Міністерство освіти і науки України

Тернопільський національний технічний університет імені Івана Пулюя

Прикладних інформаційних технологій та електроінженерії

(повна назва факультету)

Комп'ютерно-інтегрованих технологій

(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр

(назва освітнього ступеня)

на тему: Розроблення та дослідження програмного комплексу для оптимізації системи зберігання, обробки та аналізу баз даних IP-адрес.

Виконав(ла): студент(ка) 6 курсу, групи КТм-61

спеціальності 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка

(шифр і назва спеціальності)

		Третяк С. В.
	(підпис)	(прізвище та ініціали)
Керівник		к.т.н. Чихіра І. В.
	(підпис)	(прізвище та ініціали)
Нормоконтроль		к.т.н. Чихіра І.В.
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		к.т.н. Голотенко О. С.
	(підпис)	(прізвище та ініціали)
Рецензент		к.т.н. Медвідь В.Р.
	(підпис)	(прізвище та ініціали)

Тернопіль

2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет прикладних інформаційних технологій та електроінженерії
(повна назва факультету)

Кафедра комп'ютерно-інтегрованих технологій
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Голотенко О.С.
(підпис) (прізвище та ініціали)
« » 20___ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр
(назва освітнього ступеня)

за спеціальністю 174 – «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»
(шифр і назва спеціальності)

студенту Третяк Сергій Вікторович
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення та дослідження програмного комплексу для оптимізації системи зберігання, обробки та аналізу баз даних IP-адрес

Керівник роботи Чихіра Ігор Вікторович к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*12*» *11* 2025 року № *4/7-962*

2. Термін подання студентом завершеної роботи 23.12.2025

3. Вихідні дані до роботи технічна документація

4. Зміст роботи (перелік питань, які потрібно розробити)
1. Аналітична частина 2. Технологічна частина 3. Конструкторська частина 4. Науково-дослідницька частина 5.Спеціальна частина 6.Охорона праці та безпека в надзвичайних ситуаціях. Основні висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Актуальність роботи, 2. Мета роботи, 3. Структура IP-адрес, 4. Структурна схема таблиць БД

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці та	доц. Тотосько О.В.		
Безпека в надзв. Ситуаціях.			

7. Дата видачі завдання 12.11.2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналітична частина	12.11-17.11.2025	
2	Технологічна частина	18.11-27.11.2025	
3	Конструкторська частина	28.11-05.12.2025	
4	Науково-дослідна частина	06.12-10.12.2025	
5	Спеціальна частина	11.12.-15.12.2025	
6	Охорона праці та безпека в надзвичайних ситуаціях	16.12-20.12.2025	
7	Основні висновки	21.12-22.12.2025	

Студент _____ Третяк С.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Чихіра І.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота магістра складається з пояснювальної записки та графічної частини (ілюстративний матеріал – слайди).

Об’єм графічної частини кваліфікаційної роботи становить 18 слайдів.

Об’єм пояснювальної записки складає 81 друковану сторінку формату А4(210х297).

Кваліфікаційна робота складається з шести розділів, в яких нараховується 38 рисунків та 5 таблиць з даними.

В роботі використано 17 літературних джерел.

В даному проекті було розроблено автоматизовану систему взаємодії мережевого обладнання з базою даних географічного розташування IP-мереж. Було досліджено можливості оптимізації форматів зберіганні IP-адрес у базах даних для забезпечення пришвидшення пошуку потрібних даних.

В результаті буде створена база даних по встановленню географічної приналежності IP-адрес, котра дозволятиме виконувати різноманітні дії: отримання географічного положення IP-адрес, отримання статистики по поширенню та кількості IP-адрес у певному географічному об’єкті, а також створювати конфігураційні файли для мережевого обладнання на основі запитів заданого формату.

Основне завдання розробленої системи – автоматизація конфігурування мережевого обладнання, та підвищення інформативності статистики що надається мережевим обладнання адміністратору.

Ключові слова: База даних, автоматизація, MySQL, Mikrortik, PHP, IP-адреси.

Третяк С.В. Розроблення та дослідження програмного комплексу для оптимізації системи зберігання , обробки та аналізу баз даних IP-адрес.

174 – Автоматизація, комп’ютерно-інтегровані технології та робототехніка./ С.В. Третяк, - Тернопіль: ТНТУ, 2025. – 81 с.

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, ОДИНИЦЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

СУБД – система управління базами даних;

БД – база даних

IP-адреса – адреса пристрою в компютерних мережах з протоколом IP(Internet Protocol);

PHP – мова програмування PHP що використовується на веб-серверах;

HTML – Hyper Text Markup Language , мова розмітки веб-сторінок;

ROS – Routerboard Operation System, операційна система обладнання Mikrotik;

ВДТ – відео-дисплейний термінал. Робоче місце оператора ЕОМ.

ВСТУП

На сьогоднішній день глобальні компютерні мережі зустрічаються у всіх сферах життєдіяльності та виробничих процесів. Всесвітня мережа Інтернет знаходить застосування у роботі майже всіх підприємств, організацій, державних та приватних закладів. Кожна країна, кожне місто чи навіть невеликий населений пункт має своє «представлення» у мережі Інтернет.

Нажаль не всі учасники цієї мережі використовують її для законних справ, є й такі що використовують мережу з негативними намірами. І тому перед адмінстраторами мережевого обладнання з'являється все більше і більше завдань для захисту від таких атак чи спроб проникнення в мережу підприємства. Однак це вимагає багато часу і зусиль якщо блокувати конкретні адреси зловмисників, інколи доцільніше превентивно заблокувати доступ для певної мережі загалом, якщо більшість атак походять з цієї мережі. Саме тому для адміністраторів був би дуже помічним інструмент котрий автоматично показуватиме з якого населеного пункту чи країни відбувалась атака, щоб можна було не очікуючи наступних атак заблокувати превентивно трафік з цього обєкту.

Також немаловажним аспектом є використання кількох ліній доступу до мережі інтернет у організаціях. І досить актуальним є завдання маршрутизації до різних географічних обєктів через різні канали доступу в мережу Інтернет. Наприклад звернення у світові мережі маршрутизувати через провайдера №1, а звернення до адрес в Україні маршрутизувати через провайдера №2 , оскільки по Україні від дає кращу швидкість. Тому, на мою думку, автоматизована система створення маршрутів до вибраного географічного обєкту значно скоротить трудозатрати мережевих адміністраторів при виконанні таких завдань.

Виходячи з вищенаведених міркувань було прийняте рішення розробити та реалізувати систему автоматизованої взаємодії мережевого обладнання з базами даних географічного розташування IP-мереж у всесвітній мережі Інтернет.

Розділ 1: АНАЛІТИЧНА ЧАСТИНА

1.1. Аналіз стану питання за літературними та іншими джерелами

Огляд існуючих рішень.

Однією з ключових складових систем аналізу та оптимізації роботи з IP-адресами є використання достовірних джерел географічної інформації. Геолокація IP-адрес дозволяє встановити країну, регіон, місто, а іноді й провайдера або автономну систему, до якої належить певна IP-адреса або IP-мережа. Такі дані широко застосовуються у мережевій безпеці, аналізі трафіку, фільтрації доступу, контентній персоналізації та автоматизованому формуванні правил для мережевого обладнання.

На сьогодні існує низка спеціалізованих сервісів і баз даних, що надають інформацію про географічне розташування IP-адрес.

MaxMind GeoIP / GeoLite2

MaxMind є одним з найпопулярніших постачальників геолокаційних баз даних. Компанія надає як безкоштовні (GeoLite2), так і комерційні (GeoIP2) бази даних.

Основні характеристики:

- підтримка IPv4 та IPv6;
- визначення країни, регіону, міста;
- інформація про провайдера (ISP) та автономну систему (ASN);
- регулярні оновлення;
- можливість локального використання бази даних без зовнішніх запитів.

Файл саме їх списку мереж було використано в проекті для створення бази даних.

IP2Location

IP2Location — комерційний сервіс, який надає розширену інформацію про IP-адреси та мережі.

Основні можливості:

- визначення країни, регіону, міста;
- дані про часовий пояс, поштовий індекс;
- інформація про тип з’єднання (мобільний, дата-центр, проксі);
- локальні бази даних та API-доступ.

IPinfo

IPinfo — сучасний API-сервіс для отримання інформації про IP-адреси в режимі реального часу.

Надає дані про:

- країну та місто;
- ASN та організацію;
- хостинг або дата-центр;
- можливість масових запитів.

1.2. Актуальність виконання роботи

Всі вищереперілені у розділі 1.1 сервіси надають досить деталізовані та актуальні дані. Проте ключовим недоліком є те що вони надають лише стислу інформацію по конкретному запиту. Створювати свої спеціалізовані запити для отримання розширеної статистики не можливо. А тим більше нема можливості змінювати формат відповіді щоб формувати шаблони готових команд для конфігурування мережевого обладнання.Тому на мою думку, для вирішення достатньо вузькоспеціалізованих завдань, найоптимальнішим рішенням є створення власної бази даних використовуючи дані надані вищепереліченими сервісами. Так буде отримано деталізація і актуальність інформації та в той ж час

гнучкість та ширина можливостей взаємодії з базою знань. Що в подальшому дозволить гнучко нарощувати затребуваний функціонал та реалізовувати необхідні механізми взаємодії відносно поставлених завдань.

1.3. Методи вирішення поставленої задачі

Для вирішення поставленої задачі було прийняте рішення застосувати ряд технологій.

СУБД MySQL для доступу до збережних даних, а конкретно до інформації про географічне розташування IP-мереж. Дана СУБД була обрана за безкоштовність, швидкість роботи та представленість на більшості платформ хостингу в мережі Інтернет.

Для створення зручного механізму взаємодії між БД та користувачем чи мережевим обладнанням було прийнято рішення використати серверні сценарії PHP. PHP було використано з метою спрощення взаємодії між БД та користувачем. Так як користувачу не буде потреби знати як і де працює БД, не буде потрібно вміти виконувати запити SQL, лиш вказати потрібні дані в шаблоні введення необхідних даних і всі наступні дії PHP виконає за нього та видасть результат у зручному для користувача, завершеному і готовому до застосування, виді.

У якості інструменту реалізації візуального інтерфейсу користувача було використано HTML. Оскільки це дозволить легко і в той ж час гнучно проектувати зручний інтерфейс, з мінімальними трудозатратами.

В результаті має бути спроектована та реалізована система котра буде приймати від користувача чи обладнання короткий шаблонний запит, та видавати структуровану, зручну для сприйняття людиною, інформацію або готові до застосування команди конфігурації мережевого обладнання.

1.4. Висновки та постановка задач на кваліфікаційну роботу магістра.

На даний момент існує ряд рішень(найбільш відомі з них приведені у розділі 1.1.) котрі реалізовують функціонал наближений до запланованого в реалізацію, проте функціонал саме наближений та може бути суттєво покращений. Тому було прийняте рішення реалізувати запланований функціонал в програмному комплексі з відкритим вихідним кодом , щоб надати можливість всім бажаючим в подальшому покращувати і розширювати функціонал даного програмного продукту, або ж використовувати частини його вихідного коду для реалізації суміжних завдань. Також в задачі на кваліфікаційну роботу магістра було додане завдання дослідити методи покращення швидкості взаємодії з базою даних через зміну чи доповнення формату зберігання даних в таблицях даної БД. А саме – проаналізувати методи запису IP-адрес у таблицях БД. Дослідити результуючу швидкодію при різних форматах збереження даних і/або структурі таблиці БД.

Розділ 2: ТЕХНОЛОГІЧНА ЧАСТИНА

2.1.Найменування та область застосування

В даній дипломній роботі розроблено програмний комплекс для надання інформації та статистики користувачам, а також для автоматизації конфігурування мережевого обладнання на базі СУБД MySQL та інтерпритованої мови програмування PHP.

База даних містить всю необхідну географічну інформацію про IP-мережі. БД може працювати як на окремому фізичному сервері, так і розміщуватися на компютері що виконує роль веб-сервера. База даних являє собою сукупність таблиць повязаних між собою певними відносинами.

Програмний комплекс для роботи з базою даних це серверні котрі розміщуються на веб-сервері. В даному випадку для створення серверних сценаріїв було використано інерпритовану мову програмування PHP.

2.2. Структура бази даних

База даних забезпечує накопичення даних, а також має реалізовувати швидкий доступ до вищеописаних даних. Інформація в БД має бути: не надлишковою, не суперечливою, мати цілісну структуру. Виходячи з даних вимог було заплановано що база даних містимитиме таку інформацію: адресу мережі, розмір мережі, найменування країни чи міста (в залежності з яким рівнем деталізації нам будуть необхідні дані будуть використані різні таблиці).

2.3. Проектування бази даних

2.3.1. Представлення IP-адрес у MySQL/MariaDB

Існує кілька способів зберігати IPv4:

1) VARCHAR(15)

Найпростіший, але найнеефективніший підхід.

Недоліки:

- повільне порівняння рядків;
- неефективний індекс (лексикографічний, а не числовий);
- займає 7–15 байт + накладні.

2) BINARY(4)

IP зберігається як 4-байтовий масив. Функції перетворення:

- INET_ATON() → числове значення
- INET_NTOA() → рядок

3) UNSIGNED INT (найкращий варіант для IPv4)

Займає 4 байти, дає:

- швидкі порівняння,
- ефективну індексацію B-Tree,
- легке порівняння діапазонів.

Приклад зберігання:

```
CREATE TABLE ip_data (
    ip_int INT UNSIGNED NOT NULL,
    PRIMARY KEY (ip_int)
);
```

2.3.2. Зберігання CIDR-мереж у реляційних БД

Стандартних типів CIDR немає, тому мережу треба записувати у вигляді:

start_ip, end_ip

або ж:

base_ip, prefix

Найпоширеніший і найефективніший спосіб — зберігати початок і кінець діапазону, використовуючи INT (IPv4) або BINARY(16)/BIGINT UNSIGNED (IPv6).

Приклад для IPv4:

```
CREATE TABLE networks (
    start_ip INT UNSIGNED NOT NULL,
    end_ip INT UNSIGNED NOT NULL,
    prefix_length TINYINT UNSIGNED,
    PRIMARY KEY (start_ip, end_ip)
);
```

Приклад перетворення CIDR → діапазону:

192.168.1.0/24 - start = 3232235776 -end = 3232236031

Пошук IP-адреси в CIDR-діапазонах

Найчастіша задача в аналітиці — визначити, в який CIDR-блок входить IP. SQL-запит матиме наступний вид:

```
SELECT *
FROM networks
WHERE ip_int BETWEEN start_ip AND end_ip
LIMIT 1;
```

Чому працює швидко?

- індекс на (start_ip, end_ip) дозволяє миттєво відсікати непотрібні діапазони;
- операція BETWEEN для чисел — одна з найшвидших.

Проте це лише самий поширений метод. Метою цього дослідження є перевірка можливості використання інших форматів збереження адрес IP-мереж у базах даних. Тому також будуть розроблені і використані інші методи.

2.3.3 Отримання файлів географічної приналежності

Для початку робіт необхідно отримати список IP-мереж в мережі Інтернет, з розділенням по країнах і містах(бажано). Мною було обрано сервіс https://www.maxmind.com як такий що надає актуальні дані на безкоштовній основі, інтерфейс сайту відображено на рисунку 2.3.3.1

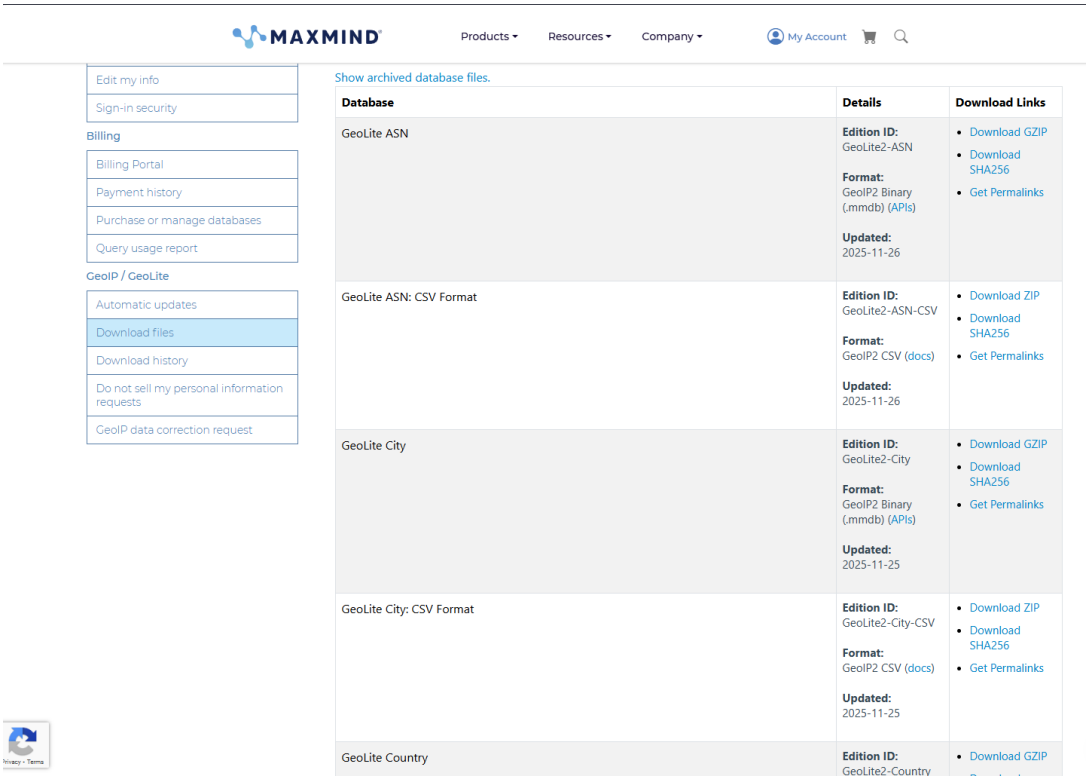
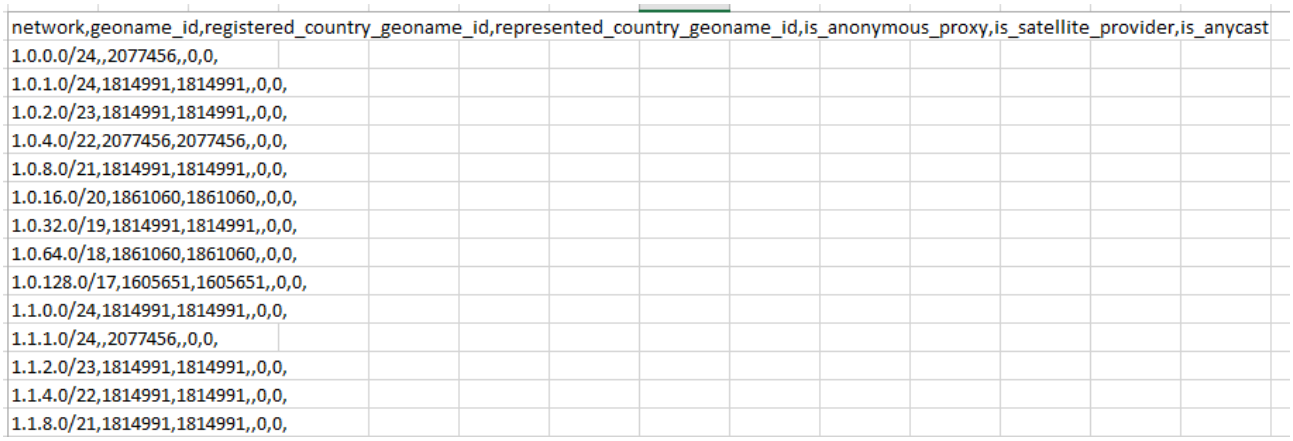


Рис. 2.3.3.1. Інтерфейс сторінки скачування сервісі Maxmind

В даному випадку нам потрібні файли GeoLite Country: CSV Format та GeoLite City: CSV Format , котрі є архівами що містять csv файли з необхідними даними.

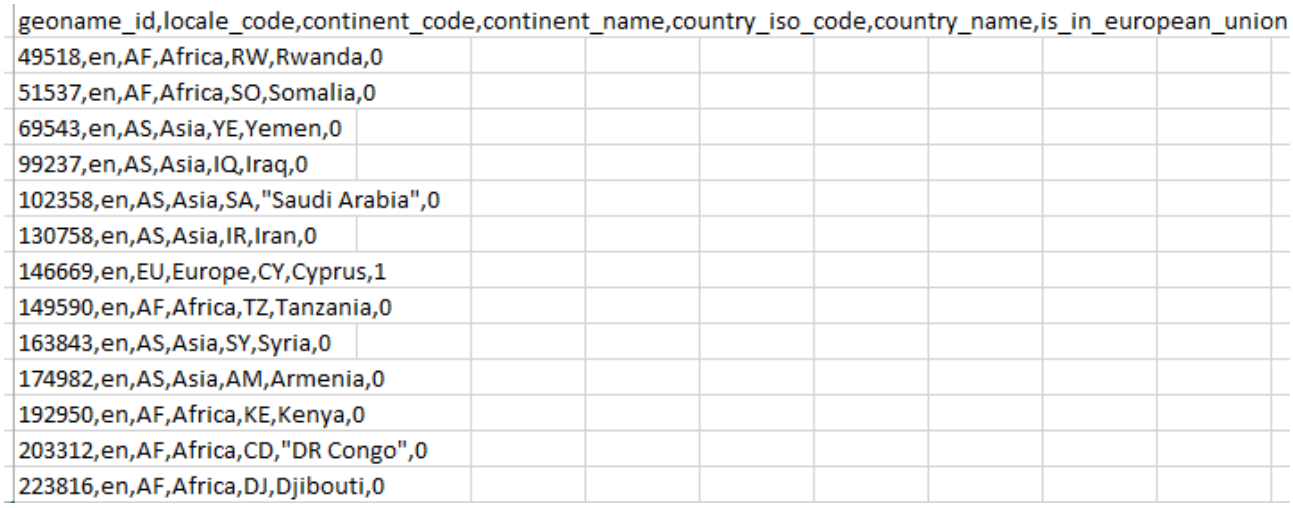
В архіві GeoLite Country ми зокрема отримуємо файл GeoLite2-Country-Blocks-IPv4.csv, вміст даного файлу проілюстровано на рисунку 2.3.4.2



network	geoname_id	registered_country_geoname_id	represented_country_geoname_id	is_anonymous_proxy	is_satellite_provider	is_anycast
1.0.0.0/24		2077456		0	0	
1.0.1.0/24	1814991	1814991		0	0	
1.0.2.0/23	1814991	1814991		0	0	
1.0.4.0/22	2077456	2077456		0	0	
1.0.8.0/21	1814991	1814991		0	0	
1.0.16.0/20	1861060	1861060		0	0	
1.0.32.0/19	1814991	1814991		0	0	
1.0.64.0/18	1861060	1861060		0	0	
1.0.128.0/17	1605651	1605651		0	0	
1.1.0.0/24	1814991	1814991		0	0	
1.1.1.0/24		2077456		0	0	
1.1.2.0/23	1814991	1814991		0	0	
1.1.4.0/22	1814991	1814991		0	0	
1.1.8.0/21	1814991	1814991		0	0	

Рис. 2.3.3.2 – зразок вмісту файлу GeoLite2-Country-Blocks-IPv4.csv

Та файл GeoLite2-Country-Locations-en.csv, вміст котрого проілюстровано на рисунку 2.3.3.3



geoname_id	locale_code	continent_code	continent_name	country_iso_code	country_name	is_in_european_union
49518	en	AF	Africa	RW	Rwanda	0
51537	en	AF	Africa	SO	Somalia	0
69543	en	AS	Asia	YE	Yemen	0
99237	en	AS	Asia	IQ	Iraq	0
102358	en	AS	Asia	SA	"Saudi Arabia"	0
130758	en	AS	Asia	IR	Iran	0
146669	en	EU	Europe	CY	Cyprus	1
149590	en	AF	Africa	TZ	Tanzania	0
163843	en	AS	Asia	SY	Syria	0
174982	en	AS	Asia	AM	Armenia	0
192950	en	AF	Africa	KE	Kenya	0
203312	en	AF	Africa	CD	"DR Congo"	0
223816	en	AF	Africa	DJ	Djibouti	0

Рис. 2.3.3.3 – зразок вмісту файлу GeoLite2-Country-Locations-en.csv

Маючи інформацію з цих двох файлів ми можемо почати проектувати таблиці баз даних для зберігання потрібних даних.

