

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

**Звіт
лабораторної роботи №1**

із дисципліни «Бази даних» на тему
«Практика роботи з запитамі SQL»

Варіант 2

Виконала:
студентка групи КМ-21
Челюскіна Ю.Ф.

Керівник:
асистент Дмитренко О. А.

Київ — 2025

Зміст

1	Мета	2
2	Постановка задачі	2
3	Хід роботи	2
3.1	Запит 1: Агрегація з додатковими функціями	2
3.2	Запит 2: Знайти студентів з однаковими оцінками в різних курсах	3
3.3	Запит 3: Повний аналіз курсів, викладачів, студентів	4
3.4	Запит 4: UNION запит з додатковими умовами	5
4	Висновки.....	2
5	Використані ресурси	3

1 МЕТА

Закріпити навички роботи з SQL таблицями на прикладі бази даних з тематики освітніх курсів університету. Набути досвіду створення таблиць для конкретних запитів та очікуваних результатів.

2 ПОСТАНОВКА ЗАДАЧІ

Створити 3 таблиці(про University Courses, Student, Professor, та Department) та наповнити їх даними, щоб відповіді при виконанні наступних запитів були як очікується.

3 ХІД РОБОТИ

3.1 ЗАПИТ 1: АГРЕГАЦІЯ З ДОДАТКОВИМИ ФУНКЦІЯМИ

Очікуваний результат:

dept_name	professor_count	total_salary	avg_salary	min_salary	max_salary	salary_deviation	mode_salary
Biology	1	85000.00	85000.000000000000	85000.00	85000.00	null	85000.00
Computer Science	1	95000.00	95000.000000000000	95000.00	95000.00	null	95000.00
Mathematics	1	88000.00	88000.000000000000	88000.00	88000.00	null	88000.00
Physics	1	92000.00	92000.000000000000	92000.00	92000.00	null	92000.00

Отриманий результат:

	dept_name character varying (50) 🔒	professor_count bigint	total_salary double precision	avg_salary double precision 🔒	min_salary double precision 🔒	max_salary double precision 🔒	salary_deviation double precision 🔒	mode_salary double precision 🔒
1	Biology	1	85000	85000	85000	85000	[null]	85000
2	Computer Science	1	95000	95000	95000	95000	[null]	95000
3	Mathematics	1	88000	88000	88000	88000	[null]	88000
4	Physics	1	92000	92000	92000	92000	[null]	92000

По-перше було необхідно створити дві таблиці. Причому таблиця Professor залежить від Departments, адже викладачі відносяться до своїх відділів.

На цьому етапі код створення таблиць виглядає наступним чином:

```
CREATE TABLE Department(  
dept_id SERIAL,  
dept_name VARCHAR(50),  
PRIMARY KEY (dept_id)  
);
```

```
CREATE TABLE Professor(  
prof_id SERIAL,  
salary DOUBLE PRECISION,  
dept_id INTEGER NOT NULL,  
PRIMARY KEY (prof_id),  
FOREIGN KEY (dept_id) references Department(dept_id)  
);
```

Це достатня база для подальшого удосконалення бази даних.



Робота здійснюється таким чином, що в коді створюються таблиці, наповнюються необхідними даними за допомогою INSERT INTO, а після основного коду є рядки, що видаляють таблицю (DROP TABLE). Це дозволяє зручно вдосконалювати код.

3.2 ЗАПИТ 2: ЗНАЙТИ СТУДЕНТІВ З ОДНАКОВИМИ ОЦІНКАМИ В РІЗНИХ КУРСАХ

Очікуваний результат:

student1	student2	grade	course1	course2
Alice	Charlie	4.00	1	3

Отриманий результат:

	student1 character varying (20) 🔒	student2 character varying (20) 🔒	grade double precision 🔒	course1 integer 🔒	course2 integer 🔒
1	Alice	Charlie	4	1	3

Після аналізу цього запиту було створено три нових таблиці. Це Course, Student та Enrollement, як пов'язує дві перші.

В процесі через неувважність було пропущено атрибут name для студента, що призвело до помилки:

```
ERROR: стовпець s1.name не існує  
LINE 3:      s1.name as student1,  
          ^  
  
ПОМИЛКА: стовпець s1.name не існує  
SQL state: 42703  
Character: 36
```

Як результат код таблиць виглядає так:

```

CREATE TABLE Course(
course_id SERIAL PRIMARY KEY
);

CREATE TABLE Student(
student_id SERIAL PRIMARY KEY,
name VARCHAR(20)
);

CREATE TABLE Enrollment(
student_id INTEGER NOT NULL,
course_id INTEGER NOT NULL,
grade Double Precision CHECK (grade<=5 AND grade >=0),
PRIMARY KEY (student_id, course_id),
FOREIGN KEY (student_id) REFERENCES Student(student_id),
FOREIGN KEY (course_id) REFERENCES Course(course_id)
);

