МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

до виконання лабораторних робіт з дисципліни
«Бази даних»
для здобувачів освітнього ступеня «бакалавр»
галузі знань F «Інформаційні технології»
спеціальності F3 «Комп’ютерні науки»
освітньої програми «Комп’ютерні науки та прикладне програмування»
усіх форм навчання

Черкаси
2026

УДК 004.65 (07)
М54

Затверджено вченою радою ФІТІС, протокол №9 від 26.02.2026 р., згідно з рішенням кафедри комп'ютерних наук та системного аналізу, протокол №9 від 05.01.2026 р.

Упорядники: Максимов А.Є., *PhD з комп'ютерних наук, викладач кафедри комп'ютерних наук та системного аналізу*
Триус Ю.В., *професор, к.ф.-м.н., д. пед. н., завідувач кафедри комп'ютерних наук та системного аналізу*

Рецензент: Миронець І.В., *к.т.н., доцент*

М54 Методичні рекомендації до виконання лабораторних робіт з дисципліни «Бази даних» для здобувачів освітнього ступеня «бакалавр» галузі знань F «Інформаційні технології», спеціальності F3 «Комп'ютерні науки», освітньої програми «Комп'ютерні науки та прикладне програмування» усіх форм навчання [Електронний ресурс] / [упоряд. Максимов А.Є., Триус Ю.В.]; М-во освіти і науки України, Черкас. держ. технол. ун-т. Черкаси: ЧДТУ, 2026. 71 с.

Методичні рекомендації спрямовані на формування у здобувачів освітнього ступеня «бакалавр» галузі знань «12 (F) Інформаційні технології», спеціальності 122 (F3) «Комп'ютерні науки», освітньої програми «Комп'ютерні науки та прикладне програмування» базових знань і практичних навичок з проєктування, моделювання та використання реляційних і нереляційних баз даних, опрацювання мов запитів (SQL та NoSQL-підходів), оптимізації структури даних і організації сховищ, а також сприяє розвитку системного, логічного та аналітичного мислення у сфері комп'ютерних наук.

УДК 004.65 (07)

Виробничо-практичне
електронне видання
комбінованого використовування

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

до виконання лабораторних робіт з дисципліни
«Бази даних»
для здобувачів освітнього ступеня «бакалавр»
галузі знань F «Інформаційні технології»
спеціальності F3 «Комп'ютерні науки»
освітньої програми «Комп'ютерні науки та прикладне програмування»
усіх форм навчання

Упорядники: **Максимов** Антон Євгенійович, **Триус** Юрій Васильович
В авторській редакції.

ВСТУП

Формування та розвиток практичних умінь студентів, що здобувають освітній рівень бакалавра, у процесі вивчення дисципліни «Бази даних» передбачає оволодіння знаннями з проєктування, моделювання та реалізації структур даних у реляційних і нереляційних системах, планування та організації зберігання інформації, а також набуття навичок у розв'язанні прикладних і навчально-наукових задач, пов'язаних з обробкою даних. Для цього необхідна система знань щодо основних понять теорії баз даних, моделей даних, концептуального, логічного та фізичного проєктування, принципів роботи з SQL та NoSQL інструментами. Підґрунтям виступає розуміння методології створення схем баз даних, принципів нормалізації, індексування, оптимізації запитів, а також знання сфер застосування сучасних СКБД, їхніх переваг, обмежень і типових сценаріїв використання.

Метою викладання дисципліни «Бази даних» є формування у студентів теоретичних знань про принципи побудови, функціонування та застосування реляційних і нереляційних баз даних у сучасних інформаційних системах, практичних навичок проєктування, розробки, адміністрування баз даних та ефективного використання мов запитів (SQL і NoSQL) для обробки даних, компетентностей із вибору, впровадження та інтеграції баз даних різних типів залежно від потреб конкретної інформаційної системи.

Основними завданнями вивчення дисципліни «Бази даних» є формування у здобувачів здатності проєктувати бази даних відповідно до вимог інформаційної системи, використовувати реляційні та нереляційні системи для зберігання й аналізу даних, виконувати оптимізацію запитів та налаштовувати системи управління базами даних, а також інтегрувати різні типи баз даних для побудови сучасних інформаційних систем.

Мета лабораторних робіт полягає у забезпеченні розуміння та засвоєння здобувачами бакалаврського освітнього рівня змісту дисципліни «Бази даних» шляхом свідомого закріплення, поглиблення й систематизації теоретичних знань, а також набуття практичних навичок проєктування, реалізації, налагодження та оптимізації реляційних і нереляційних баз даних для ефективного зберігання, обробки та аналізу даних у сучасних інформаційних системах.

Методичні рекомендації до лабораторних робіт здобувачів бакалаврського освітнього рівня із навчальної дисципліни «Бази даних» містять: вступ; чотирнадцять лабораторних робіт, які охоплюють основні теми вказаного курсу; список використаної літератури; додатки. Під час виконання робіт здобувачі вивчають матеріали окремих тем шляхом опрацювання відповідної літератури, виконують лабораторні завдання, здійснюють підготовку до лабораторних робіт та поточного контролю знань.

Цільовим призначенням видання є забезпечення навчально-методичної підтримки у вивченні дисципліни «Бази даних», підготовці й проведенні наукових досліджень для студентів за спеціальністю 122 (F3) «Комп'ютерні науки».

Лабораторна робота (ЛР) вважається виконаною за умови повного й коректного виконання всіх завдань, передбачених методичними рекомендаціями; подання коректного рішення без критичних помилок; обґрунтованого використання

інструментів і середовища розробки; належного оформлення роботи, коду та результатів роботи (див. Додаток А), а також дотримання принципів академічної доброчесності. Максимальна оцінка за виконану ЛР складає 5 балів. Бали розподіляються наступним чином: за виконання загальних (аудиторних) завдань до ЛР – 2 бали, за виконання індивідуальних завдань до ЛР – 3 бали.

ЛАБОРАТОРНА РОБОТА №1
Тема: «Встановлення та налаштування СУБД MySQL»

Мета роботи: Здійснити інсталяцію та налаштування програмного забезпечення для роботи з реляційною системою керування базами даних MySQL, ознайомитися з веб-додатком phpMyAdmin, вивчити його можливості для роботи з базами даних.

Завдання до лабораторної роботи

1. У встановленому середовищі *phpMyAdmin* створити і заповнити базу даних *schools*, що містить основну таблицю *school* із заданими полями:

id, name, address, director, telephone, email, *site.*

При створенні таблиці school вказати, які поля є обов'язковими (виділено жирним шрифтом). Структуру таблиці school подано у таблиці 1.1.

Таблиця 1.1 – Сутність (таблиця) «school»

#	Назва стовпчика (поля)	Тип даних	Обмеження	Опис
1	id	INT NOT NULL PRIMARY KEY	Первинний ключ	Ідентифікатор школи в БД
2	name	VARCHAR 50 NOT NULL		Назва школи
3	address	VARCHAR 50 NOT NULL		Адреса школи
4	director	VARCHAR 50 NOT NULL		ПІП директора школи
5	telephone	VARCHAR 13 NOT NULL		Телефон школи
6	email	VARCHAR 30 NOT NULL		Електронна пошта школи
7	site	VARCHAR 50		Посилання на сайт школи

2. До бази даних ввести не менше ніж 10 записів, що містять відомості про реальні школи м. Черкаси (їх можна одержати з мережі Інтернет) та про школу, де Ви навчались.

3. Після заповнення таблицю school зберегти.

Індивідуальні завдання до лабораторної роботи

В середовищі phpMyAdmin створити відповідну базу даних із заданими таблицями. Таблиці повинні складатися не менше ніж з 5 записів. Після заповнення таблиці зберегти і зробити копію бази даних.

Здійснити редагування створеної бази даних: додати запис, видалити запис, зробити переміщення записів у базі даних і перегляд даних.

Варіанти:

1. Створити і заповнити базу даних discipline, що містить таблиці:
   - student (прізвище, ім'я, по батькові студента, дата народження, курс, на якому навчається студент, факультет, на якому навчається студент);

– discipline (назва предмету, прізвище лектора, корпус, аудиторія, вид занять).

2. Створити і заповнити базу даних aviaticket, що містить таблиці:
   – ticket (прізвище пасажира, номер білету, дата продажу, рейс, дата вильоту, вартість квитка, місце);
   – flight (рейс, пункт призначення, платформа).
3. Створити і заповнити базу даних materials, що містить таблиці:
   – material (назва матеріалу, категорія матеріалу, інформація про матеріал, ціна);
   – storage (назва складу, назва матеріалу, кількість).
4. Створити і заповнити базу даних cars, що містить таблиці:
   – car (номер авто, марка, рік випуску, колір);
   – driver (ПІБ водія, номер водійських прав, дата народження, кількість попереджень).
5. Створити і заповнити базу даних shop, що містить таблиці:
   – buyer (прізвище, ім'я, по батькові покупця, код дисконту, назва товару);
   – product (назва товару, категорія товару, кількість на складі, термін зберігання, ціна).
6. Створити і заповнити базу даних hospital, що містить таблиці:
   – illman (прізвище, ім'я, по батькові хворого, дата народження, палата, діагноз);
   – doctor (ПІБ лікаря, кваліфікація, стаж роботи).
7. Створити і заповнити базу даних library, що містить таблиці:
   – book (автор, назва, рік видання, видавництво, кількість сторінок, кількість в читальному залі, жанр);
   – visitor (ПІБ читача, адреса, дата відвідування бібліотеки, назва книги).
8. Створити і заповнити базу даних kafedra, що містить таблиці:
   – teacher (прізвище, ім'я, по батькові викладача, науковий ступінь, вчене звання, посада, предмет, який читає);
   – discipline (назва предмету, прізвище лектора, курс, група, кількість годин, форма підсумкового контролю).
9. Створити і заповнити базу даних dbmss, що містить таблиці:
   – dbms (назва СУБД, виробник, ОС, вартість, модель даних, остання версія);
   – developer (назва фірми виробника СУБД, країна, рік заснування, назва СУБД, яку розробляє).
10. Створити і заповнити базу даних faculties, що містить таблиці:
    – faculty (назва, прізвище декана, кількість студентів, телефон, адреса);
    – kafedra (назва, прізвище завідувача, назва напряму (спеціальності), з якого кафедра є випусковою, телефон).
11. Створити і заповнити базу даних hotel, що містить таблиці:
    – guest (прізвище, ім'я, по батькові гостя, дата заселення, дата виселення, номер кімнати);

– room (номер кімнати, тип кімнати, кількість місць, вартість за добу, поверх).

12. Створити і заповнити базу даних cinema, що містить таблиці:
    – movie (назва фільму, жанр, рік випуску, тривалість, режисер);
    – session (назва фільму, дата показу, час початку, зал, вартість квитка).
13. Створити і заповнити базу даних bank, що містить таблиці:
    – client (ПІБ клієнта, дата народження, паспортні дані, адреса, телефон);
    – account (номер рахунку, ПІБ клієнта, тип рахунку, дата відкриття, баланс).
14. Створити і заповнити базу даних restaurant, що містить таблиці:
    – client (прізвище, ім’я, по батькові клієнта, номер телефону, дата відвідування, номер столика);
    – menu (назва страви, категорія, вага порції, ціна).
15. Створити і заповнити базу даних sportclub, що містить таблиці:
    – member (ПІБ відвідувача, дата народження, тип абонемента, дата початку дії);
    – trainer (ПІБ тренера, спеціалізація, стаж роботи, розклад занять).

Контрольні питання

1. Що таке СУБД і які основні функції виконує MySQL?
2. Які типи даних підтримує MySQL?
3. Яка структура типового конфігураційного файлу my.cnf (або my.ini)?
4. Які механізми автентифікації користувачів підтримує MySQL?
5. Які переваги має MySQL у порівнянні з іншими СУБД, наприклад PostgreSQL або SQLite?
6. Що таке phpMyAdmin і для чого він використовується?
7. Як встановити MySQL Server на операційну систему Windows або Linux?
8. Як створити базу даних і користувача з відповідними правами в MySQL?
9. Як перевірити стан служби MySQL у системі?
10. Як створити резервну копію бази даних за допомогою утиліти mysqldump?
11. Як змінити пароль користувача MySQL через командний рядок?
12. Як налаштувати віддалений доступ до MySQL?
13. Як встановити та запустити phpMyAdmin?
14. Як підключити phpMyAdmin до локального або віддаленого MySQL-сервера?
15. Що таке MAMP, які його основні компоненти (MySQL, Apache, PHP) і як за його допомогою розгорнути локальний сервер?

ЛАБОРАТОРНА РОБОТА №2
Тема: «Основні оператори мови SQL. Створення бази даних засобами мови SQL»

Мета роботи: Вивчення типів і форматів даних мови SQL, способів і засобів мови SQL для створення бази даних і таблиць даних, а також формування вмінь і навичок щодо створення запитів на заповнення таблиці даними за допомогою оператора INSERT, оновлення (UPDATE), вилучення (DELETE) та вибірку даних (SELECT).

Завдання до лабораторної роботи

1. Зайти в середовище phpMyAdmin для роботи з БД MySQL.
2. Відкрити або створити (завантажити) базу даних *schools*.
3. У базі даних *schools* відкрити таблицю *school*.
4. Ввести реальні дані про Першу міську гімназію м. Черкаси.
5. Виконати до бази даних *schools* такі запити:
   - **Insert:**
     - додати нову школу м. Черкаси (наприклад ЗОШ №2) з відповідними реальними даними за допомогою запиту;
     - додати дані про нову школу м. Черкаси (наприклад ЗОШ №34) з відповідними реальними даними за допомогою запиту;
     - додати нову нереальну школу з нереальними даними за допомогою запиту.
   - **Update:**
     - змінити в запису даних про Першу міську гімназію дані в полі «телефон» на номер за форматом: (0472)337121;
     - змінити в запису даних про ЗОШ №34 дані в полі «Телефон» на інший номер, наприклад: (0472)112233.
   - **Delete:**
     - видалити створену нереальну школу з всіма даними з бази даних;
     - видалити створену ЗОШ №34 з всіма даними з бази даних.
   - **Select:**
     - за допомогою запиту вивести всі дані з таблиці school;
     - за допомогою запиту вивести всі дані з таблиці school з назвами стовпчиків українською мовою;
     - вибрати інформацію про школи, які мають свій сайт;
     - вибрати школи, у яких є міський номер телефону, тобто номер, у якого є такий запис (0472);
     - вибрати школи, у яких адреса сайту містить ck.ua, використовуючи предикат site LIKE '%ck.ua%'.
6. Зберегти базу даних schools у файлі schools.sql**.**
7. Подивитися більш детально інформацію про основні оператори в MySQL за посиланням: https://www.w3schools.com/sql/default.asp

Індивідуальні завдання до лабораторної роботи

1. Розробити структуру бази даних з певної предметної області згідно з варіантом індивідуального завдання, виділивши основну таблицю даних і допоміжні таблиці даних.

2. Створити за допомогою phpMyAdmin базу даних, використовуючи оператор CREATE DATABASE.

3. Створити в базі даних основну таблицю за допомогою оператора CREATE TABLE.

4. Заповнити створену таблицю даними за допомогою оператора INSERT.

5. Розробити запити на оновлення (UPDATE), вилучення (DELETE) та просту вибірку (SELECT) даних.

6. Виконати розроблені запити та перевірити їх результати.

7. Зберегти створену базу даних і запити.

Варіанти:

1. Проєктна установа ліцензована на розробку архітектурних проєктів і складається з декількох відділів. Кожен відділ очолює завідувач і в ньому працюють співробітники на певних посадах. Відділи розташовані в певних приміщеннях і мають телефони. Співробітники працюють в конкретних приміщеннях і мають телефон. Співробітник може виконувати один або декілька проєктів. Кожний проєкт має номер, назву, дату початку і закінчення, а також керівника.

2. На кафедрі, котра готує спеціалістів з конкретної спеціальності, працюють викладачі. Викладачі займають певні посади, мають вчене звання й ступінь і читають конкретні дисципліни. Дисципліни характеризуються назвою, кількістю годин, семестром вивчення, формою контролю. Одну дисципліну можуть вести декілька викладачів, а один викладач – декілька дисциплін.

3. Клієнт отримує продукцію зі складу за накладною. В накладній вказано: номер, ім’я клієнта, назва продукції, код, кількість і вартість. В одній накладній може бути вказано декілька видів продукції. Кожен вид продукції характеризується назвою, кодом, виробником, адресою виробника. Клієнт характеризується кодом, ім’ям, адресою, рахунком.

4. Бібліотека має декілька відділів. Відділ характеризується назвою, номером, номером приміщення, номером телефону. Читач може бути записаним в декількох відділах і брати там книги. Читач характеризується номером, ПІБ, адресою, телефоном. Книга має назву, автора, УДК, ББК.

5. Клієнт має банківські рахунки в різних банках. Банк характеризується, назвою, адресою, № телефону, № ліцензії. Клієнт має ПІБ, адресу, телефон. Банківський рахунок характеризується номером, номером банку, номером клієнта, видом рахунку, сумою.

6. Акції акціонерного товариства зберігаються у зберігача. Одне акціонерне товариство може зберігати свої акції у декількох зберігачів а один зберігач може зберігати акції декількох акціонерних товариств. Акціонерне товариство характеризується назвою, адресою, № телефону. Зберігач акцій характеризується назвою, адресою, № телефону. Акція характеризується номером, номером категорії, датою емісії, номіналом.

7. Товари відправляються в контейнерах морським транспортом, певним рейсом. Рейс характеризується типом і назвою судна, портом відправлення та портом прибуття, датою відправлення і датою прибуття. Контейнер має номер, опис товару, правила розвантаження. На контейнер оформляється накладна, в яку заноситься номер контейнера, назва судна, номер рейса, вантажоодержувач.

8. Корпорація складається із декількох підприємств. Кожне підприємство випускає декілька видів продукції і поставляє на склади, які розташовані в різних містах. Підприємство характеризується назвою, адресою, видами продукції та її кількістю. Продукція характеризується назвою, кодом, упаковкою. Склад характеризується адресою, назвою. Поставка характеризується кодом продукції, датою, кількістю.

9. Покупець купує товари в декількох магазинах єдиної мережі, розраховуючись з кредитної картки. Покупець характеризується видом і номером кредитної картки, сумою кредиту, залишком кредиту. Покупка характеризується номером, вартістю, номером магазину, відміткою про сплату, датою, № кредитної картки. Магазин характеризується номером, адресою, № телефону, № розрахункового рахунку.

10. Корпоративна мережа складається з декількох сегментів локальних мереж. В кожному сегменті може бути декілька вузлів. Деякі вузли можуть належати декільком сегментам (шлюзи, мости, маршрутизатори). Мережа характеризується назвою, типом, адресою. Сегмент мережі характеризується назвою, IP-адресою, кількістю вузлів, типом доступу до мережі. Вузли (або станції) характеризуються номером, IP-адресою, локальним ім’ям, типом вузла.

11. Будівельна організація бідує різні будинки. В будівництві приймають участь декілька бригад робітників різних спеціальностей, що використовують різні матеріали. Будівля характеризується типом, номером майданчика, датою здачі. Бригада характеризується спеціальністю, бригадиром, кількістю робітників. Матеріали характеризуються назвою, специфікацією, кількістю. Одна бригада може працювати на декількох будовах.

12. Менеджер продає товари клієнтам. На продаж товару оформляється замовлення. Менеджер характеризується ПІБ, адресою, датою прийняття на роботу. Товар має назву, модель або артикул, кількість на складі, вартість одиниці. Клієнт характеризується ПІБ і адресою. Замовлення включає дату, час, назву товару, кількість, вартість.

13. Туристична фірма організовує подорожі для клієнтів. Клієнти можуть замовляти одну або декілька туристичних поїздок. Кожна поїздка проводиться за певним маршрутом і має відповідального менеджера. Маршрут характеризується країною, містом, датою початку і завершення, вартістю. Менеджер характеризується ПІБ, посадою, номером телефону. Клієнт характеризується ПІБ, паспортними даними, адресою, номером телефону.

14. Медичний центр складається з кількох відділень. У кожному відділенні працюють лікарі різних спеціалізацій, які приймають пацієнтів. Пацієнт може відвідувати декількох лікарів. Відділення характеризується назвою, номером кабінету, номером телефону. Лікар характеризується ПІБ, спеціалізацією, стажем

роботи. Пацієнт характеризується ПІБ, датою народження, адресою, номером телефону. Прийом характеризується датою і часом відвідування.

15. Інтернет-магазин продає товари покупцям через систему онлайн-замовлень. Один покупець може оформити декілька замовлень, кожне з яких містить один або декілька товарів. Товар характеризується назвою, кодом, категорією, ціною. Покупець характеризується ПІБ, електронною поштою, адресою доставки, номером телефону. Замовлення характеризується номером, датою оформлення, статусом, загальною вартістю.

Контрольні питання

1. Що таке SQL та які основні групи операторів у ньому виділяють?
2. Для чого призначені оператори DDL, DML, DQL, DCL та TCL?
3. Яким оператором створюється нова база даних у SQL?
4. Які основні параметри можна вказувати під час створення таблиці за допомогою оператора CREATE TABLE?
5. Чим відрізняються обмеження PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK та NOT NULL?
6. Для чого використовується оператор INSERT, і які існують способи вставки даних у таблицю?
7. У чому полягає різниця між операторами UPDATE та DELETE?
8. Яка роль оператора SELECT, та які основні складові його структури (SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY)?
9. Як виконується зміна структури таблиці за допомогою оператора ALTER TABLE?
10. Яким оператором видаляються таблиці та бази даних, і які ризики пов’язані з його використанням?
11. Яке призначення оператора CREATE INDEX, і в яких випадках індекси варто створювати?
12. У чому полягає різниця між логічним та фізичним видаленням даних у SQL?

ЛАБОРАТОРНА РОБОТА №3
Тема: «Створення реляційної бази даних. Робота з базою даних із пов'язаними таблицями»

Мета роботи: Навчитись проєктувати та створювати реляційну базу даних із кількома взаємопов'язаними таблицями, використовуючи первинні та зовнішні ключі.

Завдання до лабораторної роботи

Завдання 1. Спроєктувати і створити БД students у середовищі phpMyAdmin.

Опис предметної області: Студенти навчаються в університеті у групах. Кожна група має назву, форму навчання (денна, заочна), рік створення, напрям підготовки (спеціальність) і старосту. Кожний студент має ПІБ, стать, дату народження, номер телефону, адресу електронної пошти, № залікової книжки, групу, де навчається, форма оплати (за контрактом чи на бюджеті). У групі може навчатися багато студентів, але студент може одночасно навчатися тільки в одній групі.

Проєктування БД предметної області.