2.3.4 Проектування таблиць бази даних

Перш за все потрібно було визначитися з типом таблиць у котрих зберігати дані. Вибір зупинився на двох самих поширених – MyISAM та InnoDB. Кожен із них має власну архітектуру, підхід до зберігання даних, індексування та забезпечення цілісності, що безпосередньо впливає на продуктивність систем, особливо тих, які працюють із великими масивами IP-адрес, логів або мережевих даних.

Архітектура зберігання

MyISAM

- Таблиці зберігаються у трьох файлах:
  .frm — структура таблиці;
  .MYD — дані;
  .MYI — індекси.
- Відсутня підтримка транзакцій, журналювання змін та механізмів відновлення після збою.
- Підходить для задач, де потрібне максимально швидке читання.

InnoDB

- Використовує індексоване зберігання у форматі *tablespace*, усі дані та індекси можуть зберігатися в одному контейнері (ibdata) або окремо у файлах.
- Підтримує транзакції, ACID, журнали redo/undo.
- Підтримує автоматичне відновлення після збою.

Продуктивність

MyISAM

- Дуже швидке читання завдяки простій структурі.
- Швидкі повнотекстові індекси (FULLTEXT).
- Низька продуктивність при великій кількості записів або при активних записах.

InnoDB

- Висока продуктивність у змішаних режимах читання/запису.
- Використовує кластерний індекс, що прискорює пошук за первинним ключем.
- Має буферний пул, який кешує як індекси, так і дані.

Надійність та відновлення

MyISAM

- У випадку збою сервера таблиця може бути пошкоджена.
- Необхідність у ручному виконанні REPAIR TABLE.
- Немає журналів транзакцій.

InnoDB

- Автоматично відновлює стан завдяки redo/undo логам.
- Дані залишаються консистентними навіть після аварійного завершення.

Робота з IP-адресами та великими обсягами логів

MyISAM

Переваги:

- Швидкі послідовні читання.
- Підходить для статичних архівів логів.

Недоліки:

- Паралельні записи блокують таблицю.
- Ризик пошкодження при великих об’ємах логування.

InnoDB

Переваги:

- Оптимальна робота з числовими полями (зокрема перетворені IP: INET_ATON() → INT UNSIGNED).
- Висока продуктивність індексації.
- Дозволяє виконувати агрегації та аналітичні запити без блокувань.
- Значно краще підходить для систем аналізу трафіку або великих СУБД.

Недоліки:

- Дещо більші накладні витрати на зберігання.

Виходячи з вищенаведеного можна зробити попередній висновок що MyISAM є більш оптимальним вибором для статичних даних де домінує читання, дані змінюються рідко, а ключовим є просте та швидке отримання інформації.

З іншої сторони InnoDB є універсальним, безпечним та оптимізованим механізмом зберігання, який забезпечує високу продуктивність для систем із великими обсягами IP-адрес, логів, журналів подій та аналітичних даних. Проте всюди порівнюють саме послідовне читання даних, а у системі що я проектую буде широке застосування пошуків по індексах. У останній час InnoDB сильно покращив роботу з індексами, читання з використанням індексів . Підсумовуючи MyISAM швидше:

- великі статичні таблиці (>100 ГБ),
- послідовне читання,
- повнотекстовий пошук у старих MySQL,
- read-only архіви.

InnoDB швидше:

- вибірки через індекси,
- часте читання одних і тих же даних,
- сервіси з постійними INSERT/UPDATE,
- будь-які вибірки у високонавантажених системах,
- робота з IP-адресами (через індексацію).

Тому все таки вибір був зупинений на InnoDB, як такий що забезпечить кращу швидкість з індексованими записами при роботі з адресами IP-мереж.

Виходячи з наявних даних попередньо було заплановано наступну структуру таблиць країн та адрес мереж:таблиця мереж по країнах:

```
CREATE TABLE `ipbase`.`ipv4country` (
`id` MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT ,
`ipnetwork` VARCHAR(18) NOT NULL ,
`mask` TINYINT UNSIGNED NOT NULL ,
`begin` INT UNSIGNED NOT NULL ,
`end` INT UNSIGNED NOT NULL ,
PRIMARY KEY (`id`), INDEX (`mask`), UNIQUE (`begin`, `end`)
) ENGINE = InnoDB;
```

Де :

- id – ідентифікатор запису;
- ipnetwork – текстовий запис IP-мережі виду 128.128.128.128/25(максимально можлива довжина тексту 18 символів);
- mask – маска мережі у виді числа (можливі значення від 0 до 32);
- begin – початок діазону IP-мережі записаний в десятковому виді;
- end – кінець діазону IP-мережі записаний в десятковому виді;

Також створений основний ключ(Primary Key) по полю id як унікальному ідентифікатору запису. Поле ipnetwork також являється унікальним, але оскільки воно є типу Varchar то пошук за ним буде повільнішим. На мою думку використовувати основний ключ по додатковому полю типу Medium Integer буде оптимальнішим рішенням, оскільки основний ключ фігурує в структурі індексів та даних у таблиці, тому найбільш компактний і зручний для читання основний ключ пришвидшить роботу з даними навіть якщо нема звернень до нього у явному виді.

До індексів також віднесено поле mask так як в подальшому можливий пошук по ширині бітової маски, тому я одразу запланував індексування цього поля.

Адреси початку і кінця діапазону IP-мережі зроблені унікальним ключем, адже, за умови що дані структуровано і перевірено на помилки, не можливий випадок коли дві мережі мають одинакові адреси початку і кінця діапазону, так як це означатиме що це та сама мережа.

Проте при створенні таблиці мною було пропущене одне з головних полів – географічна приналежність IP-мережі. Тому створюємо ще поле з ідентифікатором країни, та створено індекс по ньому:

```
ALTER TABLE `ipv4country` ADD `geoid` MEDIUMINT UNSIGNED NOT NUL
L AFTER `end`, ADD INDEX (`geoid`);
```

Після чого структура таблиці набула виду проілюстрованого на рисунку 2.3.4.1.

#	Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
1	id	mediumint(8)		UNSIGNED	Ні	*Немає*		AUTO_INCREMENT
2	ipnetwork	varchar(18)	utf8mb4_general_ci		Ні	*Немає*		
3	mask	tinyint(3)		UNSIGNED	Ні	*Немає*		
4	begin	int(10)		UNSIGNED	Ні	*Немає*		
5	end	int(10)		UNSIGNED	Ні	*Немає*		
6	geoid	smallint(5)		UNSIGNED	Ні	*Немає*		

Рис. 2.3.4.1 – структура таблиці географічної приналежності IPv4 мереж.

Створюємо таблицю країн з вказанням континенту до котрого вони належать(буде корисним при побудові статистики та агрегації).

```
CREATE TABLE `ipbase`.`country` (
`id` MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT ,
`country` VARCHAR(100) NULL ,
`continent` VARCHAR(100) NULL DEFAULT NULL ,
PRIMARY KEY (`id`), INDEX (`continent`)
```

```
) ENGINE = InnoDB;
```

Де:

- id -ідентифікатор країни, на котрий посилається поле **geoid** таблиці **ipv4country;**
- country – назва країни англійською мовою;
- continent – назва континенту котрому належить країна.

Кінцева структура таблиці списку країн набула виду проілюстрованому на рисунку 2.3.4.2

Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
id	mediumint(8)		UNSIGNED	Ні	*Немає*		AUTO_INCREMENT
country	varchar(100)	utf8mb4_general_ci		Так	*NULL*		
continent	varchar(100)	utf8mb4_general_ci		Так	*NULL*		

Рис. 2.3.4.2 – структура таблиці списку країн

Також створюємо таблицю географічної приналежності адрес IP-мереж по містам, та таблицю міст з вказанням країни котрій дане місто належить.

```
CREATE TABLE `ipv4city` (
  `id` int(10) UNSIGNED NOT NULL,
  `ipnetwork` varchar(18) NOT NULL,
  `mask` tinyint(3) UNSIGNED NOT NULL,
  `begin` int(10) UNSIGNED NOT NULL,
  `end` int(10) UNSIGNED NOT NULL,
  `city` mediumint(8) UNSIGNED NOT NULL,
  `country` mediumint(8) UNSIGNED NOT NULL,
  `latitude` decimal(7,4) NOT NULL,
  `longitude` decimal(7,4) NOT NULL
) ENGINE=InnoDB
ALTER TABLE `ipv4city`
  ADD PRIMARY KEY (`id`),
```

```
ADD UNIQUE KEY `begin` (`begin`,`end`),
ADD KEY `mask` (`mask`),
ADD KEY `city` (`city`),
ADD KEY `country` (`country`);
```

Структура даних повторює структуру даних таблиці ipv4country , але поле geoid замінене двома полями, city та country, котрі вказують ID міста та країни котрим належить дана підмережа. Поля latitude та longitude відповідають за географічні координати ширину та довжину, буде корисним в разі візуалізації фізичного розміщення реєстрації IP-мережі. Загальна структура таблиці проілюстрована на рисунку 2.3.4.3

Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
id	int(10)		UNSIGNED	Ні	*Немає*		AUTO_INCREMENT
ipnetwork	varchar(18)	utf8mb4_general_ci		Ні	*Немає*		
mask	tinyint(3)		UNSIGNED	Ні	*Немає*		
begin	int(10)		UNSIGNED	Ні	*Немає*		
end	int(10)		UNSIGNED	Ні	*Немає*		
city	mediumint(8)		UNSIGNED	Ні	*Немає*		
country	mediumint(8)		UNSIGNED	Ні	*Немає*		
latitude	decimal(7,4)			Так	*NULL*		
longitude	decimal(7,4)			Так	*NULL*		

Рис. 2.3.4.3 – структура таблиці географічної приналежності IPv4 (міста).

Наступним кроком створюємо таблицю міст, де вказуємо ідентифікатор міста, країну приналежності, та назву міста англійською мовою.

```
CREATE TABLE `city` (
  `id` mediumint(9) NOT NULL,
  `country` varchar(100) NOT NULL,
  `city` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
ALTER TABLE `city`
  ADD PRIMARY KEY (`id`),
  ADD KEY `country` (`country`);
```

Як результат ми отримуємо структуру таблиці проілюстровану на рис. 2.3.4.4

Ім'я	Тип	Зіставлення	Атрибути	Нуль	За замовчуванням	Коментарі	Додатково
id	mediumint(9)			Ні	*Немає*		AUTO_INCREMENT
country	varchar(100)	utf8mb4_general_ci		Ні	*Немає*		
city	varchar(100)	utf8mb4_general_ci		Ні	*Немає*		

Рис. 2.3.4.4 – структура таблиці списку міст.

На цьому етап попереднього проектування системи можна вважати завершеним. Створено лише пусту базу даних з необхідною структурою. Надалі буде проведено заповнення таблиць даними та зміна структури таблиць під нові вимоги якщо такі виникнуть, результуючі структури таблиць буде наведено в додатку 2 та додатку 3. А також буде розроблено програмну частину для взаємодії з базою даних адрес IP-мереж.

Розділ 3: КОНСТРУКТОРСЬКА ЧАСТИНА.

РОЗРОБКА СИСТЕМИ ВЗАЄМОДІЇ З БАЗОЮ ДАНИХ

3.1 Вибір мови програмування розробки системи взаємодії з БД.

Вибір мови програмування для реалізації програмного комплексу є критично важливим етапом проєктування, оскільки саме від інструментарію розробника залежить продуктивність системи, стабільність, безпека, масштабованість та простота супроводу. У межах даної роботи для створення системи оптимізації зберігання, обробки та аналізу баз даних IP-адрес обрано мову програмування PHP, що є одним із найбільш доцільних та технологічно обґрунтованих рішень.

Широка підтримка роботи з реляційними СУБД

PHP має нативну та високопродуктивну підтримку взаємодії з системами керування базами даних MySQL/MariaDB — саме тими СУБД, які найбільш часто використовуються для зберігання великих колекцій IP-адрес та мереж. Мова забезпечує:

- розширення mysqli та PDO з підготовленими запитами;
- оптимізовані драйвери з мінімальними накладними витратами;
- стабільний доступ до інструментів профілювання запитів;
- підтримку складних операцій на рівні БД, включаючи транзакції, індексацію та оптимізацію JOIN-конструкцій.

Ці можливості роблять PHP ефективним середовищем для створення інструментів роботи з IP-адресами та підмережами, а також для реалізації високошвидкісних сервісів обробки мережевих даних.

Оптимальна продуктивність для серверних застосунків

Сучасні версії PHP (8.x) забезпечують суттєве зростання продуктивності завдяки:

- JIT-компіляції,
- оптимізованому Zend Engine,
- ефективній роботі з асоціативними масивами та структурованими даними,
- зниженню часу виконання типових серверних операцій.

Завдяки цьому PHP може обробляти великі обсяги мережевих даних (IP-адреси, діапазони, таблиці маршрутизації) без значних затримок і із мінімальним споживанням ресурсів.

Наявність готових бібліотек для роботи з IP-адресами та CIDR

PHP має розвинену екосистему пакетів Composer, серед яких є готові інструменти для роботи з Ipv4 та Ipv6:

- бібліотеки для конвертації IP ↔ integer;
- засоби для обчислення CIDR, діапазонів, масок;
- парсери та валідатори мережевих параметрів;
- інструменти для агрегації та оптимізації списків IP-мереж.

Це дозволяє прискорити розробку і забезпечити високу точність обчислень, необхідних для наукових досліджень у галузі оптимізації IP-простору.

Можливість тісної інтеграції з веб-інтерфейсом.

У рамках побудови системи аналізу IP-адрес важливо забезпечити:

- веб-інтерфейс перегляду та управління даними,
- віддалений доступ через REST API,
- можливість побудови інтерактивних інструментів.

PHP є однією з найкращих мов для створення таких веб-систем завдяки:

- вбудованій серверній моделі,
- наявності фреймворків (Laravel, Symfony),
- простій інтеграції з фронтендом,

- можливості обробки запитів у режимі реального часу.

Таким чином, мова дозволяє реалізувати повноцінний прикладний сервіс для роботи з IP-адресами, що поєднує базу даних, аналітику та інтерфейс користувача.

У підсумку, використання PHP є оптимальним вибором для розробки системи взаємодії з базою даних IP-мереж завдяки поєднанню продуктивності, простоти інтеграції з реляційними СУБД, наявності бібліотек для обробки IP-адрес, широкій підтримці розробниками та здатності створювати масштабовані веб-орієнтовані системи. Всі ці фактори роблять PHP повноцінною платформою для реалізації аналітичних і оптимізаційних інструментів у сфері управління IP-простором.

3.2 Створення механізму заповення таблиць бази даних.

Перш за все потрібно заповнити таблиці даними отриманими з сервісу maxmind.com у розділі 2.3.3.

Для цього мною було програму читання даних з CSV-файлу, форматування, та запису у відповідну таблицю бази даних. Було створено чотири програми, кожна програма відповідала за заповнення конкретної таблиці.

Таблиця країн

Файл списку країн має структуру зображену на рисунку 3.2.1

```
1  geoname_id,locale_code,continent_code,continent_name,country_iso_code,country_name,is_in_european_union
2  49518,en,AF,Africa,RW,Rwanda,0
3  51537,en,AF,Africa,SO,Somalia,0
4  69543,en,AS,Asia,YE,Yemen,0
5  99237,en,AS,Asia,IQ,Iraq,0
6  102358,en,AS,Asia,SA,"Saudi Arabia",0
7  130758,en,AS,Asia,IR,Iran,0
```

Рис. 3.2.1 – структура файлу списку країн.

Оскільки настільки детальної інформації про країну зберігати нема необхідності, та щоб таблиця мала якомога компактніший вид і була легша для пошуку то я в таблицю записав лиш унікальний ідентифікатор країни, її назву та континент до котрого країна належить. Результуючий код вказано у лістингу 3.2.1

Лістинг 3.2.1 - заповення таблиці країн:

```
<?php
    include ('connect_db.php');
    //вказуємо файл звідки читати дані
    $ipcountry_file=fopen('GeoLite2-Country-Locations-en.csv',
"r");
    //зчитуємо перший рядок з описом полів
    $row_ip= fgetcsv($ipcountry_file);
    //в циклі зчитуємо всі рядки з файлу
    While ($row_ip= fgetcsv($ipcountry_file))
    {
        //записуємо в БД рядок з даними
        $query_insert_country="INSERT INTO country SET
id='$row_ip[0]', country='$row_ip[5]', continent='$row_ip[3]'";
        mysqli_query($mysql, $query_insert_country);
    }
?>
```

Таблиця міст

Файл списку міст має структуру зображену на рисунку 3.2.2

```
1  geoname_id,locale_code,continent_code,continent_name,country_iso_code,country_
2  11797,en,AS,Asia,IR,Iran,02,Māzandarān,,,Afrā,,Asia/Tehran,0
3  18918,en,EU,Europe,CY,Cyprus,04,Ammochostos,,,Protaras,,Asia/Famagusta,1
4  49518,en,AF,Africa,RW,Rwanda,,,,,,,Africa/Kigali,0
5  49747,en,AF,Africa,SO,Somalia,BK,Bakool,,,Oddur,,Africa/Mogadishu,0
```

Рис. 3.2.2 – структура файлу списку міст.

Виходячи з аналогічних міркувань як для таблиці країн мною було прийняти рішення що необхідною та корисною будуть лише – ідентифікатор міста, назва міста та країна розташування міста. Даний код приведено в лістингу 4.2.2

Лістинг 4.2.2 - заповення таблиці міст:

```
<?php
    include ('connect_db.php');
    //вказуємо файл звідки читати дані
    $ipcountry_file=fopen('GeoLite2-City-Locations-en.csv', "r");
    //зчитуємо перший рядок з описом полів
    $row_ip= fgetcsv($ipcountry_file);
```

```
    //в циклі зчитуємо всі рядки з файлу
    While ($row_ip= fgetcsv($ipcountry_file))
    {
        //Готуємо дані до запису в базу даних
        $country= mysqli_real_escape_string($mysql, $row_ip['5']);
        $city= mysqli_real_escape_string($mysql, $row_ip['10']);
        //записуємо в БД рядок з даними
        $query_insert_city="INSERT INTO city SET id='$row_ip[0]',
country='$country', city='$city'";
        mysqli_query($mysql, $query_insert_city);
    }
?>
```

Таблиця адрес IP-мереж по країнах

Файл мереж по країнах має структуру зображену на рисунку 3.2.3

```
network,geoname_id,registered_country_geoname_id,represented_country_geoname
1.0.0.0/24,,2077456,,0,0,
1.0.1.0/24,1814991,1814991,,0,0,
1.0.2.0/23,1814991,1814991,,0,0,
1.0.4.0/22,2077456,2077456,,0,0,
```

Рис. 3.2.3 – структура файлу адрес мереж по країнах.

Першим(нульовим в нумерації PHP) йде поле в котрому записано адресу мережі в CIDR форматі. Формат зручний для запису і читання людиною, але незручний для розуміння базою даних. Тому для забезпечення можливості пошуку та порівняння потрібно перевести дані в формат запису у виді числа. Для цього ми відділяємо адресу мережі від маски за допомогою функції explode, котра розділяє вказану змінну по вказаному символу. Так ми по символу «/» відділяємо адресу від маски:

list($ip, $prefix) = explode('/', $row_ip['0']);

Представляємо IP-адресу в виді десяткового числа

$ip_dec = ip2long($ip);

Створюємо маску

$mask = -1 << (32 - $prefix);

Мінус один використано оскільки 11111111 11111111 11111111 11111111 у 32-бітному поданні буде представлено як «мінус один». << - це побітовий зсув вліво. Отже ми зсуваємо вліво на ширину префікса, та заповнюємо біти нулями *11111111 11111111 11111111 11111111 << 8* дасть нам 11111111 11111111 11111111 00000000 (CIDR маска /24)

Тоді адресу мережі(початок діапазону) можна вирахувати як *$network = $ip_dec & $mask;*

Де $ip_dec – це IP-адреса у десятковому вигляді(32-бітне число), а $mask – маска мережі, також 32-бітне число. & - це побітове AND, яке порівнює кожний біт двох чисел.

Аналогічно ми можемо обчислити кінець діапазону, вона ж бродкаст-адреса *$broadcast = $network | (~$mask);*

~ - це побітове NOT, вона ж інверсія всіх бітів. Тобто 0 стане 1, а 1 стане 0.

| - це побітове OR.

В результаті всі біти що йдуть після адреси мережі будуть рівні 1. Що й дасть нам максимально можливу адресу для цієї адреси мережі з заданою шириною маски мережі.

В результаті було створено файл заповнення таблиці адрес IP-мереж по країнах, вміст котрого приведено в лістингу 3.2.3

Лістинг 3.2.3. – заповнення таблиці приналежності IP мереж країнам

```
<?php
    //виставляємо ліміт часу в 3600 секунд, так як скрипт
заповнення
    //бази даних займає довгий час
    set_time_limit(3600);
    ini_set('max_execution_time', 3600);
    include ('connect_db.php');
    //вказуємо файл звідки читати дані
    $ipcountry_file=fopen('GeoLite2-Country-Blocks-IPv4.csv',
"r");
    //зчитуємо перший рядок з описом полів
    $row_ip= fgetcsv($ipcountry_file);
    //в циклі зчитуємо всі рядки з файлу
```

```
        While ($row_ip= fgetcsv($ipcountry_file))
        {
            list($ip, $prefix) = explode('/', $row_ip['0']);
            $ip_dec = ip2long($ip);
            $mask = -1 << (32 - $prefix);
            $network   = $ip_dec & $mask;
            $broadcast = $network | (~$mask);
            /*
             * заповнюємо GEOID одиницею якщо значення відсутнє.
             * зроблено на випадок відсутніх значень у файлі. щоб
уникнути
             * можливих помилок при заповненні бази даних
             */

$row_ip['1']=($row_ip['1']=='')?$row_ip['2']:$row_ip['1'];
            $query_add_addr="INSERT INTO ipv4country SET
ipnetwork='$row_ip[0]', mask='$prefix', begin='$network',
end='$broadcast', geoid='$row_ip[1]'";
            mysqli_query($mysql, $query_add_addr);
        }
    ?>
```

Таблиця адрес IP-мереж по містах

Файл мереж по країнах має структуру зображену на рисунку 3.2.4

```
network,geoname_id,registered_country_geoname_id,represented_country_geoname
1.0.0.0/24,,2077456,,0,0,,,,,
1.0.1.0/24,1814991,1814991,,0,0,,34.7732,113.7220,1000,
1.0.2.0/23,1814991,1814991,,0,0,,34.7732,113.7220,1000,
1.0.4.0/22,2077456,2077456,,0,0,,-33.4940,143.2104,1000,
1.0.8.0/21,1814991,1814991,,0,0,,34.7732,113.7220,1000,
1.0.16.0/20,1861060,1861060,,0,0,,35.6900,139.6900,500,
```

Рис. 3.2.4 – структура файлу адрес мереж по містах.