```

3.3 ЗАПИТ 3: ПОВНИЙ АНАЛІЗ КУРСІВ, ВИКЛАДАЧІВ, СТУДЕНТІВ

Очікуваний результат:

course_name	professor	dept_name	enrolled_students	avg_course_grade	max_grade	min_grade
Advanced Algorithms	Dr. Smith	Computer Science	1	4.00	4.00	4.00
Quantum Mechanics	Dr. Williams	Physics	1	4.00	4.00	4.00
Molecular Biology	Dr. Brown	Biology	1	3.80	3.80	3.80
Linear Algebra	Dr. Johnson	Mathematics	2	3.60	3.70	3.50

Отриманий результат:

	course_name character varying (20)	professor character varying (20)	dept_name character varying (50)	enrolled_students bigint	avg_course_grade numeric	max_grade double precision	min_grade double precision
1	Advanced Algorithms	Dr. Smith	Computer Science	1	4.00	4	4
2	Quantum Mechanics	Dr. Williams	Physics	1	4.00	4	4
3	Molecular Biology	Dr. Brown	Biology	1	3.80	3.8	3.8
4	Linear Algebra	Dr. Johnson	Mathematics	2	3.60	3.7	3.5


Спершу було додано атрибути назви курсу, ім'я викладача, FK викладача в Enrollment та FK ід відділу в курс.

Після цих дій, а також зміни даних в таблицях отримали такий результат:

	course_name character varying (20)	professor character varying (20)	dept_name character varying (50)	enrolled_students bigint	avg_course_grade numeric	max_grade double precision	min_grade double precision
1	Linear Algebra	[null]	Mathematics	0	[null]	[null]	[null]
2	Quantum Mechanics	[null]	Physics	0	[null]	[null]	[null]
3	Advanced Algorithms	Dr. Smith	Computer Science	1	4.00	4	4
4	Molecular Biology	Dr. Brown	Biology	1	4.00	4	4

Очевидно, що в нас наразі недостатня кількість студентів, що впливає на результат.

Отже спочатку додаємо нових студентів до таблиці Students (всього має бути 5), а потім робимо відношення між ними та курсами таким чином, щоб досягти бажаного результату.

 В процесі зміни коду та додавання нових залежностей варто звертати увагу на порядок видалення таблиць, щоб не отримати подібні помилки:



```
ERROR: обмеження enrollment_professor_id_fkey на таблиця enrollment
залежить від таблиця professor неможливо видалити таблиця professor,
тому що від нього залежать інші об'єкти
```

3.4 Запит 4: UNION запит з додатковими умовами

Очікуваний результат:

name	status
Alice	High Performer
Charlie	High Performer
David	Average Performer
Bob	Average Performer

Отриманий результат:

	name character varying (20) 	status text 
1	Alice	High Performer
2	Charlie	High Performer
3	David	Average Performer
4	Bob	Average Performer

Так як в нас вже є чимало атрибутів та таблиць, простіше було перевірити що саме не працює, ніж аналізувати кожен рядок.

Після запуску коду отримали очікувану помилку:

```
ERROR: стовпець s.gpa не існує
LINE 6: WHERE s.gpa >= 3.7
              ^

ПОМИЛКА: стовпець s.gpa не існує
SQL state: 42703
Character: 87
```

На цьому етапі стало зрозуміло, що в нас всього 4 студенти, а не 5, як здавалось, тобто один студент проходить два курси, що є логічно.

Хоча таблиця Enrollment не передбачала такого використання, тому було додано окремий первинний ключ, який не вимагає від студента, викладача чи відділу бути унікальним значенням в цій таблиці:

```
ERROR: Ключ (student_id, course_id, professor_id)=(3, 4, 4) вже існує.повторювані значення ключа порушують обмеження унікальності "enrollment_pkey"
```

```
CREATE TABLE Enrollment(  
  enrollment_id SERIAL PRIMARY KEY,  
  student_id INTEGER NOT NULL,  
  course_id INTEGER NOT NULL,  
  professor_id INTEGER NOT NULL,  
  grade Double Precision CHECK (grade<=5 AND grade >=0),  
  FOREIGN KEY (student_id) REFERENCES Student(student_id),  
  FOREIGN KEY (course_id) REFERENCES Course(course_id),  
  FOREIGN KEY (professor_id) REFERENCES Professor(prof_id),  
);
```

Також, власне повертаючись до першої помилки, було додано атрибут gpa до таблиці студентів:

```
CREATE TABLE Student(  
  student_id SERIAL PRIMARY KEY,  
  name VARCHAR(20),  
  gpa Double Precision CHECK (gpa<=5 AND gpa >=0),  
);
```

4 ВИСНОВКИ

В ході цієї лабораторної роботи було повторено та закріплено знання про створення БД в PostgreSQL.