Таблиця 3.1 – Сутність «Студент»

	student			
#	Назва стовпчика	Тип даних	Обмеження	Опис
1	student_id	INTEGER NOT NULL PRIMARY KEY	Первинний ключ	Ідентифікатор
2	name	VARCHAR		ПІБ
3	sex_id	INTEGER NOT NULL FOREIGN KEY	Зовнішній ключ	Код статі
4	birth_date	DATE		Дата народження
5	phone	VARCHAR		Телефон
6	email	VARCHAR		Електронна пошта
7	recordbook_number	VARCHAR		№ залікової книжки
8	group_id	INTEGER NOT NULL FOREIGN KEY	Зовнішній ключ	Код групи
9	pay_form_id	INTEGER NOT NULL FOREIGN KEY	Зовнішній ключ	Код форми оплати

Таблиця 3.2 – Сутність «Група»

	group			
#	Назва стовпчика	Тип даних	Обмеження	Опис
1	student_group_id	INTEGER PRIMARY KEY	Первинний ключ	Ідентифікатор
2	name	VARCHAR		Назва групи
3	education_form_id	INTEGER NOT NULL FOREIGN KEY	Зовнішній ключ	Форма навчання
4	start_year	YEAR		Рік створення

Таблиця 3.3 – Сутність «Форма навчання»

	education_form			
#	Назва стовпчика	Тип даних	Обмеження	Опис
1	education_form_id	INTEGER PRIMARY KEY	Первинний ключ	Ідентифікатор
2	name	VARCHAR		Назва

Таблиця 3.4 – Сутність «Форма оплати»

	pay_form			
#	Назва стовпчика	Тип даних	Обмеження	Опис
1	pay_form_id	INTEGER PRIMARY KEY	Первинний ключ	Ідентифікатор
2	name	VARCHAR		Назва

Таблиця 3.5 – Сутність «Стать»

	sex			
#	Назва стовпчика	Тип даних	Обмеження	Опис
1	sex_id	INTEGER PRIMARY KEY	Первинний ключ	Ідентифікатор
2	name	VARCHAR		Назва

Таблиця 3.6 – Сутність «Староста»:

	leader			
#	Назва стовпчика	Тип даних	Обмеження	Опис
1	student_group_id	INTEGER NOT NULL UNIQUE KEY FOREIGN KEY	Унікальний ключ Зовнішній ключ	Код групи
5	leader_id	INTEGER NOT NULL UNIQUE KEY FOREIGN KEY	Унікальний ключ Зовнішній ключ	Код студента – старости групи

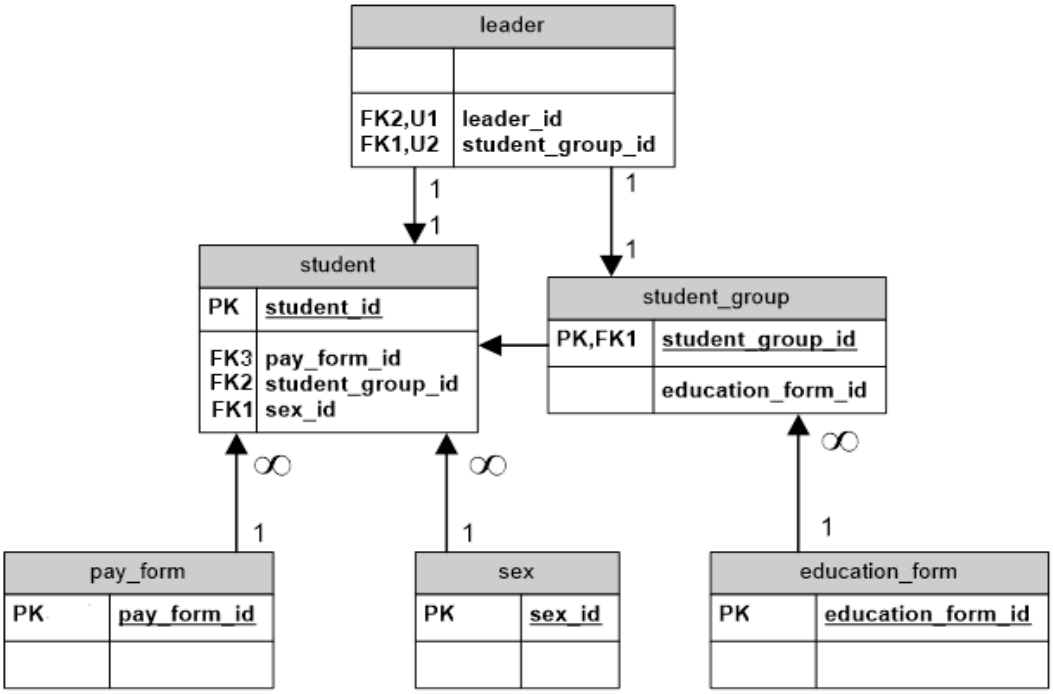

Рисунок 3.1 – Схема зв’язків бази даних

Завдання 2. Розширити БД students.

1. Створити самостійно таблицю бази даних «Староста» (leader) з двома зовнішніми ключами.
2. Спроєктувати БД предметної області у відповідності до індивідуального варіанту у вигляді сутностей. Встановити відповідні зв'язки між ними за допомогою зовнішніх ключів. Створити базу даних з таблицями у відповідності до спроєктованих сутностей.
3. Заповнити таблиці бази даними по 5-10 записів у кожній таблиці.
4. Зберегти створену БД.

Індивідуальні завдання до лабораторної роботи

Виконати *індивідуальне завдання* згідно з варіантом (у відповідності до варіантів, наведених в лабораторній роботі №2):

- визначити необхідну кількість сутностей (таблиць) у БД;
- визначити структуру кожної таблиці БД (поля, їх тип, ключові поля);
- визначити зв'язки, що існують між таблицями;
- заповнити таблиці тестовими даними;
- сформувати запити до даних, що зберігаються у БД та вивести їх результати;
- зберегти створену БД і запити до неї.

Контрольні питання

1. Що таке реляційна база даних і які її основні властивості?
2. Які етапи включає процес створення реляційної бази даних?
3. Що являє собою первинний ключ (PRIMARY KEY) і яку роль він відіграє в таблиці?
4. Що являє собою зовнішній ключ (FOREIGN KEY) та як він забезпечує зв'язки між таблицями?
5. Які типи зв'язків між таблицями існують і де вони застосовуються?
6. Як створити зв'язок між таблицями під час виконання оператора CREATE TABLE?
7. Як працює каскадне оновлення та каскадне видалення?
8. Які переваги має зберігання даних у кількох пов'язаних таблицях порівняно з однією великою таблицею?
9. Яким чином можна виконувати вибірки даних із кількох пов'язаних таблиць за допомогою JOIN?

ЛАБОРАТОРНА РОБОТА №4
Тема: «Робота з даними в мові SQL: вибірка, фільтрація, агрегація та модифікація»

Мета роботи: Навчитись застосовувати оператори INSERT, UPDATE, DELETE та різні форми оператора SELECT для формування запитів і маніпулювання даними у зв'язаних таблицях.

Завдання до лабораторної роботи

1. Відкрити створену на попередній лабораторній роботі базу даних *students* в phpMyAdmin.
2. Виконати до бази даних *students* такі запити:
   - **Select:**
     - вибрати інформацію про студентів і групи, яким вони належать, в одну таблицю;
     - вибрати студентів, які народились у 1994 році;
     - вибрати студентів жіночої статі.
   - **Insert:** додати групу за допомогою запиту.
   - **Update:**
     - перевести студентів, що вчаться за контрактом, на заочну форму навчання;
     - перевести на бюджет студентів, у яких у номері залікової книжки є 0.
   - **Delete:** видалити студентів, які не мають телефонів і електронної пошти, з бази даних.
3. Зберегти запити у файли.

Індивідуальні завдання до лабораторної роботи

1. Відкрити створену на попередній лабораторній роботі базу даних згідно варіанту індивідуального завдання з лабораторної роботи №2.
2. Наповнити таблиці, створеної бази, даними по 10-15 записів.
3. Викликати SQL-редактор phpMyAdmin.
4. Розробити запити на додавання, вилучення, оновлення та вибірку даних згідно з варіантом.
5. Виконати розроблені запити до бази даних.
6. Створити SELECT-запит на вибірку даних мінімум з двох таблиць одночасно на свій розсуд.
7. Зберегти запити у файли.
8. Оформити звіт про виконання завдань лабораторної роботи і надіслати його на перевірку разом з файлами запитів.

Варіанти:

1. *Select*:
   - вибрати інформацію про відділи і його завідувачів в одну таблицю;
   - вибрати працівників, які зайняті на проектах, які проводяться в заданий період часу;

– вибрати співробітників, які зайняті більше ніж на одному проєкті.

Insert: додати відділ за допомогою запиту.

Update:

– для кожного проекту, тривалість якого менша 10 днів, збільшити тривалість до 15 днів;
– провести зміну телефонів у відділів у зв'язку зі зміною АТС.

Delete: видалити проекти, дата початку яких була у січні місяці 2014 року.

2. *Select*:
   – вибрати інформацію про дисципліни і викладачів, які їх читають в одну таблицю;
   – вибрати дисципліни кількість годин в яких більша 108;
   – вибрати дисципліни, для яких форма контролю дорівнює заданій користувачем формі.

   Insert: додати дисципліну за допомогою запиту.

   Update:

   – для кожної дисципліни, кількість годин якої менша 108, збільшити кількість годин до 108;
   – провести зміну виду контролю із „заліку" на „екзамен" для дисциплін, у яких кількість годин не менше 144.

   Delete: видалити дисципліни, кількість годин яких менша 36.
3. Select:
   – вибрати інформацію із накладних про продані товари у задану дату. В таблиці має бути код накладної, одержувач, дата , товар, кількість;
   – вибрати товари із кодом менше 200;
   – вибрати товари для заданого виробника.

   Insert: додати товар за допомогою запиту.

   Update:

   – для кожної накладної, яка була у певний період часу, зменшити вартість товару в них на 15%. (Період задається параметрами запиту);
   – змінити адресу виробника, код якого дорівнює 10.

   Delete: видалити накладні, дата яких була у вересні місяці 2014 року.
4. *Select*:
   – вибрати інформацію про відділи бібліотеки;
   – вибрати для читача відділи в яких він записаний. (Читач задається номером);
   – вибрати для відділу читачів, в якому вони записані. (Відділ задається номером).

   Insert: додати відділ за допомогою запиту.

   Update:

   – для кожної книги змінити;
   – провести зміну телефонів у відділах у зв'язку зі зміною АТС.

   Delete: видалити читача із заданим телефоном.
5. *Select*:

– вибрати інформацію про клієнтів та банк сума на рахунку для яких більша 100 000;
– вибрати інформацію про клієнта по коду;
– вибрати банківські рахунки з заданим видом рахунку.

Insert: додати банк за допомогою запиту.

Update:
– для кожного рахунку, сума якого більша 200 000, збільшити суму на 5%;
– для банку з заданим номером ліцензії змінити назву на задану в параметрі запиту.

Delete: видалити рахунки, сума яких менша 10.

6. *Select*:
– вибрати інформацію про зберігачів акцій заданого акціонерного товариства;
– вибрати інформацію про акціонерні товариства, акції яких є у зберігача;
– вибрати акції номіналом більше 1000 і інформацію про їх зберігачів.

Insert: додати акцію за допомогою запиту.

Update:
– для кожної акції номіналом більше 5000 збільшити номінал на 5%;
– змінити адресу для заданого акціонерного товариства.

Delete: видалити акції, номінал яких менше 10.

7. *Select*:
– вибрати інформацію про контейнери та про їх порт прибуття для заданого рейсу;
– вибрати контейнери заданого вантажоодержувача;
– вибрати рейси, які відправились у заданий період.

Insert: додати рейс за допомогою запиту.

Update:
– для кожного рейсу, тривалість якого більша 90 днів, збільшити тривалість на 5 днів;
– для заданого рейсу змінити порт призначення.

Delete: видалити рейси, дата початку яких була у березні місяці 2012 року.

8. *Select*:
– вибрати інформацію про поставки, які були зроблені у певний період часу;
– вибрати поставки для заданого складу;
– вибрати підприємства і їх продукцію, якщо її кількість продукції менша 5.

Insert: додати склад за допомогою запиту.

Update:
– для кожного підприємства збільшити кількість продукції на 10;
– провести зміну назви для заданого підприємства.

Delete: видалити поставки, які відбувались у 2013 році.

9. *Select*:
– вибрати інформацію про покупки заданого покупця (за номером кредитної картки), які були зроблені ним у певний період часу;

– вибрати товари для заданого магазину (за його номером);
– вибрати магазини, у яких кількість товарів менша 5.

Insert: додати новий магазин за допомогою запиту і 3 товари в ньому.

Update:
– для кожного магазина збільшити кількість товарів на 5;
– провести зміну № розрахункового рахунку для заданого магазина.

Delete: видалити товари, які були продані до зазначеної дати.

10. *Select*:
– вибрати інформацію про сегменти мережі, кількість вузлів, в яких більше 5;
– вибрати інформацію про вузли заданого сегменту;
– вибрати вузли заданого типу.

Insert: додати новий вузол за допомогою запиту.

Update:
– для кожного сегменту збільшити кількість вузлів на 5;
– провести зміну назви для заданого вузла.

Delete: видалити сегменти з нульовою кількістю вузлів.

11. *Select*:
– вибрати інформацію про будинки, на яких працює задана бригада;
– вибрати будинки, у яких дата здачі після заданого числа;
– вибрати бригади, у яких кількість робітників менша 5;

Insert: додати новий будинок за допомогою запиту.

Update:
– у кожній бригаді організації збільшити кількість робітників на 3;
– провести зміну бригадира для заданої бигади.

Delete: видалити будинки, які були здані до жовтня 2014 році.

12. *Select*:
– вибрати інформацію про замовлення, які були зроблені у певний період часу (Дата, Час, ПІБ Менеджера, ПІБ Клієнта);
– вибрати замовлення для заданого клієнта;
– вибрати товари, кількість на складі яких менша 100.

Insert: додати клієнта за допомогою запиту.

Update:
– для кожного товару збільшити його кількість на складі у 2 рази;
– провести зміну ПІБ для заданого клієнта.

Delete: видалити товари, які мають залишок на складі 0.

13. Select:
– вибрати інформацію про поїздки, вартість яких перевищує задану суму;
– вибрати поїздки для заданого клієнта;
– вибрати маршрути, дата завершення яких пізніше заданої дати.

Insert: додати нового клієнта за допомогою запиту.

Update:
– для всіх маршрутів збільшити вартість на 10%;

– провести зміну відповідального менеджера для заданої поїздки.

Delete: Видалити поїздки, дата завершення яких раніше 2015 року.

14. Select:

– вибрати інформацію про прийоми пацієнтів у заданий період часу (дата, час, ПІБ лікаря, ПІБ пацієнта);
– вибрати лікарів заданої спеціалізації;
– вибрати пацієнтів, які відвідували більше ніж одного лікаря.

Insert: додати нового пацієнта за допомогою запиту.

Update:

– для всіх лікарів збільшити стаж роботи на 1 рік;
– провести зміну спеціалізації для заданого лікаря.

Delete: видалити прийоми, дата яких раніше 2016 року.

15. Select:

– вибрати інформацію про замовлення, оформлені у заданий період часу (номер, дата, ПІБ покупця, загальна вартість);
– вибрати замовлення для заданого покупця;
– вибрати товари, ціна яких перевищує задане значення.

Insert: додати новий товар за допомогою запиту.

Update:

– для всіх товарів збільшити ціну на 5%;
– провести зміну адреси доставки для заданого покупця.

Delete: видалити товари, кількість яких дорівнює 0.

Контрольні питання

1. Яке призначення оператора SELECT у мові SQL?
2. Для чого використовується ключове слово WHERE, і які умови можна в ньому задавати?
3. Яка різниця між WHERE та HAVING?
4. Які функції належать до агрегатних (SUM, AVG, COUNT, MIN, MAX) і що вони роблять?
5. У чому полягає призначення оператора GROUP BY, і коли він використовується?
6. Для чого потрібен оператор ORDER BY і які режими сортування він підтримує?
7. Як працюють оператори INSERT, UPDATE та DELETE, і в яких випадках вони застосовуються?
8. Що таке оператори порівняння (=, <>, >, <, BETWEEN, LIKE, IN) та як вони використовуються у фільтрації?
9. Як можна обмежити кількість результатів у вибірці (LIMIT, OFFSET / TOP)?
10. Які правила слід враховувати під час модифікації даних, щоб уникнути небажаних змін у таблиці?

ЛАБОРАТОРНА РОБОТА №5
Тема: «Збережені процедури в мові SQL»

Мета роботи: Навчитися створювати та використовувати збережені процедури для вибірки та модифікації даних у пов'язаних таблицях реляційної бази даних.

Завдання до лабораторної роботи

Студенти навчаються в університеті у групах. Кожна група має: назву, форму навчання (денна, заочна), рік створення, напрям підготовки (спеціальність) і одного старосту. Кожний студент має: ПІБ, стать, дату народження, номер телефону, адресу електронної пошти, № залікової книжки, групу, де навчається, форму оплати (за контрактом чи на бюджеті). У групі може навчатися багато студентів, але студент може одночасно навчатися тільки в одній групі

1. Відкрити створену на попередніх лабораторних роботах базу даних *students* у phpMyAdmin.
2. Створити до бази даних *students* такі збережені процедури:
   2.1. Створити збережену процедуру для пошуку кореня суми двох величин, які задаються у якості параметрів процедури.
   2.2. Створити збережену процедуру для виводу інформації про групи, її назву, рік створення для заданої форми навчання.
   2.3. Створити збережену процедуру для обчислення сумарної кількості студентів, що навчаються у певній групі та знаходження долі цих студентів від загальної кількості.
3. Створити збережену процедуру, яка обраховує кількість студентів, які навчаються на денній формі та їх % від загальної кількості студентів.
4. Зберегти результати у файли.

Індивідуальні завдання до лабораторної роботи

1. Відкрити створену на попередній лабораторній роботі базу даних згідно варіанту індивідуального завдання (теми зазначені в ЛР №2).
2. Створити збережені процедури згідно з варіантом завдання.
3. Перевірити збережені процедури на правильність виконання операцій.
4. Зберегти результати у файли.

Варіанти:

1. Завдання:
   – створити збережену процедуру для пошуку добутку двох величин, які задаються у якості параметрів процедури;
   – створити збережену процедуру для виводу інформації про проєкти та осіб, які на ньому задіяні;
   – створити збережену процедуру для підрахунку кількості годин у проєктах в заданий період.
2. Завдання:

– створити збережену процедуру для пошуку середнього значення двох величин, які задаються у якості параметрів процедури;
– створити збережену процедуру для виводу інформації про дисципліни та викладачів, які їх читають;
– створити збережену процедуру для обчислення загальної кількості годин та відсотку годин по дисципліні від загальної кількості.

3. Завдання:
– створити збережену процедуру для пошуку суми двох величин, які задаються у якості параметрів процедури;
– створити збережену процедуру для виводу інформації про накладні та їх вміст;
– створити збережену процедуру для обчислення загальної вартості товару в накладних та загальної кількості товарів в накладних.

4. Завдання:
– створити збережену процедуру для пошуку суми квадратів двох величин, які задаються у якості параметрів процедури;
– створити збережену процедуру для виводу інформації про відділ, та користувачів, які в ньому записані;
– створити збережену процедуру для обчислення кількості читачів у відділах та знайти їх відсоток від загальної кількості.

5. Завдання:
– створити збережену процедуру для пошуку модулю різниці двох величин, які задаються у якості параметрів процедури;
– створити збережену процедуру для виводу інформації про банк та рахунки, які належать цьому банку;
– створити збережену процедуру для обчислення загальної суми на рахунках по банках та їх відсоток від загальної суми.

6. Завдання:
– створити збережену процедуру для пошуку частки двох величин, які задаються у якості параметрів процедури;
– створити збережену процедуру для виводу інформації про акціонерне товариство та акції які випущенні на нього;
– створити збережену процедуру для обчислення кількості акцій на кожне акціонерне товариство та їх відсоток від загальної кількості.

7. Завдання:
– створити збережену процедуру для пошуку різниці двох величин, які задаються у якості параметрів процедури;
– створити збережену процедуру для виводу інформації про рейс та вантаж який він переводить;
– створити збережену процедуру для обчислення кількості контейнерів у кожному рейсі.

8. Завдання:
– створити збережену процедуру для пошуку суми двох величин, помноженої на значення першої, які задаються у якості параметрів процедури;

– створити збережену процедуру для виводу інформації про підприємства, види його продукції та кількість;
– створити збережену процедуру для обчислення сумарної кількості продукції, що випускається підприємством та знаходження долі продукції, що випускається від загальної кількості.

9. Завдання:
– створити збережену процедуру для пошуку суми квадратів двох величин, які задаються у якості параметрів процедури;
– створити збережену процедуру для виводу інформації про покупки заданого покупця (за номером кредитної картки), які були зроблені ним у певний період часу;
– створити збережену процедуру для обчислення сумарної кількості покупок заданого покупця (за номером кредитної картки) та їх середню вартість.

10. Завдання:
– створити збережену процедуру для пошуку квадрату різниці двох величин, які задаються у якості параметрів процедури;
– створити збережену процедуру для виводу інформації про мережі, їх назву, тип та адресу;
– створити збережену процедуру для обчислення сумарної кількості вузлів, що належать сегменту мережі та знаходження долі вузлів, що займає сегмент мережі від загальної кількості вузлів.

11. Завдання:
– створити збережену процедуру для пошуку кореня з суми квадратів двох величин, які задаються у якості параметрів процедури;
– створити збережену процедуру для виводу інформації про будинки, на яких працює задана бригада;
– створити збережену процедуру для обчислення сумарної кількості працівників у всіх бригадах і середню кількість працівників у бригадах.

12. Завдання:
– створити збережену процедуру для пошуку квадрату суми двох величин, які задаються у якості параметрів процедури;
– створити збережену процедуру для виводу інформації про замовлення, їх дату, назву товару, кількість та вартість;
– створити збережену процедуру для обчислення сумарної кількості замовлень на певний товар та знаходження долі цього товару від загальної кількості товарів у замовленнях.

13. Завдання:
– створити збережену процедуру для пошуку середнього значення двох величин, які задаються у якості параметрів процедури;
– створити збережену процедуру для виводу інформації про клієнтів та туристичні поїздки, які вони замовили;
– створити збережену процедуру для обчислення загальної кількості туристичних поїздок кожного клієнта та їх середньої вартості.