Над файлом виконуємо аналогічні дії як до попереднього файлу, лише ще заносимо в базу даних країну та географічні координати реєстрації мережі.

Результуючи файл має вміст приведений в лістингу 5.2.4:

Лістинг 3.2.4 - заповнення таблиці приналежності IP мереж містам

```
<?php
    //виставляємо ліміт часу в 3600 секунд, так як скрипт заповнення
    //бази даних займає довгий час
    set_time_limit(3600);
    ini_set('max_execution_time', 3600);
    include ('connect_db.php');
    //вказуємо файл звідки читати дані
    $ipcountry_file=fopen('GeoLite2-City-Blocks-IPv4.csv', "r");
    //зчитуємо перший рядок з описом полів
    $row_ip= fgetcsv($ipcountry_file);
    //в циклі зчитуємо всі рядки з файлу

    While ($row_ip= fgetcsv($ipcountry_file))
    {
        list($ip, $prefix) = explode('/', $row_ip['0']);
        $ip_dec = ip2long($ip);
        $mask = -1 << (32 - $prefix);
        $network   = $ip_dec & $mask;
        $broadcast = $network | (~$mask);
        /*
         *     заповнюємо     країну     реєстрації     наступним значенням(країна походження) якщо значення відсутнє.
         * зроблено на випадок відсутніх значень у файлі. щоб уникнути
         * можливих помилок при заповненні бази даних
         * також заповнюємо "заглушками" місто .
        */

$row_ip['2']=($row_ip['2']=='')?$row_ip['3']:$row_ip['2'];
        $row_ip['2']=($row_ip['2']=='')?1:$row_ip['2'];
        $row_ip['1']=($row_ip['1']=='')?1:$row_ip['1'];
        $row_ip['7']=($row_ip['7']=='')?0:$row_ip['7'];
        $row_ip['8']=($row_ip['8']=='')?0:$row_ip['8'];
        $query_add_addr="INSERT      INTO      ipv4city      SET ipnetwork='$row_ip[0]',      mask='$prefix',      begin='$network',
```

```
end='$broadcast',      city='$row_ip[1]',      country='$row_ip[2]',
latitude='$row_ip[7]', longitude='$row_ip[8]'";
            mysqli_query($mysql, $query_add_addr);
        }
    ?>
```

3.3. Створення користувацького інтерфейсу взаємодії з базою даних.

Для початку мною було створено спрощений інтерфейс взаємодії користувача з таблицями бази даних географічного розміщення IP-адрес, з метою забезпечення можливості пошуку входження введеної IP-адреси у мережі наведені в відповідних таблицях.

Інтерфейс створювався на мові розмітки HTML для спрощення проектування та можливості його зміни в подальшому без необхідності суттєвої зміни вихідного коду.

В результаті було сформовано наступний вихідний код форми пошуку:

```
<form action="find_ip.php" method="POST">
    <input type="text" name="ip" pattern="^(25[0-5]|2[0-
4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}$"
required>
    <input type="submit" value="знайти">
</form>
```

Було використано інструкцію *pattern="^(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}$"* щоб виконувалась перевірка на коректність введення IP-адреси.

Результат введення даних у формі передавався на обробку PHP-обробнику котрий здійснював пошук відповідного запису у таблиці географічного розміщення адрес по містам.Вміст файла обробника приведено в лістингу 3.3.1

Лістинг 3.3.1. – пошук адреси в таблиці міст.

```
<?php
include ('connect_db.php');
```

```
    $ip_addr=$_POST['ip'];
    $ip= ip2long($ip_addr);
    list($first_octet,$second_octet,$third_octet,
$fourth_octet)=explode('.', $ip_addr);
    $query_get_country_city="SELECT ipv4city.ipnetwork as network,
city.city as city, country.country as country, country.continent as
continent FROM ipv4city LEFT JOIN city on city.id=ipv4city.city LEFT
JOIN country ON country.id=ipv4city.country WHERE
ipv4city.first_octet='$first_octet' AND ('$ip' BETWEEN
ipv4city.begin AND ipv4city.end)";
    $geo_data= mysqli_fetch_assoc(mysqli_query($mysql,
$query_get_country_city));
    echo "<strong>Мережа</strong>: $geo_data[network]<BR>";
    echo "<strong>Місто</strong>: $geo_data[city]<BR>";
    echo "<strong>Країна</strong>: $geo_data[country]<BR>";
    echo "<strong>Континентv</strong>: $geo_data[continent]<BR>";
    ?>
```

Першочергово пошук здійснювався по таблицях ipv4country та ipv4city, але оскільки таблиці містять ID географічних об’єктів, але не їх назви то потрібно було ще доєднати до запиту таблиці з співставленням ID та назв. Тому було в запит було додано директиву

LEFT JOIN city on city.id=ipv4city.city, та *LEFT JOIN country ON country.id=ipv4city.country*

щоб у результат виконання запиту було також включено дані з відповідного рядку таблиці city та таблиці country. Враховуючи що в таблиці *ipv4city* вже присутнє поле *country*, то робити запит по таблиці ipv4country нема потреби, достатньо буде запиту по таблиці *ipv4city* котра містить дані і про місто і про країну розташування.

Результат виконання запиту зображено на рисунку 3.3.1.

Мережа: 193.168.156.0/22
Місто: Oslo
Країна: Norway
Континентv: Europe

Рис. 3.3.1 – Результат пошуку по введеній IP-адресі.

Після отримання підтвердження роботоздатності алгоритму базового пошуку було прийнято рішення про розширення функціональних можливостей. Наступним кроком була реалізація отримання списку IP-мереж для певної країни.

Перш за все в інтерфейсі користувача було створено випадаючий список з всіма наявними в базі країнами. Реалізовано це було за допомогою інструкції випадаючого списку в HTML - *<select></select>*. Заповнення списку відбувалось автоматично згідно наявних даних у таблиці country

```
<form action="find_country_networks.php" method="GET">
    <select name="country">
        <?php
            $query_get_countries="SELECT * FROM `country` Order
by continent ASC, country ASC";
            $countries_mysql= mysqli_query($mysql,
$query_get_countries);
            while ($country=
mysqli_fetch_assoc($countries_mysql))
            {
                echo "<option
value=\"$country[id]\">$country[continent] -
$country[country]</option>";
            }
        ?>
    </select>
    <input type="submit" value="показати">
</form>
```

В результаті ідентифікатор країни передавався в PHP-обробник, вміст приведено в лістингу 3.3.2, котрий з бази даних отримував необхідні дані:

Лістинг 3.3.2 – отримання з бази даних інформації по країні

```
<?php
include ('connect_db.php');
$country_id=$_GET['country'];
$query_get_name="SELECT country, continent FROM `country` WHERE
id='$country_id'";
$country=                mysqli_fetch_assoc(mysqli_query($mysql,
$query_get_name));
```

```
    $country_name=$country['country'];
    $query_get_number_networks="SELECT COUNT(*) as countt FROM `ipv4country` WHERE country='$country_id'";
    $networks= mysqli_fetch_assoc(mysqli_query($mysql, $query_get_number_networks));
    $number_networks=$networks['countt'];
    echo "В <strong>$country_name</strong> загалом є <strong>$number_networks</strong> мереж:<BR>";
    $query_get_networks="SELECT ipnetwork, mask FROM `ipv4country` WHERE country='$country_id'";
    $networks_mysql= mysqli_query($mysql, $query_get_networks);
    $available_hosts=0;
    while ($network= mysqli_fetch_assoc($networks_mysql))
    {
        echo "$network[ipnetwork]";
        $number_host_in_network=pow(2,(32-$network['mask']));
        echo "($number_host_in_network),  ";
        $available_hosts=$available_hosts+$number_host_in_network;
    }
    echo "<BR>Всього можлива кількість пристроїв в країні: $available_hosts"
    ?>
```

На сторінці відображалась назва країни, кількість глобально маршрутизованих IP-мереж в цій країні, список мереж(з вказанням доступної кількості пристроїв в цій мережі) та загальна теоретично-можлива кількість пристроїв в цих мережах. Результат виконання запиту зображено на рисунку 3.3.2

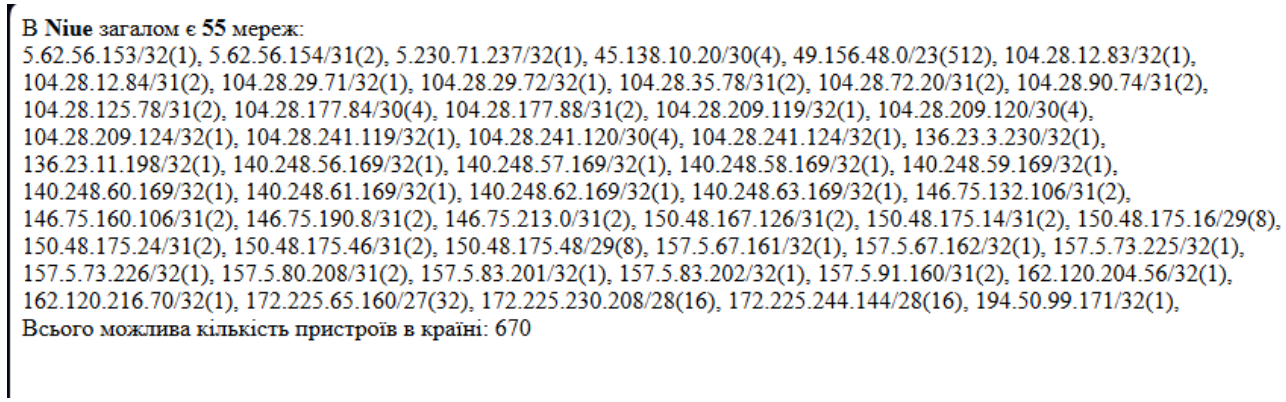

В **Niue** загалом є **55** мереж:
5.62.56.153/32(1), 5.62.56.154/31(2), 5.230.71.237/32(1), 45.138.10.20/30(4), 49.156.48.0/23(512), 104.28.12.83/32(1), 104.28.12.84/31(2), 104.28.29.71/32(1), 104.28.29.72/32(1), 104.28.35.78/31(2), 104.28.72.20/31(2), 104.28.90.74/31(2), 104.28.125.78/31(2), 104.28.177.84/30(4), 104.28.177.88/31(2), 104.28.209.119/32(1), 104.28.209.120/30(4), 104.28.209.124/32(1), 104.28.241.119/32(1), 104.28.241.120/30(4), 104.28.241.124/32(1), 136.23.3.230/32(1), 136.23.11.198/32(1), 140.248.56.169/32(1), 140.248.57.169/32(1), 140.248.58.169/32(1), 140.248.59.169/32(1), 140.248.60.169/32(1), 140.248.61.169/32(1), 140.248.62.169/32(1), 140.248.63.169/32(1), 146.75.132.106/31(2), 146.75.160.106/31(2), 146.75.190.8/31(2), 146.75.213.0/31(2), 150.48.167.126/31(2), 150.48.175.14/31(2), 150.48.175.16/29(8), 150.48.175.24/31(2), 150.48.175.46/31(2), 150.48.175.48/29(8), 157.5.67.161/32(1), 157.5.67.162/32(1), 157.5.73.225/32(1), 157.5.73.226/32(1), 157.5.80.208/31(2), 157.5.83.201/32(1), 157.5.83.202/32(1), 157.5.91.160/31(2), 162.120.204.56/32(1), 162.120.216.70/32(1), 172.225.65.160/27(32), 172.225.230.208/28(16), 172.225.244.144/28(16), 194.50.99.171/32(1),
Всього можлива кількість пристроїв в країні: 670

Рис 3.3.2 – інформація сформована скриптом формування статистики по країні

Аналогічним чином було сформовано інтерфейс взаємодії з списком міст світу. Проте в результаті список формувався надто довгий, незручний для використання, тому було прийняте рішення покращити методи взаємодії користувача з списком, через асинхронні запити з використанням JavaScript.

Ajax (Asynchronous JavaScript and XML) — це технологія, що дозволяє веб-сторінці виконувати асинхронні запити до сервера без необхідності повного перезавантаження сторінки. Основна ідея полягає в тому, що клієнтський браузер може отримувати дані у фоновому режимі, оновлювати окремі частини інтерфейсу та взаємодіяти з сервером динамічно, що забезпечує швидку та плавну роботу веб-застосунків.

Асинхронність означає, що програма не блокує виконання інших операцій під час очікування відповіді сервера. Запит відправляється у фоновому режимі, а відповідь обробляється у вигляді callback-функції або через механізм Promises/async/await. Завдяки цьому взаємодія з користувачем залишається безперервною.

На практиці Ajax використовується для таких задач, як автоматичне підвантаження даних, динамічне оновлення таблиць, формацій, чату, логів, безперервного моніторингу стану системи тощо. Сучасні формати даних включають JSON, що значно спростило обмін структурованою інформацією між клієнтом і сервером.

Таким чином, Ajax є ключовим механізмом створення інтерактивних, високопродуктивних веб-інтерфейсів і широко застосовується в сучасних веб-системах.

В конкретно цьому випадку було запропоновано дати можливість користувачу самому вводити частину імені міста англійською мовою, а система б формувала відповідний випадаючий список враховуючи введені символи. Це значною мірою б покращило зручність користування інтерфейсом взаємодії з базою даних.

Для цього було використано елемент типу ТЕКСТ у котрому користувач вводив частину назви міста.

```
    <INPUT type="text" placeholder="city of worls"
id="citynameworld" onkeyup="asuncCityWorld()">
```

onkeyup="asuncCityWorld()" – цей метод виклив JavaScript код котрий формував запит до PHP -скрипта з використанням введених даних , та результат відповіді поміщав на сторінку. Блок-схему результуючої взаємодії приведено у додатку 4.

```
    <script>
        function asuncCityWorld()
            {
                city=document.getElementById('citynameworld').value;
                request=new XMLHttpRequest();
                request.open("GET",
"select_city_of_world.php?&city="+city, true)
                request.onreadystatechange=function()
                {
                    if (this.readyState==4)
                    {
                        if (this.status==200)
                        {

document.getElementById('citiesworld').innerHTML=this.responseText;
                        }
                    }
                }
                request.send(null);
            }
    </script>
```

В результаті взаємодії цих скриптів користувач отримує можливість швидко і зручно шукати потрібні міста світу, результуючий вигляд елементу пошуку зображено на рисунку 3.3.3

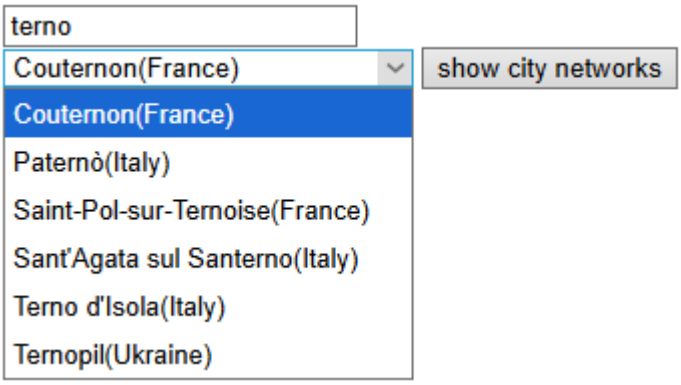


Рис. 3.3.3 – інтерфейс пошуку міст у таблиці міст

В результаті відправлення даних з веб-форми буде отримано статистику по вибраному місту, котра проілюстрована на рисунку 3.3.4

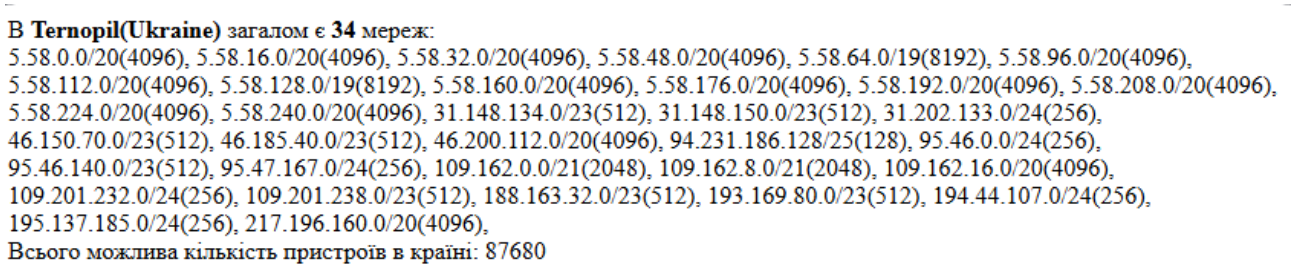

В **Ternopil(Ukraine)** загалом є **34** мереж:
5.58.0.0/20(4096), 5.58.16.0/20(4096), 5.58.32.0/20(4096), 5.58.48.0/20(4096), 5.58.64.0/19(8192), 5.58.96.0/20(4096), 5.58.112.0/20(4096), 5.58.128.0/19(8192), 5.58.160.0/20(4096), 5.58.176.0/20(4096), 5.58.192.0/20(4096), 5.58.208.0/20(4096), 5.58.224.0/20(4096), 5.58.240.0/20(4096), 31.148.134.0/23(512), 31.148.150.0/23(512), 31.202.133.0/24(256), 46.150.70.0/23(512), 46.185.40.0/23(512), 46.200.112.0/20(4096), 94.231.186.128/25(128), 95.46.0.0/24(256), 95.46.140.0/23(512), 95.47.167.0/24(256), 109.162.0.0/21(2048), 109.162.8.0/21(2048), 109.162.16.0/20(4096), 109.201.232.0/24(256), 109.201.238.0/23(512), 188.163.32.0/23(512), 193.169.80.0/23(512), 194.44.107.0/24(256), 195.137.185.0/24(256), 217.196.160.0/20(4096),
Всього можлива кількість пристроїв в країні: 87680

Рис. 3.3.4 – зразок виводу статистики по місту на прикладі Тернополя.

Наступним кроком було розширення статистичних даних по країні, а конкретно – виведення списку міст що розміщені в цій країні. Для цього було модифіковано файл що відповідає за вивід статистики по країні, було додано блок що отримує з бази даних також список міст а також кількість мереж у кожному місті.

```
SELECT id, city as name, (SELECT count(*)FROM ipv4city WHERE city=city.id) as counter_net FROM city WHERE country='$country_name' ORDER by city ASC;
```

В даному запиті ми отримуємо список міст для країни, вигляд списку проілюстровано на рисунку 3.3.5, а також підзапитом кількість мереж в цьому місті. ID міста вже проіндексоване в обидвох таблицях, покращити швидкодію запиту на цьому етапі нема можливості. Проте враховуючи результуючу швидкість отримання результатів запиту для користувача не будуть створюватися ніякі незручності повязані з очікуванням відповіді.

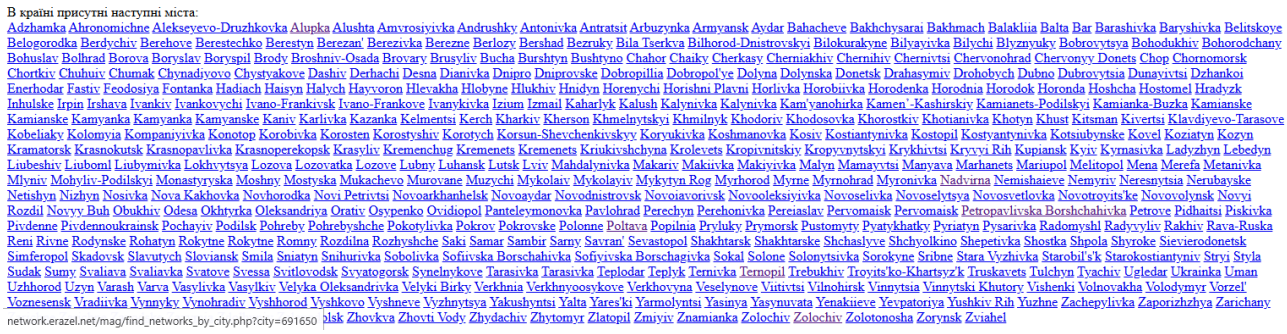


Рис. 3.3.5 – Результат виводу списку міст для вибраної країни.

Розширюючи охоплення статистики було реалізовано ще вивід 25 міст з найбільшою ймовірною кількістю пристроїв в мережах, приклад проілюстровано на рисунку 3.3.6

Рис. 3.3.6 – Результат виводу списку міст з набільшою кількістю пристроїв.

Назва кожного міста є покликанням на файл що формує статистику по даному місту. Так користувач може зручно і швидко отримувати інформацію по містах що його цікавлять.

Наступним кроком було вирішино зробити отримання аналогічної статистики по країнах , але зробити це в більш зручний для користувача спосіб, а саме створити веб-форму з котрої користувач міг би отримати список країн за певними критеріями на вибір користувача, що проілюстровано на рисунку 3.3.7

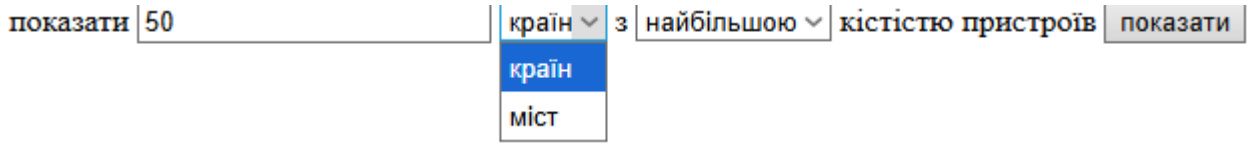


Рис. 3.3.7 – Форма вибору статистистики.

На основі даних переданих користувачем формується запит до бази даних

```
SELECT SUM(POW(2,(32-ipv4city.mask))) as hosts, city FROM ipv4city GROUP by city Order by hosts $_POST[ascening] limit $_POST[number_items]
```

Проте даний запит на таблиці з 3.5 мільйона записів займає суттєвий час, до 5с. Була висунута гіпотеза що для зменшення часу виконання запитів можна частину обрахунків провести наперед і вписати наперед обраховані значення у нову комірку, це збільшить розмір таблиці, але зменшить час виконання складних запитів. З метою підтвердження цієї гіпотези було проведено модифікацію таблиці двома SQL-запитами:

```
ALTER TABLE `ipv4city` ADD `num_hosts` INT UNSIGNED NOT NULL DEF AULT '1' AFTER `mask`;
UPDATE ipv4city SET num_hosts=POW(2,(32-mask))
```

Це заповнило поле num_hosts кількістю хостів що можлива при розмірі префіксу мережі в mask(2 в степені “32 -mask”). Після чого було виконано два заміри щоб перевірити наскільки змінився час виконання запитів з наперед обрахованим значенням.

```
SELECT SUM(POW(2,(32-ipv4city.mask))) as summ FROM ipv4city GROUP by city Order by summ DESC LIMIT 25
```

зайняв 3,8 секунди, а запит

```
SELECT SUM(num_hosts) as summ FROM ipv4city GROUP by city Order by summ DESC LIMIT 25
```

зайняв 3,9 секунди. Перевірочні виконання запитів показали що час відповіді статистично значуще не відрізняється. Отже потреби в розширені таблиці, дані в новоствореній комірці були визнані надлишковими і в той ж час не покращуючими швидкість виконання запитів, тому було проведення видаленя даної комірки `ALTER TABLE `ipv4city` DROP `num_hosts`;`

3.4 Створення конструктора правил для мережевого обладнання.

