МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет Електроніки та інформаційних технологій
(повна назва інституту/факультету)

Кафедра наноелектроніки та модифікації поверхні
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри

_____ О.Д.Погребняк
(підпис) (Ім'я та ПРІЗВИЩЕ)

15 червня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр
(бакалавр / магістр)

зі спеціальності 153 «Мікро- та наносистемна техніка»,
(код та назва)

Освітньо-професійної програми «Нанотехнології та біомедичні системи»
(освітньо-професійної / освітньо-наукової) (назва програми)

на тему: «Порівняння можливостей та ефективності використання систем керування базами даних на прикладі MySQL та Sqlite»

Здобувача (ки) групи ФЕ-91 Зарудної Ліни Володимирівни
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Л.В. Зарудна
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник доц. О.В. Ющенко _____
(посада, науковий ступінь, вчене звання Ім'я та ПРІЗВИЩЕ) (підпис)

Консультант[1)] _____
(посада, науковий ступінь, вчене звання Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2023

Сумський державний університет

(назва вузу)

Факультет Електроніки та інформаційних технологій

Кафедра наноелектроніки та модифікації поверхні

Спеціальність 153 – «Мікро та наносистемна техніка»

ЗАТВЕРДЖУЮ:

Зав. кафедрою НЕтаМП

_____ О.Д. Погребняк

« 20 » березня 2023 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Зарудній Ліні Володимирівні

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) «Порівняння можливостей та ефективності використання систем керування базами даних на прикладі MySQL та Sqlite »

затверджена наказом по університету від «_____»_____2023р. №_____

2. Термін здачі студентом закінченого проекту (роботи) 16.06.2023р.

3. Вхідні дані до проекту (роботи) 1) DB Browser for SQLite ; 2) php my admin for MySQL;

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

1) Розробка бази даних;

2) Аналіз інтерфейсу;

3) дослідження можливостей редагування;

4) розгляд синтаксису запитів;

5) створення тригерів;

6) швидкість обробки даних

5. Перелік графічного матеріалу (з точним зазначення обов'язкових креслень)

1) схематичне зображення різних типів баз даних,

2) зображення різних типів зв'язків,

3) ER-діаграма,

4) скріншоти виконання запитів

6. Дата видачі завдання _____20.03.2023 р._____

Керівник _____

(підпис)

Завдання прийняв до виконання _____

(підпис)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Ознайомлення з програмою php my admin for MySQL.	04.04 - 08.04	виконано
2	Ознайомлення з програмою DB Browser for SQLite.	09.04 - 15.04	виконано
3	Створення структури бази даних. Побудова ER-діаграми.	16.04 - 01.05	виконано
4	Створення БД та таблиць для програм, редагування, вставка даних.	02.05 - 10.05	виконано
5	Дослідження типів даних для php my admin for MySQL та DB Browser for SQLite.	11.05 - 17.05	виконано
6	Вивчення синтаксису для запитів.	18.05 - 21.05	виконано
7	Створення тригерів.	22.05 - 23.05	виконано
8	Порівняння швидкості обробки даних.	24.05 - 25.05	виконано
9	Заповнення дипломного звіту.	26.05 - 01.06	виконано

Студент-дипломник _____

(підпис)

Керівник проекту _____

(підпис)

Анотація звіту студента

Дана робота містить 44 сторінки друкованого тексту, складається із вступу, двох розділів, висновків, переліку посилань і містить 32 рисунки та 5 таблиць. У звіті були розібрані дві різні системи керування базами даних, поняття СУБД, що таке база даних та їх типи, побудована ER-діаграма.

На основі порівняння двох СУБД створені тригери та запити за допомогою мови SQL. Мною були досліджені функції редагування, типи даних, створення таблиць, швидкість обробки даних. Для ефективного порівняння створена однакова база даних клініки з надання послуг для виявлення різниці редагування та створення таблиць.

Мета роботи – показати можливості та ефективність двох систем управління баз даними під час роботи з однією структурою бази даних для клініки . На основі їх роботи визначити відмінності інструментів програмування між MySQL та SQLite.

КЛЮЧОВІ СЛОВА: БАЗА ДАНИХ, СУБД, РЕЛЯЦІЙНА БАЗА ДАНИХ, DATABASE BROWSER FOR SQLITE, PHP MY ADMIN FOR MYSQL.

ЗМІСТ

ВСТУП

Світ інформаційних технологій постійно розвивається, і системи керування базами даних (СУБД) відіграють вирішальну роль в управлінні та організації даних. У повсякденному житті ми стикаємося з базами даних всюди : веб-сайти, торгівля, банки тощо. Майже у всіх випадках де люди використовують комп’ютерні технології для збереження даних та файлів. Якщо раніше такий обсяг інформації зберігали на папері а потім в архівах, то з появою комп'ютерів з’явилась можливість зберігати інформацію в електронному вигляді.

Існує багато способів зберігання даних. Найпростіший це текстовий файл. (.txt) Але він є не дуже зручним та повільним, якщо мова йде про пошук даних у такому файлі. На запит пошуку даних у такому файлі комп’ютеру знадобиться багато часу. Для більш швидкого та зручного користування даними були створені СУБД (Системи управління базами даних).

На сьогодні існує багато видів СУБД. Найпопулярніші СКБД на сьогоднішній день є Oracle Database, Microsoft SQL server, PostgreSQL, MySQL, Redis. Вони використовуються у більшості банківських справ, торгівельних сферах, медичних та ін

Таким чином, важливо порівнювати можливості та ефективність різних СУБД, щоб визначити, яка з них найкраще підходить для конкретних випадків використання. У цій бакалаврській роботі ми порівняємо дві широко використовувані СУБД, MySQL і SQLite, з точки зору їхніх можливостей і ефективності для управління та маніпулювання даними. Метою цієї роботи є комплексний аналіз двох систем, висвітлення їхніх сильних і слабких сторін, а також надання уявлення про те, яка з них краще підходить для конкретних завдань.

У першому розділі ми дізнаємося детальніше про базу БД та її типах, у другому розглянемо процес створення бази даних за допомогою MySQL і SQLite на прикладі бази даних для клініки яка надає послуги з консультації, діагностики та терапії.

РОЗДІЛ 1. БАЗА ДАНИХ

1.1 Поняття база даних

Дані - це будь-який фрагмент інформації, який може зберігатися та оброблюватися комп'ютерною системою. Це може включати текст, числа, зображення, аудіофайли тощо.

База даних - це структурований набір даних, який зберігається в комп'ютерній системі. Він призначений для ефективної організації, керування та отримання великих обсягів даних.

Бази даних використовуються в широкому спектрі додатків, від управління запасами в малому бізнесі до обробки мільйонів транзакцій у великій фінансовій установі. У базі даних дані організовані в таблиці, які складаються з рядків і стовпців. Кожен стовпець представляє поле або атрибут даних, тоді як кожен рядок представляє запис. Наприклад, у базі даних клієнтів рядок може представляти окремого клієнта, а кожен стовпець міститиме інформацію про цього клієнта, наприклад його ім'я, адресу та номер телефону.

Важливо також усвідомлювати важливість якості даних. Дані низької якості можуть призвести до неточностей і помилок в аналізі, що може мати значні наслідки при прийнятті рішень. Таким чином, дуже важливо переконатися, що дані точні, повні та послідовні.

1.2 СУБД та їх переваги

Для ефективного управління базою даних використовується система управління базами даних (СУБД). СУБД відповідає за такі завдання, як зберігання даних, резервне копіювання та відновлення, а також безпека. СУБД також надає ряд інструментів для керування даними, наприклад мови запитів для отримання певних даних, а також форми та звіти для відображення та узагальнення даних у змістовний спосіб.

Метою СУБД є дозволити користувачам швидко й ефективно зберігати й отримувати дані. До переваг використання СУБД відносяться :

Безпека даних: СУБД дозволяють реалізувати автентифікацію користувачів і контроль доступу, що може допомогти захистити конфіденційні дані.

Цілісність даних: СУБД надають такі функції, як перевірка даних, посилальна цілісність і керування транзакціями, які можуть допомогти забезпечити точність і послідовність даних.

Спільне використання даних: СУБД дозволяють кільком користувачам отримувати доступ до даних і змінювати їх одночасно, що може покращити співпрацю та ефективність.

Масштабованість: СУБД можуть обробляти великі обсяги даних і їх можна масштабувати відповідно до потреб зростаючих програм.

Контроль над надмірністю даних: СУБД забезпечують централізоване сховище даних, що може допомогти зменшити надмірність даних і забезпечити узгодженість даних.

Загалом, СУБД відіграють вирішальну роль у розробці сучасного програмного забезпечення та необхідні для додатків, які вимагають ефективного зберігання та пошуку даних.

1.3 Типи БД

Існує кілька типів баз даних, кожен з яких підходить для певних випадків використання. Одним із найпоширеніших типів є реляційна база даних, яка організовує дані в таблиці з попередньо визначеними зв’язками. Інші типи баз даних включають бази даних NoSQL, які є більш гнучкими та дозволяють зберігати неструктуровані дані, а також об’єктно-орієнтовані бази даних, які зберігають дані як об’єкти з атрибутами та методами.

Оскільки існує багато різних типів баз даних. Якою буде найкраща база даних залежить від того як ти або організація збирається її використовувати.

Розглянемо декілька типів баз даних та їх призначення:

Реляційна база даних: Цей тип БД зберігає дані в таблицях, які мають зв'язки одна з одною. Реляційні БД є найпопулярнішим типом БД і використовуються в широкому діапазоні програм. У реляційній моделі використовуються три ключові терміни, атрибути та домени. Під терміном відношення ми розглядаємо саму таблицю, її рядки та стовбці. Атрибутами називаються іменовані стовпці відношення, а домен – це значення, які можуть приймати атрибути. Така модель БД має первинний ключ та вторинний (зовнішній). Первинний ключ – це унікальний ключ який ідентифікує таблицю. Він не може мати нульових значень. Вторинний посилається на первинний в інші таблиці. Він визначає спосіб об'єднання таблиць. Перевагами такої моделі є те, що зміни в структурі бази даних не впливають на доступ до даних, також ми можемо написати складний запит для отримання інформації.

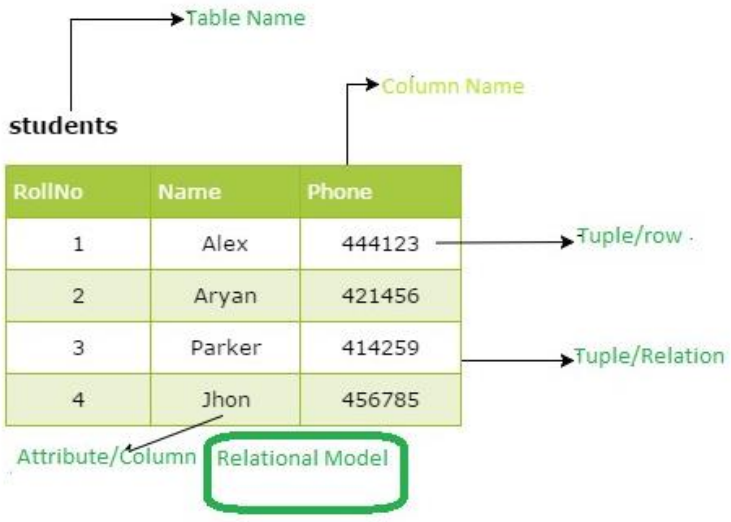


Рисунок 1.1 Схематичне зображення реляційного типу [1]

Об'єктно-орієнтована база даних: Цей тип БД зберігає дані в об'єктах, які можуть включати дані з їхніми атрибутами і поведінкою. Зазвичай використовуються в програмах, які мають справу зі складними структурами даних. Такий тип часто використовується у об'єктно-орієнтованому програмуванні. Успадкування, поліморфізм, перевантаження, ідентичність об'єкта, інкапсуляція та приховування інформації за допомогою методів надання інтерфейсу об'єктам є

одними з ключових понять об’єктно-орієнтованого програмування, які знайшли застосування в моделюванні даних. На відміну від традиційних баз даних, таких як ієрархічні, мережеві або реляційні, об’єктно-орієнтовані бази даних можуть обробляти різні типи даних, наприклад, зображення, голосове відео, включаючи текст, числа тощо.

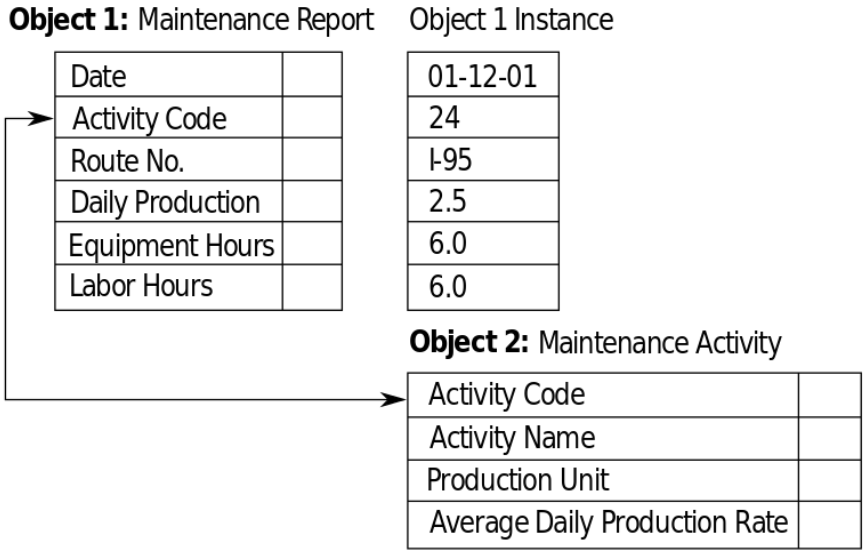


Рисунок 1.2 Схематичне зображення об’єктно-орієнтованого типу [2]

Ієрархічна база даних: цей тип БД зберігає дані в ієрархічному форматі, де кожен запис має зв’язок «батьківсько-дочірнім». Ця модель БД є одною з найстарішою і він рідше використовується у наш час. Дані у такій моделі організовані у в деревоподібну структуру. Ієрархічні СУБД зазвичай використовуються в додатках, які потребують високого ступеня організації даних. Недоліками такої модель є те що, вона може підтримувати зв’язки «один до багатьох», зв’язки «багатьох» не підтримуються.