14. Завдання:

– створити збережену процедуру для пошуку модуля різниці двох величин, які задаються у якості параметрів процедури;
– створити збережену процедуру для виводу інформації про відділення медичного центру, лікарів та пацієнтів, яких вони приймають;
– створити збережену процедуру для обчислення загальної кількості прийомів у кожному відділенні та знаходження їх відсотка від загальної кількості прийомів.

15. Завдання:
– створити збережену процедуру для пошуку квадрату суми двох величин, які задаються у якості параметрів процедури;
– створити збережену процедуру для виводу інформації про покупців, їх замовлення та товари, що входять до замовлень;
– створити збережену процедуру для обчислення загальної кількості замовлень кожного покупця та їх частки від загальної кількості замовлень.

Контрольні питання

1. Що таке збережена процедура в SQL і для чого вона використовується?
2. У чому відмінність між збереженою процедурою та звичайним SQL-запитом?
3. Які переваги використання збережених процедур у роботі з базою даних?
4. Які типи параметрів можуть мати збережені процедури (IN, OUT, INOUT) і як вони працюють?
5. Як створюється збережена процедура за допомогою оператора CREATE PROCEDURE?
6. Як виконується виклик збереженої процедури у SQL?
7. Чим відрізняється збережена процедура від функції в SQL?
8. Які обмеження або особливості функцій у SQL (наприклад, повернення значення, використання в запитах)?
9. Як можна використовувати збережені процедури та функції для роботи з даними (INSERT, UPDATE, DELETE, SELECT)?
10. Як обробляються помилки всередині збережених процедур і функцій?
11. Як створюється користувацька функція за допомогою оператора CREATE FUNCTION?

ЛАБОРАТОРНА РОБОТА №6
Тема: «Тригери в мові SQL»

Мета роботи: Навчитись створювати та використовувати тригери в SQL, розуміти принципи їх роботи, застосовувати їх для автоматизації обробки даних, контролю цілісності та реалізації логіки на рівні бази даних.

Завдання до лабораторної роботи

Студенти навчаються в університеті у групах. Кожна група має назву, форму навчання (денна, заочна), рік створення, напрям підготовки (спеціальність) і одного старосту. Кожний студент має ПІБ, стать, дату народження, номер телефону, адресу електронної пошти, № залікової книжки, групу, де навчається, форма оплати (за контрактом чи на бюджеті). У групі може навчатися багато студентів, але студент може одночасно навчатися тільки в одній групі.

1. Відкрити створену на попередніх лабораторних роботах базу даних *students* у phpMyAdmin.
2. Створити до бази даних *students* тригери.

Завдання 1. Створити тригер для ведення логу (протоколу) видалених записів таблиці «студент».

Завдання 2. Створити тригер, який видаляє всі записи про студентів групи у разі її видалення з бази.

Завдання 3. Створити тригер, який при додаванні нового студента перевіряє коректність введеного номера телефону – кількість цифр номеру повинна бути не більша 10, а також автоматично формує номер залікової книжки (якщо не введений) у форматі: «Шифр групи студента до дефісу» + «останні дві цифри року народження студента» + «код статі» + «код форми оплати навчання».

При створенні тригера:

- використати оператор вибору IF..END IF,
- ввести дві допоміжні змінні @grp – для шифру групи студента до дефісу, @bdy – для отримання останніх двох цифр року народження студента,
- використати функції LEFT, INSTR, RIGHT, YEAR, CONCAT.

Завдання 4. Доповнити тригер on_student_insert так, щоб за умови, коли поле email порожнє (має значення NULL), а поле phone не порожнє, то автоматично формується значення поля email в такому форматі: «Номер телефону студента» + «@gmail.com».

Завдання 5. Створити тригер on_student_update, який при редагуванні даних про студента за умови, коли поле email порожнє (має значення NULL), а поле phone не порожнє, автоматично формується значення поля email в такому форматі: «Номер телефону студента» + «@gmail.com».

3. Зберегти результати у файли.
4. Підготувати звіт про виконання індивідуального завдання.

Індивідуальні завдання до лабораторної роботи

1. Відкрити створену на попередній лабораторній роботі базу даних у phpMyAdmin згідно варіанту індивідуального завдання.

2. Створити тригери згідно з варіантом завдання.
3. Перевірити тригери на правильність виконання операцій.
4. Зберегти результати у файли.

Варіанти:

1. Завдання:
   - створити тригер ведення логу видалених записів про прив’язку людей до проєкту;
   - створити тригер, який видаляє всі записи про прив’язку людей до проєкту у разі видалення проєкту з бази;
   - створити тригер для перевірки коректності введення даних нового проєкту.
2. Завдання:
   - створити тригер ведення логу видалених записів про прив’язку викладачів до дисципліни;
   - створити тригер, який видаляє всі записи про прив’язку викладачів до дисципліни у разі видалення дисципліни з бази;
   - створити тригер для перевірки коректності введення даних нової дисципліни.
3. Завдання:
   - створити тригер ведення логу видалених записів про накладну;
   - створити тригер, який видаляє всі записи про накладну у разі видалення накладної з бази;
   - створити тригер для перевірки коректності введення даних нового клієнта.
4. Завдання:
   - створити тригер ведення логу видалених записів про читача;
   - створити тригер, який видаляє всі записи про читача у разі його видалення;
   - створити тригер для перевірки коректності введення даних нового читача.
5. Завдання:
   - створити тригер ведення логу видалених записів про рахунки;
   - створити тригер, який видаляє всі записи про рахунки у разі видалення банку з бази;
   - створити тригер для перевірки коректності введення даних нового банку.
6. Завдання:
   - створити тригер ведення логу видалених записів про акції;
   - створити тригер, який видаляє всі записи про акції, які належать акціонерному товариству, яке видаляється;
   - створити тригер для перевірки коректності введення даних нової акції.
7. Завдання:
   - створити тригер ведення логу видалених записів про рейс;
   - створити тригер, який видаляє всі записи про рейс у разі видалення рейсу з бази;
   - створити тригер для перевірки коректності введення даних нового рейсу.
8. Завдання:
   - створити тригер ведення логу видалених записів про продукцію;

- створити тригер, який видаляє всі записи про продукцію та кількість у разі видалення підприємства з бази;
- створити тригер для перевірки коректності введення даних нової поставки.

9. Завдання:
   - створити тригер ведення логу видалених записів про покупки;
   - створити тригер, який видаляє всі записи про покупки у разі видалення покупця, що їх здійснював, з бази;
   - створити тригер для перевірки коректності введення даних нової покупки.
10. Завдання:
   - створити тригер ведення логу видалених вузлів;
   - створити тригер, який видаляє всі вузли у разі видалення сегменту з бази;
   - створити тригер для перевірки коректності введення даних нового сегмента.
11. Завдання:
   - створити тригер ведення логу видалених записів про будівлі;
   - створити тригер, який видаляє всі записи про будівлі у разі видалення бригади, яка їх будувала;
   - створити тригер для перевірки коректності введення даних нової бригади.
12. Завдання:
   - створити тригер ведення логу видалених записів про замовлення;
   - створити тригер, який видаляє всі записи про замовлення на товар у разі видалення його з бази;
   - створити тригер для перевірки коректності введення даних нового замовлення.
13. Завдання:
   - створити тригер ведення логу видалених записів про туристичні поїздки;
   - створити тригер, який видаляє всі записи про туристичні поїздки клієнта у разі видалення клієнта з бази;
   - створити тригер для перевірки коректності введення даних нової туристичної поїздки.
14. Завдання:
   - створити тригер ведення логу видалених записів про прийоми пацієнтів;
   - створити тригер, який видаляє всі записи про прийоми у разі видалення лікаря з бази;
   - створити тригер для перевірки коректності введення даних нового прийому пацієнта.
15. Завдання:
   - створити тригер ведення логу видалених записів про замовлення;
   - створити тригер, який видаляє всі записи про замовлення покупця у разі його видалення з бази;
   - створити тригер для перевірки коректності введення даних нового онлайн-замовлення.

Контрольні питання

1. Що таке тригер у SQL та для чого він використовується?
2. Які типи тригерів існують (BEFORE, AFTER, INSTEAD OF) і в чому їх відмінності?
3. У яких подіях можуть спрацьовувати тригери (INSERT, UPDATE, DELETE)?
4. Чим відрізняється рядковий (FOR EACH ROW) тригер від операторного (FOR EACH STATEMENT)?
5. Що таке псевдотаблиці NEW та OLD, і як вони використовуються в тригерах?
6. Які обмеження та потенційні недоліки використання тригерів?
7. У яких випадках використання тригерів є доцільним, а в яких – небажаним?
8. Як тригери можуть забезпечувати цілісність даних у базі?
9. Чи можуть тригери викликати інші тригери? Які наслідки це може мати?
10. Які відмінності в реалізації тригерів у різних СУБД (MySQL, PostgreSQL, SQL Server)?

ЛАБОРАТОРНА РОБОТА №7
Тема: «ER-діаграми та нормальні форми в реляційній моделі даних (1НФ, 2НФ, 3НФ)»

Мета роботи: Ознайомитися з принципами побудови ER-діаграм та процесом нормалізації бази даних. Навчитись визначати функціональні залежності та приводити відношення до 1НФ, 2НФ і 3НФ.

Завдання до лабораторної роботи

Завдання 1. Нехай задана таблиця:

ідс	Співробітник	Номер телефону
1	Іваненко І. І.	283-56-82 390-57-34
2	Петренко П. П.	708-62-34

Проаналізувати чи знаходиться ця таблиця у 1НФ чи ні. Якщо ні, то привести задану таблицю шляхом декомпозиції до першої нормальної форми.

Завдання 2. Нехай задана початкова ненормалізована таблиця:

ідм	Менеджер	Підлеглі
1	Іваненко І. І.	Клименко О.П. Марченко В.В.
2	Петренко П. П.	Андрієнко В.О. Сидоренко Р.Б.

Проаналізувати чи знаходиться ця таблиця у 1НФ чи ні. Якщо ні, то привести задану таблицю шляхом декомпозиції до першої нормальної форми.

Завдання 3. Пропонується фрагмент відношення *Нерухомість:*

property_id	street	district	city	postal_code	property_type	rooms	rent	owner_id	staff_id	branch_id
H1111	вул. Академіка Остроградського, 76	Жовтневий	м. Полтава	36001	Квартира	2	18000.00	B33333	P22222	F22222
H2222	вул. Лесі Українки, 23	Залізничний	м. Хорол	37803	Квартира	3	15000.00	B22222	P11111	F33333
H3333	вул. Івана Мазепи, 45	Північний	м. Кременчук	39602	Будинок	4	22000.00	B44444	P66666	F11111
H4444	вул. Героїв України, 39	Західний	м. Полтава	36008	Будинок	5	26000.00	B66666	P44444	F44444
H5555	вул. Європейська, 56	Східний	м. Лубни	37500	Квартира	3	14000.00	B11111	P33333	F55555
H6666	вул. Празька, 21	Південний	м. Пирятин	37000	Будинок	6	24000.00	B77777	P77777	F33333
H7777	вул. Степана Бандери, 93	Жовтневий	м. Лохвиця	37200	Квартира	4	16000.00	B66666	P11111	F33333
H8888	вул. Авіаторська, 56	Загородній	м. Полтава	36245	Будинок	5	27000.00	B55555	P33333	F11111

1. Виконати аналіз заданого відношення на відповідність до вимог нормалізації (1НФ, 2НФ, 3НФ) та з’ясувати яким нормальним формам воно не відповідає.
2. Виконати декомпозицію відношення та запропонувати структури нових відношень.
3. Вказати зв’язки між новими відношеннями шляхом утворення первинних та зовнішніх ключів.

Завдання 4. Провести аналіз нормальних форм (1НФ, 2НФ, 3НФ) бази даних, розробленої на основі індивідуальних завдань попередніх лабораторних робіт. Створити ER-діаграму за допомогою інструменту «Designer» у phpMyAdmin для цієї бази даних до та після проведення процедури нормалізації.

Контрольні питання

1. Для чого виконується нормалізація відношень у реляційних базах даних?
2. Що таке ER-діаграма і яку роль вона відіграє при проєктуванні бази даних?
3. Які основні складові ER-діаграми?
4. Які типи зв’язків між сутностями існують і як вони позначаються на ER-діаграмі?
5. Які залежності між атрибутами називають транзитивними та функціонально повними?
6. Що таке перша нормальна форма (1НФ) і які вимоги до відношення для її виконання?
7. Що таке друга нормальна форма (2НФ) і які залежності між атрибутами враховуються при її досягненні?
8. Що таке третя нормальна форма (3НФ) і як транзитивні залежності впливають на її формування?
9. Приведіть приклади відношень у 1НФ, 2НФ та 3НФ.
10. Яка послідовність етапів нормалізації від не нормалізованої таблиці до 3НФ?
11. Як використання ER-діаграм допомагає уникнути аномалій вставки, оновлення та видалення даних?

ЛАБОРАТОРНА РОБОТА №8
Тема: «Адміністрування в SQL: резервне копіювання, відновлення та права доступу»

Мета роботи: ознайомитися з методами створення резервних копій бази даних та їх відновлення; навчитися створювати користувачів у phpMyAdmin і налаштовувати їхні права доступу; сформувати та закріпити практичні навички адміністрування реляційної бази даних.

Завдання до лабораторної роботи

1. Створити та заповнити тестову базу даних.

```
-- створення бази
CREATE DATABASE university;
USE university;

-- таблиця студентів
CREATE TABLE student (
   student_id INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(100) NOT NULL,
   birth_date DATE,
   phone VARCHAR(20),
   email VARCHAR(100)
);

-- таблиця викладачів
CREATE TABLE teacher (
   teacher_id INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(100) NOT NULL,
   degree VARCHAR(50),
   position VARCHAR(50)
);

-- таблиця дисциплін
CREATE TABLE discipline (
   discipline_id INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(100) NOT NULL,
   hours INT,
   semester INT,
   control_form VARCHAR(50)
);
-- зв’язувальна таблиця викладач ↔ дисципліна (багато-до-багатьох)
CREATE TABLE teacher_discipline (
   teacher_id INT,
   discipline_id INT,
   PRIMARY KEY (teacher_id, discipline_id),
   FOREIGN KEY (teacher_id) REFERENCES teacher(teacher_id),
   FOREIGN KEY (discipline_id) REFERENCES discipline(discipline_id)
);
```

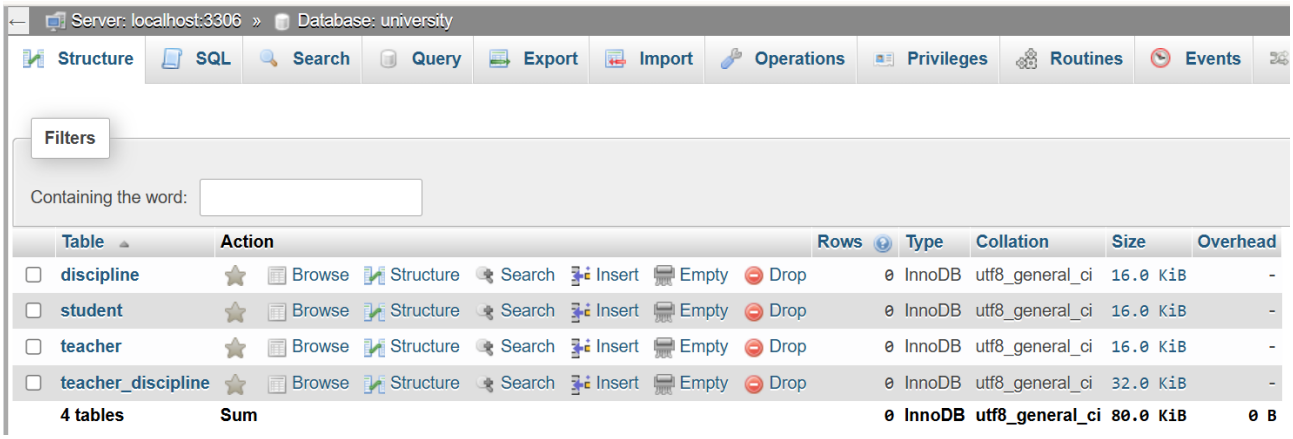

Рисунок 8.1 – Результат створення бази даних «university»

2. Провести заповнення бази даних тестовими даними.

```
-- Студенти
INSERT INTO student (name, birth_date, phone, email) VALUES
('Іваненко Іван', '2003-05-12', '+380931112233', 'ivanenko@example.com'),
('Петренко Марія', '2002-11-03', '+380671234567', 'petrenko@example.com'),
('Коваль Андрій', '2004-02-25', '+380503334455', 'koval@example.com');

-- Викладачі
INSERT INTO teacher (name, degree, position) VALUES
('Дмитренко Олег', 'кандидат наук', 'доцент'),
('Савчук Оксана', 'доктор наук', 'професор'),
('Мельник Ірина', 'кандидат наук', 'старший викладач');

-- Дисципліни
INSERT INTO discipline (name, hours, semester, control_form) VALUES
('Бази даних', 90, 3, 'іспит'),
('Алгоритми та структури даних', 120, 2, 'залік'),
('Програмування на Python', 100, 1, 'іспит');

-- Зв’язки між викладачами та дисциплінами
INSERT INTO teacher_discipline (teacher_id, discipline_id) VALUES
(1, 1), -- Дмитренко читає Бази даних
(2, 1), -- Савчук теж читає Бази даних
(2, 2), -- Савчук читає Алгоритми
(3, 3); -- Мельник читає Python
```

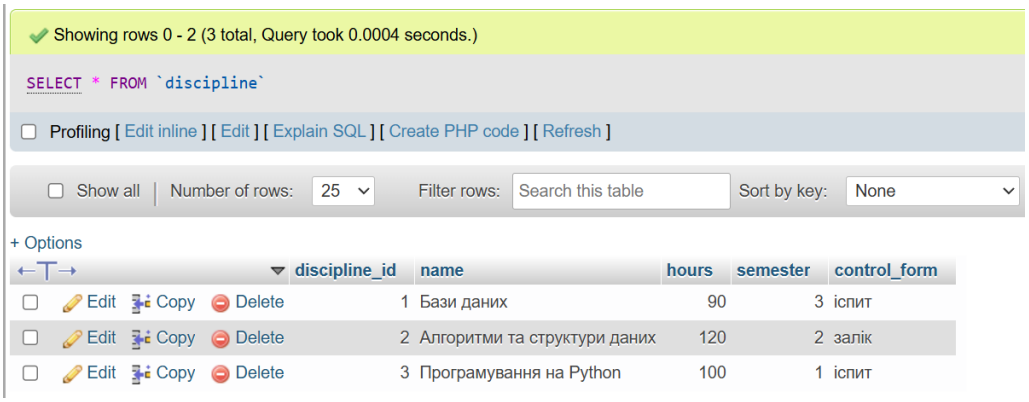

Рисунок 8.2 – Результат створення таблиці «discipline»

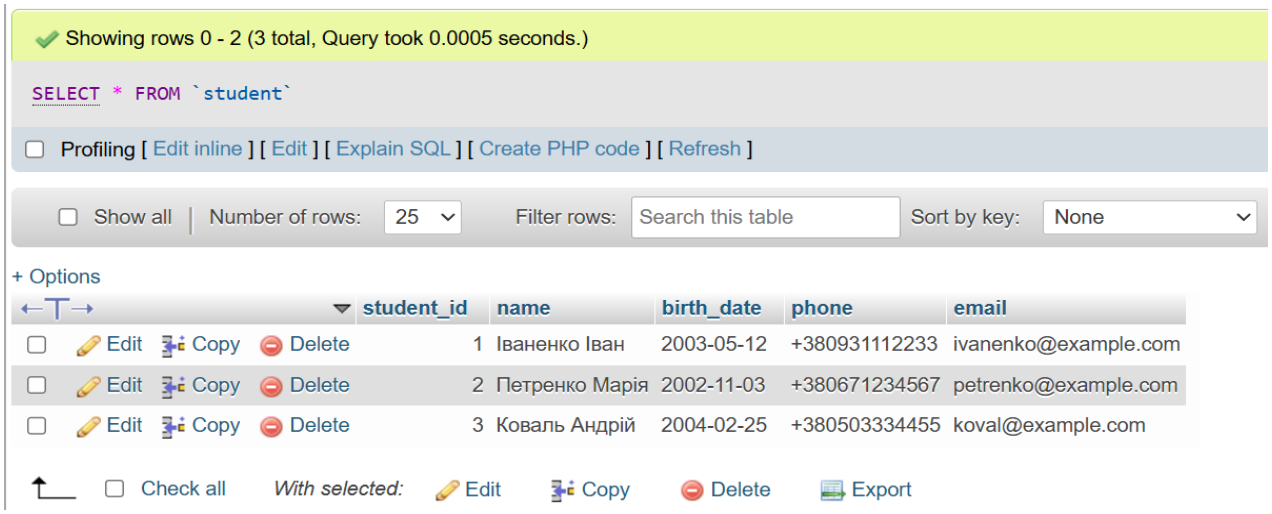

Рисунок 8.3 – Результат створення таблиці «student»

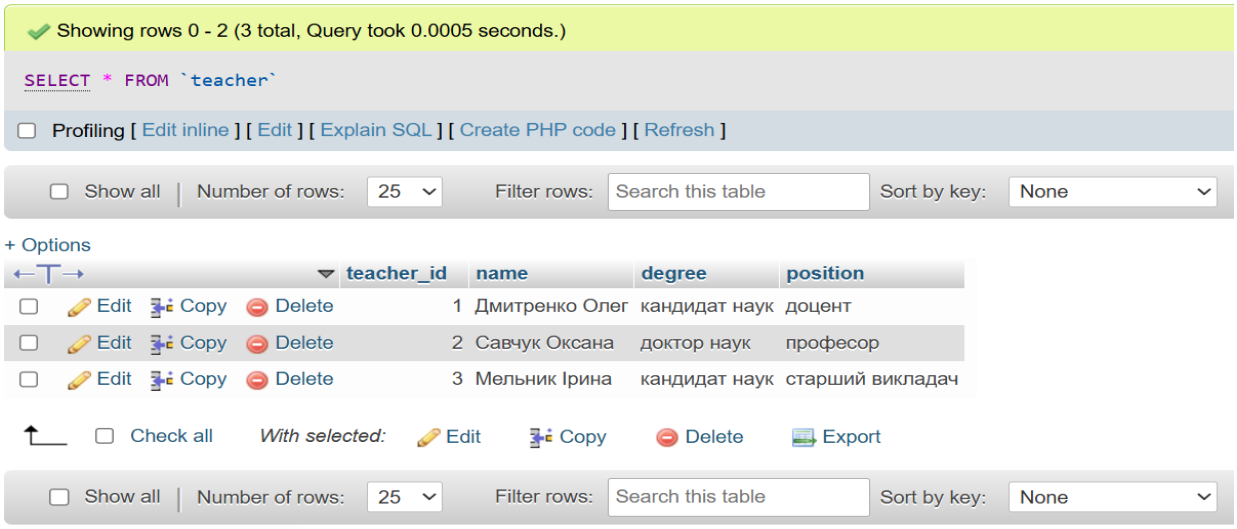

Рисунок 8.4 – Результат створення таблиці «teacher»

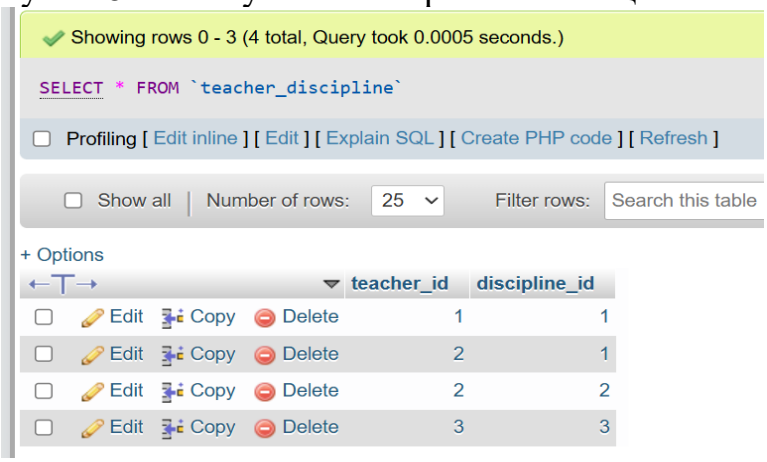

Рисунок 8.5 – Результат створення таблиці «teacher_discipline»

3. Провести резервне копіювання (експорт бази даних).

