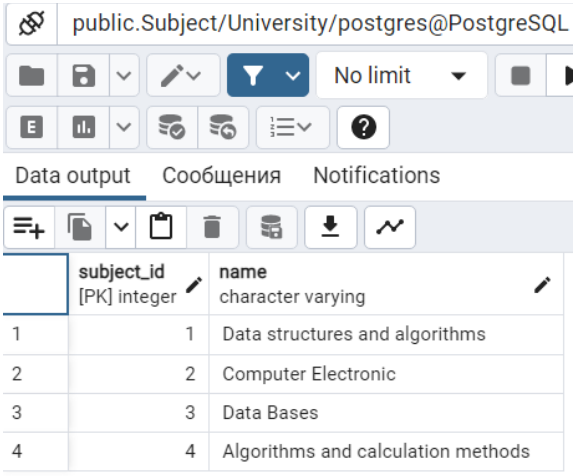
	class_id [PK] integer	number integer	building number integer
1	1	2	4
2	2	33	4
3	3	25	2
4	8	45	5

Рисунок 4. Дані таблиці Classroom



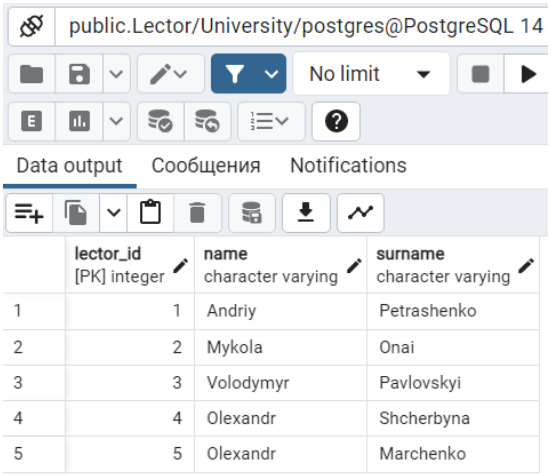
	group_id [PK] integer	code character varying (50)	number integer	capacity integer
1	1	KV	2	20
2	2	KA	24	35
3	3	FB	11	16
4	4	IT	93	19
5	5	KP	3	24
6	6	KM	13	20

Рисунок 5. Дані таблиці Group



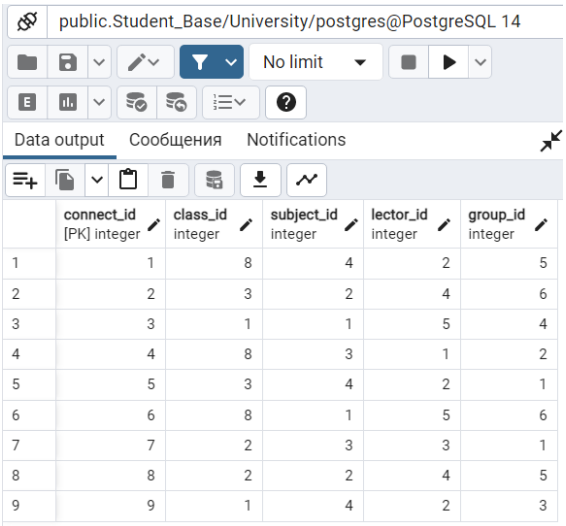
	subject_id [PK] integer	name character varying
1	1	Data structures and algorithms
2	2	Computer Electronic
3	3	Data Bases
4	4	Algorithms and calculation methods

Рисунок 6. Дані таблиці Subject



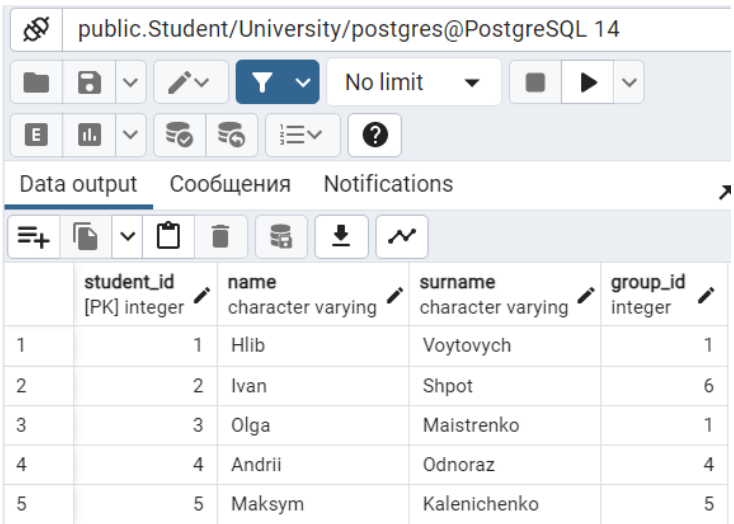
	lector_id [PK] integer	name character varying	surname character varying
1	1	Andriy	Petrashenko
2	2	Mykola	Onai
3	3	Volodymyr	Pavlovskyi
4	4	Olexandr	Shcherbyna
5	5	Olexandr	Marchenko