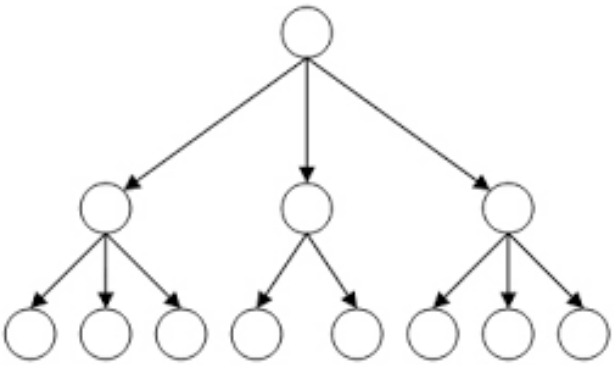


Рисунок 1.3 Схематичне зображення ієрархічного типу [1]

Мережеві бази даних: цей тип БД схожий на ієрархічні СУБД, але допускає більш складні зв’язки між записами і може мати більше ніж один батьківський вузол. Простими словами це модифікована версія ієрархічної моделі бази даних. Порівняно з більш сучасними БД ця мережева модель є не дуже зручною, через те що кожен запис може мати багато «батьків». Це ускладнює розуміння та обробку даних зі зростанням самої бази даних. Тому операції вставки, видалення та оновлення будь-якого запису можуть потребувати великої кількості налаштувань покажчиків.

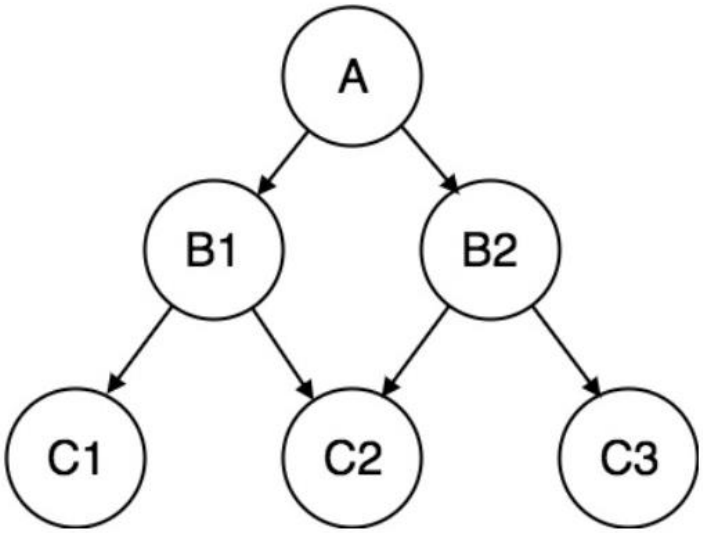


Рисунок 1.4 Схематичне зображення мережевого типу [1]

РОЗДІЛ 2. СТВОРЕННЯ БАЗИ ДАНИХ

Для того щоб повноцінно зрозуміти відмінності у можливостях а ефективності використання двох СУБД, буде створена однакова модель бази даних для MySQL та Sqlite.

Для прикладу беремо будь-яку приватну клініку, яка надає платні послуги з консультації, діагностики та терапії різних захворювань.

Для створення бази даних потрібно розподілити елементи інформації на основі сутності. База даних буде містити 7 таблиць:

I. Список послуг (service)
   - вид послуги та опис послуги (s_name)
   - ціна послуги (s_price)

II. Список пацієнтів (patient)__
   - Прізвище (p_name1)
   - Ім’я (p_name2)
   - По батькові (p_name3)
   - дата народження (p_bdate)_
   - номер телефону (p_phone_number)
   - стать Ч/Ж (p_sex)

III. Список лікарів клініки (doctor)
   - Прізвище (d_name1

- Ім’я (d_name2)
- По батькові (d_name3)
- категорія лікаря (ключ до таблиці category) (d_c_id)
- стаж лікаря (d_experience)
- графік роботи лікаря (ключ до таблиці work schedule) (d_sc_id)
- закріплений кабінет за лікарем (d_room)
- номер телефон лікаря (d_phone_numder)

IV. Список відвідування (vizit)

- дата, день відвідування (v_data)
- час (v_time)
- ID пацієнта (ключ до таблиці patient) (v_p_id)
- ID послуги (ключ до таблиці service) (v_s_id)
- ID лікаря (ключ до таблиці doctor) (v_d_id)

V. Категорії лікарів (category)

- Назва категорії (c_name)

VI. Робочій графік лікарів (schedule)

- день тижня (sc_day)
- початок та кінець робочого дня (sc_time)

VII. Зв’язок між послугою що надаються клінікою і кваліфікацією лікаря (m_2_m)

- ID послуги (ключ до таблиці service) (s_m_id)
- ID категорії (ключ до таблиці category) (c_m_id)

Таблиця 'm_2_m' була створена для того щоб можна було поєднати категорії лікарів та послуги які вони надають, бо один лікар може надавати більше одної послуги і навпаки.

Щоб розуміти які зв’язки повинні бути між різними сутностями потрібно розуміти значення поєднань. Деякі приклади типів зв’язків наведені на рисунку 2.1.

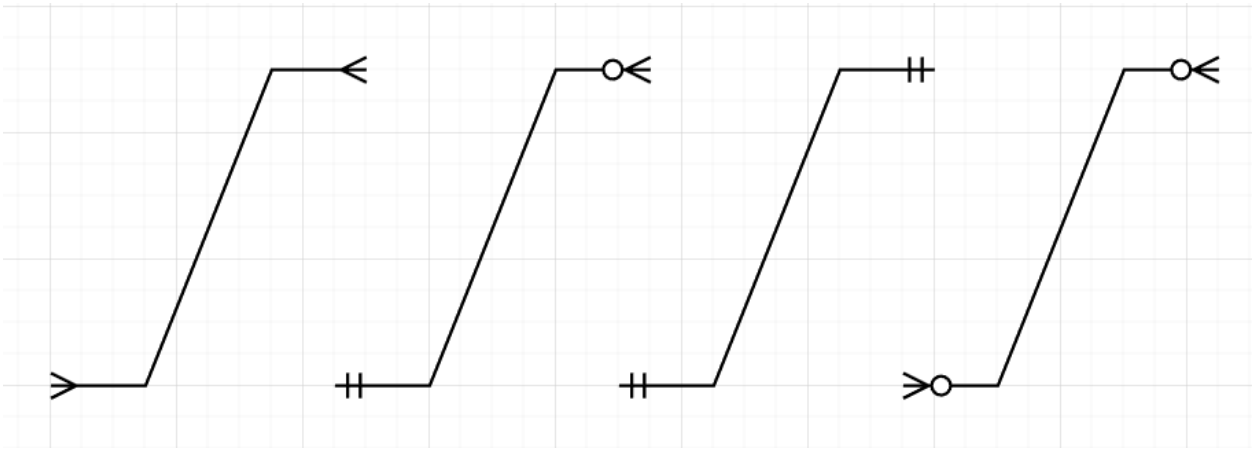


Рисунок 2.1 – Типи зв’язків: багато до багатьох, один обов’язковий до багатьох необов’язкових, один до одного, багато необов’язкових до багатьох необов’язкових.

2.1 Побудова ER-діаграми.

Для того, щоб детально мати уяву якою повинна бути база даних ми будуємо ER-діаграму. Діаграма взаємозв’язку сутностей (ER) — це тип блок-схеми, яка ілюструє, як «сутності», такі об’єкти чи поняття, пов’язані між собою в системі. Вони використовуються для проектування або налагодження реляційних баз даних. [3]

Кожен стовбець має свій унікальний ідентифікатор (PK Primary Key). Кожна таблиця може мати лише один такий первинний ключ і він обов’язково повинен мати ознаку NOT NULL. У нашому випадку наприклад для таблиці 'Patient' первинним ключем буде 'p_id', для таблиці 'Visit' - 'v_id'. І аналогічно так само для інших таблиць. Встановивши потрібні зв’язки отримуємо наступну діаграму, побудовану за допомогою інтернет сайту draw.io (рис 2.2).[4]

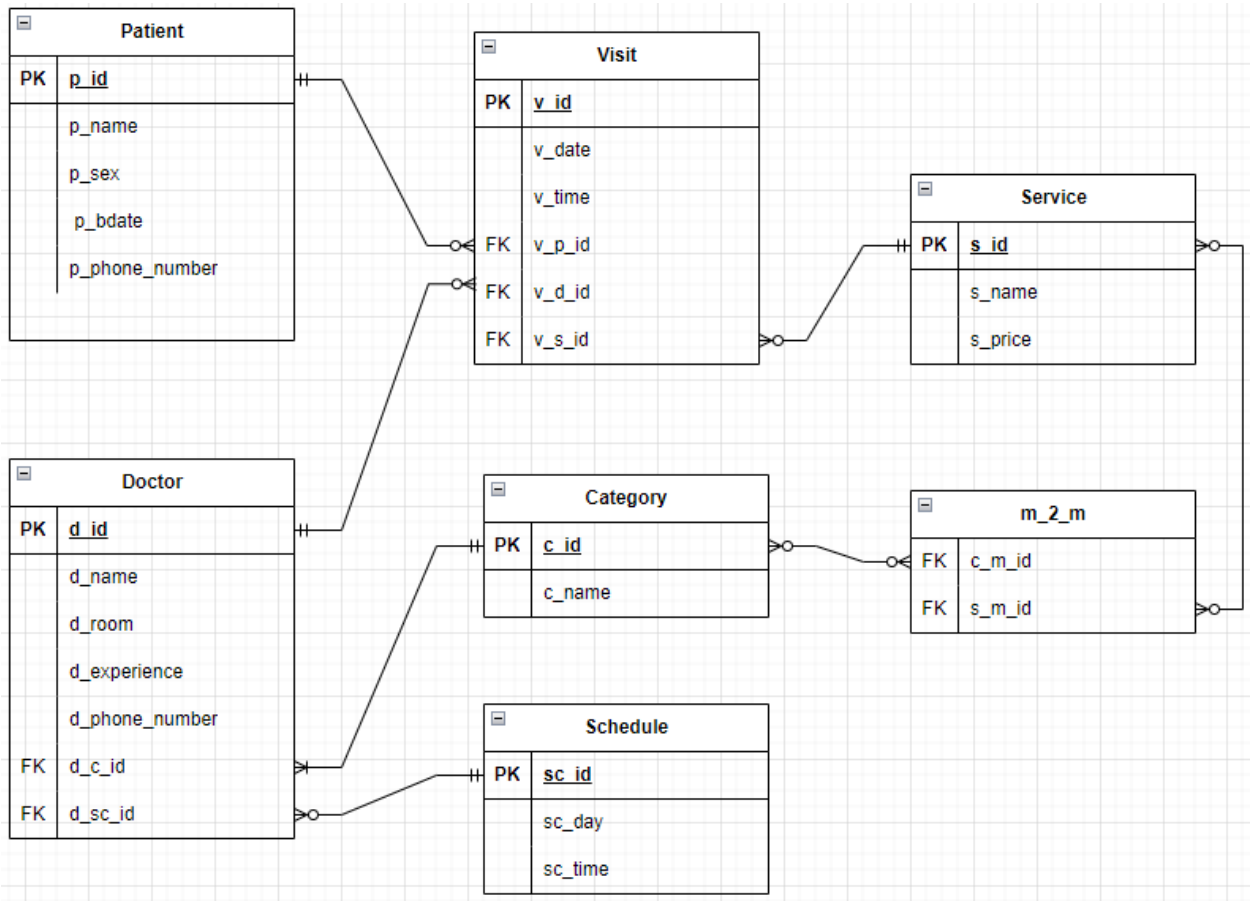


Рисунок 2.2 - ER-діаграма для клініки

2.2 Створення БД для СУБД MySQL

Для того щоб користуватися MySQL можна перейти на офіційний сайт та скачати дистрибутив останньої версії та додатково встановити програму MySQL Workbench для візуального адміністрування бази даних.

Для більш зручного користування з MYSQL можна працювати з Open Server (доступний для скачування на офіційному сайті) через інструмент адміністрування php my admin або інші.

Після встановлення Open Server на нижній панелі задач автоматично з’явиться червоний прапорець, після активації програми він стає зеленим, це означає що наш локальний веб-сервер запустився а з ним автоматично і запускається MySQL. Для роботи з MySQL і в цілому з системами управління базами даних є два способи. Перший це відкривати консоль, через нього

підключитися до MySQL та прописувати команди для редагування створеної вами БД. Такий спосіб є не легким але правильним. Другий спосіб це використання візуальних засобів для адміністрування систем управління БД. До таких програм відносяться PHPAdminer, php my admin та інші.