Після роботи з базою даних її доцільно зберегти. Для цього у головному меню робочого вікна потрібно зайти у вкладку «**Експорт**».

У вкладці «**Експорт**» потрібно обрати метод експорту, формат файлу і натиснути кнопку «**Виконати**» (рис. 8.6).

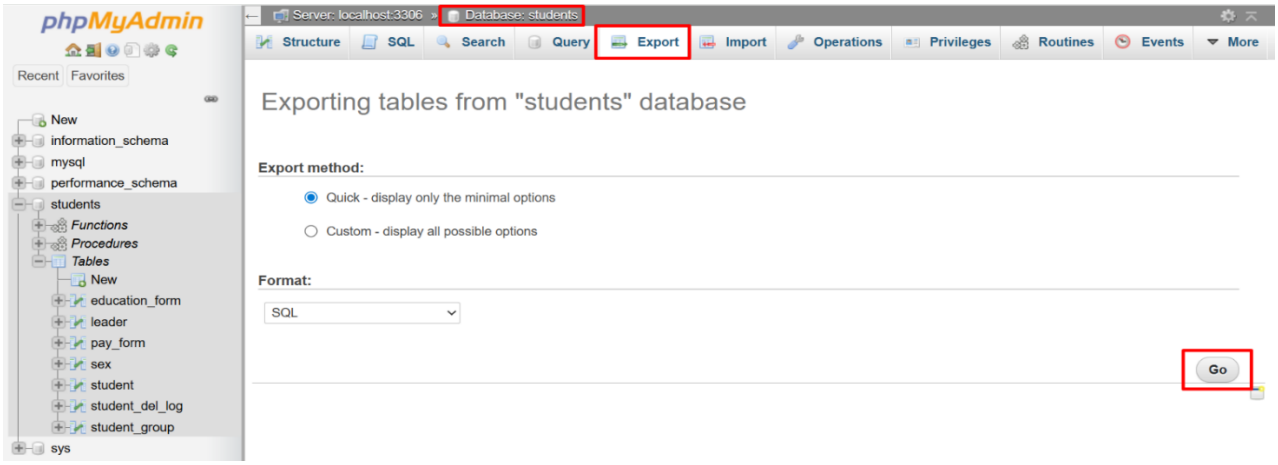

Рисунок 8.6 – Вкладка експорту бази даних в phpMyAdmin

Після цього потрібно обрати папку, де буде збережено файл з базою даних, і натиснути кнопу «**Зберегти**» (рис. 8.7).

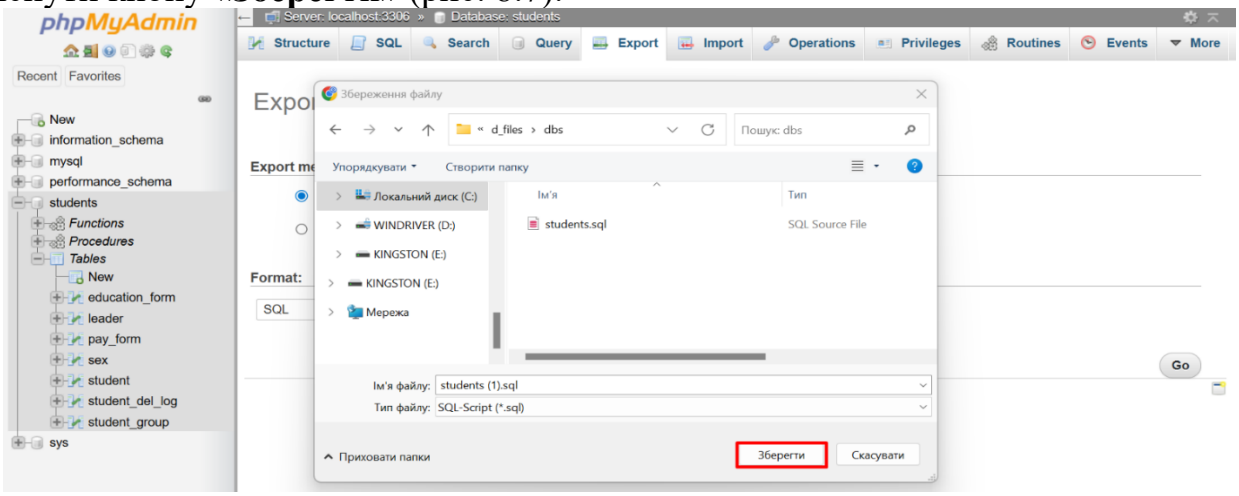

Рисунок 8.7 – Збереження файлу з базою даних

Також можна скористатись відповідною командою, якщо працюєте через командний рядок MySQL:

```
mysqldump -u root -p university > university_backup.sql
```

4. Провести імпорт бази даних.

У вкладці «**Databases**» у розділі «**Create database**» у полі «**Database name**» вказати назву бази даних, що буде завантажуватися, наприклад **students**, і натиснути на кнопку **«Create»** (рис. 8.8)**.**

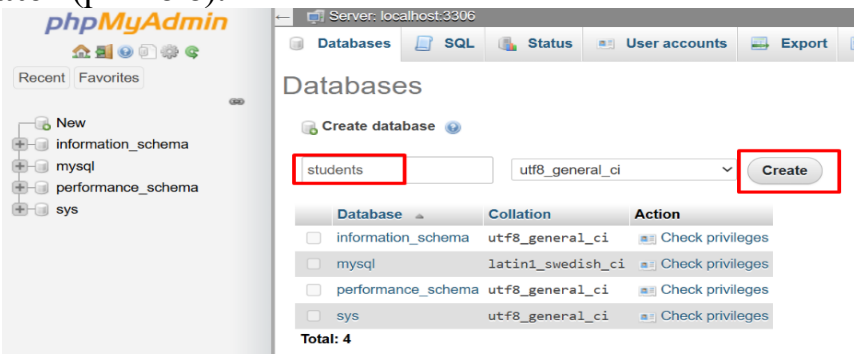

Рисунок 8.8 – Створення бази даних в phpMyAdmin

Після цього потрібно зайти у вкладку «**Import**» і розділі «**File to import**» натиснути кнопку «**Вибрати файл**» (рис. 8.9).

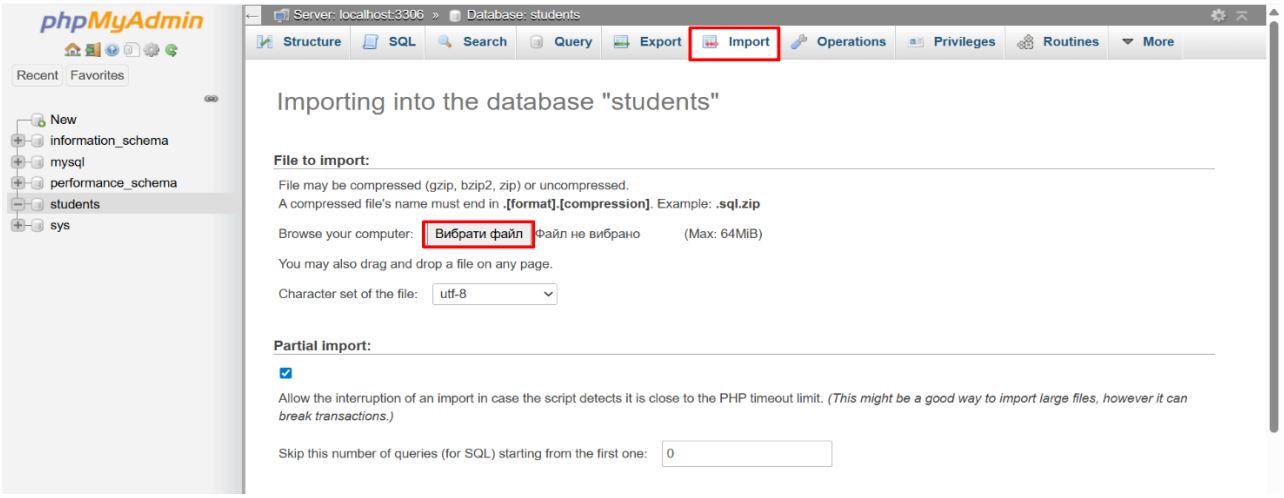

Рисунок 8.9 – Вкладка імпорту бази даних в phpMyAdmin

Далі у потрібній папці обрати необхідний файл з назвою, наприклад, **students.sql** і натиснути кнопку **«Відкрити»** (рис. 8.10).

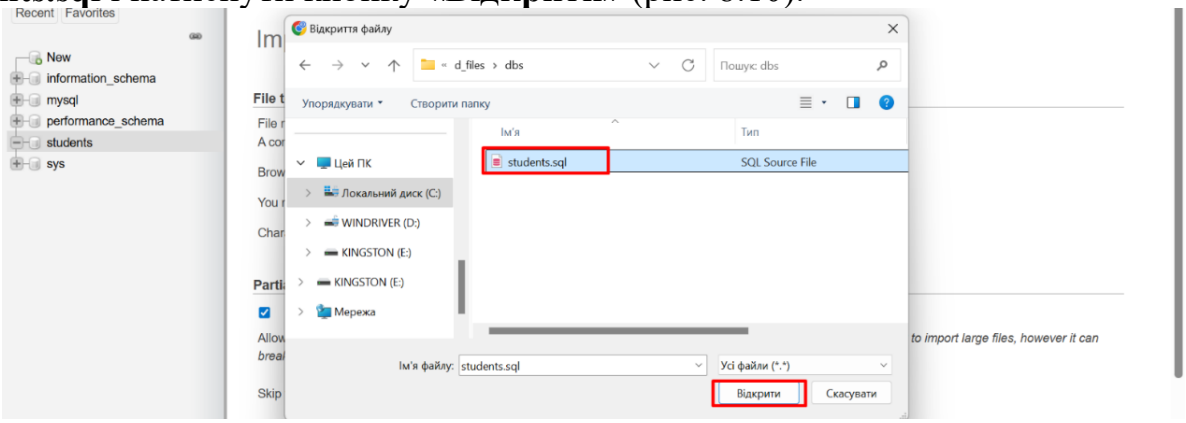

Рисунок 8.10 – Вибір файлу бази даних для імпортування

Після цього потрібно натиснути кнопку «**Go**», що знаходиться у правому нижньому куті робочого вікна (рис. 8.11).

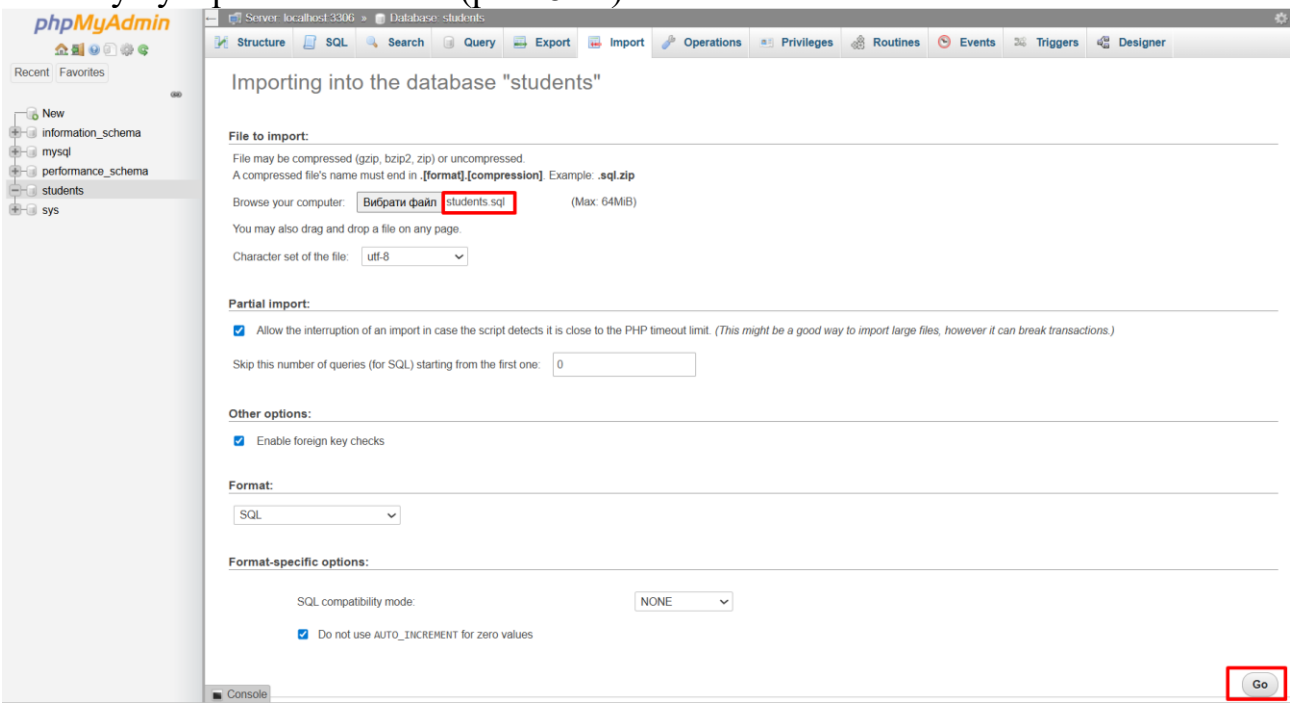

Рисунок 8.11 – Кнопка «Go» для початку процесу імпорту файлу бази даних

Якщо все виконано правильно, то у робочому вікні буде виведено результати імпортування обраної бази даних і в навігаційному меню відображено її структуру (рис. 8.12).

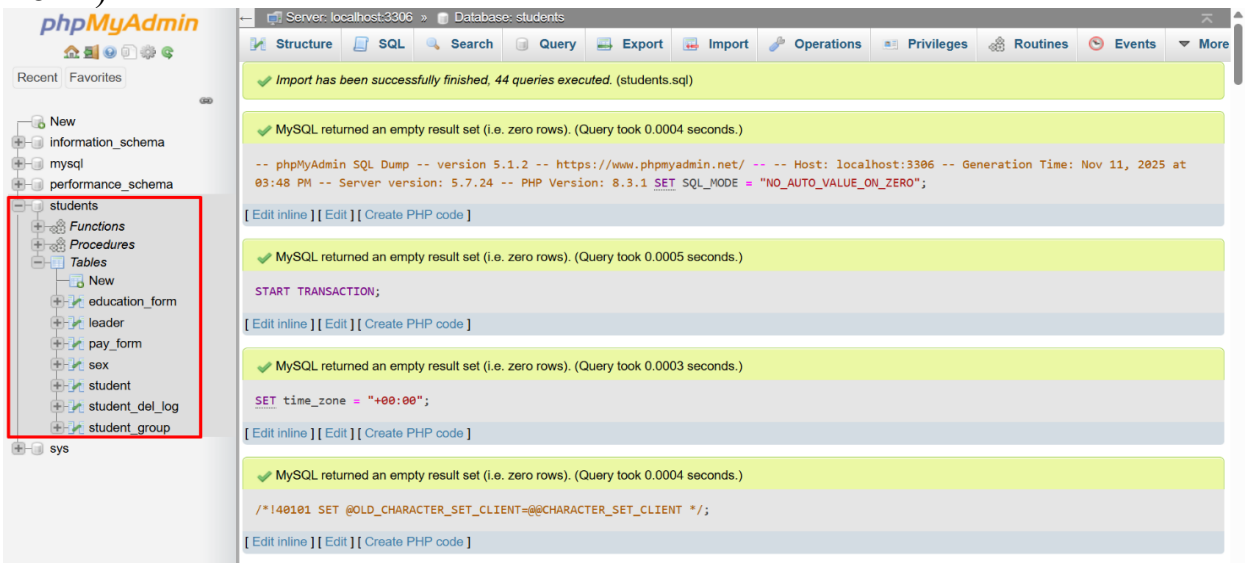

Рисунок 8.12 – Результат успішного імпорту файлу бази даних

Далі необхідно зайти у вкладку «**Structure**» (рис. 8.13), обрати яку-небуть таблицю і переконатися у наявності даних у цій таблиці через вкладку «**Browse**» (рис. 8.14).

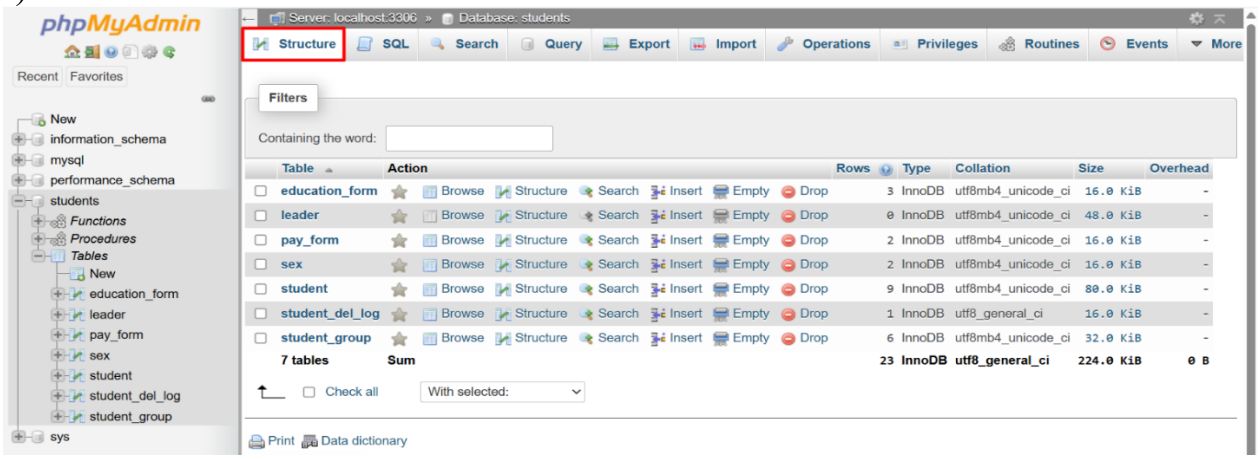

Рисунок 8.13 – Відображення імпортованої структури бази даних

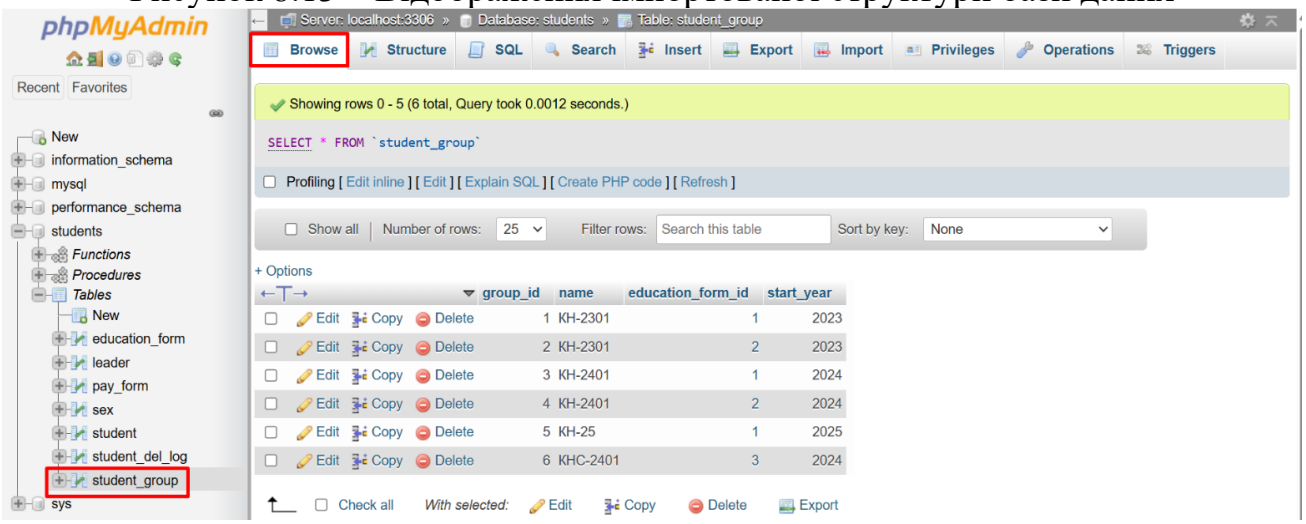

Рисунок 8.14 – Відображення імпортованих записів таблиці «student_group»

Ще одним методом імпорту бази даних є формування відповідного SQL-запиту, приклад чого продемонстровано на рисунку 8.15.