Сучасні мережеві інфраструктури характеризуються значним зростанням кількості підключень, високою динамічністю маршрутів та необхідністю швидкої реакції на зміни в глобальній мережі. У таких умовах ручне створення та підтримка правил маршрутизації, фільтрації чи пріоритезації трафіку стає трудомістким, повільним та схильним до помилок. Особливо це стосується систем, де ключову роль відіграють географічні ознаки IP-адрес, наприклад: країна, регіон, провайдер або конкретне місто походження мережного блоку.

Використання автоматизованих конструкторів правил, які працюють на основі таблиць географічного розміщення мереж, забезпечує низку суттєвих переваг, що робить їх особливо корисними у сучасних системах оптимізації маршрутизації та керування мережею.

Усунення людського фактору та зменшення кількості помилок

Ручне створення фаєрвол-правил або ACL (Access Control List) передбачає аналіз сотень або тисяч IP-адрес, що майже неминуче призводить до логічних та синтаксичних помилок.Автоматизований конструктор:

- формує правила на основі точних даних з бази IP-мереж;
- виключає ризик неправильного введення адреси;
- забезпечує ідентичність форматування правил;
- дозволяє швидко перевіряти коректність масок, діапазонів та умов фільтрації;

Таким чином, зменшується кількість помилок конфігурації, що позитивно впливає на стабільність роботи мережі.

Значне прискорення обробки великих обсягів даних

Сучасні геобази можуть містити мільйони мережних блоків. Формування правил для блокування або маршрутизації на основі таких даних вручну практично неможливе.Автоматизований конструктор має ряд переваг:

- миттєво знаходить необхідні IP-блоки;
- здатний будувати конфігурації для різних типів обладнання (Cisco, MikroTik, Juniper тощо).

Це забезпечує швидкість у десятки й сотні разів більшу, ніж ручна робота.

Покращення кібербезпеки та оперативності реагування

У разі необхідності негайно заблокувати:

- небезпечний трафік з певної країни;
- діапазони IP, пов’язані з ботнетами;
- підозрілу активність певних регіонів;

автоматична система може згенерувати правила за секунди, що критично важливо у випадку DDoS-атак або інцидентів безпеки.

Виходячи з вище-наведених висновків було прийнято рішення спробувати реалізувати систему генерації чстини кофігурації(маршрути, правила фільтру файрвола та інш.) для мережевого обладнання. В якості цільового мережевого обладнання було обрано Mikrotik як таке що дозволяє легко імпортувати файли внесення змін конфігурації та в той ж час має можливість автоматично самостійно запитувати дані з вузла керування(в випадку реалізації такого механізму адміністратором пристрою).

Тому на сторінці взаємодії користувача з базою даних було додано ще декілька наступних пунктів.

Генерація маршрутів на певний географічний об’єкт через заданий шлюз. А також генерація адреслисту включаючого адреси котрі належать певному географічному об’єкту.

Для цієї задачі було створено автоматизовану форму , котра враховуючи вибори користувача формує запит до бази даних для створення команд конфігурації мережевого обладнання. Вигляд форми для вищенаведених двох варіантів проілюстровано на рисунку 3.4.1

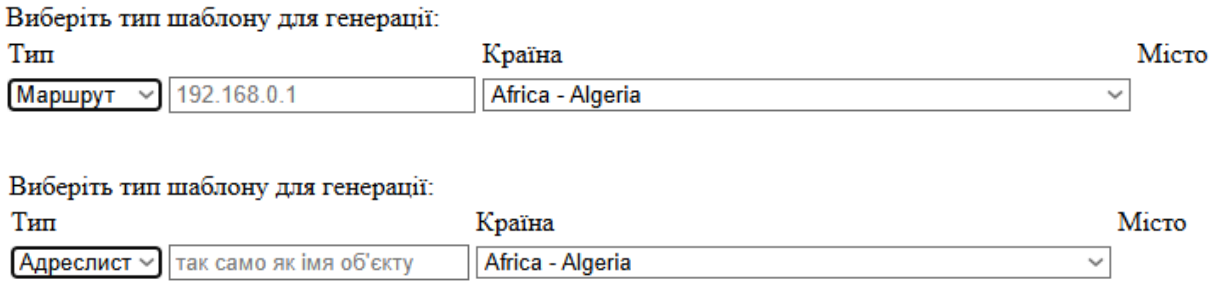


Рис. 3.4.1. – Зразки форми створення шаблону правил.

Спочатку користувач обирає тип правил – створення маршрутів чи створення адрес-листа(використовується для шаблонів у фільтрі файрволу мережевого обладнання, дозволяє вказувати не конкретну адресу, а загальний лист адрес), даний елемент форми проілюстровано на рисунку 3.4.2.

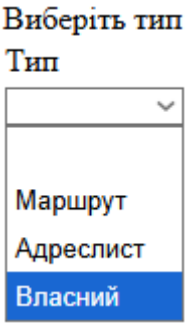


Рис. 3.4.2 – вибір типу правил

Відносно цього вибору у формі формуються наступні елементи – поле з введенням адреси шлюзу для маршруту чи поле з іменем адреслиста.

Наступним кроком користувач у випадаючому списку вибирає потрібну країну. Відно вибраної країни формується випадаючий список міст цієї країни, де користувач може обрати або всі міста країни, або конкретне місто. Результуючи вигляд елементу вибору міста згідно вибраної країни проілюстровано на рисунку 3.4.3

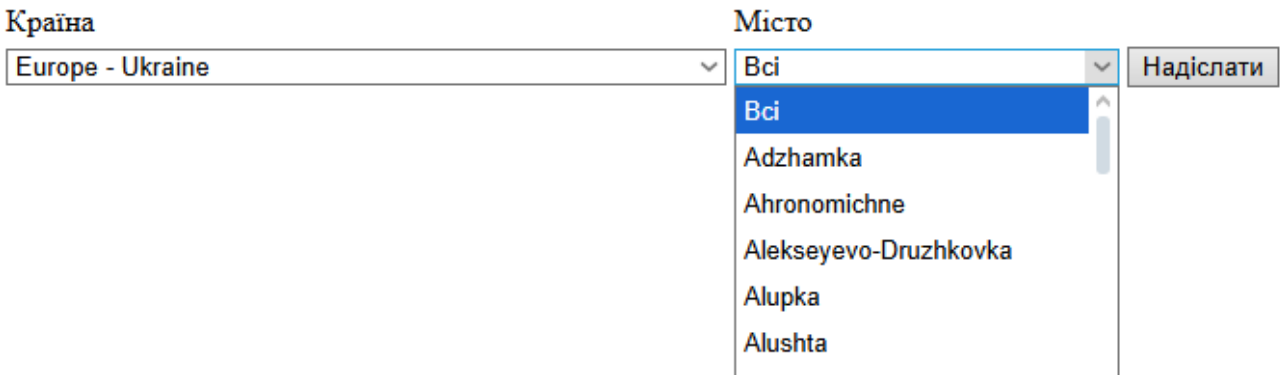


Рис. 3.4.3 – Вибір міста відносно вибраної країни

Після цього кнопкою користувач підтверджує вибір і у відповідь формується набір правил згідно шаблону, котрі користувач може самостійно скопіювати на обладнання.Результат відповіді на вибір користувача проілюстрована на рисунку 3.4.4

```
/ip firewall address-list
add address=5.58.0.0/20 list=Ternopil
add address=5.58.16.0/20 list=Ternopil
add address=5.58.32.0/20 list=Ternopil
add address=5.58.48.0/20 list=Ternopil
add address=5.58.64.0/19 list=Ternopil
add address=5.58.96.0/20 list=Ternopil
```

Рис. 3.4.4 – зразок сформованих правил.

Первинна взаємодія в формі створення шаблону ведеться лише з невеликими таблицями з списком країн та міст, котрі через свій відносно невеликий розмір забезпечують досить швидку відповідь. Тому було реалізовано динамічне перестворення списку міст на зміну вибраної країни. Це дозволило покращити зручність роботи користувача без вагомих втрат швидкодії, адже результуючий час відповіді на дії користувача вимірювався в десятитисячних долях секунди, так що можна стверджувати що користувач не спостерігатиме ніяких затримок реакції інтерфейсу на свої дії.

В даний момент реалізовано лише синтаксис для обладнання Mikrotik, але при потребі користувач може обрати шаблон «власний», зразок шаблону проілюстровано на рисунку 3.5.5, та створити власний шаблон, вписуючи текст та підставляючи в потрібних місцях адресу чи імя об’єкту:

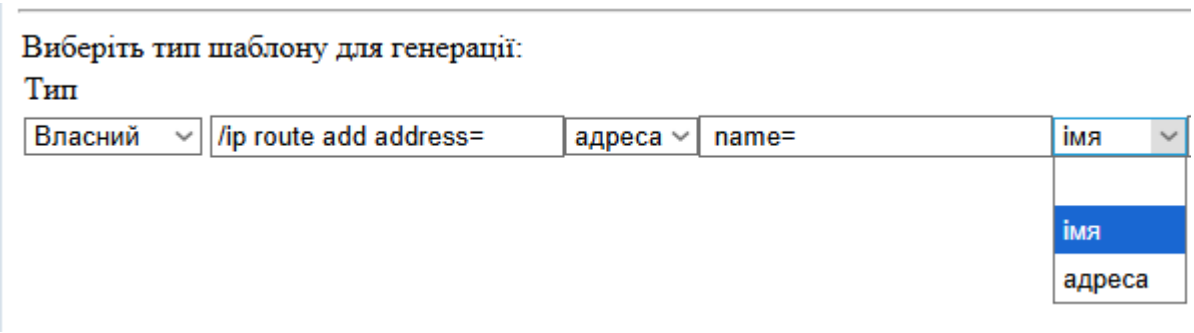


Рис. 3.4.5 – Зразок шаблону правил «власний»

Таким чином користувач зможе формувати правила для любого обладнання котре підтримує ввід текстових команд конфігурування. Лістинг файлів на мові PHP приведено у додатку 5.

3.5. Реалізація автоматизованої взаємодії з мережевим обладнанням.

Мережеве обладнання Mikrotik з операційною системою ROS має можливість виконувати запити по протоколам HTTP, HTTPS, FTP та інш. Це дозволяє при потребі отримувати дані в автоматичному режимі без необхідності вводу їх вручну лише через команду виконання запиту, або навіть ініціалізувати виконання запиту в автоматичному режимі по планувальнику завдань.

Для початку було реалізовано отримання команд створення адреслиста або маршрутів з самого мережевого пристрою, без потреби копіювання з компютера. Алгоритм дій даного застосунку наведено в додатку 7.

Користувачу надається готовий скрипт котрий він може виконати на своєму обладнанні, лише вказавши потрібну країну, скрипт конфігурування наведено в лістингу 3.5.1

Лістинг 3.5.1 – скрипт отримання адреслиста чи маршрутів для Mikrotik

```
:local country "Ukraine";
:local type "adrlist";
:local gateway "192.168.0.1";
/tool fetch
url="http://network.erazel.net/mag/mikrotik_get_country.php?t=$type&
g=$gateway&c=$country&tone=script" dst-path="getlist.rsc" mode=http;
```

```
/import filename=getlist.rsc;
delay 5s;
/file remove getlist.rsc;
```

Користувач може задаючи змінні на початку скрипта вказати країну для котрої він хоче отримати дані та вказати чи адреслист чи маршрути він хоче отримати. При виконанні цей скрипт виконає роботу в автоматичному режимі без потреба взаємодії з користувачем.

Це дозволить міняючи лише назву країни отримувати в автоматичному режимі адреслисти або маршрути для бажаних країн, без потреби переносити дані з компютера. Також пошук по країні відбувається не суворий, відбувається пошук входження введеної назви країни в назви країн в відповідній таблиці бази даних.

Наступним кроком була реалізація динамічного розпізнавання адрес в країни походження. На обладнанні мікротік адреслисти можуть формуватися динамічно, правилами файрволу згідно заданих умов. Зразок адреслиста проілюстровано на рисунку 3.5.1

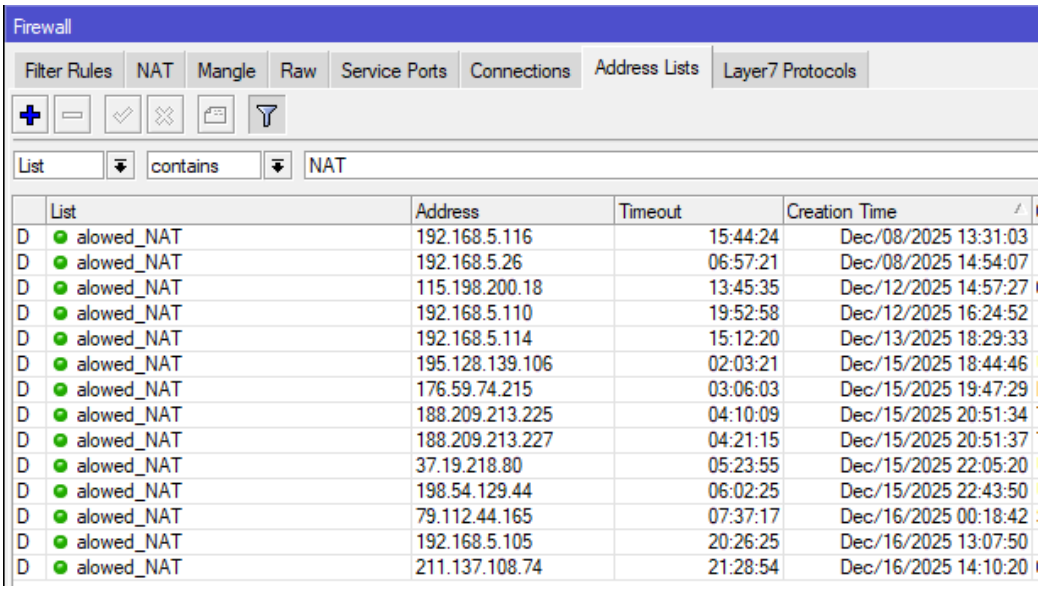


Рис. 3.5.1 – динамічно створені адреслисти на обладнанні Mikrotik

Проте інформативність даних записів можна розширити вписуючи в поле коментарів ще й країну походження адреси. Для вирішення цього завдання було створено скрипт проілюстрований на лістингу 3.5.2.

Лістинг 3.5.2. – Скрипт динамічного отримання країни походження адреси

```
:local listname "white"
:foreach i in=[/ip/firewall/address-list find list=$listname]
```

```
do={
    :global adr [/ip firewall/address-list/ get $i address ];
    :local result [/tool fetch
url="http://url="http://network.erazel.net/mag/mikrotik_get_country.
php?tone=adrlist&a=$adr mode=http as-value output=user];
    :local country ($result->"data");
    /ip firewall/address-list/set comment=$country $i
    }
```

Користувач при створенні скрипта лише задає в змінній ім’я адреслиста, а подальшу роботу скрипт виконує в автоматичному режимі при запуску.Результуючий вигляд завдання в планувальнику завдань на обладнанні Mikrotik проілюстровано на рисунку 3.5.2

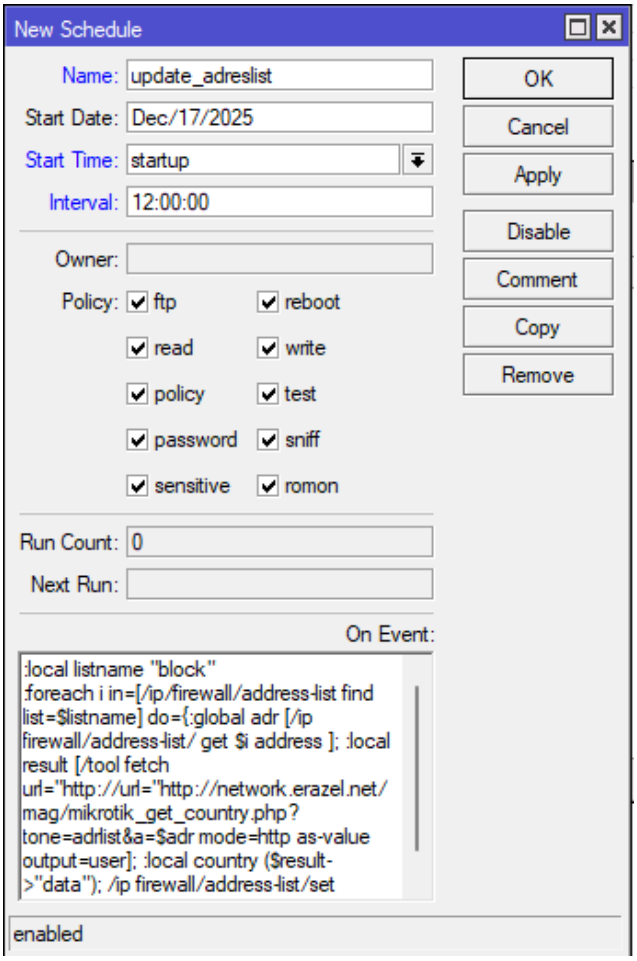


Рис. 3.5.2 – запис в планувальнику завдань Mikrotik

Скрипт проходить по всіх записах заданого адреслиста, отримує адресу, запитує у бази даних країну та вписує її в поле коментаря відповідного запису адреслиста. В результаті вигляд списку адрес стає більш наочний та зручний для

перегляду людиною, стає комфортніше розуміти географічні походження адрес в адреслисті і приймати відповідні дії. Результуючий вигляд адреслиста після виконання даного скрипта проілюстровано на рисунку 3.5.3

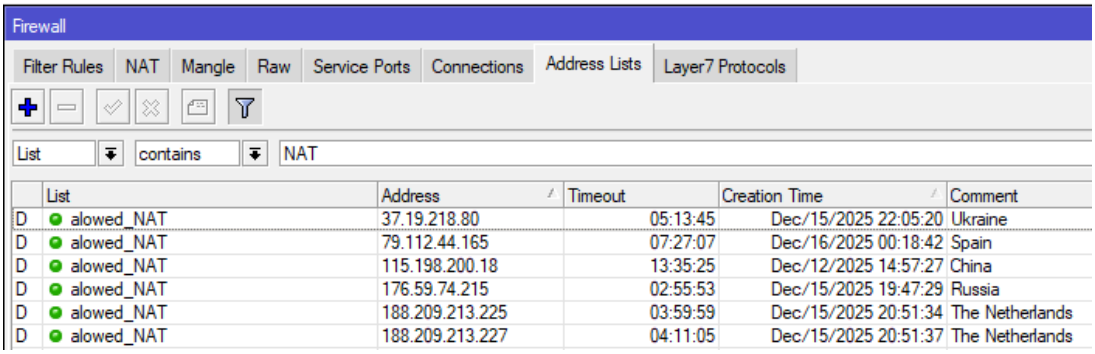


Рис. 3.5.3 – вигляд адреслиста після виконання скрипта

Лістинг серверних застосунків на мові PHP, котрі відповідають за взаємодію обладнання mikrotik з базою даних, приведено в додатку 6.

Розділ 4: НАУКОВО-ДОСЛІДНА ЧАСТИНА

4.1. Характеристика об’єкту або предмету дослідження.

Було вирішено дослідити можливість покращення швидкодії SQL-запитів при виконанні пошуку входження IP-адрес та адрес IP-мереж у діапазони адрес IP-мереж, тобто пошук входженні IP-адреси в мережу, чи входження меншої IP-мережі у більшу

Особливості зберігання IPv4-адрес у СУБД

Адреси протоколу IPv4 займають 32 біти та можуть бути представлені у реляційних базах даних у кількох форматах: рядковому (VARCHAR), багатобайтовому (BINARY(4)), або як 4-байтове ціле число (INT UNSIGNED). Найефективнішим для індексування є саме 4-байтний числовий формат, оскільки він дозволяє використовувати операції порівняння та діапазонні запити з мінімальними накладними витратами.

У системах, де спостерігається велика кількість записів з IP-адресами (журнали трафіку, лог-файли вебсерверів, геолокаційні бази), скорочення часу пошуку відіграє ключову роль. Тому важливим аспектом оптимізації є вибір правильної стратегії індексування.

Але, на мою думку, представлення та індексування по 4-байтовому цілому значенню(вся IPv4-адреса) хоч є найбільш поширенною практикою, та набуло широкого використання , може бути оптимізоване за певних умов.

4.2. Програма і методика теоретичних та експериментальних досліджень.

Перш за все мною було проведене дослідження розміру префіксів/масок у таблиці розташування IP-адрес по країнах. Для цього я зробив SQL-запит щоб відсортувало наявні маски/префікси за розміром: `SELECT mask, COUNT(mask) as countt FROM `ipv4country` GROUP BY mask ORDER by mask ASC Limit 10;`

Результат виконання скрипта зображено на рисунку 4.2.1

mask ▲ 1	countt
8	8
9	10
10	35
11	141
12	380
13	762
14	1603
15	3225
16	8441
17	5994

Рис. 4.2.1 – результат виконання SQL-запиту сортування мережевих масок.

Отже з результатів можна зрозуміти що сама «вузька», тобто конкретна мережева маска має розмір 8 біт. Виходячи з цього можна стверджувати що можна індексувати і по перших 8 бітах мережевої адреси, так як не буде випадку коли значущими будуть перші X біт але не наступні X+1 ... 8 біт.

Виходячи з цього мною було прийняте рішення створити додатковий індекс для кожного значення по перших 8 бітах адреси. 8 біт зручні ще й тим що це перший октет адреси у повному виді, так що дуже легко його отримувати з адреси через розбиття адреси методами роботи з текстовими рядками. Що в подальшому дуже спростить роботу , якщо буде використовуватися індексування по першому байту(8 бітах) адреси. Для отримання значення першого байту адреси було використано наступний PHP-код:

```
list($ip, $prefix) = explode('/', $ip_addr);
list($first_octet, $b, $c, $d) = explode('.', $ip);
```

Також було модифіковано таблиці географічної приналежності мереж, створено додаткове поле , в котрому зберігалось значення першого байту адреси:

```
ALTER TABLE `ipv4country` ADD `first_octet` TINYINT UNSIGNED NOT NUL
L DEFAULT '0' AFTER `mask`, ADD INDEX (`first_octet`);
ALTER TABLE `ipv4city` ADD `first_octet` TINYINT UNSIGNED NOT NULL D
EFAULT '0' AFTER `mask`, ADD INDEX (`first_octet`);
```

Після чого для заповнення даних виконуємо команди:

```
UPDATE ipv4country SET first_octet =
SUBSTRING_INDEX(SUBSTRING_INDEX(ipnetwork, '/', 1), '.', 1);
```

та

```
UPDATE ipv4city SET first_octet =
SUBSTRING_INDEX(SUBSTRING_INDEX(ipnetwork, '/', 1), '.', 1);
```

Ці команди оновлюють дані в відповідних комірках таблиць. SUBSTRING_INDEX(ipnetwork, '/', 1) – отримує дані до символу «/», тобто IP-адресу, а команда SUBSTRING_INDEX(..., '.', 1) – отримує з цієї IP-адреси перший октет. Після чого отримані дані записуються в відповідну комірку таблиці.