У нашому випадку ми будемо використовувати другий спосіб для візуалізації даних.

Через Open Server ми відкриваємо php my admin для подальшого користування. (рис 2.3)

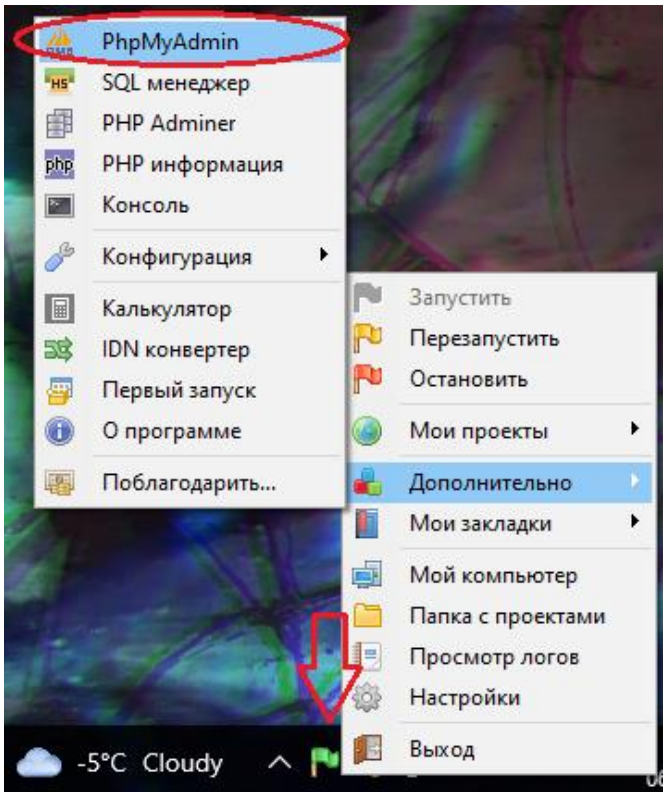


Рисунок 2.3 – Запуск php my admin через Open Server

Php my admin початково має три системних бази даних, їх не можна видалити та редагувати (рис 2.4). Для створення та користування своєю базою банних вони не потрібні.

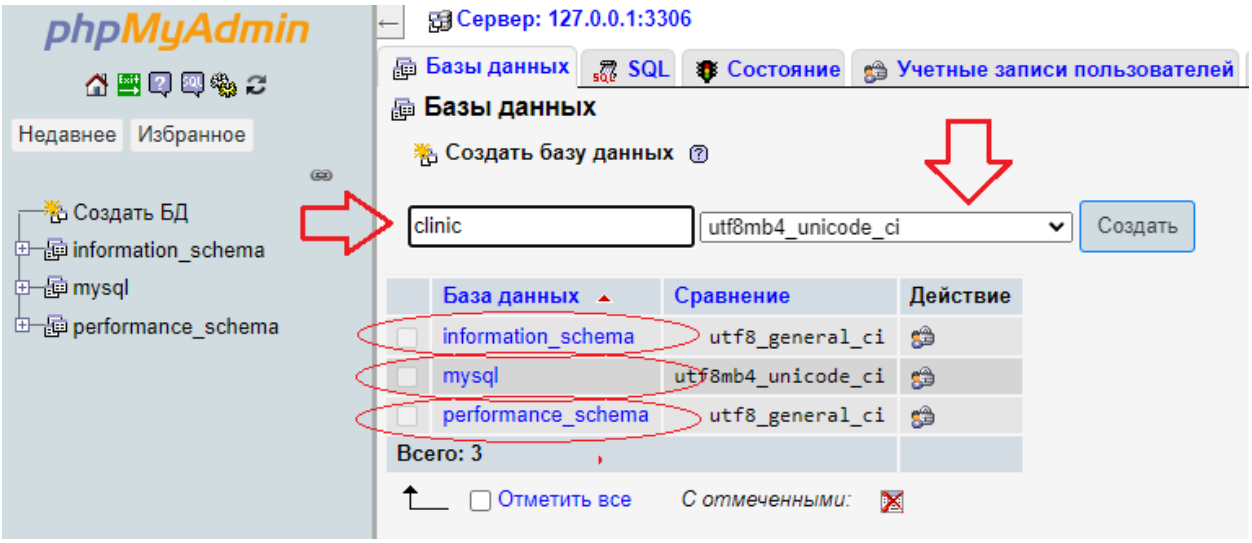


Рисунок 2.4 – Вебінтерфейс php my admin

Для створення своєї БД треба вказати лише два поля (рис 2.4). Перше це назва БД та тип кодування. Найрозповсюдженими є тип utf8_general_ci та utf8mb4_general_ci. Різниця між ними в тому що тип utf8_general_ci використовує від 1 до 3 байт для зберігання символів. Якщо ви плануєте використовувати символи до 4 байт краще обирати другий тип кодування. Після заповнення полів, створена таблиця з’явиться у списках у лівій частині програми (clinic). На одному MySQL сервері баз даних може існувати багато. Чим більше баз даних буде обслуговувати один сервер тим більшу навантаження він буду приймати.

2.2.1 Таблиці та їх редагування в MySQL

Створюємо першу таблицю в БД, яка має назву ‘Patient’ та містить такі дані: Прізвище, Ім’я, По батькові, стать, дата народження , номер телефону та ID. (рис 2.5)

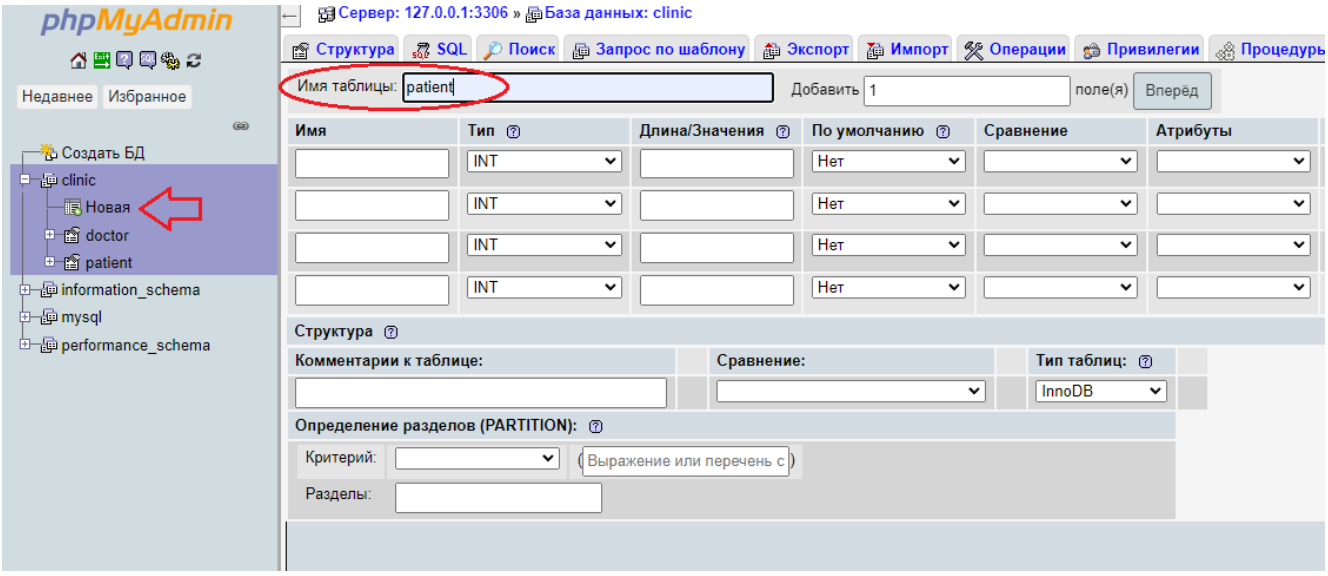


Рисунок 2.5 – Створення таблиці

Дуже важливо обрати правильний тип даних який буде приймати рядок (табл 2.1).

Для поля id бажано вказати в атрибутах UNSIGNED. Так як тип даних INT може зберігати в собі від’ємні значення, а в даному випадку тепер він буде приймати значення від 0 до 4 294 967 295. Також для id бажано вказати автоінкремент (A_I). Дана функція означає, що MySQl буде автоматично проставлятиме нумерацію і не потребуватиме ручного вводу.(рис 2.6)

Для поля phnumber обираємо тип даних VARCHAR та вказуємо обмеженість у символах (15 символів), так як номер телефону клієнта потребує мінімум 10 символів.

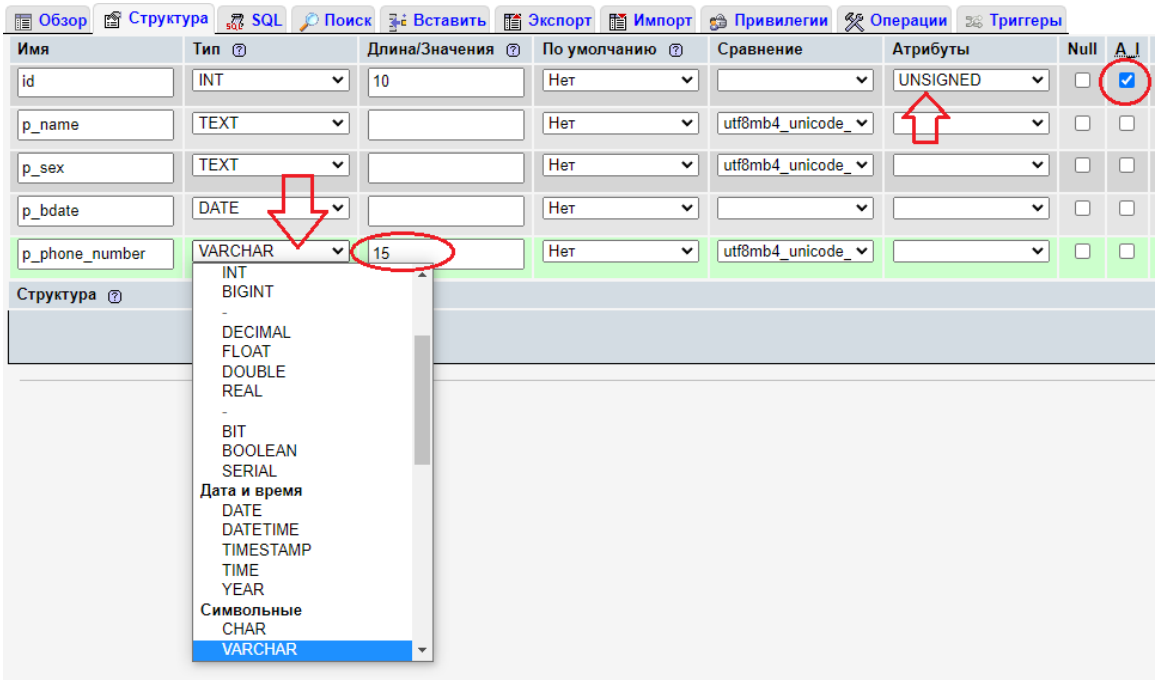


Рисунок 2.6 – Заповнення полів

Після створення таблиці можна редагувати поля або їх видаляти (рис 2.7). Для редагування доступні всі функції що і при створюванні. Можна змінювати тип даних, їх значення, атрибути та інше.

Під час додавання до таблиці нового поля можна обирати після якого поля воно повинне бути створене.

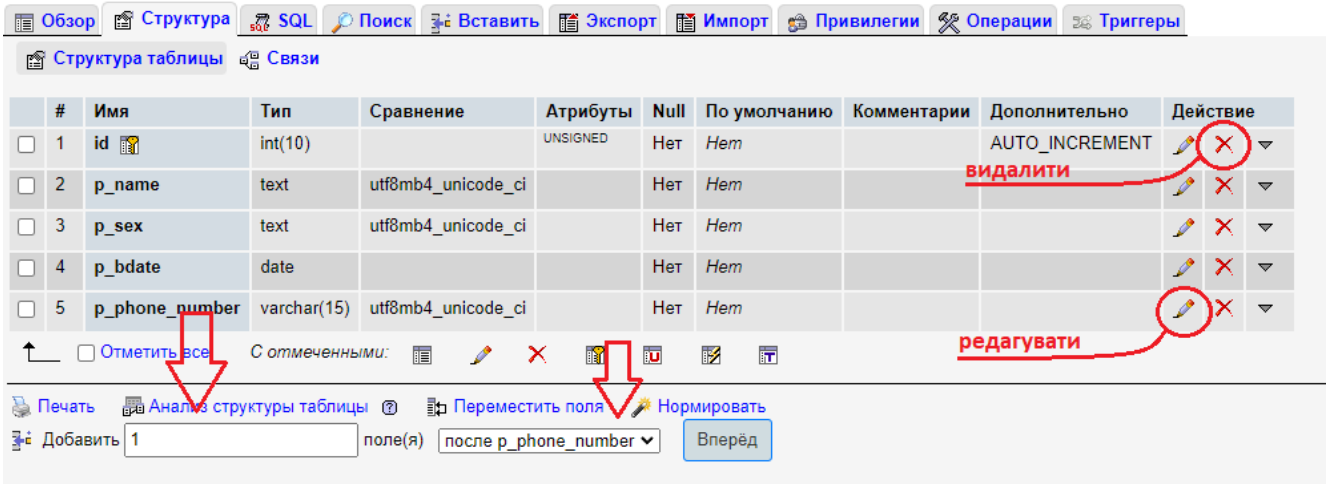


Рисунок 2.7 – Результат створеної таблиці

2.2.2 Типи даних в MySQL

Для того що рядок приймав правильні значення потрібно знати які типи існують і як правильно обрати підходящі дані для створення БД. В MySQL існує близько 30 типів даних (таб 2.1, таб 2.2., таб 2.3.). Вони потрібні для того щоб вказати який тип даних буде зберігати дане поле. Наприклад тип даних дати не дозволяє зберігати текстовий. І при такій спробі ввести текст в стовбець який може приймати лише значення з типом даних дати сервер буде показувати помилку (рис 2.8).