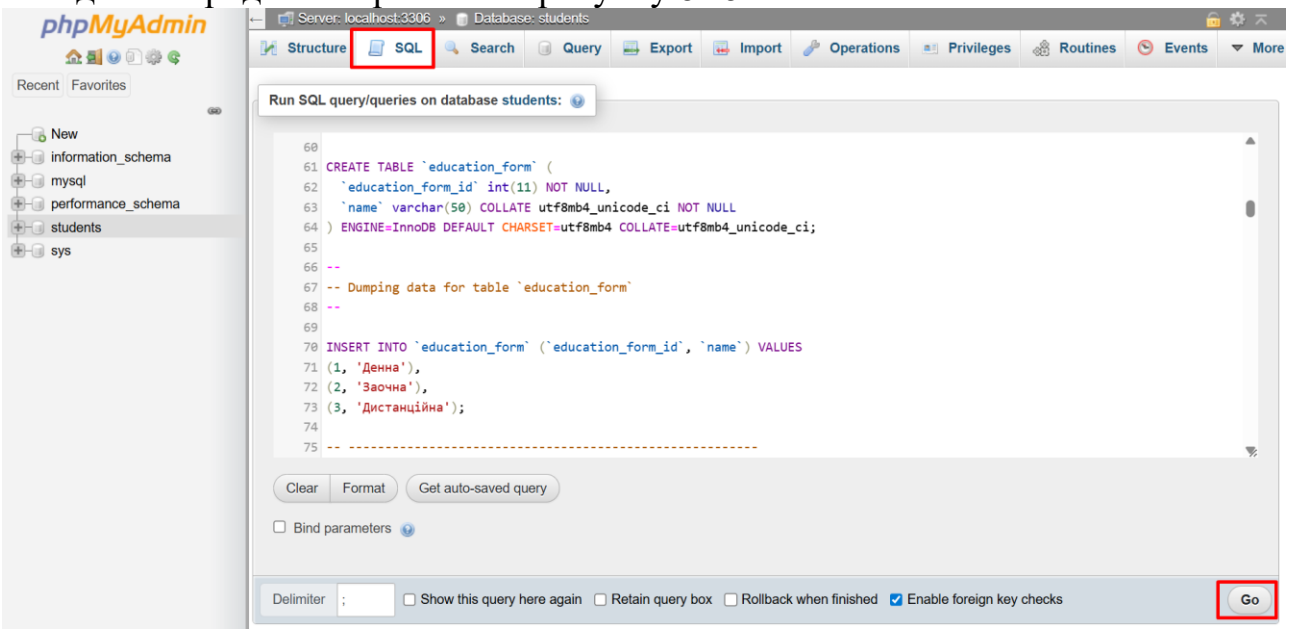

Рисунок 8.15 – Приклад імпорту бази даних за допомогою SQL-запиту

5. Додати користувачів бази даних та призначити їм права доступу до БД.

Щоб переглянути усіх користувачів поточного підключення до СКБД, необхідно виконати наступні дії:

1. Увійти до **phpMyAdmin** під користувачем **root** (або іншим адміністратором).
2. У верхньому меню вибрати **«User accounts»**, в якому буде відображено список усіх користувачів MySQL.

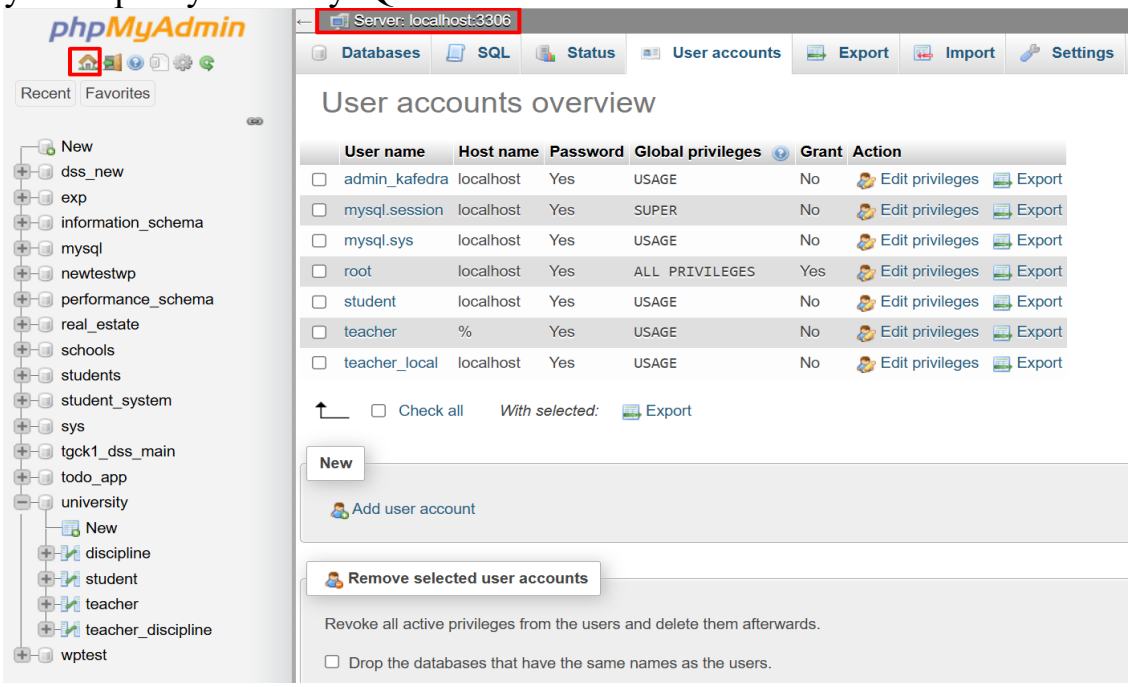

Рисунок 8.16 – Вкладка «User accounts» на домашній сторінці phpMyAdmin

Для того щоб відобразити користувачів та їх права в конкретній базі даних, необхідно вибрати її зі списку зліва та перейти у вкладку «Privileges».

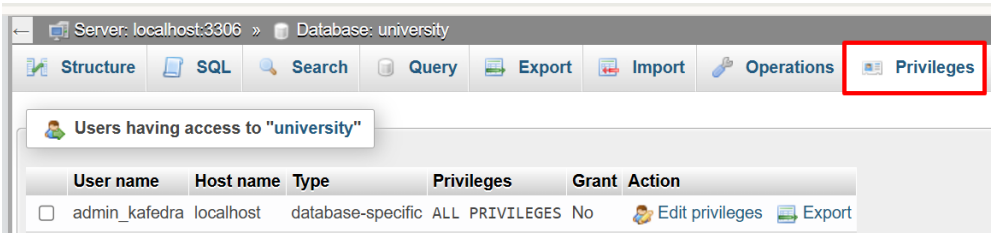

Рисунок 8.17 – Вкладка «Privileges»

Для того щоб створити *нового користувача* через графічний інтерфейс необхідно виконати наступні кроки:

1. Унизу сторінки «Privileges» натиснути **«Add user account»** (рис. 8.17).
2. Введіть дані для користувача (рис. 8.18).
3. Натисніть «Go» (Виконати) (рис. 8.19).

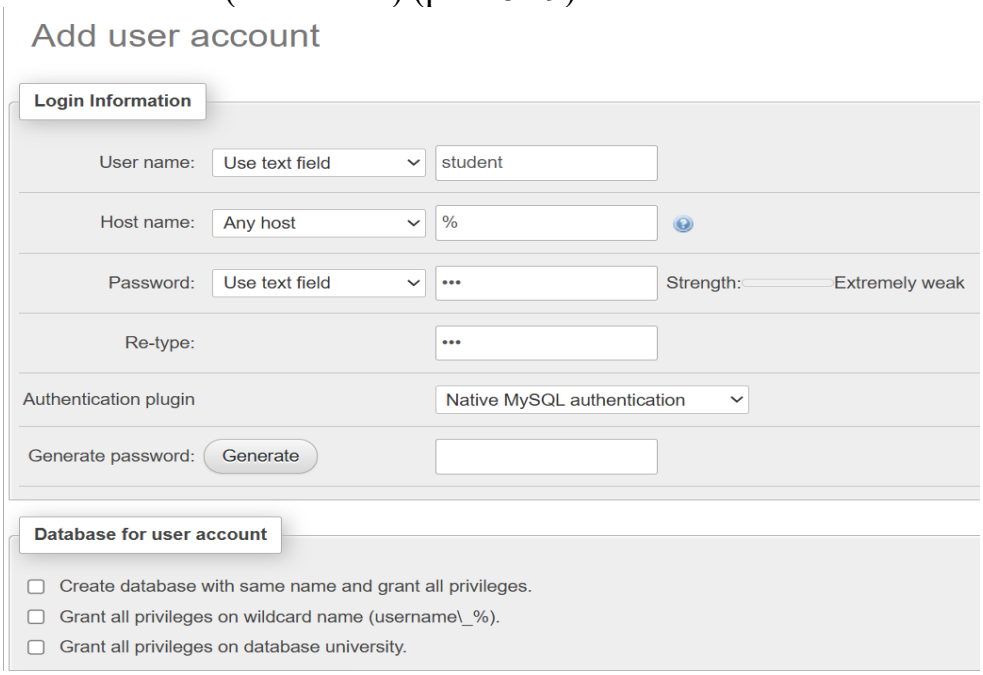

Рисунок 8.18 – Введення даних нового користувача

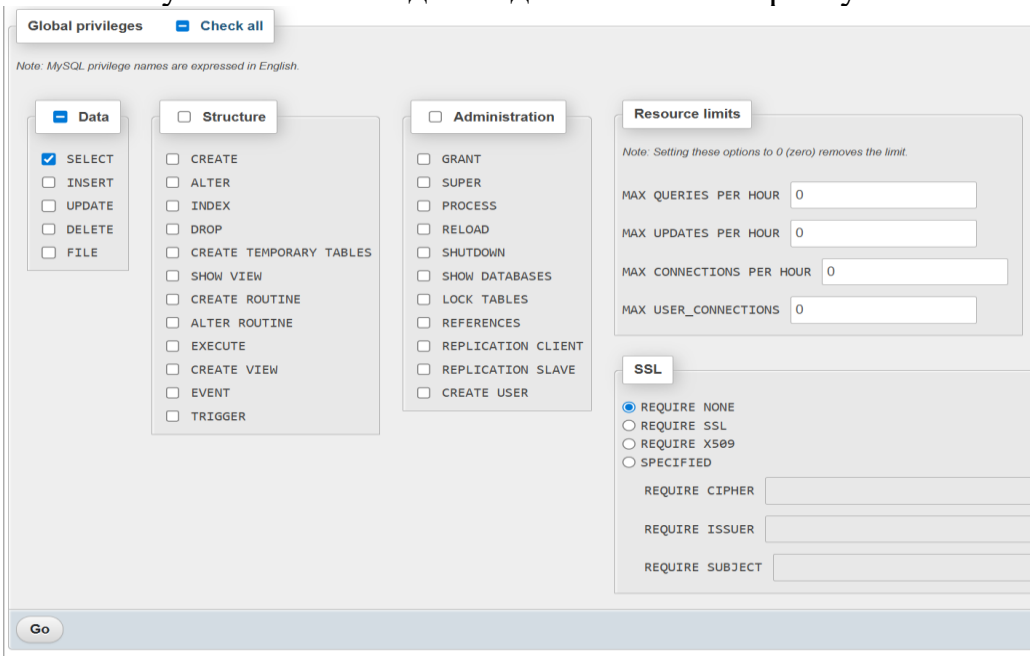

Рисунок 8.19 – Введення прав нового користувача

Результат створення нового користувача із зазначеними правами наведено на рисунку 8.20.

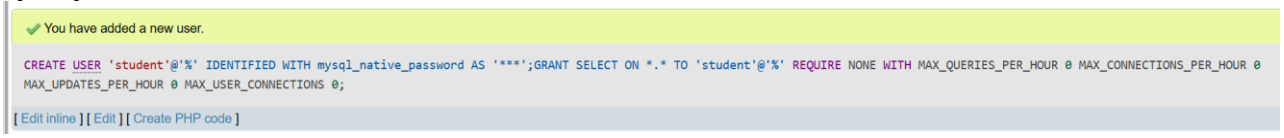

Рисунок 8.20 – Результат успішного створення нового користувача

Для того щоб створити *нового користувача* за допомогою SQL-запиту необхідно (рис. 8.21):

1. Відкрити вкладку «SQL» у phpMyAdmin.
2. Ввести команди:
   CREATE USER 'student'@'localhost' IDENTIFIED BY 'stud123';
   CREATE USER 'teacher'@'%' IDENTIFIED BY 'teach123';
   CREATE USER 'admin_kafedra'@'localhost' IDENTIFIED BY 'admin123';
3. Натиснути Go (Виконати).

Для того щоб *призначити права (GRANT)* за допомогою SQL-запиту необхідно відкрити вкладку SQL, ввести наступні команди та натиснути Go:

```
-- студент: тільки перегляд
GRANT SELECT ON university.* TO 'student'@'localhost';

-- викладач: може додавати/редагувати дисципліни
GRANT SELECT, INSERT, UPDATE ON university.discipline TO 'teacher'@'%';
GRANT SELECT, INSERT, UPDATE ON university.teacher_discipline TO 'teacher'@'%';

-- адміністратор кафедри: повний доступ до всієї бази
GRANT ALL PRIVILEGES ON university.* TO 'admin_kafedra'@'localhost';
```

Для того щоб *перевірити надані права* необхідно виконати наступні SQL-запити, phpMyAdmin покаже список дозволів для кожного користувача:

```
SHOW GRANTS FOR 'student'@'localhost';
SHOW GRANTS FOR 'teacher'@'%';
SHOW GRANTS FOR 'admin_kafedra'@'localhost';
```

SQL-запит для *відміни прав (REVOKE)*, тоді викладач не зможе редагувати записи в таблиці discipline:

```
REVOKE UPDATE ON university.discipline FROM 'teacher'@'%';
```

Після змін необхідно обов’язково *оновити права*:

```
FLUSH PRIVILEGES;
```

Приклади перевірки спрацювання прав доступу. Студент може виконати цей запит:

```
SELECT * FROM student;
```

Але при спробі виконати:

```
INSERT INTO student (name, birth_date) VALUES ('Новий Студент', '2005-01-01');
```

Отримає помилку **ERROR 1142 (42000): INSERT command denied**.

Викладач може виконати цей запит:

```
INSERT INTO discipline (name, hours, semester, control_form) VALUES ('Математичний аналіз', 120, 1, 'іспит');
```

Але при спробі виконати:

```
DELETE FROM student WHERE student_id = 1;
```

Отримає помилку **ERROR 1142 (42000): DELETE command denied**.

Індивідуальні завдання до лабораторної роботи

1. Провести підготовку середовища:

- відкрити наявну базу даних, що використовується у вашому індивідуальному варіанті з попередніх лабораторних робіт;
- перевірити наявність усіх таблиць, зв'язків, процедур і тригерів;
- за потреби виконати початкове наповнення таблиць даними.

2. Зробити резервне копіювання бази даних:

- виконати резервне копіювання вашої бази даних одним із методів:
  - через mysqldump у терміналі (для тих, хто працює через CLI);
  - або через вкладку «Експорт» у phpMyAdmin.
- зберегти отриманий дамп у форматі .sql або .sql.gz.
- перевірити, що у файлі присутні команди CREATE TABLE, INSERT INTO, а також (за наявності) CREATE PROCEDURE, TRIGGER тощо;
- зробити частковий бекап (наприклад, лише кількох таблиць або певної групи даних).

3. Зробити відновлення бази даних:

- створити нову тестову базу (наприклад, db_test або variant_test);
- виконати відновлення даних з резервної копії у нову базу. Якщо використовувався стиснутий дамп (.gz), попередньо розпакувати або використати команду gunzip -c;
- перевірити цілісність відновлених даних (наявність таблиць, зв'язків, тригерів тощо);
- продемонструвати, що вміст нової бази повністю відповідає оригінальній.

4. Додати користувачів бази даних та призначити їм права доступу до БД:

- створити *окремого користувача* (наприклад, student_user) з обмеженими правами для роботи з вашою базою;
- надати користувачу лише потрібні права (наприклад, SELECT, INSERT, UPDATE);
- перевірити, що користувач може виконувати дозволені операції та не має доступу до заборонених (наприклад, DELETE або DROP TABLE);
- створити користувача для резервного копіювання (backup_user) з мінімальними правами (SELECT, SHOW VIEW, TRIGGER);
- за потреби – створити користувача для відновлення бази (restore_user) з правами CREATE, ALTER, INSERT, CREATE ROUTINE.

5. Провести тестування безпеки та управління правами:

- зайти до СКБД через акаунт користувача з обмеженими правами. Спробувати виконати запит від імені користувача з обмеженими правами – перевірити реакцію системи;
- відкликати одне з наданих прав (наприклад, UPDATE) і переконатися, що команда тепер недоступна;
- видалити одного зі створених користувачів та перевірити реакцію системи.

Контрольні питання

1. Що таке резервне копіювання бази даних і для чого воно використовується?

2. Які існують типи резервних копій (повна, диференціальна, інкрементальна) та чим вони відрізняються?

3. Які команди SQL використовуються для створення резервної копії в різних СУБД (MySQL, PostgreSQL)?

4. У чому полягає процес відновлення бази даних із резервної копії?

5. Які параметри потрібно врахувати при плануванні регулярного резервного копіювання?

6. Що таке користувачі та ролі в SQL, і як вони пов’язані з правами доступу?

7. Які основні типи привілеїв доступу існують у SQL (SELECT, INSERT, UPDATE, DELETE, EXECUTE тощо)?

8. Яка команда SQL використовується для надання прав доступу користувачам, і як вона працює?

9. Як адміністратор може відкликати (revoke) права доступу і в яких ситуаціях це потрібно робити?

10. Які механізми безпеки додатково використовуються для захисту бази даних, окрім прав доступу (аудит, шифрування, обмеження підключень)?

ЛАБОРАТОРНА РОБОТА №9
Тема: «Встановлення MongoDB і виконання базових запитів»

Мета роботи: Ознайомитись із базами даних NoSQL, зокрема MongoDB, встановити середовища для роботи з MongoDB, виконати базові CRUD-операцій.

Програмне забезпечення: Операційна система (Windows, Linux або macOS), MongoDB Community Server, MongoDB Shell або MongoDB Compass, Python.

Завдання до лабораторної роботи

1. Встановити MongoDB:

- завантажити останню версію MongoDB Community Server з офіційного сайту https://www.mongodb.com/try/download/community (рис. 9.1);
- завантажити MongoDB Shell з офіційного сайту https://www.mongodb.com/try/download/shell;
- встановити MongoDB, слідуючи інструкціям;
- запуск MongoDB.

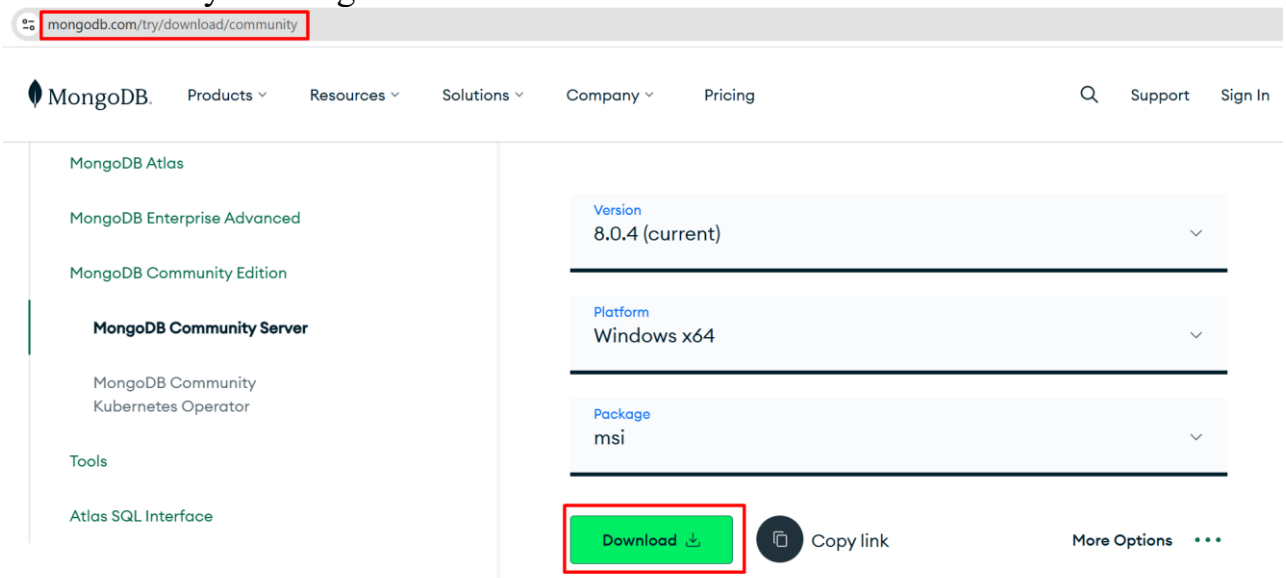

Рисунок 9.1 – Завантаження MongoDB з офіційного сайту

MongoDB можна завантажити кількома способами. Для Windows доступне завантаження встановлювача msi, а також завантаження архіву zip. Достатньо завантажити zip-архів і розпакувати його в потрібній нам папці.

Якщо шлях встановлення не було змінено під час встановлення, стандартне розташування програми у Windows:

C:\Program Files\MongoDB\Server\8.0\bin

Для Windows запуск відбувається через *mongod.exe*, для Linux/macOS можна скористатись командою *systemctl start mongod*.

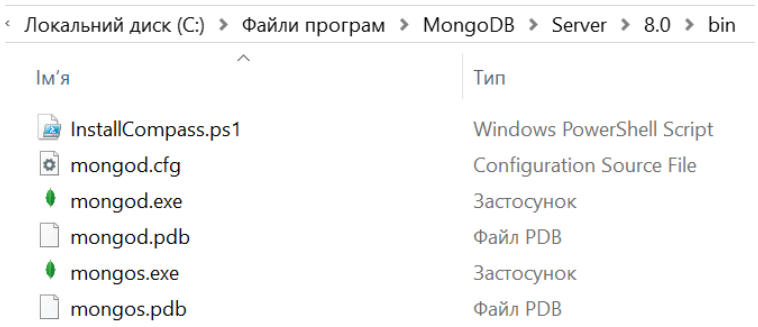

Рисунок 9.2 – Директорія з встановленою MongoDB

Переконатися, що сервер працює, виконавши команду *mongosh* або запустивши MongoDB Compass.