Виконуючи завдання, вдалося набути досвіду аналізу вже створених запитів до БД. Всі запити були добре зрозумілі та логічні, тому особливих проблем не викликали. Найкраще запам'ятались команди Join та Group By, адже саме вони найбільше впливали на очікуваний результат.

Фінальний вміст БД виглядає так:

```
INSERT INTO Department (dept_id, dept_name)
VALUES
(1, 'Biology'),
(2, 'Computer Science'),
(3, 'Mathematics'),
(4, 'Physics');
```

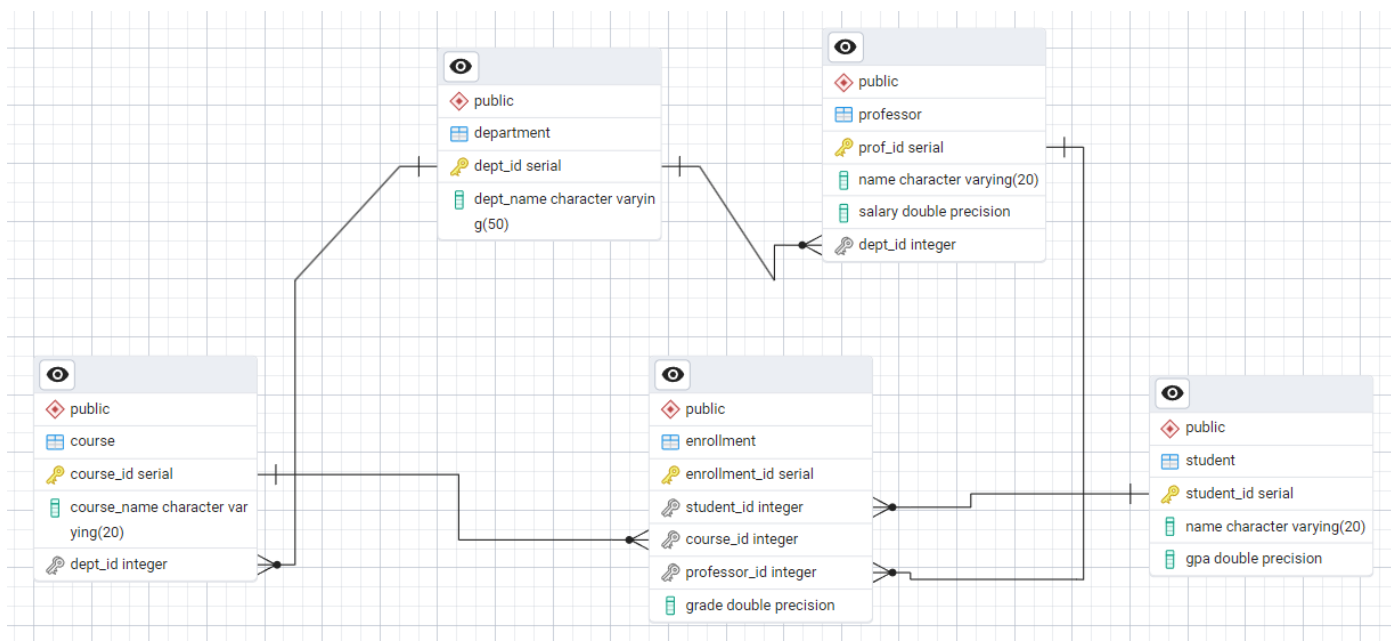
```
INSERT INTO Professor(prof_id, name, salary, dept_id) VALUES
(1, 'Dr. Smith', '85000.00', 1),
(2, 'Dr. Williams', '95000.00', 2),
(3, 'Dr. Brown', '88000.00', 3),
(4, 'Dr. Johnson', '92000.00', 4);
```

```
INSERT INTO Course(course_id, course_name, dept_id) VALUES
(1, 'Advanced Algorithms', 2),
(3, 'Quantum Mechanics', 4),
(2, 'Molecular Biology', 1),
(4, 'Linear Algebra', 3);
```

```
INSERT INTO Student(student_id, name, gpa) VALUES
(1, 'Alice', 4),
(2, 'David', 3.6),
(3, 'Bob', 3.5),
(4, 'Charlie', 3.8)
;
```

```
INSERT INTO Enrollment (student_id, course_id, professor_id, grade) VALUES
(1, 1, 1, 4),
(3, 4, 4, 3.5),
(2, 4, 4, 3.7),
(4, 3, 2, 4),
(2, 2, 3, 3.8);
```


ERD виглядає так:



5 ВИКОРИСТАНІ РЕСУРСИ

1. Інформація по SQL запитам - <https://www.geeksforgeeks.org/sql-tutorial/?ref=lbp>
2. Синтаксис PostgreSQL та приклади використання - <https://www.w3schools.com/postgresql/index.php>