Зразок вмісту таблиць після виконання цих команд проілюстровано на рисунках 4.2.2 та 4.2.3

id	ipnetwork	mask	first_octet	begin	end	city	country	latitude	longitude
2983777	185.29.80.0/23	23	185	3105705984	3105706495	721472	719819	47.5232	21.6334
2983778	185.29.82.0/23	23	185	3105706496	3105707007	3054643	719819	47.5000	19.0412
2983779	185.29.84.0/22	22	185	3105707008	3105708031	756280	798544	49.4266	22.5880
2983780	185.29.88.0/24	24	185	3105708032	3105708287	2781371	2782113	47.4194	15.2842

Рис. 4.2.2 – вміст таблиці ipv4city

id	ipnetwork	mask	first_octet	begin	end	geoid
463026	183.177.79.0/24	24	183	3081850624	3081850879	1819730
463027	183.177.80.0/24	24	183	3081850880	3081851135	1819730
463028	183.177.81.0/26	26	183	3081851136	3081851199	1819730
463029	183.177.81.64/26	26	183	3081851200	3081851263	1861060

Рис. 4.2.3 – вміст таблиці ipv4country

Також було створено поле first_octet_chr типу varchar з розміро 3 символи , куди теж записував перший октет адреси, але вже у текстовому виді, а не числовому. Дане поле було утворено для перевірки впливу формату зберіганні даних в полі на швидкість виконання пошуку при використанні індексу по цьому полю.

Наступним кроком було заплановане дослідження порівняння швидкості вибірки з бази даних з різними режимами індексування даних:

1. **Без індексу**
2. **Індексування за повною IPv4-адресою (4-байтовий integer)**
3. **Комбіноване індексування за першим октетом (3 байти) та повною адресою**
4. **Комбіноване індексування за першим октетом (1 байт) та повною адресою**

Комбіноване індексування передбачає наявність двох індексів:

- індекс по полю first_octet (1 байт) , або індекс по полю first_octet_chr (3 байти)
- індекс по полю ip4 (4 байти).

Цей підхід дозволяє здійснювати попередній швидкий відбір потенційних кандидатів, зменшуючи навантаження на основне B-дерево.

Висота індексного дерева та кількість порівнянь

Довжина ключа та кількість елементів напряму впливають на висоту B-дерева, що визначає кількість порівнянь при пошуку. Для таблиці обсягом понад 50 млн записів у форматі InnoDB індекс зазвичай має висоту 3–4 рівні.

Індексування по одному октету фактично розбиває всю таблицю на 256 незалежних сегментів. Це скорочує обсяг пошуку у кожному сегменті приблизно до:

$$\frac{N}{256}$$

що суттєво зменшує кількість операцій порівняння.

Селективність при діапазонних запитах

У більшості реальних задач (логування, аналітика трафіку, виділення CIDR-діапазонів) фільтрація виконується не за однією конкретною IP-адресою, а за мережами, наприклад:

- 192.168.0.0/16,
- 10.0.0.0/8,
- 172.16.0.0/12.

За наявності індексу лише по 4-байтовому значенню СУБД змушена виконувати діапазонний пошук у масштабі всієї таблиці. Індекс по першому октету дозволяє здійснити перший етап швидкого групування, різко зменшуючи обсяг даних.

Покращення роботи кешу та зменшення I/O

Індекси з меншим ключем:

- ефективніше кешуються у buffer pool,

- займають менше сторінок B-дерева,
- рідше спричиняють disk seeks,
- менше навантажують L2/L3 кеш CPU.

Отже, попередній фільтр по першому октету зменшує обсяг операцій з пам’яттю та диском.

4.3. Обробка результатів досліджень.

Було проведено тестування швидкості виконання запитів:

- без використання проіндексованих полів(рядок 1 таблиць);
- з використанням Between і двох 4байтових цілих значень на котрі було створено індекс(рядок 2 таблиць);
- з використанням індексу полю типу varchar(3) (перший октет) та додатково використанням Between і двох 4байтових цілих значень на котрі було створено індекс(рядок 3 таблиць);
- з використанням індексу по 1байтовому полю(перший октет) та додатково використанням Between і двох 4байтових цілих значень на котрі було створено індекс(рядок 4 таблиць);

Результати замірів представлені в рисунках 4.3.1, 4.3.2 та таблицях 4.3.1 і 4.3.2.

Табл. 4.3.1 – Час виконання запитів у талиці бази даних на 617 тис. записів

1	`SELECT * FROM `ipv4country` WHERE `ipnetwork`='183.177.79.0/24';`	0,2347
2	`SELECT * FROM `ipv4country` WHERE (INET_ATON('183.177.79.25') BETWEEN`begin` AND `end`);`	0,1804
3	`SELECT * FROM `ipv4country` WHERE `first_octet_chr`='183' AND (INET_ATON('183.177.79.25') BETWEEN`begin` AND `end`)`	0,0047
4	`SELECT * FROM `ipv4country` WHERE `first_octet`='183' AND (INET_ATON('183.177.79.25') BETWEEN`begin` AND `end`)`	0,0019

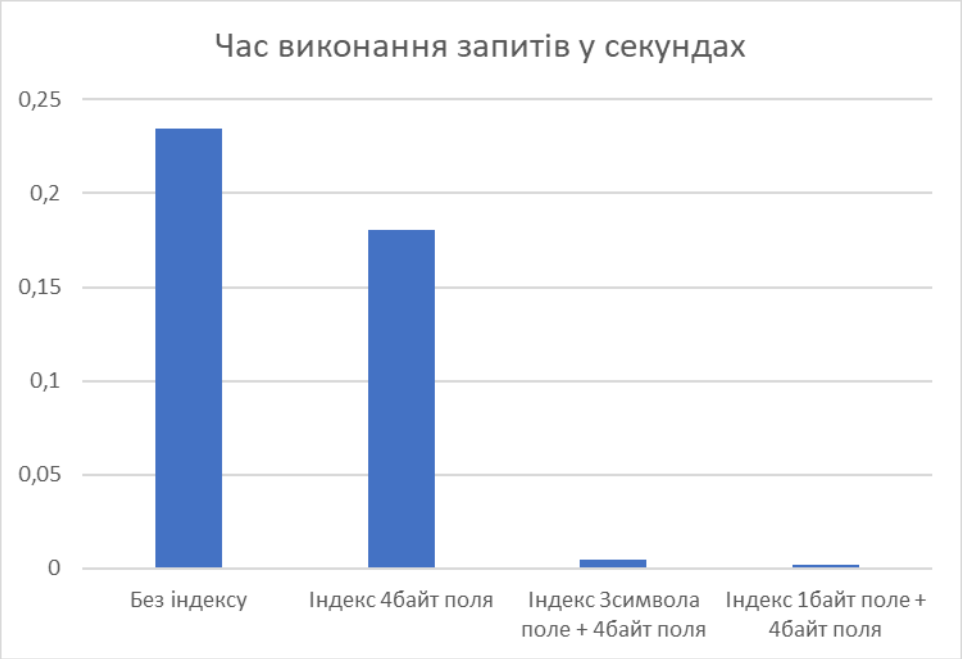


Рис. 4.3.1 - Час виконання запитів у талиці бази даних на 617 тис. Записів

Табл. 4.3.2 – Час виконання запитів у талиці бази даних на 3.5 млн. записів

1	`SELECT * FROM ipv4city WHERE ipnetwork='108.195.188.0/22';`	5,51230
2	`SELECT * FROM ipv4city WHERE (INET_ATON('108.195.188.45') BETWEEN begin AND end);`	0,34620
3	`SELECT * FROM ipv4city WHERE (first_octet_chr='108' AND (INET_ATON('108.195.188.45') BETWEEN begin AND end));`	0,1176
4	`SELECT * FROM ipv4city WHERE (first_octet='108' AND (INET_ATON('108.195.188.45') BETWEEN begin AND end));`	0,07290

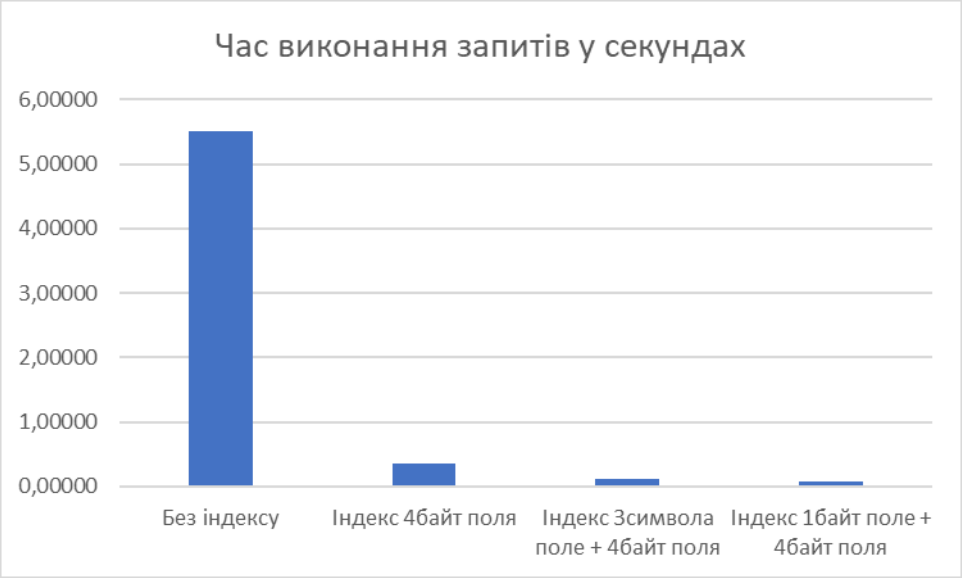


Рис. 4.3.2 - Час виконання запитів у талиці бази даних на 3.5 млн. записів

4.4. Аналіз і узагальнення отриманої інформації.

Проведений аналіз показує, що комбінована стратегія індексування IPv4-адрес за першим октетом та повним 4-байтовим значенням є найбільш ефективною для застосувань, пов’язаних із високонавантаженими системами зберігання логів, мережевою аналітикою та обробкою трафіку. Основними причинами підвищеної продуктивності є:

1. Зниження кардинальності пошуку – поділ таблиці на 256 підгруп.
2. Зменшення висоти B-дерева – скорочення кількості порівнянь та доступів до сторінок InnoDB.
3. Покращене кешування – менший розмір ключів та сегментованість даних.
4. Оптимізація діапазонних запитів – істотне прискорення при роботі з CIDR-блоками.

Комбіноване індексування може забезпечувати прискорення виконання типових запитів у 5–100 разів порівняно з використанням одного лише індексу за повною IPv4-адресою.

Таким чином для зберігання IP-адрес та IP-мереж є оптимальним використовувати надлишкові дані малого розміру. Це збільшує розмір таблиці, але значно, інколи на порядок, зменшує час пошуку при використанні комбінованих індексів. Що, на мою думку, є досить вагомою перевагою при

роботі з таблицями великого розміру(100 000 записів і більше), оскільки збільше витрат місця на накопичувачав вимірюється відсотками, а приріст швидкодії сягає сотень, а в окремих випадках і тисяч, відсотків.

Тому є практичний зміст витатити час на невелике ускладнення структури даних у таблиці БД, але в результаті досить вагомо зекономити час при використанні у готовому програмному засобі.

РОЗДІЛ 5. СПЕЦІАЛЬНА ЧАСТИНА.

У даному проекті, як було зазначено в попередніх розділах, використовуються мова програмування PHP, СУБД MySQL, та мережеве обладнання Mikrotik під керуванням операційної системи ROS v7.

5.1. Використовуєма мова програмування PHP

Програмна частина проекту реалізована на мові програмування PHP. Для роботи з даною мовою було застосовано середовище розробки Apache NetBeans v27, котре є вільно поширюваним програмним засобом. Вигляд базового вікна продемонстровано на рисунку 5.1.1

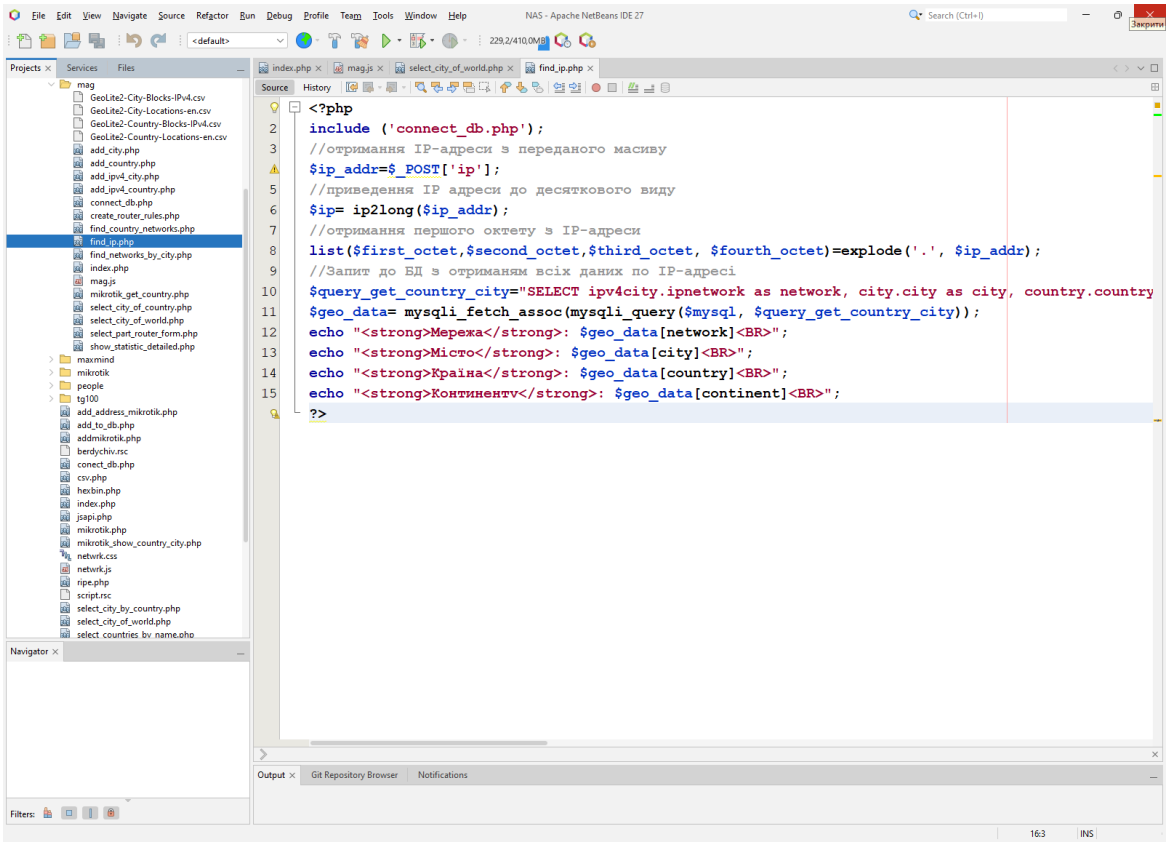


Рис. 5.1.1 – Вигляд вікна IDE Apache NetBeans 27

Дане середовище розробки дозволяє працювти з цілим рядом мов програмування, а саме:

- Java
- C/C++
- HTML

- CSS
- JavaScript
- PHP

Оскільки PHP є інтерпритованою, а не компільованою мовою то середовищу розробки потрібно вказати де знаходиться файл інтерпритатора для потрібної версії мови програмування. Після цього всі помилки в коді будуть автоматично підсвічуватися і їх можна буде одразу виправити без необхідності завантажувати код на веб-сервер і переглядати його звіти про помилки. Цей механізм автоматичного підсвічування синтаксисичних помилок має досить великий ергономічний і економічний ефекти, оскільки економить час програміста, а також покращує зручність роботи з кодом, що своєю чергою позитивно впливає на продуктивність роботи програміста.

Як було вказано в переліку підтримуємих мов програмування мова програмування JavaScript теж підтримується даним середовищем, що зображено на рисунку 5.1.2

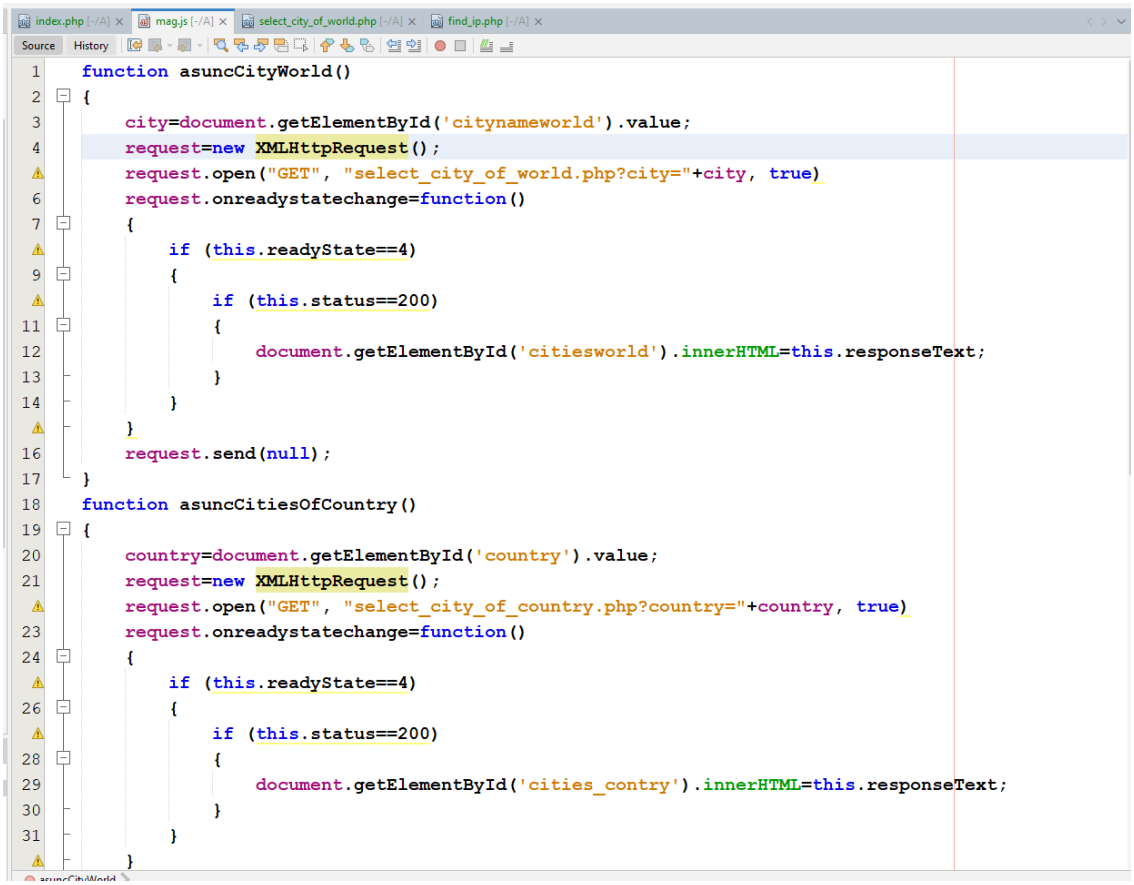


Рис. 5.1.2 – вікно програми з файлом mag.js

Це підвищило зручність, оскільки в проекті певною мірою теж застосовувались скрипти на мові JavaScript і не було потреби у використанні кількох середовищ розробки. Вся робота велась в одному середовищі, з уніфікованим набором файлів.

Головною перевагою було те що середовище не тільки має підсвітку синтаксису а й автозавершення процедур, операторів, функцій, а також імен змінних, що дозволяло уникати помилок введення при активній роботі. Адже найбільша проблема це одруківки в іменах змінних, так як це не висвічується як помилка і дуже складно знайти таку помилку в вихідному коді програми.

5.2. Робота з СУБД MySQL

У якості БД для проекту було застосовано MySQL. Ця БД має швидку роботу, легкість налаштувань, гнучкість конфігурування, але вона не має зручного графічного інтерфейсу для взаємодії з нею. Вся робота проводиться через текстову консоль , що досить незручно при роботі з великими проектами. Тому користувачами була розроблена реалізація інтерфейсу взаємодії з СУБД MySQL на мові PHP, котра отримала назву PhpMyAdmin. Вигляд вікна цього інтерфейсу продемонстровано на рисунку 5.2.1

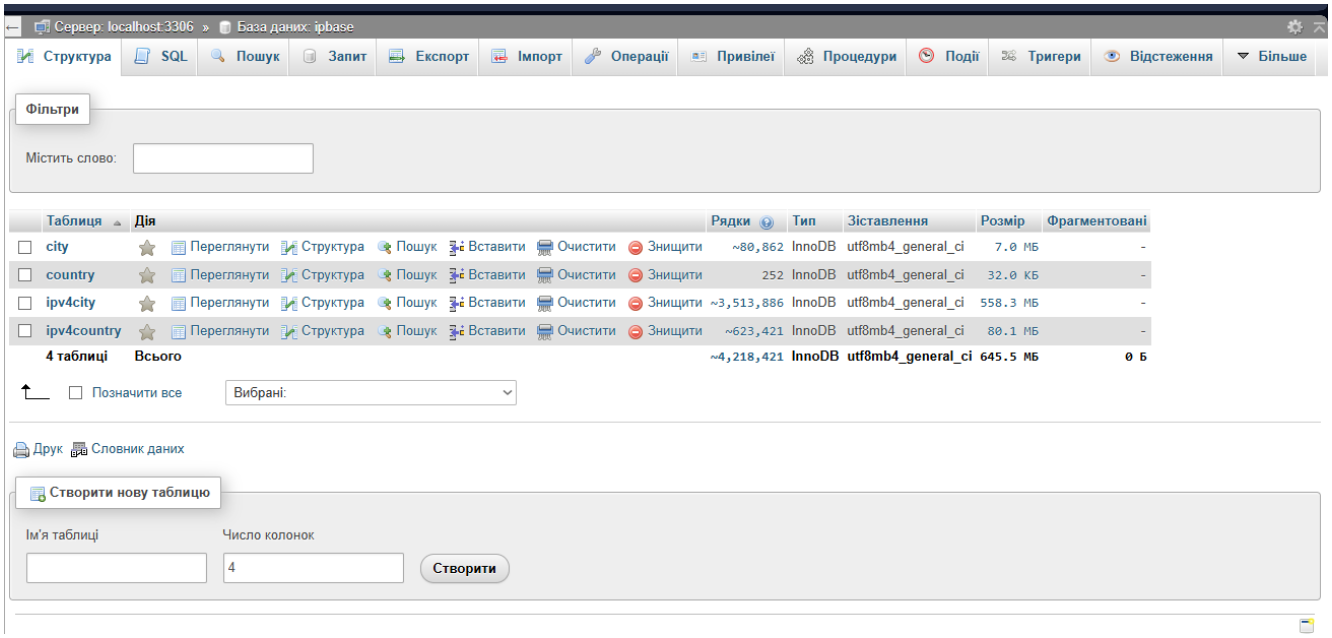


Рис. 5.2.1 – вигляд вікна PhpMyAdmin.