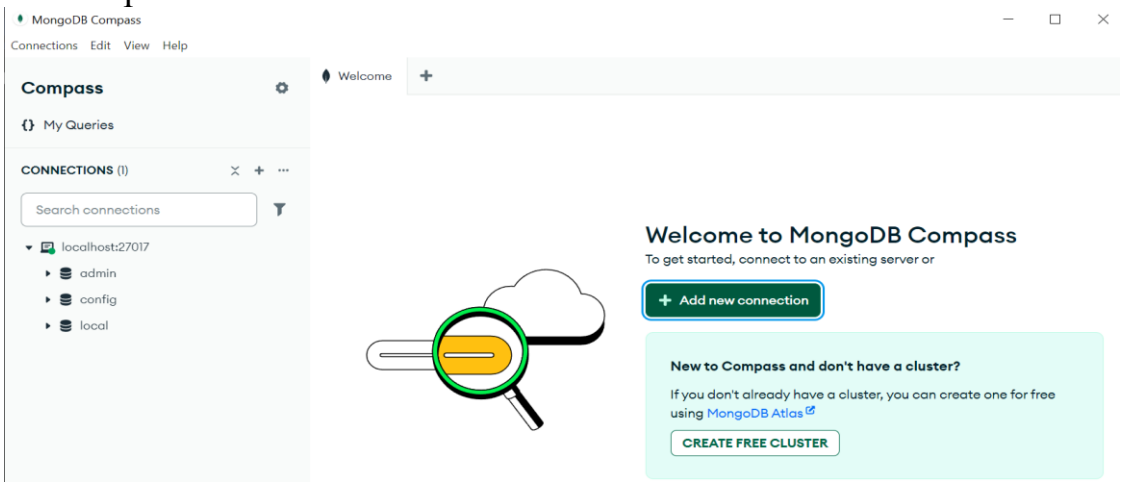

Рисунок 9.3 – Результат запуску MongoDB Compass

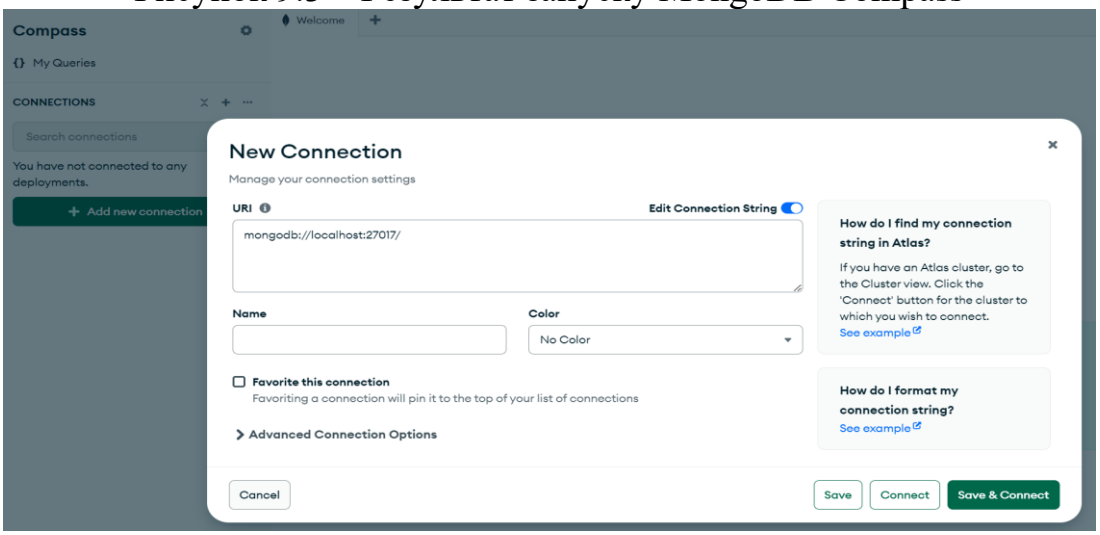

Рисунок 9.4 – Створення нового з'єднання в MongoDB Compass

Після натиснення на кнопку «Save & Connect» отримуємо інформацію про підключення. Для створення нової бази даних потрібно натиснути на «+» біля поля з URL підключення (рис. 9.5).

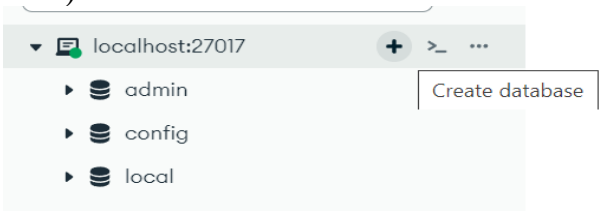

Рисунок 9.5 – Створення бази даних в MongoDB Compass

Для успішного створення бази даних необхідно ввести її назву та назву колекції, результат чого продемонстровано на рисунку 9.6.

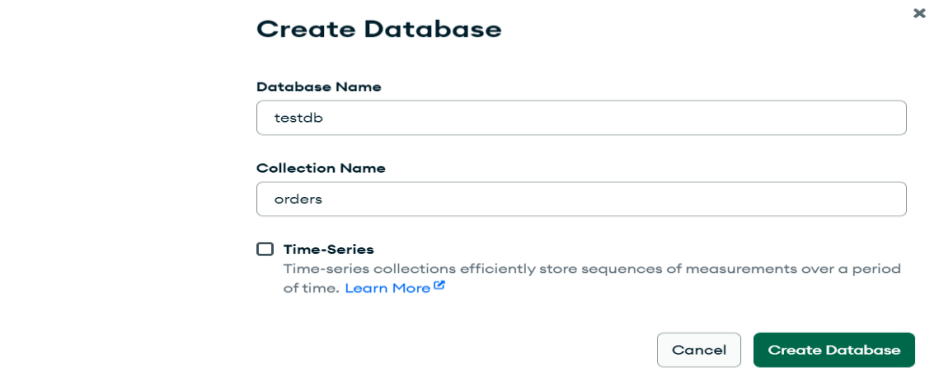

Рисунок 9.6 – Назва бази даних та колекції

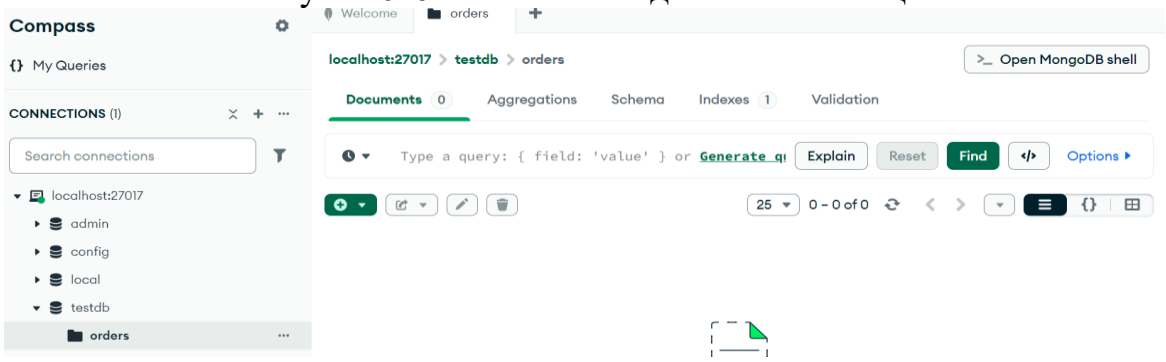

Рисунок 9.7 – Результат створення бази даних та колекції

Щоб наповнити колекцію даними натиснемо на «Add data (+)» (рис. 9.8).

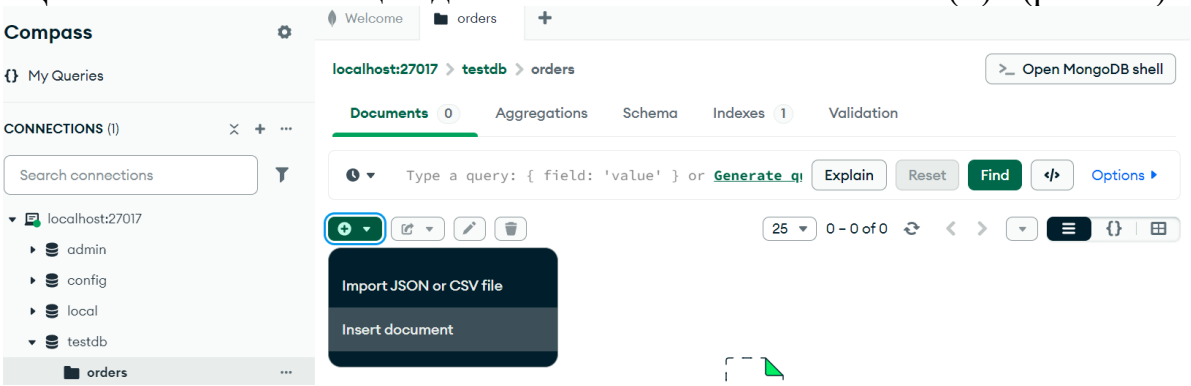

Рисунок 9.8 – Додавання даних до колекції

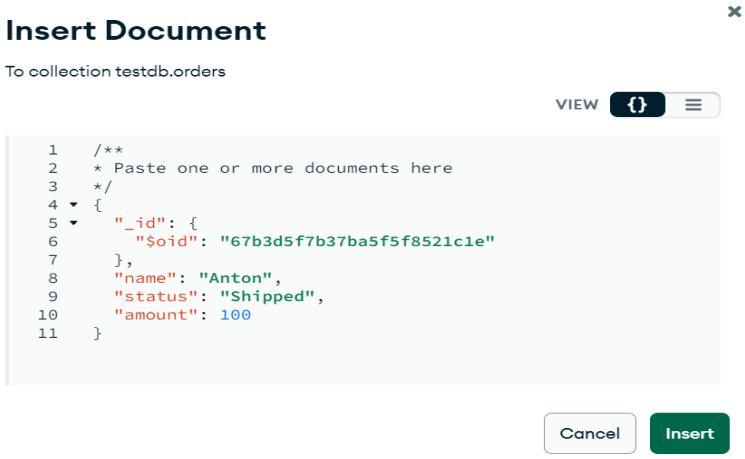

Рисунок 9.9 – Створення нового запису в колекції

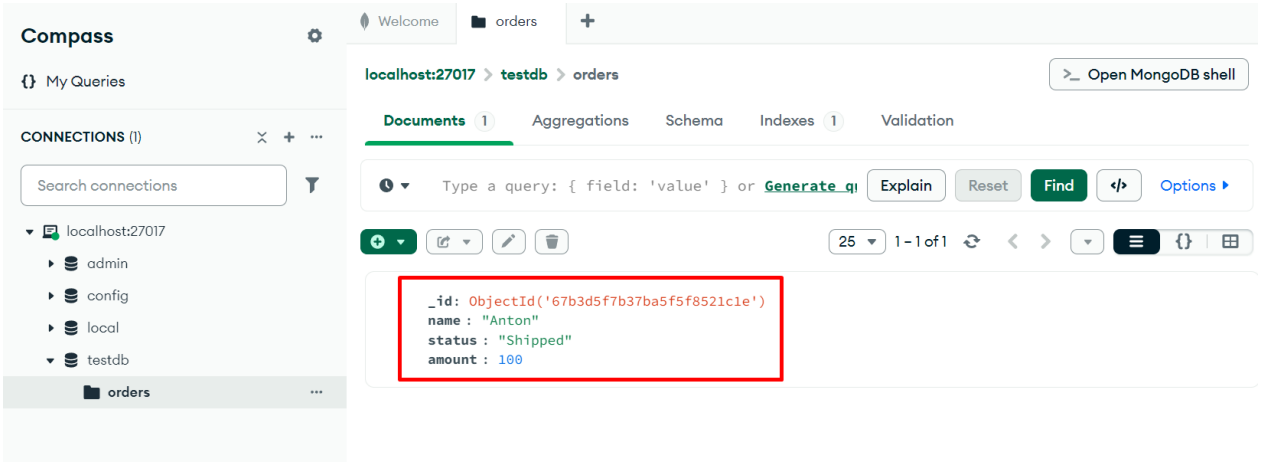

Рисунок 9.10 – Результат створення нового запису в колекції

Перевірити які дані містить колекція можна через MongoDB shell, який можна відкрити безпосередньо з MongoDB Compass (рис. 9.11), або через командний рядок.

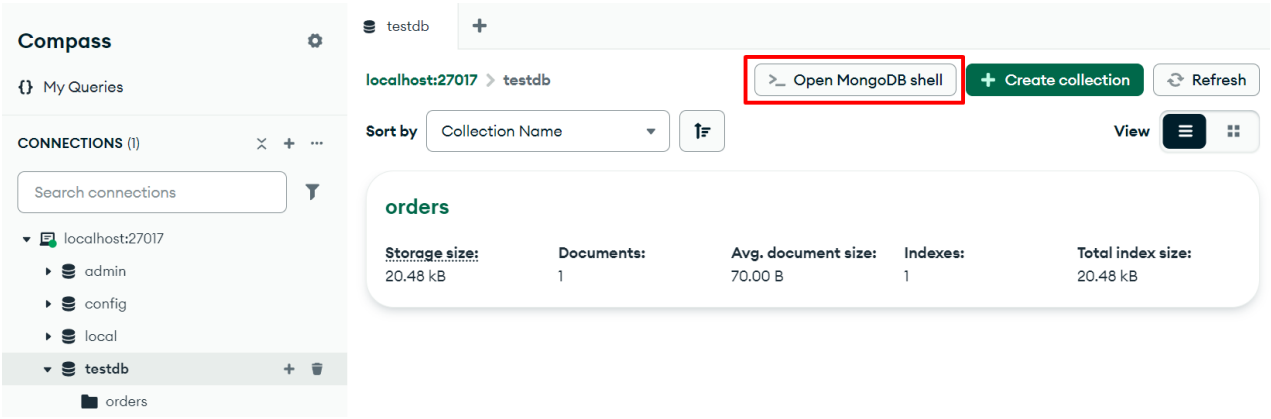

Рисунок 9.11 – Відкриваємо MongoDB Shell через MongoDB Compass

На рисунку 9.12 продемонстровано як отримати дані з колекції «orders» за допомогою команди: db["orders"].find()

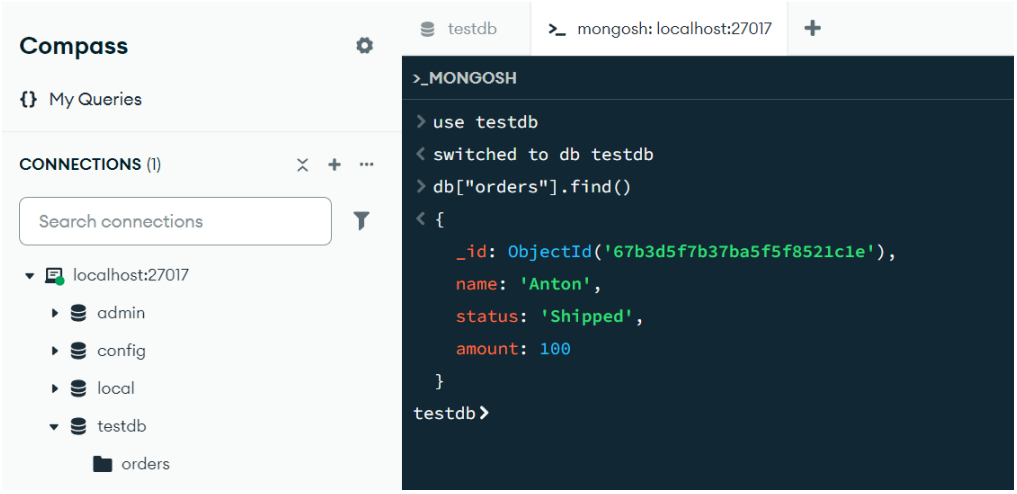

Рисунок 9.12 – Результат отримання даних з колекції «orders» через MongoDB Shell

Індивідуальні завдання до лабораторної роботи

Створити базу даних на задану тему та виконати базові CRUD-запити (Create, Read, Update, Delete).

Варіанти:

1. Бібліотека (книги, автори, читачі).
2. Кінотеатр (фільми, сеанси, квитки).
3. Інтернет-магазин (товари, категорії, замовлення).
4. Система бронювання готелів (готелі, номери, бронювання).
5. Онлайн-курси (курси, викладачі, студенти).
6. Автосалон (автомобілі, моделі, продажі).
7. Лікарня (пацієнти, лікарі, прийоми).
8. Спортивний клуб (спортсмени, тренери, секції).
9. Виставка мистецтв (експонати, художники, виставки).
10. Банківська система (клієнти, рахунки, транзакції).
11. Соціальна мережа (користувачі, пости, коментарі).
12. Школа (учні, вчителі, предмети).
13. Парк розваг (атракціони, квитки, відвідувачі).
14. Туристичне агентство (тури, клієнти, путівки).
15. Логістична компанія (склади, перевезення, товари).
16. Ресторан (страви, замовлення, клієнти).
17. Фермерське господарство (продукція, фермери, продажі).
18. IT-компанія (працівники, проєкти, завдання).
19. Музей (експонати, відвідувачі, екскурсії).
20. Благодійний фонд (донори, проєкти, фінансування).

Контрольні питання

1. Що таке NoSQL та чим MongoDB відрізняється від реляційних баз даних?
2. Яка команда використовується для запуску MongoDB у терміналі чи командному рядку?
3. Які основні типи даних підтримує MongoDB?
4. Як створити нову базу даних у MongoDB?
5. Яка команда вставки документа у колекцію використовується в MongoDB?
6. Як переглянути всі бази даних у MongoDB?
7. Як переглянути всі колекції у конкретній базі даних?
8. Що таке документ у MongoDB і як він структурований?
9. Як виконати простий запит для вибірки документів за певним полем у колекції?
10. Яка команда видаляє документ або колекцію у MongoDB?
11. Як виконати базові CRUD-операції у MongoDB?
12. Як підключитися до MongoDB через Python?

ЛАБОРАТОРНА РОБОТА №10
Тема: «Використання MongoDB: пошук даних (find), агрегація ($match, $group, $sort)»

Мета роботи: Ознайомитись з основними операціями пошуку та агрегації даних у MongoDB, навчитися використовувати метод find() та агрегатні оператори $match, $group, $sort для обробки інформації у NoSQL базах даних. Розглянути способи встановлення зв'язків між колекціями та виконання запитів для роботи з багатьма таблицями за допомогою оператора $lookup.

Програмне забезпечення: Операційна система (Windows, Linux або macOS), MongoDB Community Server, MongoDB Shell або MongoDB Compass, Python.

Завдання до лабораторної роботи

1. **Створити базу даних та колекції:**
   - створити базу даних university та колекцію students;
   - додати у колекцію щонайменше 5 документів з інформацією про студентів (ім'я, вік, курс, середній бал);
   - створити колекцію courses та додати дані про курси (назва курсу, викладач);
   - створити колекцію enrollments для зв'язку між студентами та курсами (Many-to-Many).
2. **Виконати пошук (find):**
   - знайти всіх студентів, що навчаються на 3 курсі;
   - вибрати студентів із середнім балом більше 85;
   - знайти всі записи у enrollments, щоб перевірити, які студенти записані на курси.
3. **Виконати агрегацію ($match, $group, $sort):**
   - відфільтрувати студентів, які навчаються на 2 та 3 курсах ($match);
   - групувати студентів за курсом та обчислити середній бал для кожного курсу ($group);
   - відсортувати результати за середнім балом у порядку спадання ($sort);
   - вивести імена студентів, які відповідають параметрам агрегації;
   - використати $lookup, щоб отримати список курсів, на які записаний студент.

Приклад коду з автоматичним присвоєнням id:

```
// 1. Додавання документів у колекцію
use university;
db.students.insertMany([
   {name: "Андрій", age: 20, course: 3, gpa: 88},
   {name: "Марія", age: 19, course: 2, gpa: 92},
   {name: "Олексій", age: 21, course: 4, gpa: 75},
   {name: "Ірина", age: 20, course: 3, gpa: 81},
   {name: "Сергій", age: 22, course: 4, gpa: 95}
```

```
]);

// 2. Пошук студентів на 3 курсі
db.students.find({course: 3});

// 3. Пошук студентів із середнім балом більше 85
db.students.find({gpa: {$gt: 85}});

// 4. Агрегація: фільтрація, групування та сортування
db.students.aggregate([
   { $match: { course: { $in: [2, 3] } } },
   { $group: { _id: "$course", avgGPA: { $avg: "$gpa" } } },
   { $sort: { avgGPA: -1 } }
]);

// 5. Додавання курсів
db.courses.insertMany([
   { name: "Математика", instructor: "Петренко" },
   { name: "Фізика", instructor: "Іваненко" },
   { name: "Програмування", instructor: "Сидоренко" }
]);

// 6. Отримання згенерованих _id
let students = db.students.find().toArray();
let courses = db.courses.find().toArray();

// 7. Додавання зарахувань студентів на курси (Many-to-Many)
db.enrollments.insertMany([
   { student_id: students[0]._id, course_id: courses[2]._id }, // Андрій ->
Програмування
   { student_id: students[0]._id, course_id: courses[0]._id }, // Андрій -> Математика
   { student_id: students[1]._id, course_id: courses[1]._id }, // Марія -> Фізика
   { student_id: students[2]._id, course_id: courses[2]._id }, // Олексій ->
Програмування
   { student_id: students[3]._id, course_id: courses[0]._id }, // Ірина -> Математика
   { student_id: students[3]._id, course_id: courses[1]._id }, // Ірина -> Фізика
   { student_id: students[4]._id, course_id: courses[2]._id } // Сергій ->
Програмування
]);

// 8. Отримання списку курсів, на які записаний студент (наприклад, Андрій)
db.enrollments.aggregate([
   {
      $lookup: {
         from: "students",
```

```
        localField: "student_id",
        foreignField: "_id",
        as: "student"
      }
    },
    {
      $lookup: {
        from: "courses",
        localField: "course_id",
        foreignField: "_id",
        as: "course"
      }
    }
]).pretty();

// 9. Отримання студентів, які записані на конкретний курс (наприклад, Програмування)
db.enrollments.aggregate([
    {
      $lookup: {
        from: "students",
        localField: "student_id",
        foreignField: "_id",
        as: "student"
      }
    },
    {
      $lookup: {
        from: "courses",
        localField: "course_id",
        foreignField: "_id",
        as: "course"
      }
    }
]).pretty();
```

Приклад коду з присвоєнням id вручну:

```
// 1. Додавання студентів
use university;
db.students.insertMany([
    { _id: 1, name: "Андрій", age: 20 },
    { _id: 2, name: "Марія", age: 19 },
    { _id: 3, name: "Олексій", age: 21 },
    { _id: 4, name: "Ірина", age: 20 },
    { _id: 5, name: "Сергій", age: 22 }
```

```
]);

// 2. Додавання курсів
db.courses.insertMany([
   { _id: 101, name: "Математика", instructor: "Петренко" },
   { _id: 102, name: "Фізика", instructor: "Іваненко" },
   { _id: 103, name: "Програмування", instructor: "Сидоренко" }
]);
// 3. Додавання зарахувань студентів на курси (Many-to-Many)
db.enrollments.insertMany([
   { student_id: 1, course_id: 103 },
   { student_id: 1, course_id: 101 },
   { student_id: 2, course_id: 102 },
   { student_id: 3, course_id: 103 },
   { student_id: 4, course_id: 101 },
   { student_id: 4, course_id: 102 },
   { student_id: 5, course_id: 103 }
]);
// 4. Отримання списку курсів, на які записаний студент (наприклад, Андрій)
db.enrollments.aggregate([
   { $match: { student_id: 1 } },
   {
      $lookup: {
         from: "courses",
         localField: "course_id",
         foreignField: "_id",
         as: "course_details"
      }
   }
]);
// 5. Отримання студентів, які записані на конкретний курс (наприклад,
Програмування)
db.enrollments.aggregate([
   { $match: { course_id: 103 } },
   {
      $lookup: {
         from: "students",
         localField: "student_id",
         foreignField: "_id",
         as: "student_details"
      }
   }
]);
```

Індивідуальні завдання до лабораторної роботи

Виконати аналогічні операції із завдання 2 у відповідності до створеної бази даних з лабораторної роботи №9 за допомогою Mongo Shell та Python.