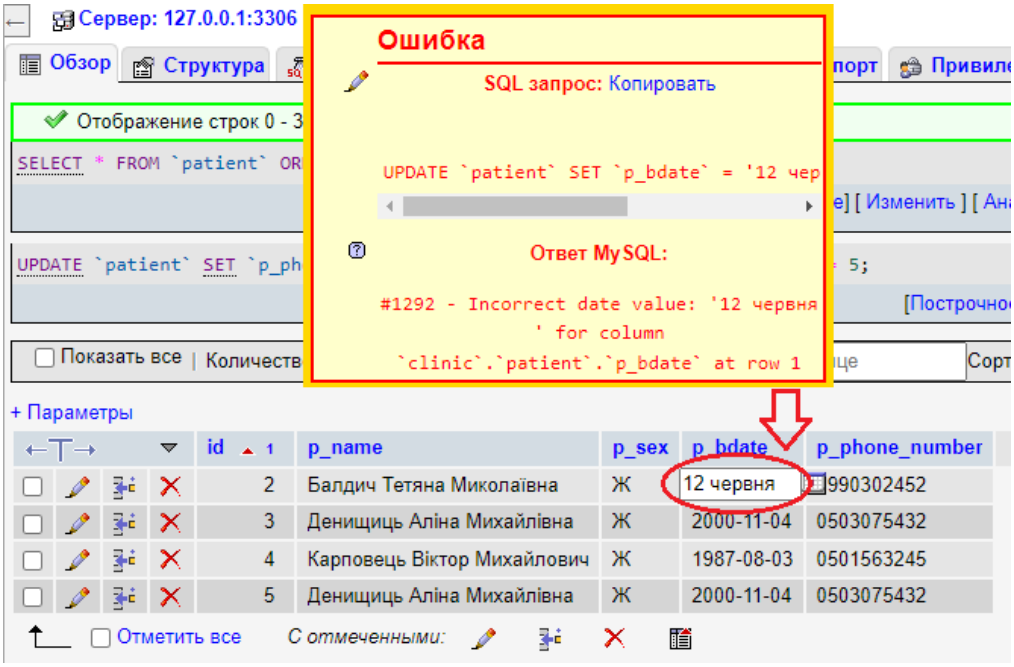


Рисунок 2.8 – Помилка при неправильному вводі типу даних

Тип даних TEXT від VARCHAR відрізняється ще тип, що у типі TEXT ми не можемо вказувати обмеженість у символах.

Таблиця 2.1 - Текстові дані [5]

Тип	Опис
CHAR()	Рядок тексту сталої довжини, заданої у дужках, до 255 символів
VARCHAR()	Рядок тексту змінної довжини до заданої у дужках, до 255 символів
TEXT	Рядок тексту до 65 535 символів

BLOB	Двійковий об'єкт до 65 535 байт даних

Продовження до таблиці 2.1

MEDIUMTEXT	Рядок тексту до 16 777 215 символів
MEDIUMBLOB	Двійковий об'єкт до 16 Мегабайт даних
LONGTEXT	Рядок тексту до 4 294 967 295 символів
LONGBLOB	Двійковий об'єкт до 4 Гігабайт даних
ENUM(x,y,…)	Список можливих значень, до 65 535 різних. Якщо значення, що вставляють у поле, не перелічене у списку, буде вставлено порожнє значення. Упорядкування — у порядку запису
SET(x,y,…)	Подібно до ENUM, але може містити до 64 значень у списку. Комірка може містити довільну підмножину множини перелічених значень

Для роботи з невід'ємними цілими числами потрібно вписати службове слово UNSIGNED після назви типу.

Максимальну кількість цифр чисел з рухомою крапкою вказують як параметр n. Максимальну кількість цифр після десяткової крапки вказують як параметр k.

Тип BOOL — синонім до TINYINT, а сталі TRUE та FALSE — до 1 та 0 відповідно.

Таблиця 2.2 - Числові дані [5]

Тип	**Опис**
TINYINT	Ціле від –128 до 127 або від 0 до 255
SMALLINT	Ціле від –32 768 до 32 767 або від 0 до 65 535
MEDIUMINT	Ціле від –8 388 608 до 8 388 607 або від 0 до 16 777 215
INT	Ціле від –2 147 483 648 до 2 147 483 647 або від 0 до 4 294 967 295
BIGINT	Ціле –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807 або від 0 до 18 446 744 073 709 551 615
FLOAT(n,k)	Число з рухомою крапкою (4 байти) з додатним модулем від $1.2 \cdot 10^{-39}$ до $3.4 \cdot 10^{38}$
DOUBLE(n,k)	Число з рухомою крапкою подвійної точності (8 байт) з додатним модулем від $2.2 \cdot 10^{-308}$ до $1.8 \cdot 10^{308}$
DECIMAL(n,k), NUMERIC(n,k) або DEC(n,k)	Число з рухомою крапкою, збережене як рядок, $(M + 2)$ байти

Типи DATETIME та TIMESTAMP повертають однакові формати дати. Але в запиті INSERT або UPDATE формат TIMESTAMP передбачає автоматичне встановлення поточного часу й дати. Також TIMESTAMP приймає різні формати: YYYYMMDDHHMMSS, YYMMDDHHMMSS, YYYYMMDD і YYMMDD.

Таблиця 2.3 – Типи дати і й часу [5]

Тип	Опис
DATE	Дата у форматі YYYY-MM-DD (3 байти). Підтримано діапазон від '1000-01-01' до '9999-12-31'
DATETIME	Формат YYYY-MM-DD HH:MM:SS (8 байт). Підтримано діапазон від '1000-01-01 00:00:00' до '9999-12-31 23:59:59'
TIMESTAMP	Кількість секунд з початку епохи Unix у форматі YYYY-MM-DD HH:MM:SS (4 байти). Підтримано діапазон від '1970-01-01 00:00:01' до '2038-01-09 03:14:07'
TIME	Час у форматі HH:MM:SS (3 байти). Підтримано діапазон від '-838:59:59' до '838:59:59'
YEAR(M)	Рік у M-цифровому форматі (M = 2, 4). Значення, дозволені в 4-цифровому форматі: від 1901 до 2155. Значення дозволені у 2-цифровому форматі: від 70 до 69, що відповідає 1970 та 2069.

Сервер для адміністрування MySQL має близько 40 типів даних на відміну він DB Browser (SQLite), який має лише 5 типів.

2.2.3 Внесення даних та їх функції редагування

Почати вносити дані в таблицю можна двома способами. Або через запит SQL, де потрібно буде прописати команду або через відповідну кнопку «Вставити» в верхньому меню (рис 2.9).

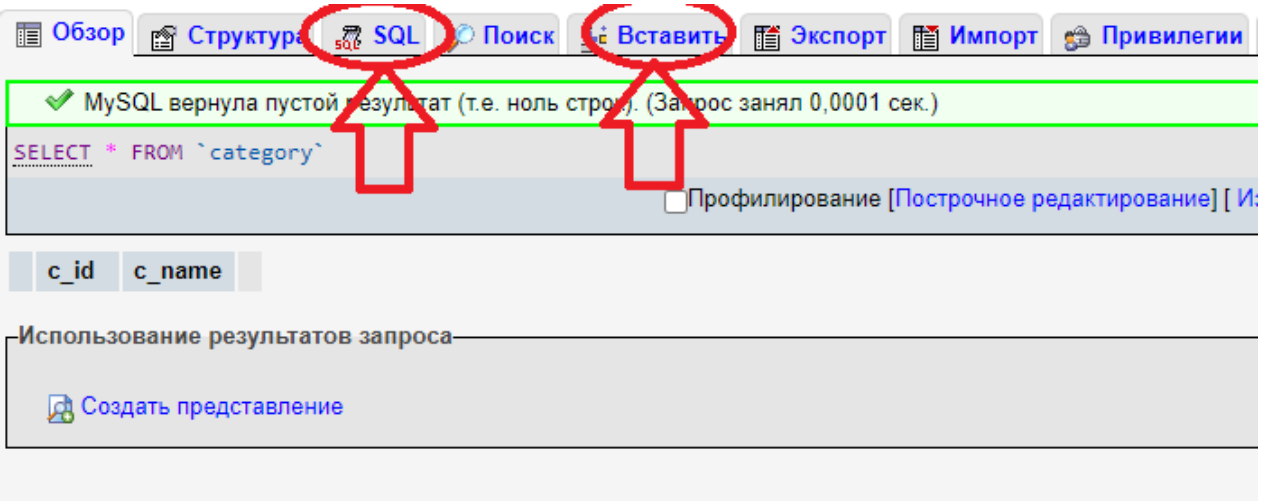


Рисунок 2.9 – Способи внесення даних

Наша таблиця потребує ПІБ клієнта, стать, його дату народження номер телефону. Після натискання пункту «SQL» в верхньому меню відкривається консоль для вводу запиту.

Додавання записів виконується командою INSERT INTO після вказується назва таблиці до якої додаються дані, в дужках вказуються поля для даних, виконується команда VALUES, в дужках та одинарних лапках вводяться через кому потрібні вам дані.

MySQL дозволяє вносити дані використовуючи подвійні лапки, але у більшості інших СБД використовуються одинарні та є універсальними, ставимо одинарні для зручності. Для зручності та пояснення запитів можна використовувати закоментовані рядки. Символ на початку /* та вкінці */ означає коментар всередині рядка, запит SQL його не бачить та не використовує. Додавати дані до таблиці можна використовуючи повторно одну команду та вносити дані для одного клієнта чи пацієнта, або ж виконати запит INSERT INTO та занести в таблицю одразу декілька рядків (рис 2.10).

Якщо в запиті використовується декілька команд, то після першої обов’язково треба ставити крапку з комою. У іншому випадку виникає помилка.

Запит SQL:

```
INSERT INTO `doctor` ( `d_name`, `d_room`, `d_experience`, `d_phone_number`) VALUES ( 'Кушнір Марина Миколаївна ', '106', '21 рік', '0912462973');
/*додавання однієї строки через команду*/
INSERT INTO `doctor` ( `d_name`, `d_room`, `d_experience`, `d_phone_number`) VALUES /*додавання одраузу декількох строк через команду*/
( 'Березюк Євгенія Василівна  ', '103', '13 років', '0634145573'),
( 'Гіленко Сергій Олександрович', '102', '3.5 років', '0966908828'),
( 'Дмітрова Євгенія Василівна ', '107', '14 років', '0947623572'),
 'Качковська Владислава Володимирівна ', '108', '9 років', '0938274868')
```

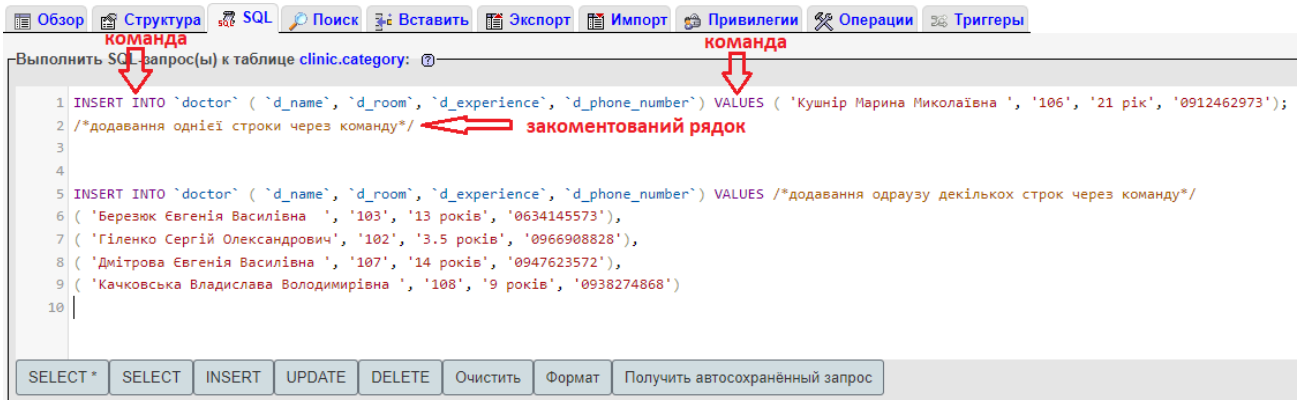


Рисунок 2.10 – Заповнення рядка через запит SQL

Другий спосіб дай змогу зробити внесення даних швидше. Після обраної кнопки меню «Вставити» відкривається нове вікно (рис2.11). Відповідні поля потрібно заповнити відповідними записами, які може приймати цей рядок. Рядок d_id заповнюється автоматично через автоінкремент, функцію яка була обрана при створювання таблиці (рис2.6). Користуючись таким способом сервер дозволяє додати не більше двох рядків. Тому якщо потрібно зробити внесення для більше чим двох рядків, потрібно використати запити SQL.