Дане середовище виконується на стороні вебсервера, тому потрібно мати веб-сервер з доступом до потрібної бази даних. Після вказаня параметрів підключення до СУБД MySQL отримується доступ до інструментів керування та взаємодії з базою даних. Можна створювати і видаляти бази даних, створювати, редагувати, видаляти таблиці в потрібних базах даних, а також виконувати SQL запити (зразок побудови запиту проілюстровано на рисунку 5.2.2)

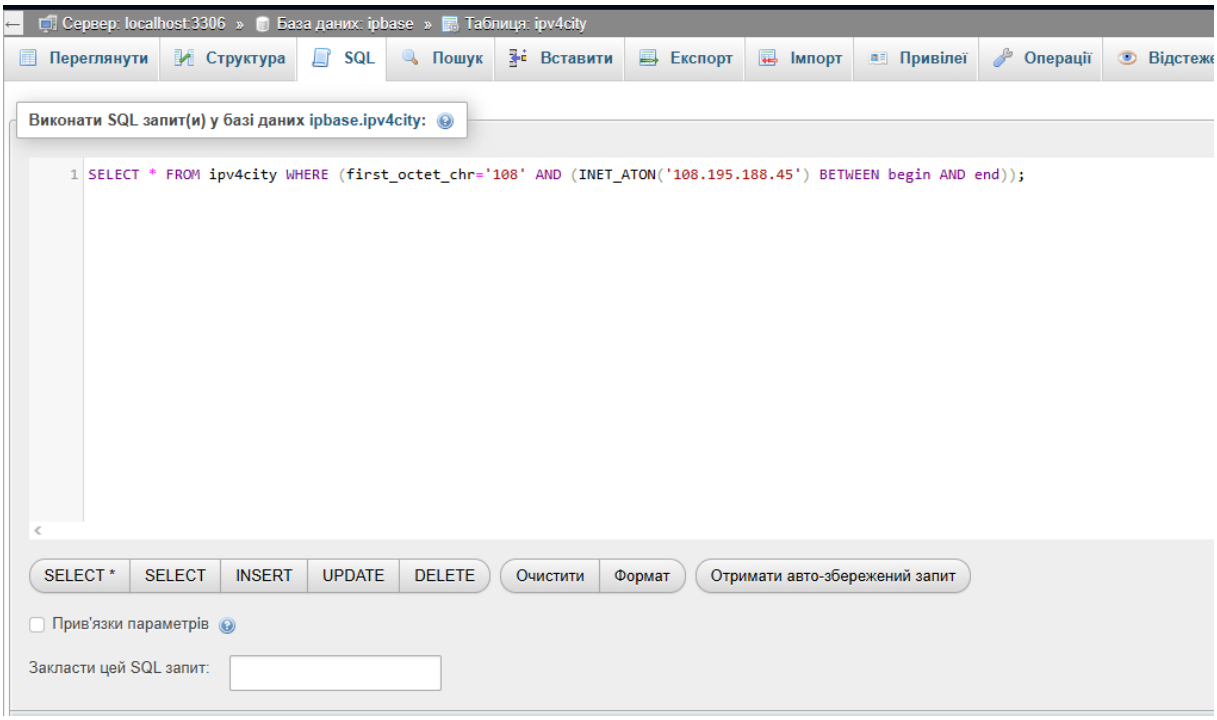


Рис. 5.2.2 – Зразок форми SQL-запиту.

Як і середовище розробки з попереднього розділу середовище PhpMyAdmin пропонує зручну можливість автозавершення імен баз даних, таблиць, полів в таблицях. Це дозволяє уникати одруківок та не вимагає знати напамять імена всіх полів в таблицях. Що досить пришвидшує роботу з складними багаторівневими запитами, оскільки.

Також до переваг даного середовища варто віднести можливість легко і швидко робити резервні копії потрібних даних перед важливими змінами в структуру таблиць або даних в таблицях, що продемонстрованор на рисунку 5.2.3.

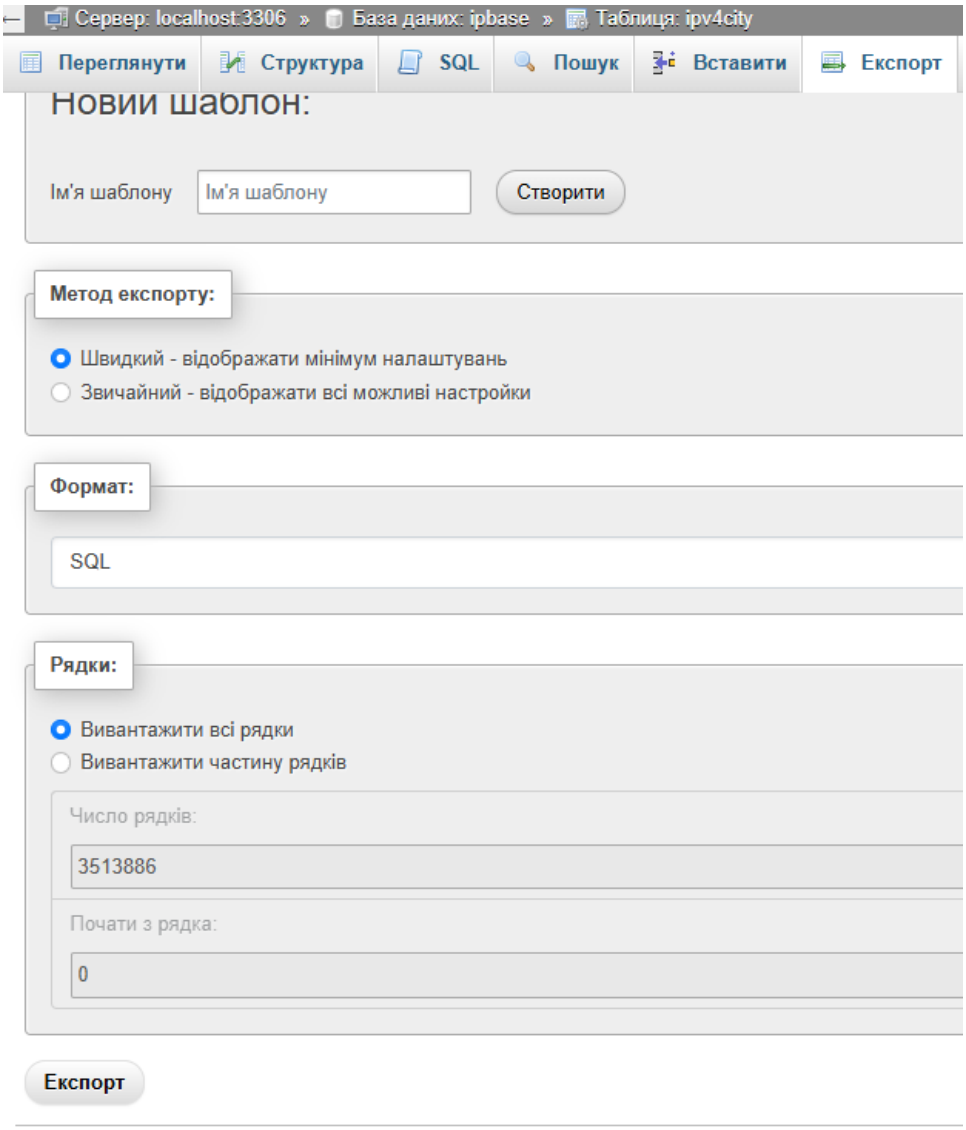


Рис. 5.2.3 – експорт даних в PhpMyAdmin

Загалом дане середовище надає всі можливості повноцінних систем взаємодії з базами даних, але без необхідності встановлювати на локальному компютері, вся робота відбувається централізовано на сервері, що можна віднести як до переваг так і до недоліків цього програмного засобу.

5.3. Мережеве обладнання Mikrotik під керуванням ROS v7.

В даній роботі було реалізовано автоматизовану взаємодію мережевого обладнання з базою даних. В якості екземпляра продвинутого мережевого обладнання виступив маршрутизатор Mikrotik RB5009 під керуванням операційної системи ROS V 7.20.4.

У виробника мережевого обладнання Mikrotik присутні дві операційні системи, SWOS та ROS. SWOS – це проста операційна система з достатньо базовим функціоналом котра використовується на комутаторах початкового і середнього рівня. Взаємодія відбувається черед веб-інтерфейс. Великої цікавості дана операційна система не представляє , адже надає досить обмежений функціонал. На відміну від неї ROS – Router Board Operation System, має куди ширший функціонал та використовується у всій лінійці випускаємої продукції Mikrotik, від найдешевших пристроїв ціною в 20 доларів США, до пристроїв корпоративного рівня ціною в 2-3 тисячі доларів США.

Однією з найбільш цікавих можливостей, в розрізі даного проекту, була можливості введення текстових команд конфігурації (рисунок 5.3.1)

Рис. 5.3.1 – ввід команд в Mikrotik

Та створення скриптів на основі команд конфігурації (рисунок 5.3.2)

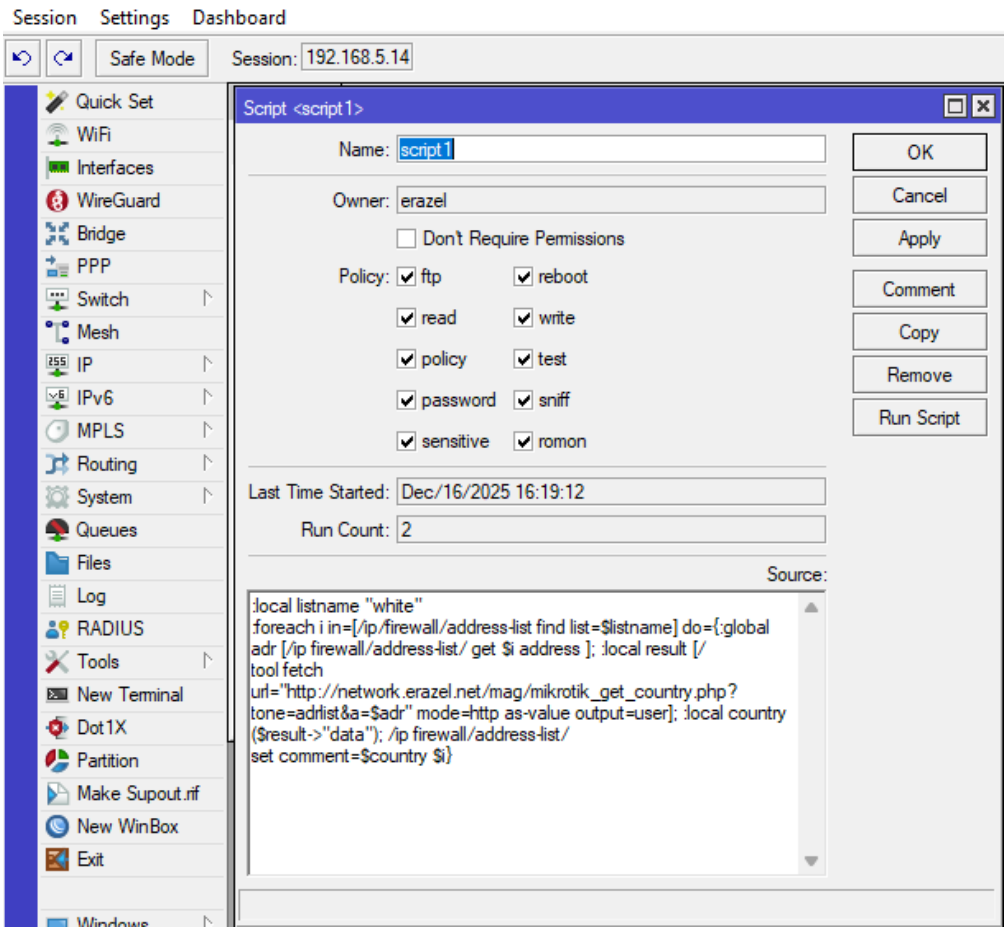
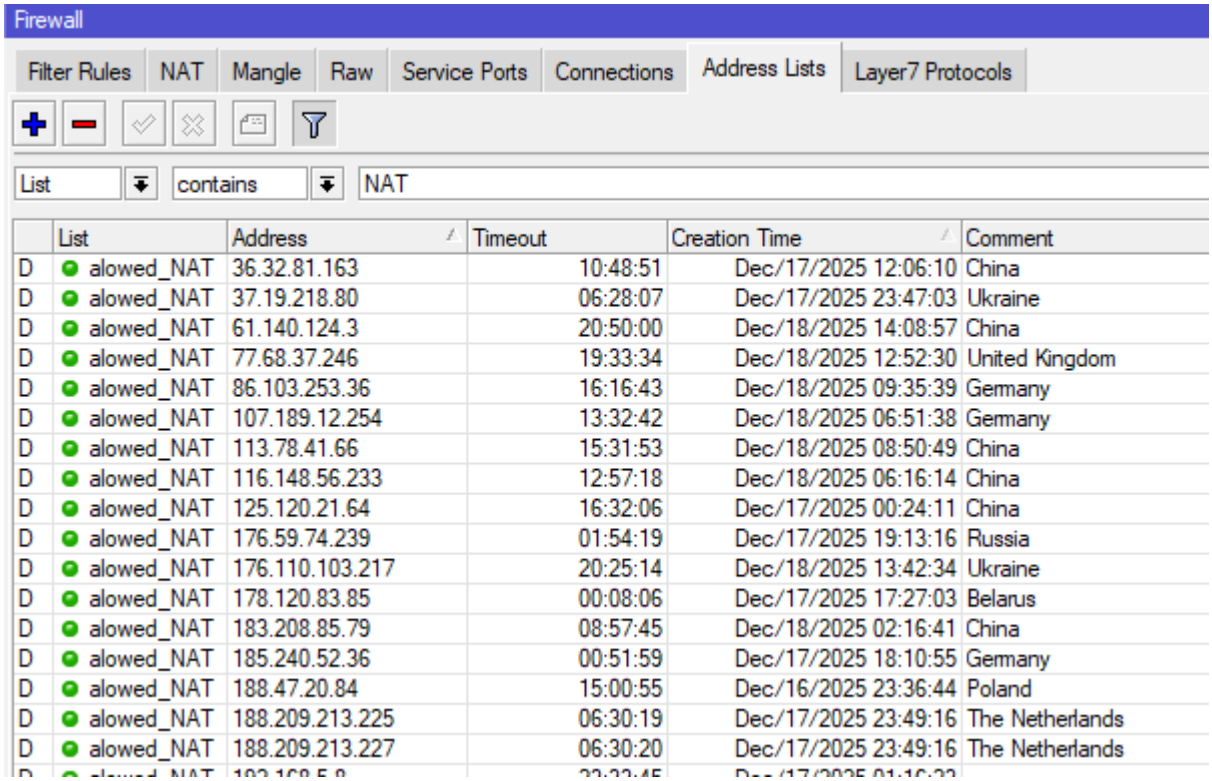


Рис. 5.3.2 – Створення скриптів на Mikrotik.

А також команда запиту до вебсервера:

```
/tool fetch url="http://network.erazel.net/mag/index.php" dst-path="getlist.rsc" mode=http;
```

Використовуючи ці можливості в проекті вдалось реалізувати автоматизовану взаємодію мережевого обладнання з базою даних. Обладнання надсилало певні запити до бази даних і на основі відповідей виконувало у себе відповідні дії. Зокрема надавало IP-адресу, а у відповідь отримувало дані якій країні належить дана адреса, та вносило ці дані у потрібні поля правил файрволу. Це дозволило автоматизовано періодично оновлювати дані на обладнанні Mikrotik, так що адміністратор не лише бачив з яких адрес відбувався доступ, а й з яких країн чи міст. Що дозволяло зручно і швидко помічати несанкціоновані доступи до сервісів котрі розміщені за мережевим обладнанням.Зразок результату виконання такого завдання проілюстровано на рисунку 5.3.3.



	List	Address	Timeout	Creation Time	Comment
D	alowed_NAT	36.32.81.163	10:48:51	Dec/17/2025 12:06:10	China
D	alowed_NAT	37.19.218.80	06:28:07	Dec/17/2025 23:47:03	Ukraine
D	alowed_NAT	61.140.124.3	20:50:00	Dec/18/2025 14:08:57	China
D	alowed_NAT	77.68.37.246	19:33:34	Dec/18/2025 12:52:30	United Kingdom
D	alowed_NAT	86.103.253.36	16:16:43	Dec/18/2025 09:35:39	Germany
D	alowed_NAT	107.189.12.254	13:32:42	Dec/18/2025 06:51:38	Germany
D	alowed_NAT	113.78.41.66	15:31:53	Dec/18/2025 08:50:49	China
D	alowed_NAT	116.148.56.233	12:57:18	Dec/18/2025 06:16:14	China
D	alowed_NAT	125.120.21.64	16:32:06	Dec/17/2025 00:24:11	China
D	alowed_NAT	176.59.74.239	01:54:19	Dec/17/2025 19:13:16	Russia
D	alowed_NAT	176.110.103.217	20:25:14	Dec/18/2025 13:42:34	Ukraine
D	alowed_NAT	178.120.83.85	00:08:06	Dec/17/2025 17:27:03	Belarus
D	alowed_NAT	183.208.85.79	08:57:45	Dec/18/2025 02:16:41	China
D	alowed_NAT	185.240.52.36	00:51:59	Dec/17/2025 18:10:55	Germany
D	alowed_NAT	188.47.20.84	15:00:55	Dec/16/2025 23:36:44	Poland
D	alowed_NAT	188.209.213.225	06:30:19	Dec/17/2025 23:49:16	The Netherlands
D	alowed_NAT	188.209.213.227	06:30:20	Dec/17/2025 23:49:16	The Netherlands

Рис. 5.3.3 – Список підключень на обладнанні Mikrotik

Як видно на прикладі адміністратору нема необхідності знати якій країні належить та чи інша адреса, йому одразу в зручному виді повідомляє звідки було здійснене підключення і можна одразу помітити підозрілу активність.

Також досить корисною стала, реалізована в даному проекті , можливість на обладнанні зробити запит до бази даних з вказанням країни і у відповідь одтримати готовий конфігураційний файл з створенням маршрутів чи адреслистів для даної країни. Адміністратору нема необхідності на компютері шукати потрібні дані в інтернеті а потім копіювати на мережеве обладнання. Достатньо в скрипті на мережевому обладнанні замінити назву країни та виконати скрипт і автоматично будуть створені потрібні правила. Це значно пришвидшує роботу якщо необхідно завести правила доступу для низки країн(наприклад заблокувати доступ з певного переліку країн).

Розділ 6. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Оскільки даний проект розроблений для автоматизації завдань системних і мережевих адміністраторів то норми охорони праці та безпеки в надзвичайних ситуаціях будуть використані ті що застосовуються до операторів персональних компютерів.

6.1. Вплив шуму на організм людини і розроблення заходів по зниженню рівня шуму.

Деякі ВДТ є потенційними джерелами цілого ряду звуків, що містять як коливання, які можна почути, так і коливання ультразвукового діапазону. Цей шум справляє негативний вплив на функціональний стан користувачів.

Відомо, що шум несприятливо діє на людину, особливо при тривалому впливі. У користувача, діяльність якого пов'язана з переробкою інформації, що часто супроводжується елементами творчості, це виражається у зниженні розумової працездатності (наприклад, швидкість обробки тексту зменшується на 10— 15%, зростає кількість помилок), у прискоренні розвитку зорового втомлення, зміні відчуття кольорів, підвищенні витрати енергії (на 17%), появі головного болю, розвитку безсоння, послабленні уваги та ін.

Шум може бути фактором, що сприяє розвитку стресу. Відзначено взаємозв'язок між скаргами на шум від ВДТ, з одного боку, та емоційними порушеннями і поганим настроєм — з другого. Вплив шуму на вегетативну нервову систему може проявлятися при рівнях, близьких до припустимого, і призводити до порушення периферійного кровообігу за рахунок спазму капілярів шкіри та слизових оболонок, а також до інших негативних наслідків.

Для вимірювання шуму застосовують різні шумо-вимірювачі, частотні аналізатори та інші прилади. Частотні аналізатори служать для виділення будьякої смуги частот для подальшої направленої корекції шуму як за об'єктивними показниками, так і згідно з суб'єктивним сприйняттям користувача.

Вимірювання шуму на робочих місцях здійснюється згідно з ДСТУ 12.1.050-2001 та ДСТУ 23941:2004.

Нормованими параметрами шуму на робочих місцях є рівні середньоквадратичних звукових тисків (дБ) та рівні звуку (дБА), що вимірюються по шкалі «А» шумовимірювача. Останні найбільш близькі до фізіологічного сприйняття людиною.

Згідно з ДСТУ 12.1.003-83 шум у приміщенні, де виконують роботу, пов'язану з виробленням концепцій, створенням нових програм, викладацькою роботою, творчістю, не повинен перевищувати 40 дБА. Праця керівників виробництва, пов'язана з контролем групи людей, що виконують переважно розумову роботу, не повинна супроводжуватися шумом вище 50 дБА. Висококваліфікована розумова робота, що вимагає зосередженості, може проводитись у приміщеннях, де рівень шуму не перевищує 55дБА. Під час виконання розумової роботи за особистим графіком з інструкцією (операторська та близькі до неї види діяльності) і точних зорових робіт рівень шуму не повинен перевищувати 65 дБА.

Сумарний вплив численних джерел шуму у приміщенні у результаті багаторазового відбиття звукових хвиль може значно перевищити енергію прямого звуку від тих же джерел. Шум від окремих приладів не повинен перевищувати фоновий більше ніж на 5 дБ.

Найчастіше рівні акустичного випромінювання, які виходять від ВДТ, охоплюють діапазон частот від 6,3 до 40 кГц. Домінуючими є частоти від 16 до 40 кГц, пов'язані з частотою горизонтальної розгортки. Шум, можливо, виникає у осерді перетворювача горизонтальної розгортки. Не виключено, що всередині ВДТ існують вторинні джерела шуму.

Рівні звукового тиску на відстані приблизно 50 см від багатьох ВДТ у напрямі максимуму випромінювання знаходяться у межах від 30 до 68 дБ (середнє значення — 51 дБ). В діапазоні 16 - 20 кГц максимальний зареєстрований рівень склав 61 дБ (середнє значення 53 дБ).

Основними заходами боротьби з шумом є усунення або ослаблення причин шуму в самому його джерелі у процесі проектування, використання засобів звукопоглинання, раціональне планування виробничих приміщень.

6.2. Заходи, що покращують умови праці оператора

Впровадженню режимів праці та відпочинку оператора ЕОМ повинна передувати робота щодо наукового обґрунтування тривалості та порядку проведення перерв, заснована на урахуванні змісту праці та факторів, які обумовлюють умови праці. Кількість, тривалість та структура перерв, а також їх «наповнення» визначаються характером прояву втоми та відповідають періодам зниження працездатності (пов'язаних з роботою, що виконується, та підвладних впливу добової періодичної зміни фізіологічних та психофізіологічних функцій людини).

Тривалість безперервної роботи за ВДТ без регламентованої перерви не повинна перевищувати 2 год. Тривалість обідньої перерви визначається чинним законодавством про працю та правилами внутрішнього трудового розпорядку підприємства (організації, установи).

При 8-го динній робочій зміні регламентовані перерви доцільно встановити:

для I категорії робіт за ВДТ через 2 год від початку зміни та через 2 год після обідньої перерви, кожна тривалістю 10 хв;

для II категорії робіт за ВДТ через 2 год від початку зміни тривалістю 15 хв, через 1,5 та 2,5 год після обідньої перерви тривалістю 15 та 10 хв відповідно або тривалістю 5—10 хв через кожну годину роботи, залежно від характеру технологічного процесу;

для III категорії робіт за ВДТ через 2 год від початку зміни, через 1,5 та 2,5 год після обідньої перерви тривалістю 20 хв кожна або тривалістю 5—15 хв через кожну годину роботи, залежно від характеру технологічного процесу.

Під час роботи за ВДТ у нічну зміну, незалежно від групи та категорії робіт, тривалість регламентованих перерв збільшується на 60 хв.

Після розробки раціонального режиму праці та відпочинку (визначення тривалості перерв на відпочинок, послідовності чергування проміжків праці та відпочинку, проектування змісту відпочинку) на підприємствах здійснюється експериментальне впровадження нового режиму праці та відпочинку протягом 3—4 місяців. Потім проводяться фізіологічні та соціально-економічні дослідження для виявлення ефективності нового режиму праці та відпочинку.