Рисунок 7. Дані таблиці Lector



	connect_id [PK] integer	class_id integer	subject_id integer	lector_id integer	group_id integer
1	1	8	4	2	5
2	2	3	2	4	6
3	3	1	1	5	4
4	4	8	3	1	2
5	5	3	4	2	1
6	6	8	1	5	6
7	7	2	3	3	1
8	8	2	2	4	5
9	9	1	4	2	3

Рисунок 8. Дані таблиці Student_Base



	student_id [PK] integer	name character varying	surname character varying	group_id integer
1	1	Hlib	Voytovych	1
2	2	Ivan	Shpot	6
3	3	Olga	Maistrenko	1
4	4	Andrii	Odnoraz	4
5	5	Maksym	Kalenichenko	5

Рисунок 9. Дані таблиці Student

SQL-текст опису БД

```
-- Database: University

-- DROP DATABASE IF EXISTS "University";

CREATE DATABASE "University"
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LC_COLLATE = 'Ukrainian_Ukraine.1251'
    LC_CTYPE = 'Ukrainian_Ukraine.1251'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1
    IS_TEMPLATE = FALSE;
-- Table: public.Classroom

-- DROP TABLE IF EXISTS public."Classroom";

CREATE TABLE IF NOT EXISTS public."Classroom"
(
    class_id INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1
MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    "number" INTEGER NOT NULL,
    "building number" INTEGER NOT NULL,
    CONSTRAINT "Classroom_pkey" PRIMARY KEY (class_id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Classroom"
    OWNER TO postgres;
-- Table: public.Group

-- DROP TABLE IF EXISTS public."Group";

CREATE TABLE IF NOT EXISTS public."Group"
(
    group_id INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1
MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    code CHARACTER VARYING(50) COLLATE pg_catalog."default" NOT NULL,
    "number" INTEGER NOT NULL,
    capacity INTEGER NOT NULL,
    CONSTRAINT "Group_pkey" PRIMARY KEY (group_id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Group"
    OWNER TO postgres;
-- Table: public.Lector

-- DROP TABLE IF EXISTS public."Lector";

CREATE TABLE IF NOT EXISTS public."Lector"
(
    lector_id INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1
MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    name CHARACTER VARYING COLLATE pg_catalog."default" NOT NULL,
    surname CHARACTER VARYING COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Lector_pkey" PRIMARY KEY (lector_id)
)

TABLESPACE pg_default;
```

```
ALTER TABLE IF EXISTS public."Lector"
    OWNER TO postgres;
-- Table: public.Subject

-- DROP TABLE IF EXISTS public."Subject";

CREATE TABLE IF NOT EXISTS public."Subject"
(
    subject_id INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1
MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    name CHARACTER VARYING COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Subject_pkey" PRIMARY KEY (subject_id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Subject"
    OWNER TO postgres;
-- Table: public.Student_Base

-- DROP TABLE IF EXISTS public."Student_Base";

CREATE TABLE IF NOT EXISTS public."Student_Base"
(
    connect_id INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1
MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    class_id INTEGER NOT NULL,
    subject_id INTEGER NOT NULL,
    lector_id INTEGER NOT NULL,
    group_id INTEGER NOT NULL,
    CONSTRAINT "Student_Base_pkey" PRIMARY KEY (connect_id),
    CONSTRAINT fk_class_id FOREIGN KEY (class_id)
        REFERENCES public."Classroom" (class_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk_group_id FOREIGN KEY (group_id)
        REFERENCES public."Group" (group_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk_lector_id FOREIGN KEY (lector_id)
        REFERENCES public."Lector" (lector_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk_subject_id FOREIGN KEY (subject_id)
        REFERENCES public."Subject" (subject_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Student_Base"
    OWNER TO postgres;
-- Table: public.Student

-- DROP TABLE IF EXISTS public."Student";

CREATE TABLE IF NOT EXISTS public."Student"
(
    student_id INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1
MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    name CHARACTER VARYING COLLATE pg_catalog."default" NOT NULL,
    surname CHARACTER VARYING COLLATE pg_catalog."default" NOT NULL,
```

```
    group_id INTEGER,
    CONSTRAINT "Student_pkey" PRIMARY KEY (student_id),
    CONSTRAINT fk_group_id FOREIGN KEY (group_id)
        REFERENCES public."Group" (group_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."Student"
    OWNER TO postgres;
```