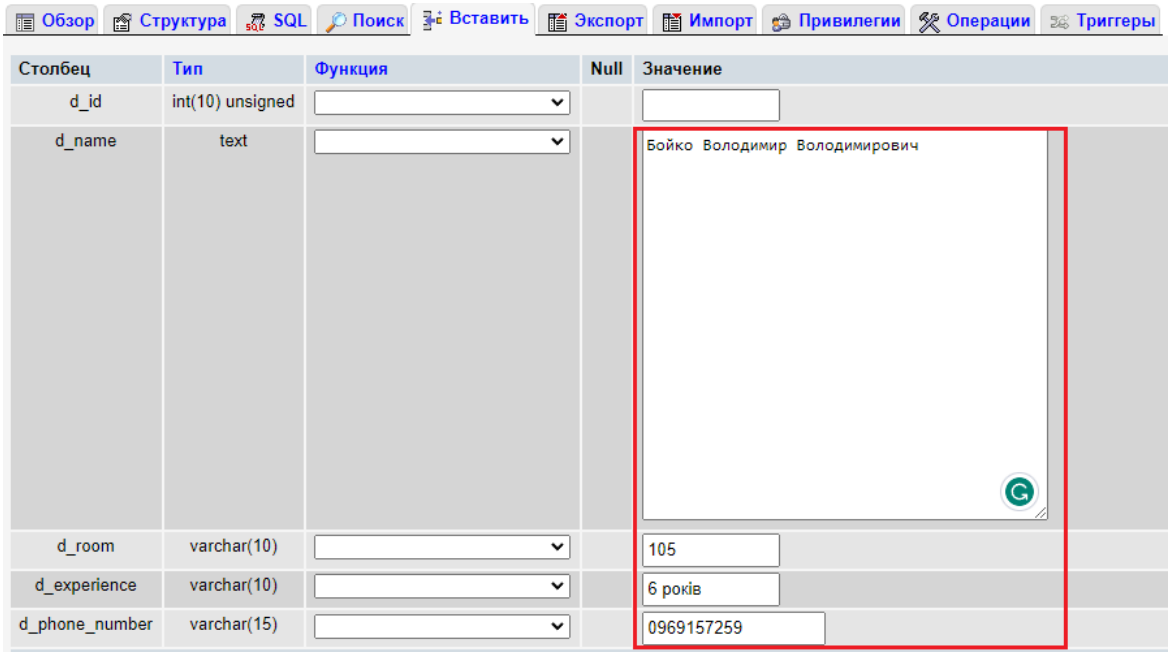


Рисунок 2.11 – Заповнення рядка через верхнє меню

Редагування рядка доступне через «олівець», який автоматично відкриває вікно з даними. Або можна редагувати не покидаючи таблицю, клацнувши два рази павою кнопкою на потрібне вікно з даними (рис2.12).

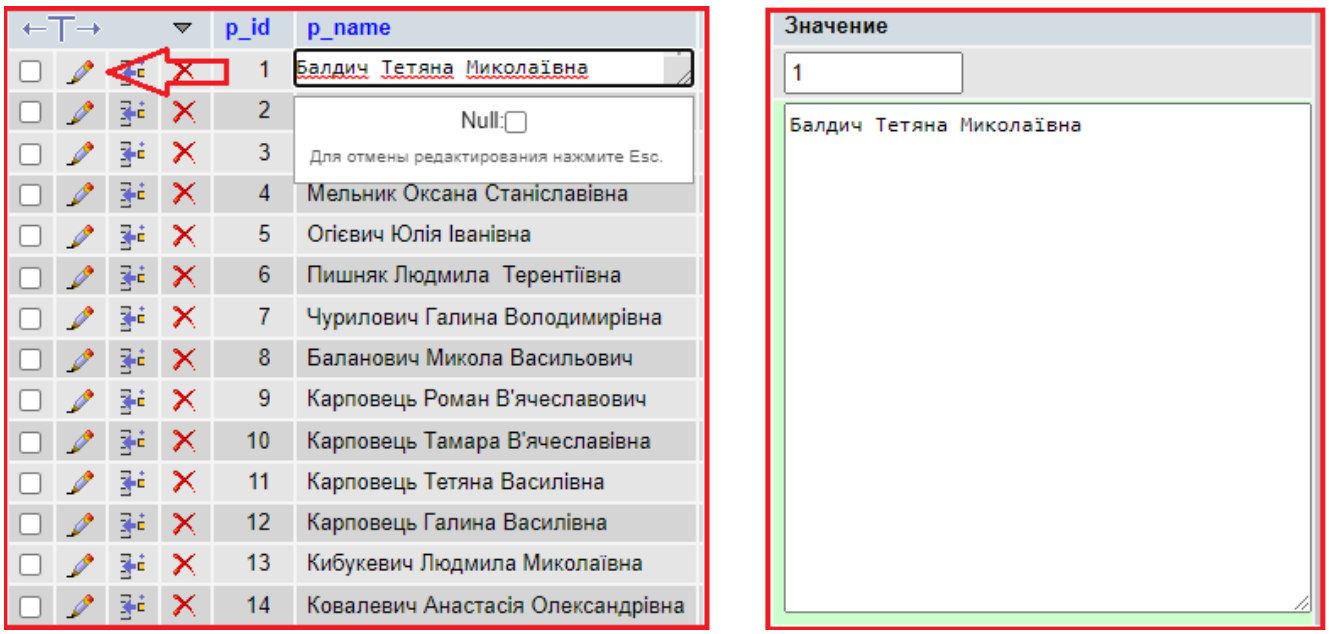


Рисунок 2.12 – Редагування рядка через «олівець або вікно з даними

2.2.4 Запити в MySQL

Інструкція SQL складається з речень. Кожне речення в інструкції SQL виконує свою функцію. Деякі речення в інструкції SELECT є обов'язковими. У таблиці нижче наведено найпоширеніші речення SQL (таб 2.4).

Таблиця 2.4 – Команди SQL

Речення SQL	**Функції**	**Обов'язкове**
SELECT	Перелічує поля, які містять потрібні дані.	Так
FROM	Перелічує таблиці, у яких містяться поля, зазначені в реченні SELECT.	Так
WHERE	Визначає умови для поля, яким мають відповідати всі записи, щоб бути включеними до результатів.	Ні
ORDER BY	Визначає спосіб сортування результатів.	Ні
ORDER BY	В інструкції SQL, яка містить агрегатні функції, перелічує поля, не зведені в реченні SELECT.	Лише за наявності таких полів
HAVING	В інструкції SQL, яка містить агрегатні функції, визначає умови, які застосовуються до полів, зведених в інструкції SELECT.	Ні

«SELECT» одна з основних команд, яку найчастіше використовують в SQL. В перекладі з англійської означає «вибрати».

Виконуємо запит для надання інформація щодо пацієнтів у яких рік народження в проміжку з 2000 року по 2004. При чому вказуємо на зміну назви стовбців для name нехай буде тепер «PATIENT», а для birth - «DATE of BIRTH». Для цього використовуємо оператор AS. Виконуємо сортування даних у спадному порядку.

Код для запиту:

SELECT p_name AS 'PATIENT', p_bdate AS 'DATE of BIRTH', p_id FROM patient

WHERE p_bdate > '2000.01.01' AND p_bdate < '2003.12.31' ORDER by p_bdate DESC

Після реалізації запиту сервер MyQSL видає таблицю, яка повністю відповідає вибірці за даним запитом (рис 2.13). Важливо правильно вказувати тип дати під час запиту, так як рядок приймає лише тип «рр-мм-дд». Якщо змінити послідовність даних під час виконання запиту, сервер поверне неправильний результат таблиці. Але на відміну від SQLite, MyQSL може зчитувати тип даних дати в незалежності від того записаний від через крапку, кому чи тире і автоматично його записувати тільки в форматі «рр-мм-дд».

	PATIENT	DATE of BIRTH	p_id
	Мертенс Олег Валерійович	2003-11-27	22
	Мельник Оксана Станіславівна	2003-10-10	4
	Ковалевич Анастасія Олександрівна	2003-08-31	14
	Мисечко Юлія Анатоліївна	2003-02-27	17
	Карповець Галина Василівна	2002-03-03	12
	Денищиць Аліна Михайлівна	2000-11-04	2

Рисунок 2.13 – Запит SQL для серверу MyQSL

2.2.5 Тригери

У SQL тригери — це спеціальні типи збережених процедур, які автоматично виконуються у відповідь на певні події або дії, що відбуваються в таблиці. Тригери пов’язані з таблицею та запускаються операторами мови обробки даних (DML), такими як INSERT, UPDATE або DELETE. Коли відбувається зазначена подія, код тригера виконується, що дозволяє виконувати додаткові дії або перевірки.

Тригери можна класифікувати на два основні типи залежно від часу їх виконання:

Тригери Before: ці тригери виконуються до того, як у таблиці відбудеться подія запуску (наприклад, INSERT, UPDATE або DELETE). Вони зазвичай використовуються для перевірки або зміни даних перед їх записом у таблицю.

Якщо тригер до викликає помилку або виконує відкат, інструкція запуску скасовується.

Тригери After: ці тригери виконуються після того, як відбулася подія ініціювання та дані були записані в таблицю. Вони зазвичай використовуються для дій, які повинні виконуватися після операції маніпулювання даними, наприклад, журналювання або оновлення інших таблиць. Тригери After не можуть змінювати дані, які їх ініціювали.

Тригери можна додатково класифікувати залежно від кількості разів, коли вони виконуються. Існують тригери, які виконуються один раз для кожного рядка, на який впливає подія запуску. Наприклад, якщо інструкція UPDATE змінює кілька рядків, тригер на рівні рядка, пов’язаний із цією таблицею, буде виконано для кожного постраждалого рядка. А тригери на рівні оператора виконуються лише один раз для кожного оператора запуску, незалежно від кількості рядків, на які впливає оператор. Наприклад, якщо оператор INSERT вставляє кілька рядків, тригер на рівні оператора, пов’язаний із цією таблицею, буде виконано лише один раз для всього оператора. Для видалення тригера виконується функція DROP TRIGGER ‘назва тригеру’.

Для виявлення різниці синтаксису між двома СУБД мною був створений однаковий тригер за своєю функцією.

Код для створення тригеру для php my admin for MySQL:

```
DELIMITER $$
CREATE TRIGGER wrong_p_phone_number BEFORE INSERT ON patient
FOR EACH ROW
BEGIN
    IF NEW.p_phone_number NOT LIKE '0_____' THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'невірно набраний номер';
    END IF;
END $$
```

DELIMITER ;

У даному коді описана функція при якій під час додавання нового рядку в таблицю, заповнювати дані номеру телефону дозволятиме лише починаючи з 0. Також даний тип може містити лише 10 обов'язкових символів. Під час помилкового вводу номеру телефону менше або більше 10 символів спрацьовує тригер, який видає помилку « невірно набраний номер». Тригер також працює якщо перший символ номеру телефону не 0 (рис2.14).

Код для внесення нового рядка з некоректним номером телефону:

INSERT INTO patient (p_id, p_name, p_sex, p_bdate, p_phone_number)

VALUES ('27','Михно Степан Миколайович', 'Ч' , ' 2000-02-12', '9990100388')

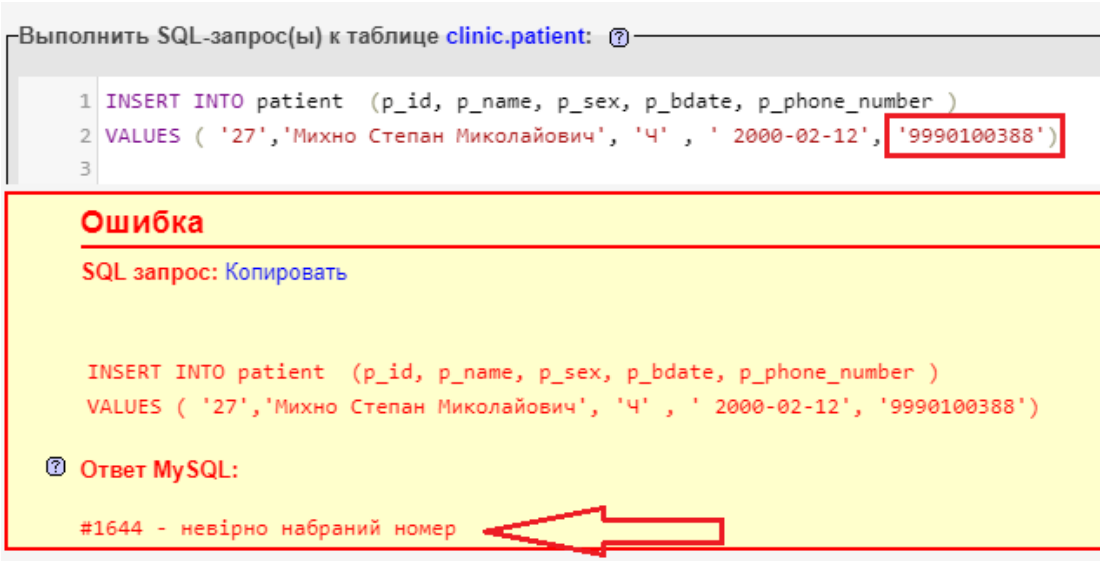


Рисунок 2.14 – Спрацювання тригеру при помилці вводу номеру

Також php my admin for MySQL у своєму головному меню окремо має вікно для створення та перегляду вже всіх раніше створених тригерів.