Навколишнє робоче середовище повинно формуватися у тісній взаємодії з працівниками таким чином, щоб врахувати особливості користувачів з різними фізичними та розумовими якостями. Тому умови роботи мають бути досить варіабельними.

В положенні сидячи основне навантаження припадає на м'язи, що підтримують хребетний стовп та голову. При цьому тиск більшої частини маси тіла припадає на стегна, перешкоджаючи проникненню крові у нижню його частину. У зв'язку з цим при тривалому сидінні час від часу необхідно змінювати фіксовані робочі пози. До того ж при роботі сидячи природне прогинання поперекової ділянки хребетного стовпа уперед змінюється на прогинання назад, що часто є причиною появи болю у попереку.

Для фізіологічно правильно обґрунтованої робочої пози сидячи мають бути забезпечені оптимальні положення частин тіла: корпус випрямлений, зберігаються природний вигин хребта та кут нахилу таза. Необхідно уникати сильних нахилів торса, поворотів голови та крайніх положень суглобів кінцівок.

Літературні дані про оптимальні кути між сусідніми сегментами тіла, що забезпечують зручність пози, неоднозначні. Для пози сидячи частіше за все рекомендують такі значення кутів:

кут, створений положенням осі торса та шиї, змінюється залежно від роботи, що виконується; при значенні його більше 25° виникають хворобливі відчутгя у задній частині шиї; ближчим до оптимального вважається кут, що наближається до 15°;

вимоги до значення кута, утвореного положенням осі торса та осі стегна, дещо розходяться; за одними даними, він має бути прямим, тобто, 90°, за іншими—тупим (110-115°);

кут, створений віссю стегна та гомілки, може бути у діапазоні 90—120°, при куті більше 120° можлива рання втома розтягнутих згинаючих м'язів стегон;

вважається, що для кута, створеного віссю гомілки та підошви ступні, оптимальне значення 90—100°, можливе його збільшення до 115°;

кращим положенням руки визнано таке, при якому вона звисає вздовж тіла, тобто кут, створений віссю плече-ліктьового сегмента та вертикаллю торса, дорівнює нулю;

при роботі, коли передпліччя підтримуються підлокітниками або площиною столу, рука може утворювати досить великий кут з вертикаллю (до 45°), а коли маса руки утримується плечем і точкою опори служить кисть руки, то максимальне значення кута не повинно перевищувати 35°;

кут, утворений віссю плеча та передпліччя, може бути від 40° при згинанні та до 180° при максимальному витяганні; кут у 90° наближається до оптимального, оскільки згинаючі та розгинаючі м'язи стискаються однаковою мірою, а умови Кровообігу найбільш сприятливі;

кут утворений віссю передпліччя та кистю, рівний 180° вважається краіДим, оскільки при цьому м язи, що приводять у рух кисть, знаходяться у стані рівного скорочення, а кисть є прямим продовженням передпліччя, припустиме латеральне (бічне) відхилення - 10° .

Виходячи з загальних принципів організації робочого місця у нормативно-методичних документах сформульовані вимоги до його конструкції.

Основним обладнанням робочого місця користувача ВДТ є відеомонітор, клавіатура, робочий стіл, стілець (крісло); допоміжним — пюпітр, підставка для ніг, шафи, полиці та ін. Вимоги до них відображені у нормативних документах: ДСТУ 8604:2015; ДСТУ 7299:2013.

Під просторовою орієнтацією робочого місця розуміється розміщення у певному порядку елементів основного та допоміжного обладнання відносно одне одного та працюючої людини. Просторова організація робочого місця в основному визначається розмірами та формою сенсорного та моторного простору, формою та параметрами елементів робочого місця та просторовим розташуванням елементів відносно працюючого.

Робочі місця з ВДТ повинні розташовуватися на відстані не менше як 1,5 м від стіни з віконними прорізами, від інших стін — на відстані 1 м, між собою на відстані не менше як 1,5 м. При розміщенні робочих місці» необхідно виключити можливість прямого засвічування екрана джерелом природного освітлення. Джерело природного освітлення (вікно) не повинно також потрапляти у зону прямого спостереження користувача. Відносно світлових отворів робочі місця доцільно розташовувати таким чином, щоб природне світло падало на нього збоку, переважно зліва.

При розміщенні ВДТ на робочому місці потрібно забезпечити простір для користувача величиною не менше як 850 мм з урахуванням виступаючих частин обладнання та застосування (при необхідності) спецодягу. Для стоп має бути передбачено простір по глибині та висоті не менше як 150 мм, по ширині — не менше як 530 мм.

Розташовувати ВДТ на робочому місці необхідно так, щоб поверхня екрана знаходилася на відстані 400— 700 мм від очей користувача. Рекомендується розміщувати елементи робочого місця таким чином, щоб витримувалася однакова відстань очей користувача від екрана, клавіатури, тримача документів.

Залежно від виду роботи та зручності користувача доцільно користуватися можливістю повороту та регулюванням нахилу екрана. Ця вимога тим більш важлива, чим численнішими та різноманітнішими є заплановані випадки застосування ВДТ. Установка рівня екрана над столом та його розташування повинні забезпечуватися за допомогою вторинних пристроїв на робочому місці.

Необхідно стало розташовувати клавіатуру на робочому столі, не допускаючи її хитання. Разом з тим має бути передбачена можливість її поворотів та переміщень. Положення клавіатури та кут її нахилу повинні відповідати побажанням користувача.

Принтер треба розташовувати так, щоб доступ до нього користувача та його колег був зручним. Конструкція робочого столу повинна забезпечувати можливість оптимального розміщення на робочій поверхні обладнання, що використовується, з урахуванням його кількості, розмірів, конструктивних особливостей та характеру роботи, яка виконується. Корисно мати модульне,

рухоме робоче місце. Площа столу залежить від всіх необхідних для роботи компонентів, що розміщуються, та повинна допускати можливість вільного переміщення пристроїв. Поверхня столу має бути матовою з малим відбиттям та тепло ізолюючою.

Якщо конструкція робочого місця передбачає протікання трудового процесу у позі сидячи, то висота робочої поверхні столу повинна регулюватися у межах 680— 800 мм, у середньому вона повинна становити 725 мм. Робочий стіл повинен мати простір для ніг висотою не менше як 600 мм, шириною не менше як 500 мм, глибиною на рівні колін, але не менше як 450 мм та на рівні витягнутої ноги — не менше як 650 мм.

Робоче крісло забезпечує підтримання робочої пози у положенні сидячи, і чим триваліше це положення протягом робочого дня, тим жорсткішими мають бути вимоги до створення зручних та правильних робочих сидінь.

Існує цілий ряд публікацій щодо конструювання різних тигів робочих крісел. Незважаючи на розбіжність думок дослідників у визначенні деяких параметрів виділяють загальні рекомендації конструювання крісла: необхідність реулювання найбільш важливих його елементів — висоти сидіння, висоти спинки сидіння та кута нахилу спинки. Причому процес регулювання має бути нескладним. Не слід надмірно збільшувати кількість регульованих параметрів крісла, оскільки це позначатиметеся на його стійкості. Для надання більшої стійкості та попередження перекидання при відхиленні тіла на спинку крісла у багатьох європейських країнах використовують стільці на п'яти ніжках.

Встановлення правильної висоти сидіння є першочерговим завданням під час організації робочого місця. Цей параметр визначає інші просторові параметри — висоту положення екрана, клавіатури, поверхні для записів тощо.

Висота поверхні сидіння визначається висотою підколінної ямки над підлогою, виміряної у положенні сидячи при куті згинання коліна 90°. Висоту сидіння необхідно регулювати.

Зручність невеликих переміщень у просторі робочої зони, зумовлених характером виробничої діяльності, може бути забезпечена за наявності

спеціальних коліщаток на ніжках стільця (звичайних або гальмівних) або шляхом ковзання по поверхні підлоги, що залежить від матеріалу її покриття.

Робоче місце має бути обладнане стійкою підставкою для ніг, параметри якої просто регулюються. Вона має бути розташована по всій ширині ділянки, що відводиться для ніг. Підставка повинна мати ширину не менше як 300 мм, глибину не менше як 400 мм, регулювання по висоті до 150 мм та по куту нахилу опорної поверхні підставки до 20°. Поверхня підставки має бути рифленою, а по передньому краю мати бортик висотою 10 мм.

Дотримання цих правил при організації робочого місця оператора ЕОМ значно покращить продуктивність його праці та зменшить ймовірність виникнення різноманітних професійних захворювань.

6.3 Заходи щодо охорони навколишнього середовища.

При експлуатації використаного у дипломному проекті виробничого обладнання з метою запобігання виникнення аварійних ситуацій, які мо-жуть призвести до забруднення навколишнього середовища, повинні забезпечуватись:

- надійність і безпека роботи всього основного і допоміж-ного устаткування;
- можливість досягнення номінальної продуктивнос-ті виробничого обладнання;
- економний режим роботи виробничого обладнання, встановлений на основі випробувань і заводських інструкцій;
- регулювальний діапазон навантажень;
- викиди шкідливих речовин в атмосферу в межах допустимих зна-чень.

Поряд з автоматизованим контролем найбільш відповідальних параметрів, надмірне відхилення яких від встановленого значення викликає порушення нормального технологічного процесу, передбачаються автома-тичні системи захисту обладнання від пошкоджень.

Пристрої технологічної безпеки повинні бути в повній готовності, але спрацьовувати лише в тому випадку, коли можливості автоматичного та

дистанційного керування щодо запобігання відхилень параметрів від встановлених значень вичерпані, а оператор не може вчасно на це зреагувати.

Тобто технологічні захисти покликані впливати на об'єкт керування лише у виняткових випадках, зокрема у передаварійному (аварійному) стані, або при несподіваних стрімких зростаннях величин навантажень.

Частіше всього технологічні захисти служать для попередження ава-рій обладнання при відхиленнях параметрів від допустимих значень. Вплив захистів пов'язаний з відкриттям (закриттям) запірних органів, зу-пинкою основного та допоміжного обладнання або відключенням його ре-зерву.

6.4. Розрахунок аерації виробничого приміщення.

Оскільки питання забезпечення нормованих параметрів мікроклімату у виробничому приміщенні є одним із найважливіших у даному випадку із точки зору техніки безпеки та охорони праці, то виконаємо розрахунок аерації виробничого приміщення за умови, що температура зовнішнього повітря становить вище +10 °С, тобто для теплого періоду року.

Середня температура зовнішнього повітря для теплого періоду року (згідно нормативних документів для даного кліматичного поясу) становить: $t_з$ = +20 °С. Густина чистого свіжого зовнішнього повітря $\rho_з$:

$\rho_з$ = 1,199 кг/м3.

Розрахункове значення температури $t_{р.з.}$ в робочій зоні виробничого приміщення визначаємо за виразом:

$t_{р.з.} = t_з + \Delta t_{р.з}$,

де $\Delta t_{р.з.}$ – допустимий перепад температури в робочій зоні.

Значить:

$t_{р.з} = 20 + 8 = 28$ °С.

За рекомендованим нормативним співвідношенням еквівалентних площ для даної категорії виробничих приміщень та використовуваного виробничого обладнання визначаємо коефіцієнт аерації m:

m = 1.

Температура повітря, що видаляється із виробничого приміщення:

tвих. = tр.з. / m = 28 / 1 = 28 °С.

Перепад тисків, при якому можлива асиміляція надлишкової теплоти, складає:

Δр1,2 = h·(ρвих - ρз) g,

де h – висота виробничого приміщення;

ρвих - густина забрудненого повітря при tвих.

Значить:

h = 4 м;

ρвих = 1,25 кг / м3 (згідно замірів на базовому підприємстві).

Тоді:

Δр1,2 = 4·(1,25 - 1,199)·9,8 = 2 Па.

Для даної категорії виробничих приміщень тиск Δр1 для забезпечення можливості надходження свіжого зовнішнього повітря у виробниче приміщення визначається за виразом:

Δр1 = n·Δр1,2,

де n – частина тиску, що витрачається на рух повітря через віконні пройми (для розглядуваних виробничих умов n = 0,15).

Тоді:

Δр1 = 0,15·2 = 0,3 Па.

Тиск Δр2 для витяжки забрудненого повітря через аераційний ліхтар визначається за формулою:

Δр2 = Δр1,2 – Δр1 = 2 - 0,3 = 1,7 Па.

Для визначення площ вентиляційних пройм використаємо формули для визначення тисків:

$$\Delta p_1 = \xi_1 \frac{v_1^2 \rho_з}{2}$$

$$\Delta p_2 = \xi_2 \frac{v_2^2 \rho_{вих}}{2}$$

де ξ1, ξ2 – коефіцієнти місцевих опорів відповідно для припливних і витяжних пройм:

ξ1 = 3,2;

ξ2 = 5,8;

v1 – швидкість чистого зовнішнього повітря у припливних проймах;

v2 – швидкість забрудненого повітря у витяжних проймах;

Значення швидкостей v1 і v2 визначаємо відповідно за виразами:

$$v_1 = \sqrt{2\Delta p_1 /(\xi_1 \cdot \rho_н)}$$

$$v_2 = \sqrt{2\Delta p_2 /(\xi_2 \cdot \rho_{вих})}$$

і підставляємо їх у формули для визначення площ відповідно припливних та витяжних пройм.

У результаті отримуємо:

$$F_1 = (G / \rho_з) / (3600 \cdot \sqrt{2\Delta p_1 / (\xi_1 \cdot \rho_з)}) = G / (3600\sqrt{2\Delta p_1 \rho_з / \xi_1})$$

$$F_2 = (G / \rho_{вих}) / (3600 \cdot \sqrt{2\Delta p_2 / (\xi_2 \cdot \rho_{вих})}) = G / (3600\sqrt{2\Delta p_2 \rho_{вих} / \xi_2})$$

де G – нормовані масові витрати повітря на функціонування аерації для даної категорії виробничих приміщень та даного їх об’єму і забезпечення нормованої кратності повітрообміну, рівної 8.

Тобто сумарні площі становитимуть:

припливних пройм:

$$F_1 = 52000 / (3600 \cdot \sqrt{2 \cdot 0{,}3 \cdot 1{,}199 / 3{,}2}) = 52000 / 1707 = 31 \ \text{м}^2$$

витяжних пройм:

$$F_2 = 52000 / (3600 \cdot \sqrt{2 \cdot 1{,}7 \cdot 1{,}25 / 5{,}8}) = 52000 / 3082 = 16{,}9 \ \text{м}^2$$

6.5. Пожежна профілактика

Виробничі умови, системи вентиляції і кондиціонування повітря повинні відповідати протипожежним вимогам будівельних норм.

Повинні бути встановлені терміни проведення профілактичних огля-дів та очищення повітроводів, фільтрів, вогнезатримуючих клапанів, іншого обладнання вентиляційних систем, а також визначений порядок відключення вентиляційних систем і дій обслуговуючого персоналу в разі виникнення пожежі або аварії. Особа, призначена відповідальною за технічний стан та справність вентиляційних систем, зобов'язана забезпечити додержання вимог пожежної безпеки під час їх експлуатації.

На підприємстві відповідно до ДБН В.2.5-74:2013 та ДБН В.2.5-64:2012 необхідно передбачати систему протипожежного водопостачання, яка є джерелом подачі води для пересувної пожежної техніки та установок пожежегасіння. Протипожежний водогін, як правило, об'єднується з господарчо-питтєвим чи виробничим водогоном.

Витрату води на гасіння пожежі при об'єднаному водогоні для спринклерних чи дренчерних установок, внутрішніх пожежних кранів та зов-нішніх гідрантів протягом 1 год. з моменту початку пожежегасіння необхі-дно приймати як суму найбільших витрат, що визначаються у відповідності з вимогами „Інструкції з проектування установок автоматичного пожежегасіння", ДБН В.2.5-74:2013 та ДБН В.2.5-64:2012.

При розрахунку протипожежного водопостачання тривалість гасіння пожежі приймається 3 год.; для будівель I та II ступеня вогнестійкості категорій Г та Д - 2 год.

Час роботи пожежних кранів приймається 3 год. При встановленні пожежних кранів на системах автоматичного пожежегасіння час їх роботи необхідно приймати рівним часу роботи систем автоматичного пожежегасіння.

Передбачаємо наступні заходи щодо евакуації персоналу у випадку виникнення пожежі:

а) повинні бути передбачені евакуаційні виходи, число яких прийма-ється згідно вимог протипожежної профілактики не менше двох;

б) ширина шляхів евакуації повинна бути не меншою одного метра, дверей - 0,8 метра;

в) двері повинні відкриватися у напрямі виходу із будівлі;

г) не допускаються на шляхах евакуації перепади висот, більші сорока п'яти сантиметрів і виступи в місцях перепаду висот;

д) не допускається влаштування на шляхах евакуації персоналу приміщень будь-якого призначення, газо- і паропроводів тощо;

є) відстань від найбільш віддаленого робочого місця до евакуаційного виходу не повинна бути більшою двадцяти метрів для даної категорії будівлі;

ж) повинно бути передбачене евакуаційне освітлення.

Згідно з вимогами НАПБ А.01.001-2014 «Правила пожежної безпеки в Україні» всі виробничі приміщення повинні бути забезпечені первинними засобами пожежегасіння, в якості яких можна застосовувати хімічно-пінні вогнегасники ОХП-10, ОП-14, ОП-9М, повітрянопінні вогнегасники ОВА-5, ОШ1-10, ОВП-250 та інші засоби.

ВИСНОВКИ

В результаті виконання магістерської роботи було розроблено програмний комплекс для взаємодії користувача з базою даних географічного розташування IP-мереж. Також було реалізовано механізми автоматичної взаємодії мережевого обладнання Mikrotik з вищеописаною базою даних.

Також в процесі роботи, з метою покращення швидкості роботи, було досліджено різні типи зберігання IP-адрес, оскільки на мою думку використовуємі зараз є досить оптимальними, але мають можливість покращення швидкодії.

В проекті повністю реалізовано запланований функціонал. Але оскільки проект є проектом з відкритим вихідним кодом то можливість розширення функціоналу, згідно нових потреб та умов, є доступна в любий момент часу. Тому можна стверджувати що проект не є фінально завершеним, він є проектом з перманентним розширенням та/або зміною функціоналу.

ПЕРЕЛІК ПОСИЛАНЬ

1. І. В. Чихіра, І. С. Дідич (2022), Конспект лекцій з дисципліни «Системи управління базами даних» напряму підготовки 151«Автоматизація та комп’ютерно-інтегровані технології
2. А.Г. Микитишин, М.М. Митник, П.Д. Стухляк, В.В. Пасічник(2013) Навчальний посібник для технічних спеціальностей вищих навчальних закладів «Комп’ютерні мережі» ISBN 978-617-574-087-3
3. Jesper Wisborg Krogh (2020), MySQL 8 Query Performance Tuning: A Systematic Method for Improving Execution Speeds
4. Paul Dubois (2013), MySQL 5th Edition
5. Д. І. Могилевич, І. В. Кононова (2019) КПІ ім. Ігоря Сікорського Електронне мережне навчальне видання «Адресації в ір-мережах теоретичні основи та приклади розв'язання задач»
6. Світлана Суліма (2023) «Метод оптимізації sql запитів системи управління базами даних» https://www.researchgate.net/publication/372259794_METOD_OPTIMIZACII_SQL_ZAPITIV_SISTEMI_UPRAVLINNA_BAZAMI_DANIH
7. Ігор Чихіра к.т.н, доцент, Віталій Левицький, к.т.н., Артур Микитишин (2019) «Етапи оптимізації баз даних»
8. Житецький В.Ц., Джигерей В.С.(2000) «Основи охорони праці» – 4-е видання.
9. Гаврик Є.О.(2003) «Охорона праці. Навчальний посібник».
10. Заплатинський В.М. (2003) «Безпека життєдіяльності людини»
11. Положення про Цивільну оборону України (затверджене постановою КМУ від 10 травня 1994 року №299)
12. Закон «Про цивільну оборону України» (від 24 березня 1999 року).
13. Мусієнко М.М., Серебряков В.В., Брайон, К (2002) «Екологія. Охорона природи: Словник – довідник»і
14. Вебсайт - https://help.mikrotik.com/docs/spaces/ROS/pages/8978514/Fetch
15. Вебсайт - https://help.mikrotik.com/docs/spaces/ROS/pages/47579229/Scripting

16. Методичні рекомендації з виконання, оформлення та захисту кваліфікаційних робіт магістрів спеціальності 174 "Автоматизація, комп’ютерно-інтегровані технології та робототехніка" / ТНТУ ім. І. Пулюя; уклад. А.Г. Микитишин, М.М. Митник., В.В. Левицький, Р.І. Королюк – Тернопіль: ТНТУ, 2024. – 82 с.

17. Методичні вказівки до виконання курсового проєкту з дисципліни «Основи автоматизованого проєктування складних об’єктів та систем» для здобувачів освітнього рівня магістр за спеціальністю 174"Автоматизація, комп’ютерно-інтегровані технології та робототехніка" / Уклад. В.В. Левицький, І.С. Дідич – Тернопіль: ТНТУ, 2023. - 44 с.