Контрольні питання

1. У чому відмінність між find() та $match у MongoDB?
2. Як працює оператор $group? Які агрегатні функції можна використовувати?
3. Як здійснюється сортування результатів у MongoDB?
4. Які основні типи зв’язків можна реалізувати в MongoDB?
5. Як реалізувати зв’язок One-to-Many у MongoDB?
6. Як реалізувати зв’язок Many-to-Many у MongoDB?
7. Для чого використовується оператор $lookup?
8. Як можна отримати всі курси, на які записаний конкретний студент?
9. Як можна отримати список студентів, зарахованих на певний курс?
10. Які плюси та мінуси використання окремої колекції (enrollments) для збереження зв’язків у порівнянні з вкладеними документами?

ЛАБОРАТОРНА РОБОТА №11
Тема: «Робота з графовою базою даних Neo4j: основи мови Cypher»

Мета роботи: Ознайомитися з принципами роботи графової бази даних Neo4j, навчитися створювати, модифікувати та здійснювати запити до бази даних за допомогою мови Cypher. Закріпити навички моделювання зв'язків між об'єктами та роботи з графовими структурами даних.

Програмне забезпечення: Операційна система (Windows, Linux або macOS), Neo4j Desktop або Neo4j AuraDB (хмарна версія), Neo4j Browser (вбудований графічний інтерфейс для виконання запитів), Neo4j Bloom (інтерфейс для візуалізації графових даних), Docker (для запуску Neo4j у контейнері, за потреби).

Завдання до лабораторної роботи

1. Завантажити та встановити Neo4j Desktop (https://neo4j.com/download) або запустити Neo4j за допомогою Docker (https://hub.docker.com/_/neo4j).

2. Створити нову базу даних та підключитися до неї через браузерний інтерфейс Neo4j Browser.

3. Створити графову модель «Соціальна мережа».

Створіть вузли для кількох людей:

```
CREATE (:Person {name: "Іван", age: 30}), (:Person {name: "Оксана", age: 28})
```

Додайте між ними зв'язки (наприклад, "дружба"):

```
MATCH (a:Person {name: "Іван"}), (b:Person {name: "Оксана"})
CREATE (a)-[:FRIENDS_WITH]->(b)
```

4. Створити запити для пошуку інформації.

Отримайте список усіх людей у базі:

```
MATCH (p:Person) RETURN p
```

Знайдіть усіх друзів Івана:

```
MATCH (:Person {name: "Іван"})-[:FRIENDS_WITH]->(friend) RETURN friend.name
```

5. Оновіть та видаліть дані.

Змініть вік Оксани на 29 років:

```
MATCH (p:Person {name: "Оксана"}) SET p.age = 29
```

Видаліть зв'язок «дружба» між Іваном та Оксаною.

```
MATCH (a:Person {name: "Іван"})-[r:FRIENDS_WITH]->(b:Person {name: "Оксана"}) DELETE r
```

Приклад створення бази даних пісень для рекомендаційної системи:

```
CREATE (:Artist {name:"David Bowie"})
CREATE (:Person {name:"Alice"})
CREATE (:Song {name:"Space"})
MATCH (p:Person{name:"Alice"}), (s:Song{name:"Space"}) CREATE (p)-[:LIKES]->(s)
```

```
MATCH (p:Artist{name:"David Bowie"}), (s:Song{name:"Space"}) CREATE (p)-[:CREATED]->(s)
MATCH (p:Artist{name:"David Bowie"}) CREATE (p)-[:CREATED]->(s:Song{name:"Return"})
MATCH (s:Song{name:"Return"}) CREATE (p:Artist{name:"Freddy"})-[:CREATED]->(s)
MATCH (a:Artist{name:"Freddy"}) SET a.name = "Freddy Mercury"
MATCH (p:Artist{name:"David Bowie"}) CREATE (p)-[:CREATED]->(s:Song{name:"Again"})
MATCH (a:Song{name:"Again"}) DELETE a (рис. 1)
MATCH (a:Song{name:"Again"}) DETACH DELETE a
```

Індивідуальні завдання до лабораторної роботи

Виконати аналогічні операції із завдання 2 у відповідності до створеної бази даних з лабораторної роботи №10.

Контрольні питання

1. Що таке графові бази даних? У чому їх переваги для моделювання зв’язків?
2. Що таке мова Cypher у Neo4j? Наведіть приклади базових запитів.
3. Які основні елементи структури графової бази даних Neo4j?
4. Що таке вузол (node) і як його створити за допомогою мови Cypher?
5. Що таке ребро (relationship) у Neo4j і як його створити між двома вузлами?
6. Як задати властивості (properties) вузлів і ребер у Cypher?
7. Яка команда використовується для вибірки всіх вузлів у графі?
8. Як виконати запит на пошук вузлів за певною властивістю у Cypher?
9. Як оновити або додати властивості вузла у Neo4j за допомогою Cypher?
10. Яка команда видаляє вузли та/або ребра у графі і які обмеження слід враховувати при видаленні?

ЛАБОРАТОРНА РОБОТА №12
Тема: «Реалізація графових запитів у Neo4j»

Мета роботи: Навчитися створювати графову базу даних у Neo4j з нуля, використовувати мову Cypher для виконання складних запитів, включаючи пошук шаблонів та оптимізацію запитів. Ознайомитися з методами аналізу графів, такими як визначення центральності вузлів, пошук найкоротших та найдовших шляхів.

Програмне забезпечення: Операційна система (Windows, Linux або macOS)**,** Neo4j Desktop або Neo4j AuraDB (хмарна версія), Neo4j Browser (вбудований графічний інтерфейс для виконання запитів), Neo4j Bloom (інтерфейс для візуалізації графових даних), Docker (для запуску Neo4j у контейнері, за потреби).

Завдання до лабораторної роботи

1. Виконати проєктування графа, що відображає взаємозв’язки між людьми, компаніями та проєктами.

Початкові дані
- *Люди*: `Alice`, `Bob`, `Charlie`.
- *Компанії*: `TechCorp`, `InnovateLtd`.
- *Проєкти*: `ProjectX`, `ProjectY`.

Взаємозв’язки:
- `Alice` працює в `TechCorp`.
- `Bob` працює в `InnovateLtd`.
- `Charlie` працює в `TechCorp`.
- `Alice` бере участь у `ProjectX`.
- `Bob` бере участь у `ProjectY`.
- `Charlie` бере участь у `ProjectX`.

2. Створити граф за допомогою команд:

```
// Створення вузлів (люди)
CREATE (:Person {name: 'Alice', age: 30, city: 'New York'})
CREATE (:Person {name: 'Bob', age: 28, city: 'San Francisco'})
CREATE (:Person {name: 'Charlie', age: 35, city: 'Los Angeles'})
CREATE (:Person {name: 'David', age: 40, city: 'Chicago'})
CREATE (:Person {name: 'Eve', age: 29, city: 'Boston'})
CREATE (:Person {name: 'Frank', age: 33, city: 'Seattle'});

// Створення вузлів (компанії)
CREATE (:Company {name: 'TechCorp'})
CREATE (:Company {name: 'InnovateLtd'})
CREATE (:Company {name: 'HealthPlus'})
CREATE (:Company {name: 'GreenEnergy'});

// Створення вузлів (проєкти)
```

```
CREATE (:Project {name: 'ProjectX'})
CREATE (:Project {name: 'ProjectY'})
CREATE (:Project {name: 'ProjectZ'})
CREATE (:Project {name: 'ProjectAlpha'});

// Створення зв’язків (працевлаштування)
MATCH (a:Person {name: 'Alice'}), (c:Company {name: 'TechCorp'})
CREATE (a)-[:WORKS_AT]->(c);

MATCH (b:Person {name: 'Bob'}), (c:Company {name: 'InnovateLtd'})
CREATE (b)-[:WORKS_AT]->(c);

MATCH (c:Person {name: 'Charlie'}), (co:Company {name: 'TechCorp'})
CREATE (c)-[:WORKS_AT]->(co);

MATCH (d:Person {name: 'David'}), (h:Company {name: 'HealthPlus'})
CREATE (d)-[:WORKS_AT]->(h);

MATCH (e:Person {name: 'Eve'}), (g:Company {name: 'GreenEnergy'})
CREATE (e)-[:WORKS_AT]->(g);

// Створення зв’язків (участь у проєктах)
MATCH (a:Person {name: 'Alice'}), (p:Project {name: 'ProjectX'})
CREATE (a)-[:PARTICIPATES_IN]->(p);

MATCH (b:Person {name: 'Bob'}), (p:Project {name: 'ProjectY'})
CREATE (b)-[:PARTICIPATES_IN]->(p);

MATCH (c:Person {name: 'Charlie'}), (p:Project {name: 'ProjectX'})
CREATE (c)-[:PARTICIPATES_IN]->(p);

MATCH (d:Person {name: 'David'}), (p:Project {name: 'ProjectZ'})
CREATE (d)-[:PARTICIPATES_IN]->(p);
```

3. Провести виконання запитів.

3.1. Пошук осіб, які працюють у певній компанії

```
MATCH (p:Person)-[:WORKS_AT]->(c:Company {name: 'TechCorp'})
RETURN p.name AS PersonName;
```

3.2. Підрахунок кількості учасників у проєкті

```
MATCH (p:Person)-[:PARTICIPATES_IN]->(pr:Project {name: 'ProjectX'})
RETURN pr.name AS ProjectName, COUNT(p) AS ParticipantCount;
```

3.3. Пошук найкоротшого шляху між двома людьми:

```
MATCH path = shortestPath((a:Person {name: 'Alice'})-[:WORKS_AT|PARTICIPATES_IN*]-(b:Person {name: 'Bob'}))
RETURN path;
```

3.4. Використання індексів для оптимізації:

```
CREATE INDEX FOR (p:Person) ON (p.name);
CREATE INDEX FOR (c:Company) ON (c.name);
CREATE INDEX FOR (pr:Project) ON (pr.name);
```

4. Виконати аналіз графа.

Neo4j має бібліотеку Graph Data Science (GDS), яка дозволяє виконувати алгоритми для аналізу графів.

4.1. Завантаження графа в GDS. Цей запит створює граф із вузлів Person, Company та Project, а також зв’язків: WORKS_AT та PARTICIPATES_IN.

```
CALL gds.graph.project(
  'myGraph',  // Назва графа в пам'яті
  ['Person', 'Company', 'Project'],  // Вузли, які включаємо в граф
  {
    WORKS_AT: {orientation: 'UNDIRECTED'},
    PARTICIPATES_IN: {orientation: 'UNDIRECTED'}
  });
```

4.2. Запуск алгоритму PageRank. Цей запит обчислює PageRank для всіх вузлів у графі та повертає їх у порядку спадання важливості.

```
CALL gds.pageRank.stream('myGraph')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS Name, score
ORDER BY score DESC;
```

4.3. Видалення графа, якщо потрібно відбувається наступним чином:

```
CALL gds.graph.drop('myGraph');
```

Це корисно, якщо необхідно створити новий граф із іншою структурою.

4.4. Рекомендаційна система (спільні зв’язки)

```
MATCH (p1:Person)-[:PARTICIPATES_IN]->(pr:Project)<-[:PARTICIPATES_IN]-(p2:Person)
WHERE p1 <> p2
RETURN p1.name AS Person1, p2.name AS Person2, COUNT(pr) AS CommonProjects
ORDER BY CommonProjects DESC;
```

Індивідуальні завдання до лабораторної роботи

Виконати аналогічні операції із завдання 1 у відповідності до індивідуального варіанту створеної бази даних з лабораторної роботи №11.

Контрольні питання

1. Що таке графовий запит у Neo4j і яку роль відіграє мова Cypher?
2. Які основні переваги використання алгоритмів аналізу графів у Neo4j?
3. Як створити граф для алгоритмів GDS у Neo4j?
4. Як створити підграф (graph projection) у GDS і для чого він використовується?
5. Яка різниця між in-memory графом у GDS і звичайним графом у Neo4j?
6. Як у GDS виконати алгоритм PageRank і що він показує для вузлів графа?
7. Що таке центральність (centrality) у контексті GDS і які типи центральності підтримуються?
8. Як у GDS знаходяться спільні шляхи або компоненти зв’язності графа?
9. Яким чином можна запускати алгоритми прогнозування зв’язків (link prediction) у GDS?
10. Як у GDS відновити результати алгоритмів у базовий граф Neo4j для подальшого використання?
11. Які обмеження та рекомендації слід враховувати при роботі з великими графами у GDS?

ЛАБОРАТОРНА РОБОТА №13
Тема: «Інтеграція реляційних і нереляційних баз даних у спільному проєкті. Використання API для доступу до баз даних»

Мета роботи: Ознайомитися з принципами інтеграції реляційних (SQL) і нереляційних (NoSQL) баз даних у межах єдиного проєкту, розглянути підходи до їх взаємодії та реалізувати збереження й обробку даних з використанням обох типів БД.

Програмне забезпечення: MySQL або PostgreSQL (реляційна база даних), MongoDB або Firebase Firestore (нереляційна база даних), Середовище розробки (Visual Studio Code, PyCharm, IntelliJ IDEA тощо), Мова програмування (Python, Node.js або Java).

Завдання до лабораторної роботи

Розробити застосунок, який використовує одночасно SQL і NoSQL бази даних для збереження різних типів інформації.

1. Виконати налаштування середовища:
   - встановити MySQL/PostgreSQL та MongoDB;
   - створити необхідні таблиці та колекції.
2. Створити SQL-базу:
   - створити таблицю Users з полями id, name, email;
   - виконати операції INSERT, SELECT, UPDATE.
3. Створити NoSQL-базу:
   - створити колекцію UserLogs, де кожен запис містить user_id, action, timestamp;
   - реалізувати запис та отримання даних.
4. Провести інтеграцію SQL і NoSQL БД:
   - написати програму, яка при додаванні користувача в SQL-БД автоматично створює відповідний запис у NoSQL-БД;
   - реалізувати вибірку користувача разом із його логами дій.

Приклад коду на Python для роботи з MongoDB:

```
import mysql.connector
from pymongo import MongoClient

# Підключення до MySQL
mysql_conn = mysql.connector.connect(
   host="localhost",
   user="root",
   password="password",
   database="test_db"
)
cursor = mysql_conn.cursor()

# Підключення до MongoDB
```

```
    mongo_client = MongoClient("mongodb://localhost:27017/")
    mongo_db = mongo_client["test_db"]
    logs_collection = mongo_db["UserLogs"]

    # Додавання користувача в MySQL
    def add_user(name, email):
        cursor.execute("INSERT INTO Users (name, email) VALUES (%s, %s)", (name, email))
        mysql_conn.commit()
        user_id = cursor.lastrowid

        # Запис у MongoDB
        logs_collection.insert_one({"user_id": user_id, "action": "Created", "timestamp": "2025-03-31"})
        print("User added and logged successfully!")

    # Вибірка користувачів та логів
    def get_users():
        cursor.execute("SELECT * FROM Users")
        users = cursor.fetchall()
        for user in users:
            logs = logs_collection.find({"user_id": user[0]})
            print(f"User: {user}")
            for log in logs:
                print(f" Log: {log}")

    # Додавання нового користувача
    add_user("John Doe", "johndoe@example.com")
    # Отримання даних
    get_users()
```

Приклад інтеграції SQL і NoSQL БД**:**

1. Створення користувача у PostgreSQL (SQL):

```
INSERT INTO users (id, name, email) VALUES (1, 'Іван Петренко', 'ivan@example.com');
```

2. Отримання списку користувачів із PostgreSQL:

```
SELECT * FROM users;
```

3. Створення замовлення у MongoDB (Python):

```
from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')
db = client['shop']
orders = db['orders']
```

```
order = {"user_id": 1, "items": ["Ноутбук", "Мишка"], "date": "2025-03-31"}
orders.insert_one(order)
```

4. Отримання всіх замовлень із MongoDB:

```
for order in orders.find():
    print(order)
```

Індивідуальні завдання до лабораторної роботи

Виконати операції із практичної частини та доповнити їх у відповідності до індивідуального варіанту створеної бази даних з лабораторних робіт №9-12.

Контрольні питання

1. Які основні відмінності між SQL та NoSQL базами даних?
2. У яких випадках доцільно використовувати комбіноване рішення?
3. Які методи існують для інтеграції SQL і NoSQL?
4. Як відбувається зв’язок між реляційною та нереляційною БД у програмі?
5. Які переваги та недоліки має гібридний підхід до збереження даних?
6. Які компанії застосовують підхід Polyglot Persistence і для яких задач?

ЛАБОРАТОРНА РОБОТА №14
Тема: «Проєктування системи управління базами даних для конкретного кейсу»

Мета роботи: Навчитися перетворювати реляційну модель бази даних, створену в попередньому семестрі, у нереляційну (NoSQL) архітектуру. Ознайомитися з принципами денормалізації, документо-орієнтованого зберігання, вкладених структур та оптимізації під конкретні запити.

Завдання до лабораторної роботи

1. Проаналізувати структуру таблиць, зв'язків та основних сценаріїв використання з індивідуального завдання реляційної БД, створеної в попередньому семестрі навчання.
2. Розробити концептуальну нереляційну модель: визначити колекції, типи документів, вкладені структури.
3. Перетворити реляційні зв'язки (1:1, 1:N, N:M) у NoSQL-представлення.
4. Навести приклади документів у форматі JSON.
5. Запропонувати оптимальні індекси для MongoDB.
6. Продемонструвати приклади операцій CRUD та агрегації.
7. Обґрунтувати прийняті рішення щодо структури документів.
8. Необхідно створити зв'язок між MongoDB та Neo4j:
   - визначити, які дані залишаться в MongoDB (документи);
   - які зв'язки будуть зручно представлені в Neo4j (граф);
   - показати можливий сценарій інтеграції.

Вимоги до даних визначають, які сутності, атрибути та обсяги інформації потрібні, і саме це формує структуру майбутньої бази.

Процес проєктування бази даних для конкретного застосунку включає аналіз вимог, побудову концептуальної моделі, створення логічної моделі, розробку програмної реалізації та подальшу оптимізацію.

Концептуальна модель описує предметну область, логічна модель визначає структури даних у вибраній моделі, а фізична модель конкретизує, як це буде реалізовано в СКБД.

Нормалізація усуває дублювання та аномалії даних шляхом поділу таблиць на логічно узгоджені структури.

NoSQL доцільно використовувати при потребі гнучких схем, високої масштабованості, швидкої обробки великих даних або роботи з вкладеними/неструктурованими об'єктами.

Загальні принципи перетворення реляційних моделей на нереляційні:

1. Відмова від жорстких зв'язків – у NoSQL немає FOREIGN KEY.
2. Використання вкладених документів там, де в SQL було 1:N.
3. Зберігання масивів там, де було N:M.
4. Денормалізація – дублювання даних для прискорення читання.
5. Орієнтація на запити, а не на структуру.

Приклад відповідностей:

- SQL таблиця → MongoDB колекція.

- Рядок таблиці → JSON-документ.
- JOIN → або вкладений документ, або окремий запит.

Приклад побудови нереляційної моделі

1. Визначення основних сутностей

Необхідно визначити:

- які таблиці можуть бути об'єднані у документи;
- які дані варто зберігати вкладено;
- які дані повинні залишатися окремими.

Наприклад, у реляційній базі були таблиці:
Orders, OrderItems, Customers, Products.
У NoSQL можливий варіант:
orders, customers, products

У NoSQL OrderItems нікуди не зник – він став частиною документа orders. У реляційній моделі Order ↔ OrderItems – це зв'язок 1:N, тому потрібні дві таблиці. У документоорієнтованій моделі (MongoDB) такий зв'язок зазвичай перетворюється у вкладений масив об'єктів усередині одного документа.

Всередині кожного замовлення з'являється масив items, який замінює таблицю OrderItems:

```
{
 "order_id": 101,
 "customer_id": 5,
 "date": "2024-11-12",
 "items": [
  { "product_id": 12, "qty": 2, "price": 300 },
  { "product_id": 44, "qty": 1, "price": 190 }
 ]
}
```

Причини чому так варто робити:

1. У 99% випадків замовлення потрібне разом з товарами, тому логічно зберігати їх в одному документі.
2. У MongoDB немає класичних JOIN, тому краще об'єднати «батьківський» і «дочірні» записи.
3. 1:N – класичний випадок для вкладених документів, оскільки один Order → багато OrderItems → ідеально перетворюється в масив.
4. Читання стає швидшим, оскільки потрібні дані отримуються однією операцією з цілого документа, без звернення до кількох таблиць.

OrderItems не варто виносити всередину якщо:

- товарів у замовленні може бути **дуже багато** (тисячі);
- кожен елемент має бути **окремою сутністю**, яку часто оновлюють незалежно;
- потрібна аналітика по OrderItems на рівні всіх замовлень (хоча це теж можливо через $unwind);

Приклад перетворення зв'язку N:M із реляційної моделі у нереляційну. У реляційних базах даних зв'язок «багато-до-багатьох» (N:M) реалізується через третю проміжну таблицю, яка містить зовнішні ключі на дві головні таблиці. Наприклад, у системі навчального процесу:

- student – список студентів,
- course – список курсів,
- student_course – зв'язок, що зберігає, які студенти вивчають які курси і (за потреби) додаткову інформацію, як-от: оцінка, статус, семестр.

У NoSQL, зокрема MongoDB, немає необхідності створювати проміжну таблицю. Замість цього інформація про зв'язок може бути вбудована всередину документа студента у вигляді масиву об'єктів, де кожен об'єкт описує один курс.

Такий підхід працює завдяки гнучкій структурі документів і денормалізації, яка дозволяє зберігати всі пов'язані дані разом.