2.2.5 Швидкість обробки в MySQL

На швидкість обробки даних в MySQL впливають різні фактори.

Продуктивність основного обладнання відіграє значну роль. Це включає такі фактори, як процесор (обчислювальна потужність), пам’ять (RAM), дискове сховище і підключення до мережі.

Правильна індексація теж має значення для ефективної обробки даних у MySQL. Створюючи індекси для стовпців, які часто використовуються в умовах пошуку та операціях об’єднання, можна прискорити виконання запиту. Добре розроблені індекси зменшують потребу у скануванні великих обсягів даних, що забезпечує швидший пошук даних. Об'єм структури таблиці включаючи типи даних і зв’язки між таблицями, може впливати на швидкість обробки даних. Зі збільшенням обсягу даних такі операції, як пошук даних, сортування та агрегація, можуть тривати довше. Правильні стратегії розподілу даних, архівування чи очищення можуть допомогти пом’якшити вплив великих обсягів даних.

Параметри, пов’язані з розподілом пам’яті, розміром буфера, параметрами кешу та паралельністю потоків, рекомендовано налаштувати для оптимізації швидкості обробки даних на основі конкретного робочого навантаження та доступних ресурсів.

Важливо зазначити, що ці фактори взаємопов’язані, і оптимізація одного аспекту може мати наслідки для іншого. Тому для досягнення найкращої швидкості обробки даних у MySQL, як правило, потрібен цілісний підхід до налаштування продуктивності з урахуванням усіх відповідних факторів.

В залежності від об’єму обробки даних та складності запиту швидкість обробки запитів та даних для MySQL буде різною, але незважаючи на це запити обробляються дуже швидко. Зазвичай час для видачі результату становить від 0,0002 до 0,1 секунди (рис 2.15).

Рисунок 2.15 – Швидкість обробки даних в MySQL

2.3Створення БД для СУБД Sqlite

Одна з популярних графічних клієнтів для SQLite є програма DB Browser for SQLite. DB Browser — це безкоштовний графічний інтерфейс користувача з відкритим кодом для керування базами даних SQLite. Вона доступна для скачування на офіційному сайті. [6] Для правильної установки програми потрібно обрати підходящий пакет. Якщо у вас наприклад операційна система - Windows 64x, то обирайте інсталятор програми для 64-бітної Windows. Після встановлення запустіть програму.

Інтерфейс DB Browser дуже простий. За допомогою цієї програми можемо користуватися не тільки графічними можливостями але й запитами SQL. Коли ви запускаєте DB Browser, перше, що ви побачите, — це екран-заставка, який містить інформацію про версію програмного забезпечення, яке ви використовуєте, а також посилання на веб-сайт проекту та посібник користувача. Коли ви закриваєте екран-заставку, ви побачите систему головного меню браузера БД (рис 2.16). Система меню складається з кількох розділів, кожен із яких містить набір пов’язаних команд.

Основні розділи системи меню:

Файл: містить команди для створення, відкриття, збереження та друку файлів бази даних. Він також містить параметри для імпорту та експорту даних у та з різних форматів файлів.

Редагувати: цей розділ містить команди для редагування вмісту бази даних, включаючи додавання та видалення таблиць, полів і записів. Він також містить параметри для пошуку та заміни тексту в базі даних.

Перегляд: містить параметри для налаштування зовнішнього вигляду інтерфейсу браузера БД, наприклад змінення розміру шрифту або приховання панелі інструментів.

База даних: включає команди для керування структурою бази даних, включаючи створення та зміну таблиць та індексів. Він також містить параметри для виконання запитів SQL і керування підключеннями до бази даних.

Інструменти: цей розділ містить різні інструменти для роботи з базою даних, наприклад можливість обчислювати статистичні дані щодо даних або генерувати вибіркові дані.

Довідка: приховує посилання на посібник користувача та довідкові ресурси в Інтернеті, а також інформацію про версію програмного забезпечення та ліцензію.[7]

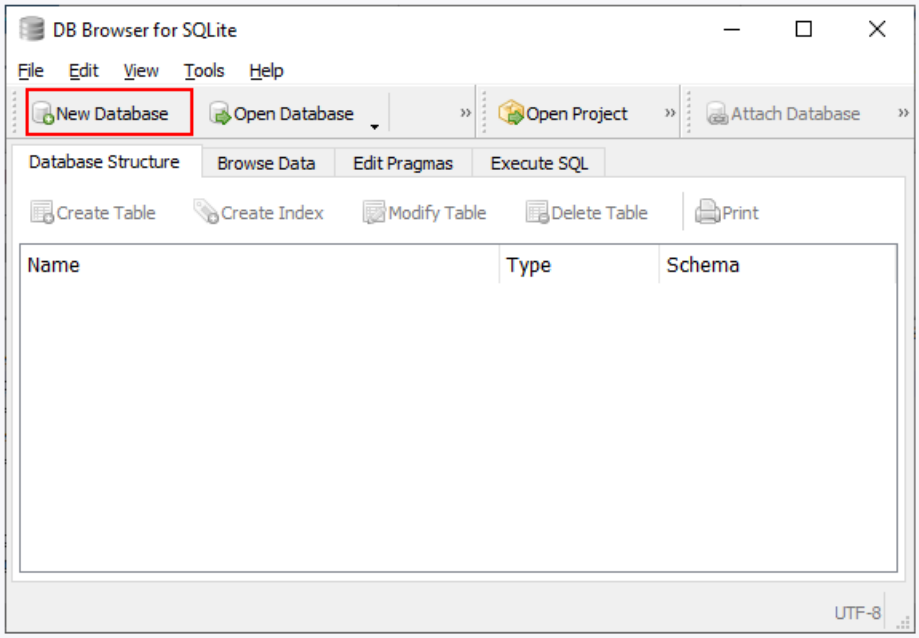


Рисунок 2.16 – Інтерфейс DB Browser for SQLite

Для того щоб створити нову базу даних обираємо вікно New Database, де програма одразу пропонує нам її місце збереження та назву файлу. Згодом ви зможете створити ще одну базу даних або ж відкрити за допомогою кнопки Open Database раніше вже збережену БД.

2.3.1 Таблиці та їх редагування для DB Browser for SQLite

Під час створення нової БД програма одразу пропонує нам обрати назву самої БД та таблиці для неї (рис2.17).

У панелі ‘Тип’ обираємо тип даних. Вони потрібні для того щоб вказати який тип даних буде зберігати дане поле. Вказуємо яке поле повинне мати унікальний ключ. Обов’язково обираємо NOT NULL для тих стовбців яким це потрібно. Це означає, що поле не може приймати нульових значень і завжди повинне містити значення.

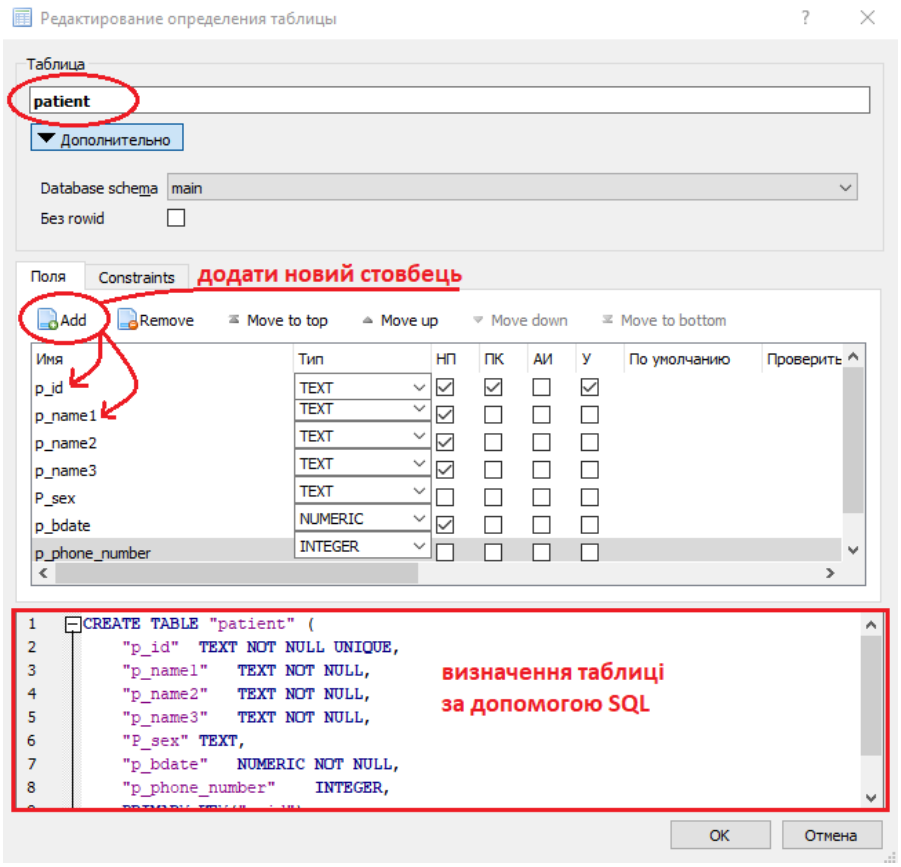


Рисунок 2.17 – Створення таблиці в DB Browser

Після створення таблиці її можна також редагувати, змінювати тип даних, які приймаються, видаляти та додавати нові поля, перейменовувати все існуючі поля (рис 2.18). Для цього потрібно обрати кнопку Модифікувати Таблицю, де з’явиться нове вікно з усіма полями цієї таблиці.

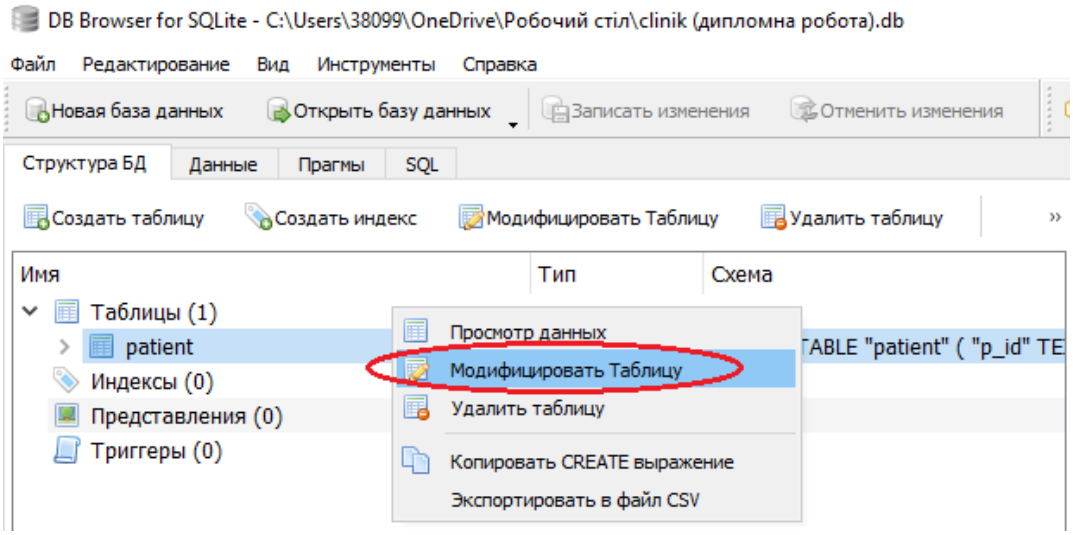


Рисунок 2.18 – Редагування таблиці

За таким же принципом створюємо інші 6 таблиці для бази даних.

Результат повноцінної бази даних наведений на рисунку 2.19. Початково DB Browser має системну таблицю даних. У даному випаду це sqlite_sequence (рис 2.19). Вона не підлягає редагуванню та видаленню. Вона потрібна для реалізації автоінкремента, яке автоматично генерує послідовні числа і записує числове значення при додаванні нового запису.

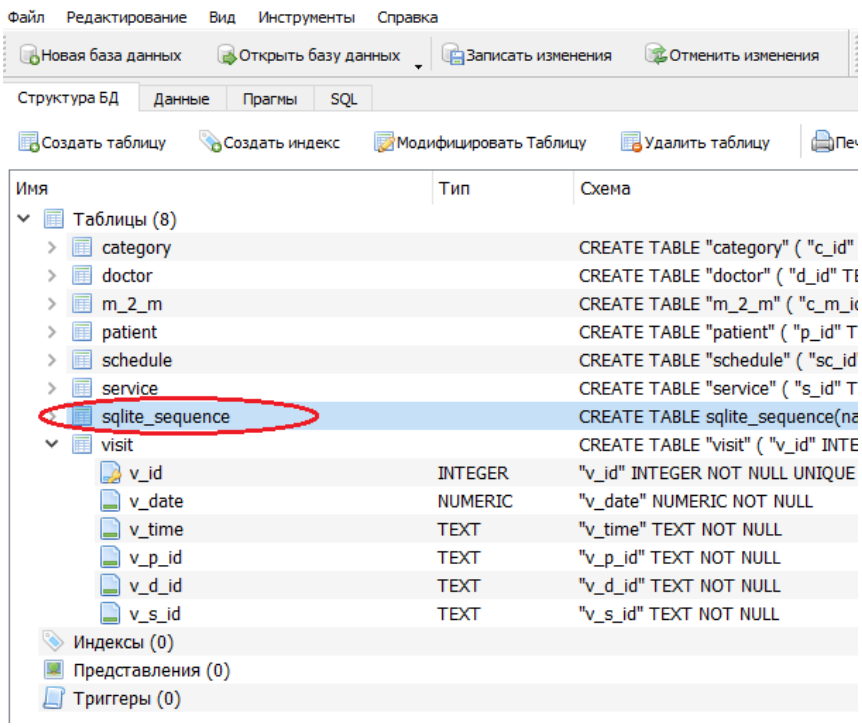


Рисунок 2.19 – Результат створеної БД для DB Browser for SQLite

2.3.2 Типи даних для DB Browser for SQLite

На вдміну від php my admin for MySQL DB Browser має лише 5 типів даних (таб 2.5). Також як і php my admin він має значення NULL, що означає відсутнє або невідоме значення.