Додатки: Лістинг файлів проекту.
add_city.php

```
<?php
    include ('connect_db.php');
    //вказуємо файл звідки читати дані
    $ipcountry_file=fopen('GeoLite2-City-Locations-en.csv', "r");
    //зчитуємо перший рядок з описом полів
    $row_ip= fgetcsv($ipcountry_file);
    //в циклі зчитуємо всі рядки з файлу
    While ($row_ip= fgetcsv($ipcountry_file))
    {
       if ($row_ip['10']=="" OR $row_ip['10']==" ")
       {
           $row_ip['10']=$row_ip['7'];
       }
       if ($row_ip['10']=="" OR $row_ip['10']==" ")
       {
           $row_ip['10']=$row_ip['9'];
       }
       if ($row_ip['10']=="" OR $row_ip['10']==" ")
       {
           list($temp, $city_temp) = explode('/', $row_ip['12']);
           $row_ip['10']=$city_temp."_2";
       }
        //Готуємо дані до запису в базу даних
        $country= mysqli_real_escape_string($mysql, $row_ip['5']);
        $city= mysqli_real_escape_string($mysql, $row_ip['10']);
        //записуємо в БД рядок з даними
        $query_insert_city="INSERT INTO city SET id='$row_ip[0]',
country='$country', city='$city'";
        mysqli_query($mysql, $query_insert_city);
    }
?>
```

add_country.php

```
<?php
    include ('connect_db.php');
    //вказуємо файл звідки читати дані
    $ipcountry_file=fopen('GeoLite2-Country-Locations-en.csv', "r");
    //зчитуємо перший рядок з описом полів
    $row_ip= fgetcsv($ipcountry_file);
    //в циклі зчитуємо всі рядки з файлу
    While ($row_ip= fgetcsv($ipcountry_file))
    {
        //записуємо в БД рядок з даними
        $query_insert_country="INSERT INTO country SET
id='$row_ip[0]', country='$row_ip[5]', continent='$row_ip[3]'";
        mysqli_query($mysql, $query_insert_country);
    }
?>
```

add_ipv4_city.php

```
<?php
    //виставляємо ліміт часу в 3600 секунд, так як скрипт заповнення
    //бази даних займає довгий час
    set_time_limit(3600);
    ini_set('max_execution_time', 3600);
    include ('connect_db.php');
    //вказуємо файл звідки читати дані
    $ipcountry_file=fopen('GeoLite2-City-Blocks-IPv4.csv', "r");
    //зчитуємо перший рядок з описом полів
    $row_ip= fgetcsv($ipcountry_file);
    //в циклі зчитуємо всі рядки з файлу

    While ($row_ip= fgetcsv($ipcountry_file))
    {
        list($ip, $prefix) = explode('/', $row_ip['0']);
        $ip_dec = ip2long($ip);
        $mask = -1 << (32 - $prefix);
        $network   = $ip_dec & $mask;
        $broadcast = $network | (~$mask);
        /*
         * заповнюємо країну реєстрації наступним значенням(країна
походження) якщо значення відсутнє.
         * зроблено на випадок відсутніх значень у файлі. щоб
уникнути
         * можливих помилок при заповненні бази даних
         * також заповнюємо "заглушками" місто .
        */
        $row_ip['2']=($row_ip['2']=='')?$row_ip['3']:$row_ip['2'];
        $row_ip['2']=($row_ip['2']=='')?1:$row_ip['2'];
        $row_ip['1']=($row_ip['1']=='')?1:$row_ip['1'];
        $row_ip['7']=($row_ip['7']=='')?0:$row_ip['7'];
        $row_ip['8']=($row_ip['8']=='')?0:$row_ip['8'];
        $query_add_addr="INSERT INTO ipv4city SET
ipnetwork='$row_ip[0]', mask='$prefix', begin='$network',
end='$broadcast', city='$row_ip[1]', country='$row_ip[2]',
latitude='$row_ip[7]', longitude='$row_ip[8]'";
        mysqli_query($mysql, $query_add_addr);
    }
?>
```

add_ipv4_country.php

```
<?php
    //виставляємо ліміт часу в 3600 секунд, так як скрипт заповнення
    //бази даних займає довгий час
    set_time_limit(3600);
    ini_set('max_execution_time', 3600);
    include ('connect_db.php');
    //вказуємо файл звідки читати дані
    $ipcountry_file=fopen('GeoLite2-Country-Blocks-IPv4.csv', "r");
    //зчитуємо перший рядок з описом полів
    $row_ip= fgetcsv($ipcountry_file);
    //в циклі зчитуємо всі рядки з файлу
    While ($row_ip= fgetcsv($ipcountry_file))
    {
        list($ip, $prefix) = explode('/', $row_ip['0']);
        $ip_dec = ip2long($ip);
        $mask = -1 << (32 - $prefix);
        $network   = $ip_dec & $mask;
        $broadcast = $network | (~$mask);
        /*
         * заповнюємо GEOID одиницею якщо значення відсутнє.
         * зроблено на випадок відсутніх значень у файлі. щоб
уникнути
         * можливих помилок при заповненні бази даних
         */
        $row_ip['1']=($row_ip['1']=='')?$row_ip['2']:$row_ip['1'];
        $query_add_addr="INSERT INTO ipv4country SET
ipnetwork='$row_ip[0]', mask='$prefix', begin='$network',
end='$broadcast', geoid='$row_ip[1]'";
        mysqli_query($mysql, $query_add_addr);
    }
?>
```

create_router_rules.php

```
<?php
    include ('connect_db.php');
    //Якщо вибрано місто ВСІ то запитуємо всю країну, а не окремі
мережі міст
    if ($_POST['city']=='all')
    {
        $query_get_name="SELECT country FROM country WHERE
id='$_POST[country]'";
        $geo= mysqli_fetch_assoc(mysqli_query($mysql,
$query_get_name));
        $name=$geo['country'];
        $query_get_items="SELECT ipnetwork from ipv4country WHERE
country='$_POST[country]'";
    }
    else
    {
        $query_get_name="SELECT city FROM city WHERE
id='$_POST[city]'";
        $geo= mysqli_fetch_assoc(mysqli_query($mysql,
$query_get_name));
        $name=$geo['city'];
        $query_get_items="SELECT ipnetwork FROM ipv4city WHERE
city='$_POST[city]'";
    }
    $item_mysql= mysqli_query($mysql, $query_get_items);
    //Якщо в меню було вибрано "маршрут" то виводимо команди
маршруту
    //якщо адреслист то відповідно адреслисту
    switch ($_POST['rule_type']) {
    case 'route':
            echo "/ip route<br>\n";
            while ($item= mysqli_fetch_assoc($item_mysql))
            {
                echo "add check-gateway=ping comment=$name dst-
address=$item[ipnetwork] gateway=$_POST[gateway]<br>\n";
            }
            break;
    case 'addrlist':
            echo "/ip firewall address-list<br>\n";
            $name=$_POST['adrlist']==''?$name:$_POST['adrlist'];
            while ($item= mysqli_fetch_assoc($item_mysql))
            {
                echo "add address=$item[ipnetwork] list=$name
<br>\n";
            }
            break;
    case 'own':
            $name=$_POST['adrlist']==''?$name:$_POST['adrlist'];
            while ($item= mysqli_fetch_assoc($item_mysql))
            {
                foreach ($_POST['sel'] as $key=>$value)
```

```
                switch ($value) {
                    case "":
                        break;
                    case "name":
                        $_POST['sel'][$key]=$name;
                        break;
                    case "adr":
                        $_POST['sel'][$key]=$item['ipnetwork'];
                        break;
                    default:
                        break;
                }

                echo
$_POST['p']['1'].$_POST['sel']['1'].$_POST['p']['2'].$_POST['sel']['
2'].$_POST['p']['3'].$_POST['sel']['3'].$_POST['p']['4'].$_POST['sel
']['4']."<br>\n";
            }
            break;
    default:
        break;
}
?>
```

find_country_networks.php

```
<?php
include ('connect_db.php');
//отримуємо ІД країни
$country_id=$_GET['country'];
//отримуємо її імя з БД
$query_get_name="SELECT country, continent FROM `country` WHERE
id='$country_id'";
$country= mysqli_fetch_assoc(mysqli_query($mysql, $query_get_name));
$country_name=$country['country'];
//отримуємо з БД кількість мереж для цієї країни
$query_get_number_networks="SELECT COUNT(*) as countt  FROM
`ipv4country` WHERE country='$country_id'";
$networks= mysqli_fetch_assoc(mysqli_query($mysql,
$query_get_number_networks));
$number_networks=$networks['countt'];
echo "В  <strong>$country_name</strong> загалом є
<strong>$number_networks</strong> мереж:<BR>";
//Отримуємо з БД список мереж країни
$query_get_networks="SELECT ipnetwork, mask FROM `ipv4country` WHERE
country='$country_id'";
$networks_mysql= mysqli_query($mysql, $query_get_networks);
$available_hosts=0;
//В циклі виводимо інфорацію по кожній мережі в списку
while ($network= mysqli_fetch_assoc($networks_mysql))
{
    echo "$network[ipnetwork]";
    $number_host_in_network=pow(2,(32-$network['mask']));
    echo "($number_host_in_network),  ";
    $available_hosts=$available_hosts+$number_host_in_network;
}
echo "<BR>Всього можлива кількість пристроїв в країні:
$available_hosts<BR>";
//Отримуємо з БД список міст котрі розташовані в даній країні.
$query_get_cities_by_country="SELECT id, city as name, (SELECT
count(*)FROM ipv4city WHERE city=city.id) as counter_net FROM city
WHERE country='$country_name' ORDER by city ASC;";
$cities_mysql= mysqli_query($mysql, $query_get_cities_by_country);
echo "<br>В країні присутні наступні міста:<BR>";
//в циклі виводимо дані міста
while ($city= mysqli_fetch_assoc($cities_mysql))
{
    echo "<a href=\"find_networks_by_city.php?city=$city[id]\"
title=\"$city[counter_net]\">$city[name]</a> ";
}
echo "<BR><BR><BR>25 міст з найбільшою ймовірною кількістю пристроїв
в них:<BR>";
//Отримуємо з БД 25 міст з найбільшою кількістю пристроїв в них.
$query_get_hosts_in_city="SELECT id, city as name, (SELECT
SUM(POW(2,(32-ipv4city.mask))) FROM ipv4city  WHERE city=city.id) as
hosts FROM city WHERE country='$country_name' ORDER by hosts DESC
Limit 25";
```

```
$bigest_host_cities_mysql= mysqli_query($mysql,
$query_get_hosts_in_city);
echo "<table>";
while ($host_city= mysqli_fetch_assoc($bigest_host_cities_mysql))
{
    echo "<tr><td><a
href=\"find_networks_by_city.php?city=$host_city[id]\">-
$host_city[name]</a></td><td>$host_city[hosts]</td></tr>";
}
echo "</table>";
?>
```

find_ip.php

```
<?php
include ('connect_db.php');
//отримання IP-адреси з переданого масиву
$ip_addr=$_POST['ip'];
//приведення IP адреси до десяткового виду
$ip= ip2long($ip_addr);
//отримання першого октету з IP-адреси
list($first_octet,$second_octet,$third_octet,
$fourth_octet)=explode('.', $ip_addr);
//Запит до БД з отриманям всіх даних по IP-адресі
$query_get_country_city="SELECT ipv4city.ipnetwork as network,
city.city as city, country.country as country, country.continent as
continent FROM ipv4city LEFT JOIN city on city.id=ipv4city.city LEFT
JOIN country ON country.id=ipv4city.country WHERE
ipv4city.first_octet='$first_octet' AND ('$ip' BETWEEN
ipv4city.begin AND ipv4city.end)";
$geo_data= mysqli_fetch_assoc(mysqli_query($mysql,
$query_get_country_city));
echo "<strong>Мережа</strong>: $geo_data[network]<BR>";
echo "<strong>Місто</strong>: $geo_data[city]<BR>";
echo "<strong>Країна</strong>: $geo_data[country]<BR>";
echo "<strong>Континентv</strong>: $geo_data[continent]<BR>";
?>
```

find_networks_by_city.php

```
<?php
include ('connect_db.php');
$city_id=$_GET['city'];
$query_get_name="SELECT city, country FROM `city` WHERE
id='$city_id'";
$city= mysqli_fetch_assoc(mysqli_query($mysql, $query_get_name));
$city_name=$city['city'];
$country_name=$city['country'];
$query_get_number_networks="SELECT COUNT(*) as countt  FROM
`ipv4city` WHERE city='$city_id'";
$networks= mysqli_fetch_assoc(mysqli_query($mysql,
$query_get_number_networks));
$number_networks=$networks['countt'];
echo "В  <strong>$city_name($country_name)</strong> загалом є
<strong>$number_networks</strong> мереж:<BR>";
$query_get_networks="SELECT ipnetwork, mask FROM `ipv4city` WHERE
city='$city_id'";
$networks_mysql= mysqli_query($mysql, $query_get_networks);
$available_hosts=0;
while ($network= mysqli_fetch_assoc($networks_mysql))
{
    echo "$network[ipnetwork]";
    $number_host_in_network=pow(2,(32-$network['mask']));
    echo "($number_host_in_network),  ";
    $available_hosts=$available_hosts+$number_host_in_network;
}
echo "<BR>Всього можлива кількість пристроїв в місті:
$available_hosts"
?>
```

index.php

```
<HTML>
    <BODY style="background-color: #EFFEFF">
        <div style="font-size: 24pt">Пошук адрес та вивід
статистики:</div>
<?php
    include ('connect_db.php');
?>
<SCRIPT src="mag.js">
</script>
<form action="find_ip.php" method="POST">
    <LEGEND>
        Пошук місцезнаходження конкретної IP-адреси
    </LEGEND>
    <input type="text" name="ip" pattern="^(25[0-5]|2[0-
4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}$"
required>
    <input type="submit" value="знайти">
</form>
<HR>
<form action="find_country_networks.php" method="GET">
    <LEGEND>
        Відображення списку адрес IP-мереж вибраної країни
    </LEGEND>
    <select name="country">
        <?php
            $query_get_countries="SELECT * FROM `country` Order by
continent ASC, country ASC";
            $countries_mysql= mysqli_query($mysql,
$query_get_countries);
            while ($country= mysqli_fetch_assoc($countries_mysql))
            {
                echo "<option
value=\"$country[id]\">$country[continent] -
$country[country]</option>";
            }
        ?>
    </select>
    <input type="submit" value="показати">
</form>
<HR>
<LEGEND>
        Пошук міста за частиною назви, та виведення мереж що йому
налажать
</LEGEND>
<INPUT type="text" placeholder="city of worls" id="citynameworld"
onkeyup="asuncCityWorld()">
<div id="citiesworld">
</div>
<HR>
<form action="show_statistic_detailed.php" method="POST">
    <LEGEND>
```

```
        Відображення міст/країн за кількістю мереж
    </LEGEND>
    показати
    <input type="number" min="1" step="1" maxlength="3" value="10"
name="number_items">
    <SELECT name="item">
        <option value="country">країн</option>
        <option value="city">міст</option>
    </SELECT>
    з
    <SELECT name="ascening">
        <option value="DESC">найбільшою</option>
        <option value="ASC">найменшою</option>
    </SELECT>
     кістістю пристроїв
    <INPUT type="submit" value="показати">
</form>
<HR>
<div style="font-size: 24pt">Створення команд конфігурації
обладнання:</div>
<form action="create_router_rules.php" method="POST">
    <LEGEND>
        Виберіть тип шаблону для генерації:
    </LEGEND>
    <TABLE>
        <tr>
            <td>Тип</td>
            <td></td>
            <td>Країна</td>
            <td>Місто</td>
        </tr>
        <tr>
            <td>
                <SELECT name="rule_type" onchange="asuncRouteForm()"
id="rule_type">
                    <OPTION value="-"></OPTION>
                    <OPTION value="route">Маршрут</OPTION>
                    <OPTION value="addrlist">Адреслист</OPTION>
                    <OPTION value="own">Власний</OPTION>
                </SELECT>
            </td>
            <td>
                <span id="routerformtwo"></span>
            </td>
            <td>
                <?php
                echo "<select name=\"country\"
onchange=\"asuncCitiesOfCountry()\" id=\"country\">";
                $query_get_countries="SELECT * FROM `country` Order
by continent ASC, country ASC";
                $countries_mysql= mysqli_query($mysql,
$query_get_countries);
                while ($country=
mysqli_fetch_assoc($countries_mysql))
                {
```

```
                    echo "<option
value=\"$country[id]\">$country[continent] -
$country[country]</option>";
                }
                echo "</select>";
                ?>
            </td>
            <td>
                <span id="cities_contry"></span>
            </td>
        </tr>
    </TABLE>
</form>
<HR>
Ви можете автоматично генерувати правила на обладнанні Мікротік
виконуючи скрипт:<BR>
<pre>
:local country "Ukraine";
:local type "adrlist";
:local gateway "192.168.0.1";
/tool fetch url="http://<?php echo
$_SERVER['SERVER_NAME'].$_SERVER['REQUEST_URI'].'mikrotik_get_countr
y.php?t=$type&g=$gateway&c=$country&tone=script'?>" dst-
path="getlist.rsc" mode=http;
/import filename=getlist.rsc;
delay 5s;
/file remove getlist.rsc;
</pre>
<BR>
Де: <BR>country - назва або частина назви країни, <BR>type - тип
відповіді (adrlist|route), <BR>якщо тип маршрут то вказати шлюз
gateway
<div style="font-size: 24pt">Автоматизований резолвінг адрес в
країни:</div>
<BR>Також можна поставити у шедулер наступний скрипт:<BR>
<pre>
:local listname "white"
:foreach i in=[/ip/firewall/address-list find list=$listname]
do={:global adr [/ip firewall/address-list/ get $i address ]; :local
result [/tool fetch url="http://url="http://<?php echo
$_SERVER['SERVER_NAME'].$_SERVER['REQUEST_URI'].'mikrotik_get_countr
y.php?tone=adrlist&a=$adr'?> mode=http as-value output=user]; :local
country ($result->"data"); /ip firewall/address-list/set
comment=$country $i}
</pre>
Він в коментарях до записів адреслисту впише країну походження IP-
адреси. Необхідно лише вказати імя адреслиста для обробки та
поставити виконання в планувальник задач, щоб оновлював дані
регулярно.
    </BODY>
</HTML>
```

Mag.js

```
function asuncCityWorld()
{
    city=document.getElementById('citynameworld').value;
    request=new XMLHttpRequest();
    request.open("GET", "select_city_of_world.php?city="+city, true)
    request.onreadystatechange=function()
    {
        if (this.readyState==4)
        {
            if (this.status==200)
            {

document.getElementById('citiesworld').innerHTML=this.responseText;
            }
        }
    }
    request.send(null);
}
function asuncCitiesOfCountry()
{
    country=document.getElementById('country').value;
    request=new XMLHttpRequest();
    request.open("GET",
"select_city_of_country.php?country="+country, true)
    request.onreadystatechange=function()
    {
        if (this.readyState==4)
        {
            if (this.status==200)
            {

document.getElementById('cities_contry').innerHTML=this.responseText
;
            }
        }
    }
    request.send(null);
}
function asuncRouteForm()
{
    type=document.getElementById('rule_type').value;
    request=new XMLHttpRequest();
    request.open("GET", "select_part_router_form.php?type="+type,
true)
    request.onreadystatechange=function()
    {
        if (this.readyState==4)
        {
            if (this.status==200)
            {
```

```
document.getElementById('routerformtwo').innerHTML=this.responseText
;
                }
            }
        }
        request.send(null);
}
```

mikrotik_get_country.php

```
<?php
    include ('connect_db.php');
    //якщо тип_один дорівнює скрипт:
    if ($_GET['tone']=="script")
    {
        $type=$_GET['t'];
        $country=$_GET['c'];
        //якщо шлюз не задано то заміняємо "підстановочним"
значенням
        $gateway=isset($_GET['g'])?$_GET['g']:"192.168.0.1";
        $query_get_country="SELECT ipnetwork FROM ipv4country WHERE
ipv4country.country IN (SELECT id FROM country WHERE country.country
like '%$country%');";
        $country_mysql= mysqli_query($mysql, $query_get_country);
        while ($item= mysqli_fetch_assoc($country_mysql))
        {
            //відносно того маршрут чи адреслист затребувані
виводимо шаблон відповіді
            switch ($type) {
                case "route":
                    echo "/ip route add check-gateway=ping
comment=$country dst-address=$item[ipnetwork] gateway=$gateway\n";
                    break;
                case "adrlist":
                    echo "/ip firewall address-list add
address=$item[ipnetwork] list=$country\n";
                    break;
                default:
                    break;
            }
        }
    }
    //якщо тип_один дорівнює автоматизований адресліст
    if ($_GET['tone']=="adrlist")
    {
        $ipadr=$_GET['a'];
list($firstoctet,$secondoctet,$thirdoctet,$fourthoctet)=explode(".",
$ipadr);
        $dec_ip=ip2long($ipadr);
        //з БД отримуємо країну котрій належить запитувана адреса
        $query_get_country="SELECT country.country as name FROM
ipv4country LEFT JOIN country on country.id=ipv4country.country
WHERE ipv4country.first_octet='$firstoctet' AND ('$dec_ip' BETWEEN
ipv4country.begin and ipv4country.end)";
        $item= mysqli_fetch_assoc(mysqli_query($mysql,
$query_get_country));
        $country=$item['name'];
        //виводимо назву країни
        echo $country;
    }
?>
```

select_city_of_country.php

```
<?php include ('connect_db.php'); ?>
<select name="city">
    <option value="all" selected>Bci</option>
<?php
    $country=$_GET['country'];
    $query="SELECT city.city, city.id FROM city LEFT JOIN country ON
country.country=city.country WHERE (country.id ='$country') Order by
city ASC";
    echo $query;
    $result= mysqli_query($mysql, $query);
    while ($city= mysqli_fetch_assoc($result))
        {
            echo "<option value=$city[id]>$city[city]</option>";
        }
?>
</select>
<input type="submit">
```

select_city_of_world.php

```
<?php include ('connect_db.php'); ?>
<form method="GET" action="find_networks_by_city.php"
target="_blanc">

<select name="city" id="city">
<?php
    $query="SELECT city, id, country FROM city WHERE (city like
'%$_GET[city]%') Order by city.city ASC LIMIT 20";
    $result= mysqli_query($mysql, $query);
    while ($city= mysqli_fetch_assoc($result))
        {
            echo "<option
value=$city[id]>$city[city]($city[country])</option>";
        }
?>
</select>
<input type="submit" value="Показати мережі міста">
</form>
```

select_part_router_form.php

```
<?php include ('connect_db.php'); ?>
<?php
$type=$_GET['type'];
switch ($type) {
    case "route":
        echo "<input type=\"text\" placeholder=\"192.168.0.1\"
title=\"Вкажіть шлюз\" name=\"gateway\">";
        break;
    case "addrlist":
        echo "<input type=\"text\" placeholder=\"так само як імя
об'єкту\" title=\"Вкажіть імя адреслиста\" name=\"adrlist\">";
        break;
    //створюємо шаблон щоб користувач міг сам його заповнити
    case "own":
        echo "<input type=\"text\" placeholder=\"частина1\"
title=\"Вкажіть початок правила\" name=\"p[1]\" size=20>";
        echo "<select name=sel[1]><option value=\"\">
</option><option value=name>імя</option><option
value=adr>адреса</option></select>";
        echo "<input type=\"text\" placeholder=\"частина2\"
title=\"Вкажіть другу частину правила\" name=\"p[2]\" size=20>";
        echo "<select name=sel[2]><option value=\"\">
</option><option value=name>імя</option><option
value=adr>адреса</option></select>";
        echo "<input type=\"text\" placeholder=\"частина3\"
title=\"Вкажіть третю частину правила\" name=\"p[3]\" size=20>";
        echo "<select name=sel[3]><option value=\"\">
</option><option value=name>імя</option><option
value=adr>адреса</option></select>";
        echo "<input type=\"text\" placeholder=\"частина4\"
title=\"Вкажіть кінець правила\" name=\"p[4]\" size=20>";
        echo "<select name=sel[4]><option value=\"\">
</option><option value=name>імя</option><option
value=adr>адреса</option></select>";
        break;
    default:
        break;
}
?>
```

show_statistic_detailed.php

```
<HTML>
<BODY style="background-color: #EFFEFF">
<style>
    table tr:hover
{
    background: #ffd1da; /* фон строки при наведении */
    border: solid #786b59;
}
</style>
<?php
include ('connect_db.php');
//згідно вибору користувача обираємо по кому отримувати статистику,
містах чи країнах
switch ($_POST['item']) {
    case "city":
        $table='ipv4city';
        $query_get_items="SELECT SUM(POW(2,(32-ipv4city.mask))) as
hosts, city FROM ipv4city GROUP by city Order by hosts
$_POST[ascening] limit $_POST[number_items]";
        break;
    default:
        $table='ipv4country';
        $query_get_items="SELECT SUM(POW(2,(32-ipv4country.mask)))
as hosts, country FROM ipv4country GROUP by country Order by hosts
$_POST[ascening] limit $_POST[number_items]";
        break;
}
$items_mysql= mysqli_query($mysql, $query_get_items);
echo "<table border=1 >";
echo "<tr><td>Назва</td><td>К-ть пристроїв</td></tr>";
while ($item= mysqli_fetch_assoc($items_mysql))
{
    if($_POST['item']=='city')
    {
        $query_get_name="SELECT city, country FROM city WHERE
id='$item[city]'";
    }
    else
    {
        $query_get_name="SELECT country FROM country WHERE
id='$item[country]'";
    }

    $iten_name= mysqli_fetch_assoc(mysqli_query($mysql,
$query_get_name));
    echo "<tr><td>$iten_name[city]
$iten_name[country]</td><td>$item[hosts]</td><tr>";
}
echo "</table>";
?>
</BODY>
</HTML>
```