Приклад документа у MongoDB:

```
{
  "student_id": 42,
  "firstname": "Анна",
  "courses": [
    { "course_id": 1, "name": "DBMS", "score": 97 },
    { "course_id": 5, "name": "AI Basics" }
  ]
}
```

Переваги такого підходу:

- простота структури: дві таблиці + проміжна → один документ з масивом;
- швидке читання: інформація про студента та всі його курси завантажується одним запитом;
- менше JOIN: у MongoDB немає складних JOIN, тому краще зберігати пов'язані дані разом;
- гнучкість: кожен курс може мати свою структуру (оцінка, прогрес, дата початку), і це не потребує зміни схеми.

Коли варто зберігати масив, а не робити окрему колекцію:

- коли об'єкти тісно пов'язані з «батьківською» сутністю;
- коли кількість елементів у масиві помірна (до кількох тисяч);
- коли інформація найчастіше читається разом.

Приклади можливих колекцій. В MongoDB колекція зазвичай відповідає основній сутності або групі пов'язаних сутностей, які доцільно зберігати разом.

Приклади колекцій для різних варіантів:

1. Медичний центр:

- patients – дані про пацієнтів;
- visits або вбудовані visits у документ пацієнта;
- doctors – інформація про лікарів;

- prescriptions – призначення та рецепти.

Приклад фрагмента документа:

```
{
 "patient_id": 12,
 "name": "Олег",
 "visits": [
   { "date": "2024-03-11", "doctor": "Кардіолог", "diagnosis": "Гіпертонія" }
 ]
}
```

2. Бібліотека:

- books – книги;
- authors – автори;
- readers – читачі;
- loans або вкладене loans у читачів.

Приклад фрагмента документа:

```
{
 "reader_id": 7,
 "fullname": "Марія Іваненко",
 "loans": [
   { "book_id": 44, "loan_date": "2024-01-14", "return_date": null }
 ]
}
```

3. Транспортна система:

- routes – маршрути;
- buses – автобуси або інший транспорт;
- tickets – квитки (можуть бути вкладені в пасажира або окремими документами).

Приклад фрагмента документа:

```
{
 "route_id": "A12",
 "stops": ["Центр", "Університет", "Вокзал"]
}
```

Приклади CRUD, агрегації та індексації

Створення документа:

```
db.orders.insertOne({
  order_id: 101,
  customer_id: 5,
  items: [ { product_id: 12, qty: 2 } ]
});
```

Пошук:

```
db.orders.find({ customer_id: 5 });
```

Оновлення вкладених елементів:

```
db.orders.updateOne(
  { order_id: 101 },
  { $set: { "items.0.qty": 3 } }
);
```

Агрегація:

```
db.orders.aggregate([
  { $unwind: "$items" },
  { $group: { _id: "$customer_id", total: { $sum: "$items.qty" } } }
]);
```

Індексація:

```
db.orders.createIndex({ customer_id: 1 });
db.products.createIndex({ name: "text" });
```

Використання графової бази даних Neo4j у проєкті

Хоча основна мета роботи – перетворення реляційної БД у документоорієнтовану модель, студенти також повинні визначити, які частини їхньої системи можуть природно відображатися у вигляді графа.

Neo4j корисний тоді, коли система містить:

- складні зв’язки між об’єктами;
- сценарії з багатьма переходами (hops);
- пошук рекомендацій;
- ієрархії або мережі.

Приклади можливих графових моделей для різних варіантів

1. Інтернет-магазин.

Вузли: Customer, Product, Order, Category

Зв’язки: PURCHASED, BELONGS_TO, VIEWED, RECOMMENDS

Приклад Cypher:

```
MATCH (c:Customer)-[:PURCHASED]->(p:Product)
RETURN c, p;
```

2. Бібліотека.

Вузли: Reader, Book, Author, Genre

Зв’язки: READ, WRITTEN_BY, HAS_GENRE, SIMILAR_TO

3. Соціальна мережа.

Вузли: User, Post, Tag

Зв’язки: FOLLOWS, LIKED, TAGGED, COMMENTED

4. Система бронювання.

Вузли: User, Ticket, Flight/Room

Зв’язки: BOOKED, CONNECTS_TO, TRANSFER

Приклад гібридної архітектури:

- MongoDB зберігає замовлення або профілі користувача;

– Neo4j зберігає рекомендаційні зв’язки між товарами або взаємодії між користувачами.

Приклад створення графових зв’язків:

```
CREATE (c:Customer {id: 5, name: "Ірина"});
CREATE (p:Product {id: 12, name: "Ноутбук"});
CREATE (c)-[:VIEWED {date: "2024-03-11"}]->(p);
```

Контрольні питання

1. Як змінюється логіка проєктування даних при переході з реляційної моделі до документ-орієнтованої?
2. Чому зв’язки 1:N часто реалізують через вкладені документи у MongoDB, і які ризики або обмеження виникають при такому підході?
3. У чому фундаментальна різниця між реалізацією зв’язку N:M через проміжну таблицю в SQL та через масиви вкладених документів у NoSQL, і які компроміси це створює?
4. Як змінюється стратегія читання і запису даних при переході від JOIN у SQL до зчитування одного агрегованого документа в NoSQL?
5. Яку роль відіграє модель доступу (read-heavy, write-heavy, mixed) під час вибору між реляційним, документним та графовим способами зберігання даних?
6. У яких випадках дані, що були однією таблицею в SQL, доцільно перетворювати на окрему колекцію у MongoDB, а коли – робити їх частиною вкладеного масиву?
7. Як побудувати еквівалент SQL-схеми у Neo4j, і які випадки використання найбільше виграють від графової моделі?
8. Чому не всі SQL-зв’язки доцільно переносити в Neo4j як ребра, і за якими критеріями визначають, що повинно бути вузлом, атрибутом або зв’язком?
9. Які труднощі виникають при інтеграції реляційної БД, документної БД і графової БД у межах спільного застосунку, і як організувати цілісність даних між ними?
10. Як зміни у структурі даних при переході до NoSQL впливають на API-рівень застосунку, і які підходи дозволяють мінімізувати зміну бізнес-логіки при міграції?

ВИКОРИСТАНА ЛІТЕРАТУРА

1. Організація баз даних : навч. посібник / О. Г. Трофименко, Ю. В. Прокоп, Н. І. Логінова, І. М. Копитчук. 2-ге вид. виправ. і доповн. Одеса : Фенікс, 2019. 246 с. 2. Бази даних та інформаційні системи: навчальний посібник / Н. О. Харів. Рівне : НУВГП, 2018. 127 с.
2. Організація баз даних : навч. посібник / О. Г. Трофименко, Ю. В. Прокоп, Н. І. Логінова, І. М. Копитчук. 2-ге вид. виправ. і доповн. Одеса : Фенікс, 2019. 246 с. 2. Бази даних та інформаційні системи: навчальний посібник / Н. О. Харів. Рівне : НУВГП, 2018. 127 с.
3. Анісімов А.В., Кулябко П.П. Інформаційні системи та бази даних: Навчальний посібник для студентів факультету комп'ютерних наук та кібернетики. Київ. 2017. 110 с.
4. Гогерчак Г.І. Інформаційні системи та бази даних: навч. посіб. Київ: Лікей, 2019. 400 с.
5. Сагайда П. І., Зорі А. А., Тарасов О. Ф. Організація комп'ютерних систем для інтелектуальної обробки даних на основі опрацювання формалізованих знань: монографія. Краматорськ. ДДМА, 2020. 191 с.
6. Харів Н. О. Бази даних та інформаційні системи: навчальний посібник / Н. О. Харів. Рівне : НУВГП, 2018. 127 с.
7. *Elmasri R., Navathe S.* Fundamentals of Database Systems. Pearson, 2021. 1360 p.
8. Бази даних та інформаційні системи. Навчальний посібник / С.В. Шаров, В.В. Осадчий. Мелітополь: Вид-во МДПУ ім. Б. Хмельницького, 2014. 352 с.
9. Костенко О. Б. Організація баз даних та знань : конспект лекцій (для студентів денної та заочної форм навчання першого (бакалаврського) рівня вищої освіти за спеціальністю 126 – Інформаційні системи та технології) / О. Б. Костенко, І. О. Гавриленко ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. Харків : ХНУМГ ім. О. М. Бекетова, 2021. 92 с.
10. Організація реляційних баз даних: навч. посіб. Куваєв Я.Г., Жукова О.А., Сечкін І.А. 2-ге вид., допов. та переробл. Дніпро : НГУ, 2017. 157 с.
11. Системи баз даних та знань. Книга 1. Організація баз даних та знань: Підручник. Берко А.Ю., Верес О.М., Пасічник В.В. 3-тє вид., стер. Львів: "Магнолія 2006", 2024. 675 с.
12. Системи баз даних та знань. Книга 2. Організація баз даних та знань: Підручник. Берко А.Ю., Верес О.М., Пасічник В.В. 3-тє вид., стер. Львів: "Магнолія 2006", 2024. 584 с.
13. Костенко О. Б. Організація баз даних та знань: конспект лекцій. О. Б. Костенко, І. О. Гавриленко. Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова, 2021. 92 с.
14. Павловський, В. І. Бази даних та засоби управління: підручник для здобувачів ступеня бакалавра за освітньо-професійною програмою підготовки бакалаврів спеціальності 123 «Комп'ютерна інженерія». В. І. Павловський, А. В. Петрашенко, КПІ ім. Ігоря Сікорського. Київ, 2024. 326 с.
15. Bill Karwin. SQL Antipatterns: Avoiding the Pitfalls of Database Programming. Pragmatic Bookshelf, 2021. 300 p.

16. Boris Scholl, Trent Swanson, Peter Jausovec. Cloud Native: Using Containers, Functions, and Data to Build Next-Generation Applications. O'Reilly Media, 2019.
17. Michael Hunger, Corinne Klinger, Andreas Kollegger. Graph Databases for Beginners. O'Reilly Media, 2020.
18. Martin Fowler. NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley, 2013. 192 p.
19. Luc Perkins, Eric Redmond, Jim Wilson. Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement. Pragmatic Bookshelf, 2018. 350 p.
20. Eugenio Culurciello. Deep Learning for Database Developers. Packt Publishing, 2022.
21. SQL Tutorial. URL: https://www.w3schools.com/sql/default.asp (дата звернення: 10.01.2026).
22. SQL Підручник. URL: https://w3schoolsua.github.io/sql/index.html (дата звернення: 10.01.2026).
23. NoSQL Databases. URL: https://web.archive.org/web/20180517011249/http://www.christof-strauch.de/nosqldbs.pdf (дата звернення: 10.01.2026).
24. Graph Databases, NOSQL and Neo4j. URL: https://www.infoq.com/articles/graph-nosql-neo4j (дата звернення: 10.01.2026).
25. MongoDB. Tutorial. URL: https://www.w3schools.com/mongodb/ (дата звернення: 10.01.2026).
26. MongoDB. Підручник. URL: https://w3schoolsua.github.io/mongodb/index.html (дата звернення: 10.01.2026).
27. ДСТУ 8302:2015 Інформація і документація. Бібліографічне посилання. Загальні положення та правила складання. Чинний від 01.07.2016. Вид. офіц. Київ : ДП «УкрНДНЦ», 2016. 30 с.
28. ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура та правила оформлювання / Нац. Стандарт України. Вид. офіц. [На заміну ДСТУ 3008-95; чинний від 2017-07-01]. Київ : ДП «УкрНДНЦ», 2016. 31 с. (Інформація та документація).

ДОДАТОК А

Міністерство освіти і науки України
Черкаський державний технологічний університет

Факультет інформаційних технологій і систем

Кафедра комп'ютерних наук та системного аналізу

Дисципліна:
«Бази даних»

З В І Т

з лабораторної роботи № 1
Тема: «Встановлення та налаштування СУБД MySQL»

студента групи КН-2401
спеціальності F3 «Комп'ютерні науки»

Петренка Дмитра Олександровича

_____ _____
(Дата) (Підпис студента)

Оцінка _____

Перевірено _____
(Дата)
Викладач _____ / Максимов А.Є. /
(Підпис) (Прізвище та ініціали)

Черкаси – 2026 р.

З В І Т

про виконання завдань до лабораторної роботи 1

Тема: «Встановлення та налаштування СУБД MySQL»

Мета роботи: Здійснити інсталяцію та налаштування програмного забезпечення для роботи з реляційною системою керування базами даних MySQL, ознайомитися з веб-додатком phpMyAdmin, вивчити його можливості для роботи з базами даних.

Протокол розв'язування завдань

Завдання 1. У середовищі **phpMyAdmin** створити базу даних **Computers**, що містить таблицю **computer** з заданими полями:

id, type, processor, motherboard, ram, hd, monitor, keyboard, mouse, speakers.

Визначити основні характеристики зазначених полів і первинний ключ таблиці.

Заповнити таблицю **computer** реальними даними про комп'ютери, що є, наприклад, у деякому інтернет-магазині (https://comfy.ua/ua/).

Розв'язок завдання 1.

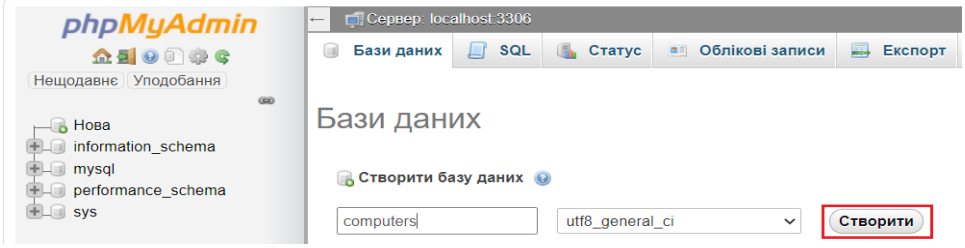

Рисунок 1 – Створення нової бази даних

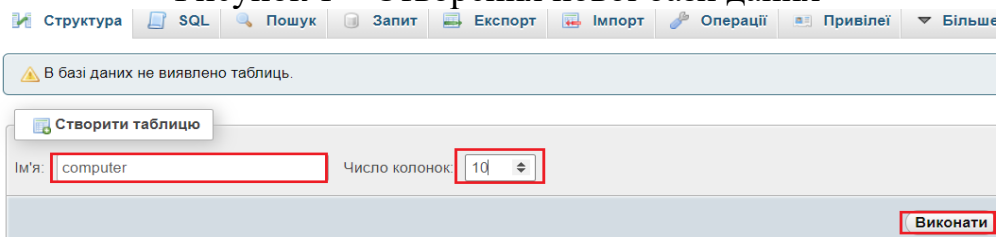

Рисунок 2 – Створення таблиці «computer»

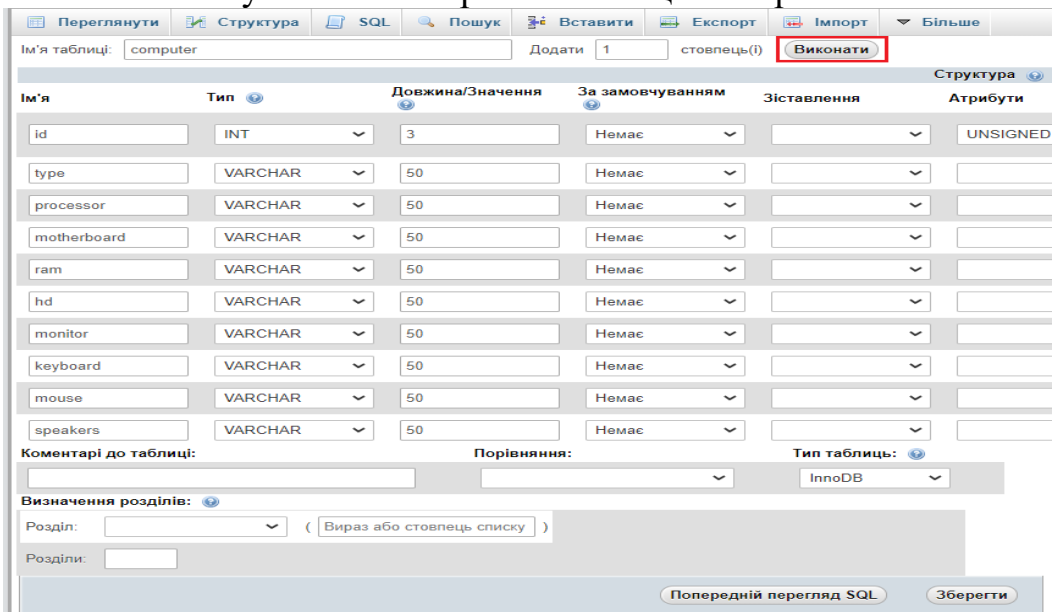

Рисунок 3 – Структура таблиці «computers»

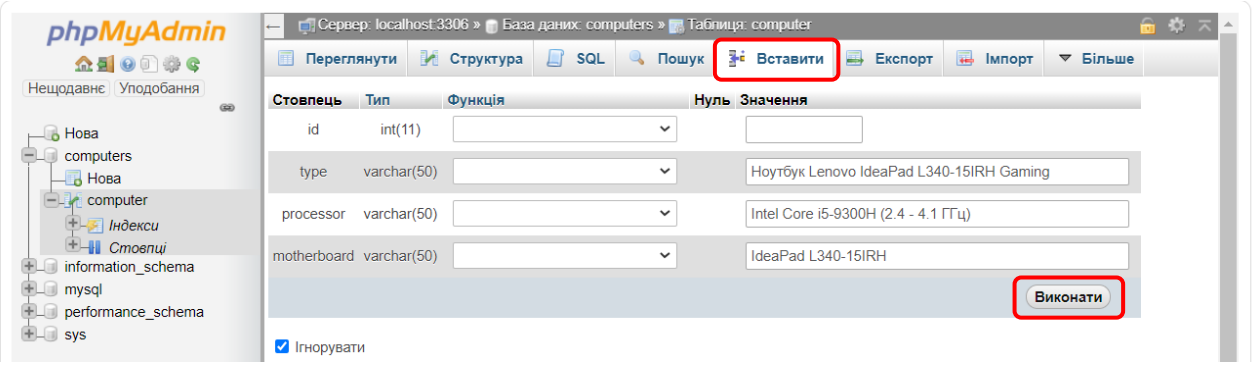

Рисунок 4 – Задання значень полів до таблиці «computers»

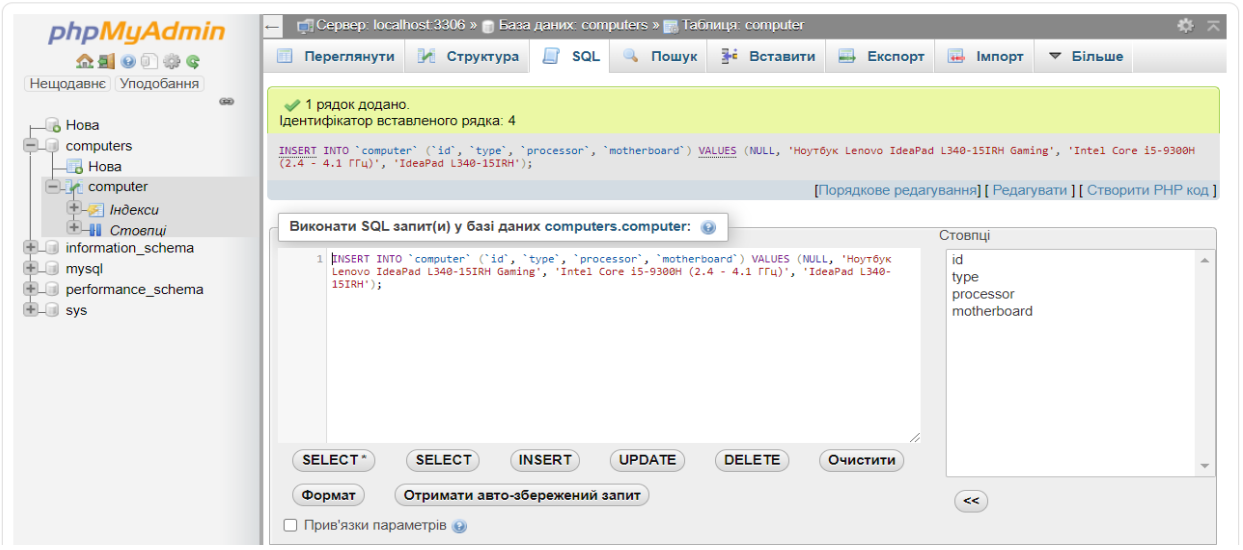

Рисунок 5 – Результат виконання SQL-запиту створення нового запису

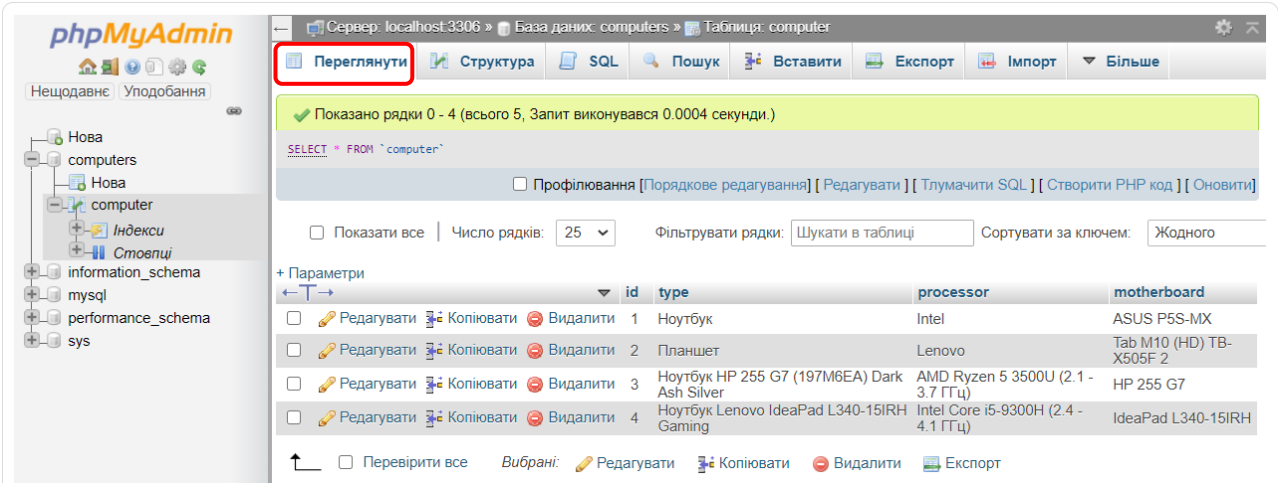

Рисунок 6 – Записи таблиці «computer»

Висновки

Під час виконання лабораторної роботи мною здійснено інсталяцію та налаштування програмного забезпечення для роботи з реляційною системою керування базами даних MySQL. У результаті виконання лабораторної роботи проведено ознайомлення з веб-додатком phpMyAdmin, розглянуто його можливості для роботи з базами даних.

ЗМІСТ