Важливо зазначити, що SQLite використовує динамічну типізацію, тобто тип даних стовпця явно не визначено. Натомість SQLite призначає спорідненість кожному стовпцю на основі значення, яке зберігається в ньому. Спорідненість впливає на те, як дані зберігаються та як вони обробляються під час операцій. Детально на прикладі описано у розділі 2.3.4

Таблиця 2.5 – Типи даних в DB Browser for SQLite

Тип	Опис
INTEGER	Представляє ціле число, яке може бути позитивним та негативним і в залежності від свого значення може займати 1, 2, 3, 4, 6 або 8 байт
REAL	Представляє число з точкою, що плаває, займає 8 байт в пам'яті

TEXT	Зберігає текстові дані, наприклад рядки символів. Може приймати як цифрові так і текстові символи, спеціальні символи і навіть двійкові дані.
BLOB	Бінарні дані. Може зберігати зображення, документи або мультимедійні дані.
NUMERIC	Представляє числові дані, включаючи цілі числа, десяткові.

Простими словами усі 5 типів даних скоріше можна назвати класами зберігання аніж типами. Оскільки концепція класів набагато ширша, чим тип даних. Наприклад ідентифікатор NUMERIC може зберігати в собі одразу всі 5 типів класу. І при заповненні рядка текстовими даними з ідентифікатором NUMERIC програма не видаватиме помилки (рис 2.20).

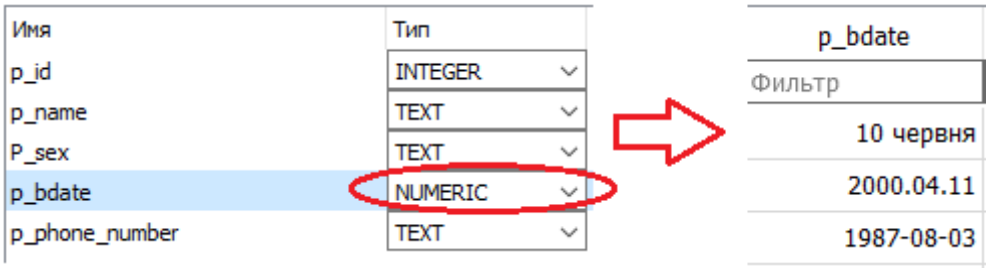


Рисунок 2.20 – Тип даних NUMERIC

2.3.3 Внесення даних та їх функції редагування

Існує два способи внесення нових записів до таблиці. Перший дозволяє зробити це через додаткове вікно меню для запиту SQL, прописавши код для додавання даних у таблицю. Другий створює додаткове пусте поле для заповнення в кінці таблиці через кнопку меню «Вставити рядок» (рис 2.21).

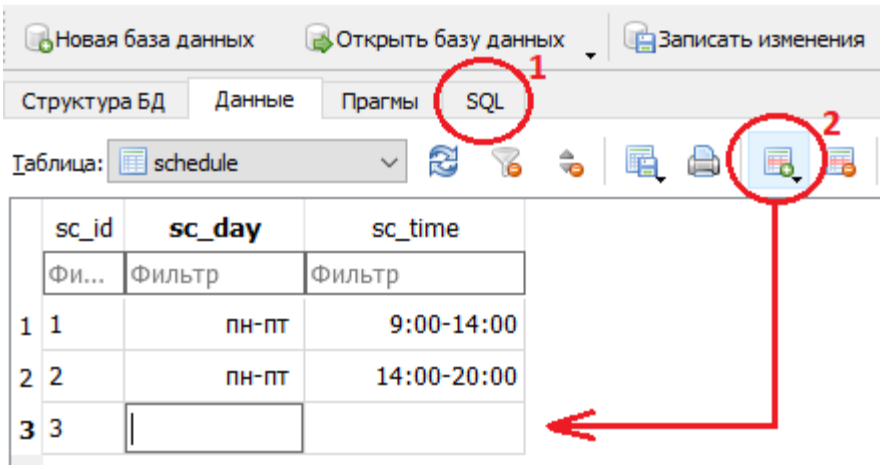


Рисунок 2.21 – Додавання записів. 1-через запит SQL, 2 – через меню «Вставити рядок»

Внесення записів за запитом SQL можна зробити за тим самим кодом, що і для php my admin for MySQL, описаного в підрозділі 2.2.3 (рис2.22). Оскільки команди для обох СУБД майже не відрізняються, не враховуючи деякі особисті вимоги до розділових знаків у обох СУБД.

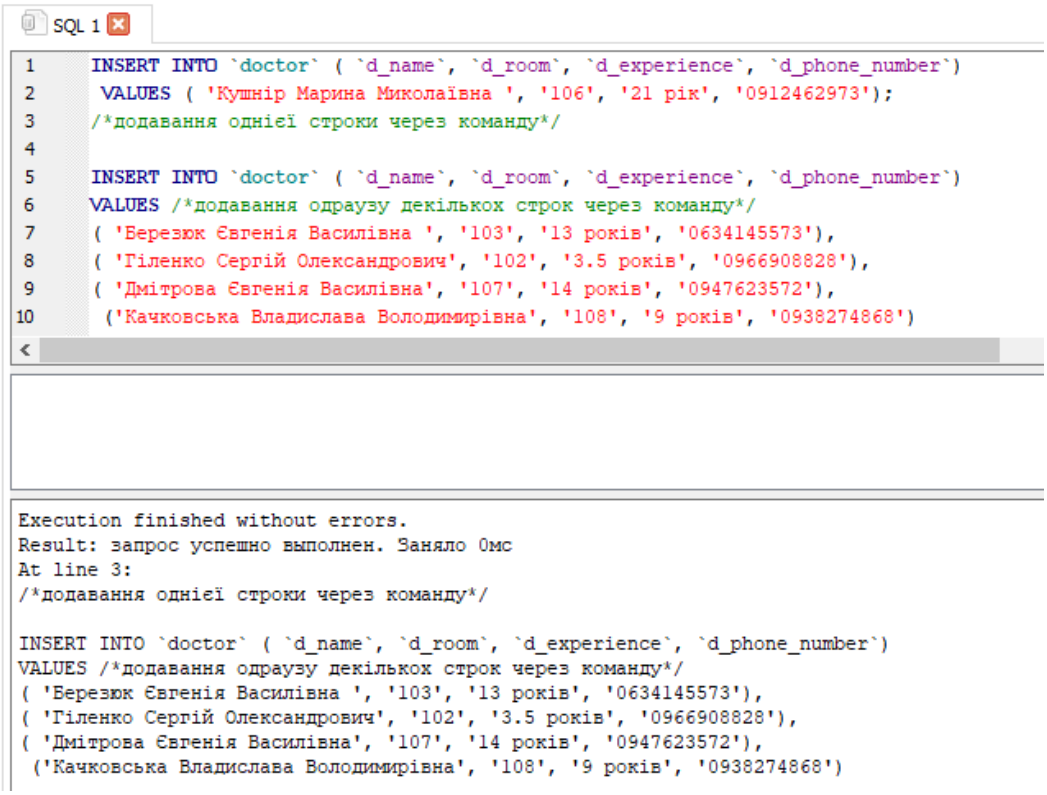


Рисунок 2.22 – Код лдя запиту на додавання нових записів

Результат запиту показаний на рисунку 2.23.

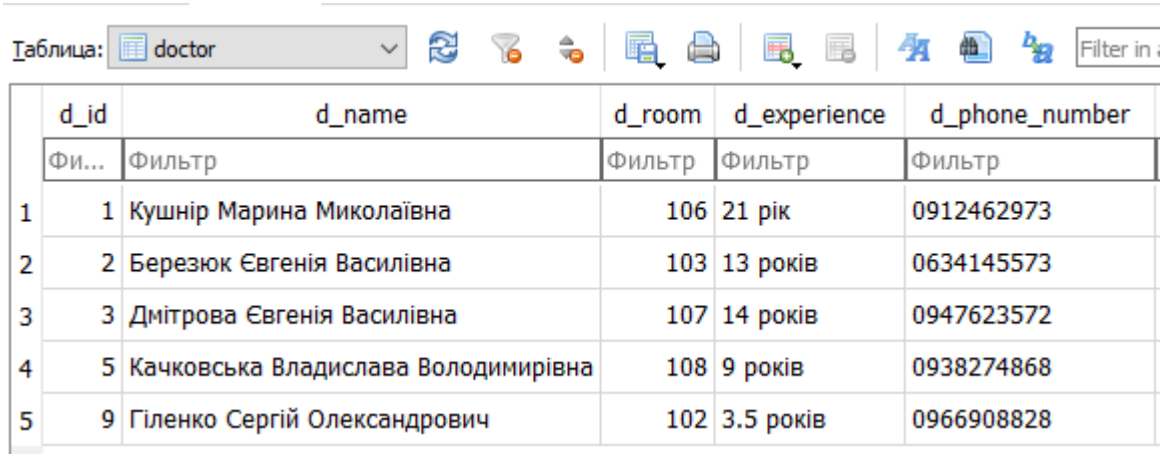


Таблица: doctor

	d_id	d_name	d_room	d_experience	d_phone_number
	Фи...	Фильтр	Фильтр	Фильтр	Фильтр
1	1	Кушнір Марина Миколаївна	106	21 рік	0912462973
2	2	Березюк Євгенія Василівна	103	13 років	0634145573
3	3	Дмітрова Євгенія Василівна	107	14 років	0947623572
4	5	Качковська Владислава Володимирівна	108	9 років	0938274868
5	9	Гіленко Сергій Олександрович	102	3.5 років	0966908828

Рисунок 2.23 – Результат заповненої таблиці

Редагування записів здійснюється одразу у таблиці, при цьому відкривається додаткове вікно «Редагування комірки» для зручності, якщо запис має багато символів через що їх може бути повністю не видно у самій таблиці (рис 2.24).

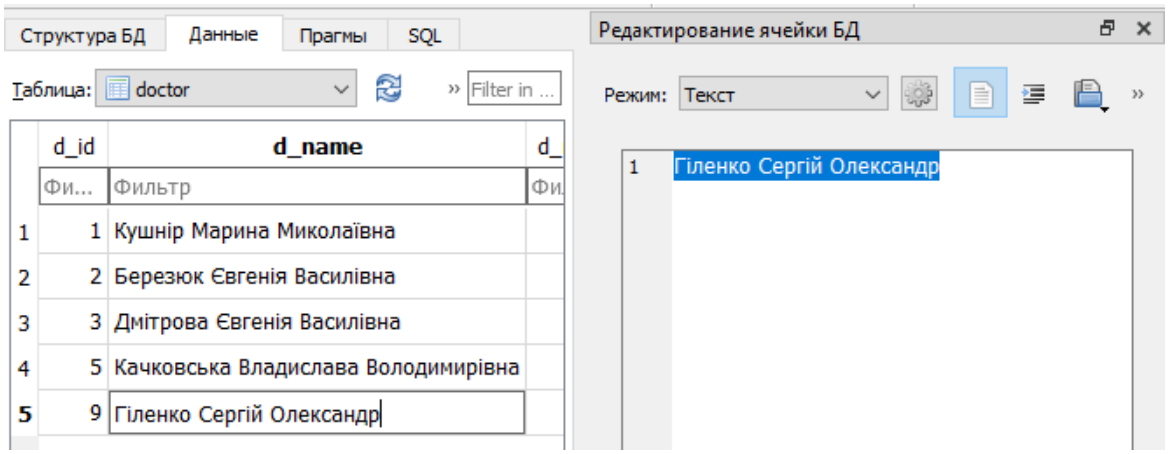


Рисунок 2.24 – Редагування комірки

2.3.4 Запити SQL в DB Browser for SQLite

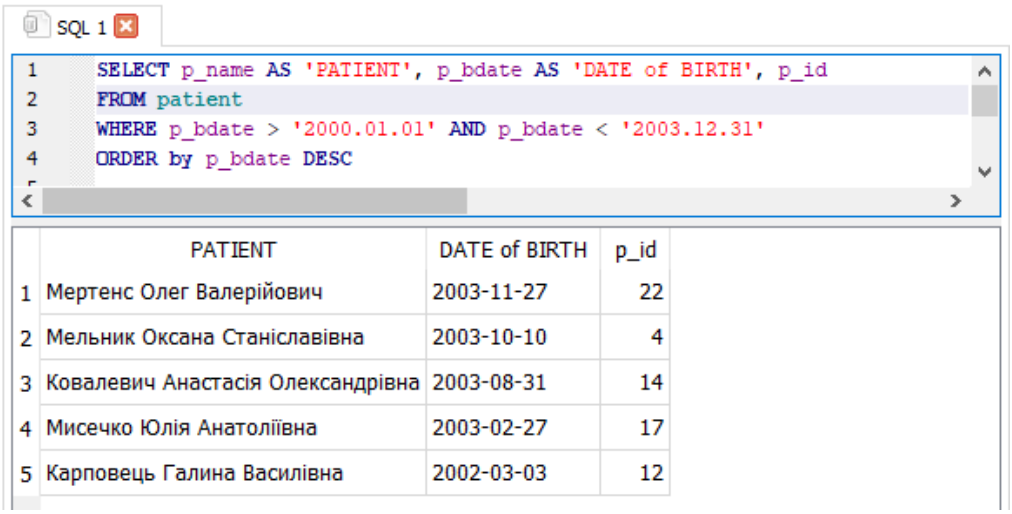
У DB Browser для SQLite запити SQL виконуються через вбудований редактор (рис 2.21, рис 2.22).

Слід зазначити, що DB Browser for SQLite не має певних типів даних для зберігання дат і часу.

Однак ви можете використовувати тип TEXT або INTEGER, щоб зберігати значення дати й часу як рядки або мітки часу. Саме через відсутність такого тип даних при деяких запитах SQL DB Browser може видавати неточну відповідь.

Оскільки він може зчитувати дату як дату тільки у форматі «рр.мм.дд». Для уникнення такої помилки слід бути уважним при розділових знаках під час запису дати.

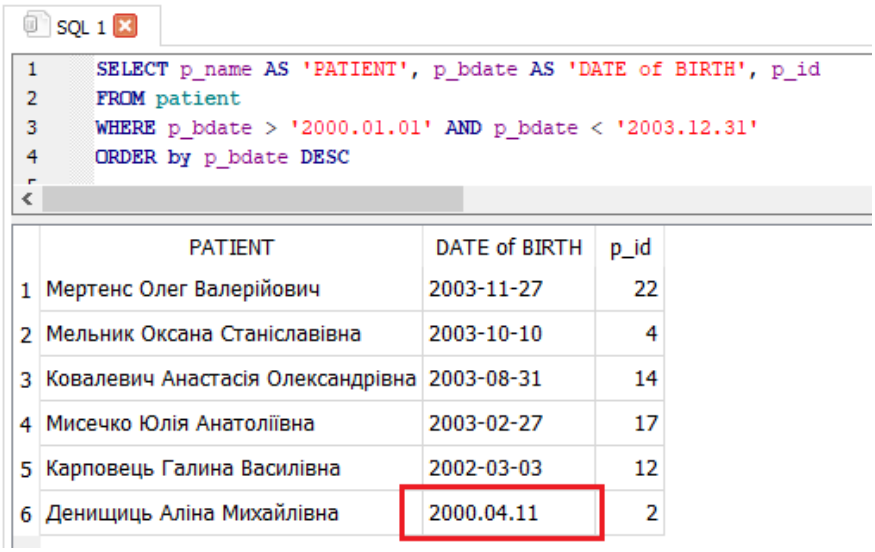
Так як дані в таблиці «patient» записані в однакому в обох СУБД при однаковому запиті SQL DB Browser видає таблицю не враховуючи запис з id – 2. Оскільки DB Browser аналізує запис типу дати «рр.мм.дд.» до першого розділового знаку. Для порівняння дивитися рисунки 2.13 і 2.25.



	PATIENT	DATE of BIRTH	p_id
1	Мертенс Олег Валерійович	2003-11-27	22
2	Мельник Оксана Станіславівна	2003-10-10	4
3	Ковалевич Анастасія Олександрівна	2003-08-31	14
4	Мисечко Юлія Анатоліївна	2003-02-27	17
5	Карповець Галина Василівна	2002-03-03	12

Рисунок 2.25 – Незадовільний результат запиту коду

Для того, щоб результат був задовільний потрібно враховувати розділові знаки. Якщо записувати дані дати через крапку система DB Browser правильно зчитає дані (рис 2.26). На відміну від php my admin for MySQL, DB Browser є більш вимогливий в коректності запису даних.



	PATIENT	DATE of BIRTH	p_id
1	Мертенс Олег Валерійович	2003-11-27	22
2	Мельник Оксана Станіславівна	2003-10-10	4
3	Ковалевич Анастасія Олександрівна	2003-08-31	14
4	Мисечко Юлія Анатоліївна	2003-02-27	17
5	Карповець Галина Василівна	2002-03-03	12
6	Денищиць Аліна Михайлівна	2000.04.11	2

Рисунок 2.26 - Задовільний результат запиту коду

2.3.5 Тригери

Тригери для DB Browser for SQLite працюють за таким же принципом, що і для php my admin. Більш детально описано у підрозділі 2.2.5.

Раніше створений тригер для php my admin for MySQL, який не дає можливості вносити дані номеру телефону число якого не починається з 0 або має більше чи менше 10 символів, при цьому викликає помилку з текстом «невірно набраний номер», не може працювати за тим самим кодом для DB Browser for SQLite.

Синтаксис написання коду для тригерів має деякі відмінності у командах.

Код для створення тригеру для DB Browser:

```
CREATE TRIGGER wrong_p_phone_number
        BEFORE INSERT on patient
BEGIN
        SELECT
                CASE
                WHEN NEW.p_phone_number NOT like '0_________' THEN
                RAISE (ABORT, 'невірно набраний номер')
```

END;

END;

Якшо аналізувати код для DB Browser в порівнянні з кодом у підрозділі 2.2.5 для php my admin, можна зробити висновок, що команди RAISE не існує взагалі для php my admin. Тому переробивши код за правилами написання для DB Browser отримуємо результат за запитом створення нового рядка (рис 2.27).

Код для внесення нового рядка з некоректним номером телефону:

INSERT INTO patient (p_id, p_name, p_sex, p_bdate, p_phone_number)

VALUES ('27','Михно Сепан Миколайович', 'Ч' , ' 2000-02-12', '99901003884')

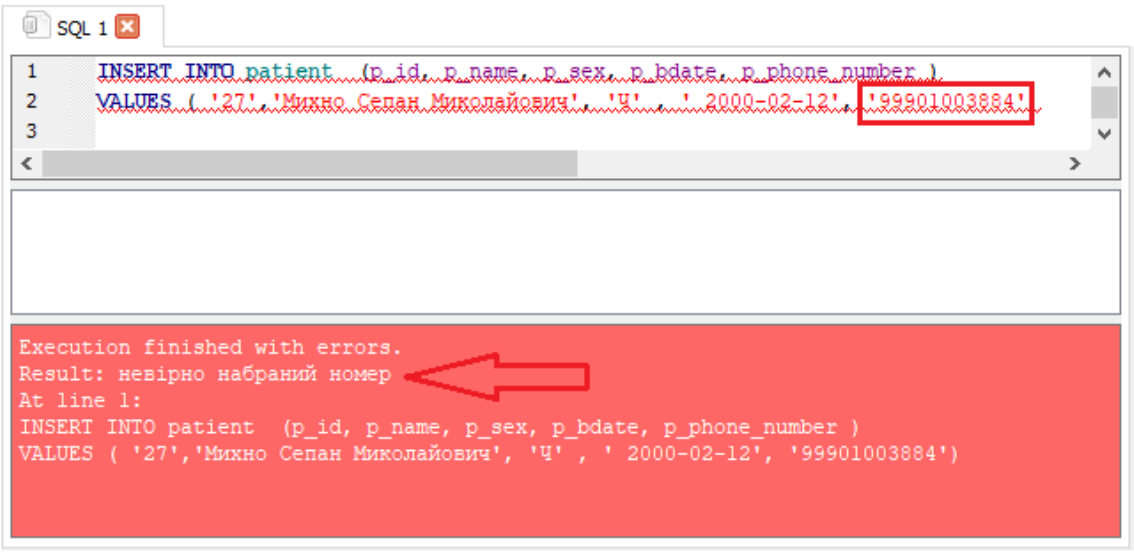


Рисунок 2.27 - Спрацювання тригеру при помилці вводу номеру

2.3.6 Швидкість обробки даних

Швидкість обробки даних в DB Browser for SQLite залежить також від апаратного забезпечення, об’єму структури даних та обсягу даних в таблицях, запитах які виконуються та об’єму даних які обробляються під час запитів, правильна індексація таблиць. Детальніше описано в підрозділі 2.2.6.

Приблизна швидкість обробки становить від 0.0001с до 0.012с (рис 2.27)

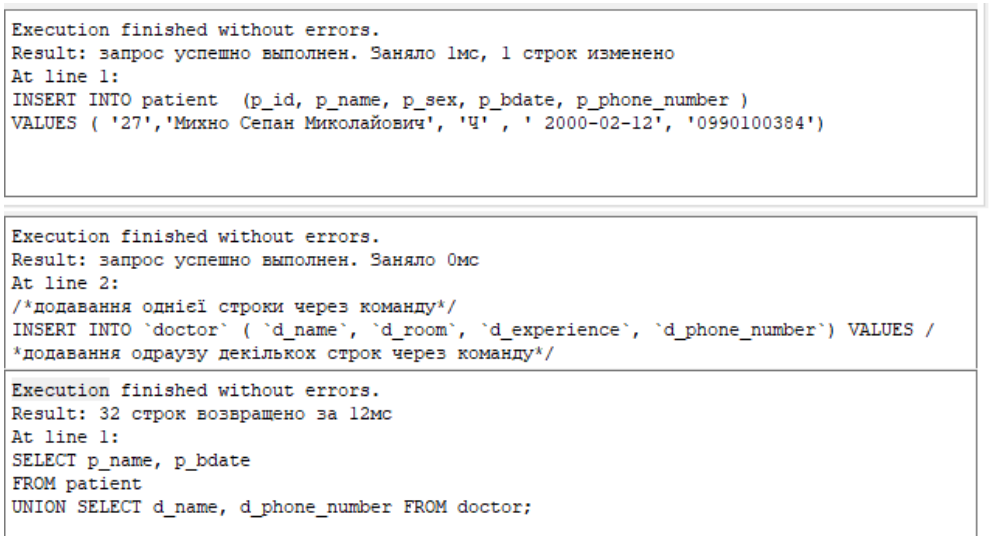


Рисунок 2.27 – Швидкість оброки даних для DB Browser for SQLite

В порівнянні php my admin for MySQL, DB Browser робить обробку швидше, тому що не потребує окремого підключення до серверу та не залежить від підключення до мережі інтернет.

ВИСНОВКИ

Дана робота присвячена одній з найбільш актуальних тем на сьогодні – база даних. Оскільки сфера систем управління базами даних продовжує розвиватися, професіонал у цій галузі повинен бути в курсі останніх тенденцій і технологій. Це включає розуміння відмінностей між різними СУБД, а також переваг і обмежень кожної з них.

Головною метою роботи було дослідження MySQL та Sqlite, для знаходження їх особистих відмінностей під час реалізації бази даних для клініки.

Підсумувавши можна сказати, що база даних є ефективним механізмом упорядкування даних ніж електронні таблиці, вона дає змогу швидко зробити пошук та використовувати її для декількох користувачів. Мною були розглянуті зовсім два різних інтерфейса. Користуючись налаштуваннями MySQL та Sqlite можна зробити висновок, що інтерфейс Sqlite є простішим, що дає змогу швидше створювати базу даних, але кількістю своїх налаштувань та можливостей він поступається MySQL.

Під час створення запитів, я дійшла висновку, що синтаксис для MySQL та Sqlite не має конкретних відмінностей, окрім того що DB Browser for SQLite є більш

вимогливим до коректності внесення даних , особливо пунктуації. Оскільки він має набагато менший перелік можливих типів даних ніж php my admin for MySQL.

Для реалізації тригерів, я окремо ознайомилась з командами для кожної СУБД. Хоча синтаксис створення тригерів подібний у MySQL і SQLite, різниця полягає у функціональності, де MySQL надає більше можливостей, а SQLite має більш обмежені можливості. Також php my admin for MySQL дає змогу створювати тригери у окрему вікні і переглядати вже існуючі тригери.

Аналізуючи швидкість обробки даних я зробила висновки, що DB Browser for SQLite швидше оновлює дані, завдяки тому, що не потребує підключення до серверу та інтернет мережі, але загалом швидкість обробки для MySQL та SQLite не має дуже значних відмінностей.

Загалом php my admin for MySQL є більше зручним та функціональним для обробки великих об’ємів даних ніж DB Browser. Тим не менш DB Browser є простішим для початкового вивчення та опанування навичок SQL.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Types of Databases [Електронний ресурс] // Learntek. – 2018. – Режим доступу до ресурсу: https://www.learntek.org/blog/types-of-databases/
2. Data Models [Електронний ресурс] // JavaTPoint – Режим доступу до ресурсу: https://www.javatpoint.com/data-models.
3. What is an Entity Relationship Diagram (ERD)? [Електронний ресурс] // Lucidchart – Режим доступу до ресурсу: https://www.lucidchart.com/pages/er-diagrams.
4. https://www.drawio.com/ - офіційний сайт Draw.io
5. Типи даних MySQL [Електронний ресурс] – Режим доступу до ресурсу: http://www.kievoit.ippo.kubg.edu.ua/kievoit/2016/58_SQL/types.html.
6. DB Browser for SQLite [Електронний ресурс] – Режим доступу до ресурсу: https://sqlitebrowser.org/.
7. Using DB Browser for SQLite [Електронний ресурс] // Data Management with SQL for Social Scientists *alpha*. – 2023. – Режим доступу до ресурсу: https://datacarpentry.org/sql-socialsci/02-db-browser.