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

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Легковагові керовані бази даних в хмарній інфраструктурі

Виконав: студент IV курсу, групи СП-41

спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

_____ Висоцький Н.І.
(підпис) (прізвище та ініціали)

Керівник _____ Гладьо Ю.Б.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Петрик М.Р.
(підпис) (прізвище та ініціали)

Рецензент _____ Лещишин Ю.З.
(підпис) (прізвище та ініціали)

Тернопіль - 2025

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

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М.Р.
(підпис) (прізвище та ініціали)
«__» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Студенту Висоцькому Назарію Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Легковагові керовані бази даних в хмарній інфраструктурі

Керівник роботи Гладьо Юрій Богданович, к.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «18» 04 2025 року № 4/7-333

2. Термін подання студентом завершеної роботи 18.06.2025 р.

3. Вихідні дані до роботи наукові літературні джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ
1. Огляд предметної області.
2. Теоретична частина
3. Реалізація програмного рішення
4. Безпека життєдіяльності, основи охорони праці
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титулка. 2. Актуальність, мета, задачі дослідження. 3. Прогнозоване зростання ринку
4. Порівняльний аналіз популярних баз даних. 5. Архітектура системи
6. Адміністративна панель. 7. Архітектура адміністративної панелі.
8. Програмні засоби та технології. 9. Діаграма класів системи.
10. Результати експерименту. 11. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці			

7. Дата видачі завдання 18.04.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	19.04 – 20.04.25	*Виконано*
2.	Підбір джерел про керування легковаговими базами		
	даних в хмарній інфраструктурі	21.04 – 24.04.25	*Виконано*
3.	Опрацювання джерел про СУБД в хмарній		
	інфраструктурі	25.04 – 29.04.25	*Виконано*
4.	Проведення дослідження щодо розробки системи		
	керування легковаговими базами даних в хмарній		
	інфраструктурі	30.04 – 04.05.25	*Виконано*
5	Розроблення програмного коду	05.05 – 12.05.25	*Виконано*
6.	Оформлення розділу «Огляд предметної області»	13.05 – 18.05.25	*Виконано*
7.	Оформлення розділу «Теоретична частина»	19.05 – 24.05.25	*Виконано*
8.	Оформлення розділу «Реалізація програмного рішення»	25.05 – 31.05.25	*Виконано*
9.	Виконання завдання до підрозділу «Безпека		
	життєдіяльності, основи хорони праці»	01.06 – 05.06.25	*Виконано*
10.	Оформлення кваліфікаційної роботи	06.06 – 09.06.25	*Виконано*
11.	Нормоконтроль	10.06 – 13.06.25	*Виконано*
12.	Перевірка на плагіат	12.06 – 16.06.25	*Виконано*
13.	Попередній захист кваліфікаційної роботи	17.06 – 20.06.25	*Виконано*
14.	Захист кваліфікаційної роботи	25.06.25	

Студент _____ Висоцький Н.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Гладьо Ю.Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота бакалавра. Тернопільський національний технічний університет імені Івана Пулюя, кафедра програмної інженерії, спеціальність 121 «Інженерія програмного забезпечення». ТНТУ, 2025, Сторінок 51, рисунків 7, джерел 22.

Ключові слова: адміністрування, демон, СУБД, хмарні технології, PostgreSQL, serverless

У роботі виконано огляд предметної галузі, проаналізовано архітектури баз даних. Порівняно можливості PostgreSQL та схожих систем керування базами даних.

Описано особливості втілення serverless баз даних, виконана оцінка можливості оптимізації роботи PostgreSQL, досліджено методи та програмні рішення адміністрування баз даних (традиційні, serverles, Open Source, горизонтальне масштабування, інформаційна безпека).

Розроблено архітектуру програмного рішення для керування легковаговими базами даних із застосуванням хмарних технологій. Архітектура передбачає легкість втілення нових функцій із гарантуванням високої надійності і продуктивності. Наведено особливості розробки демона управління та адміністративної панелі.

Для оцінки ефективності системи було проведено експеримент на віртуальній машині. Результати виконаного тестування показують значне прискорення у виконанні завдань адміністрування, що дає змогу зробити висновок про ефективність та перспективність розробленої системи.

ABSTRACT

Bachelor's thesis. Ivan Puluj Ternopil National Technical University, Department of Software Engineering, specialty 121 "Software Engineering". TNTU, 2025, Pages 51, figures 7, sources 22.

Key words: administration, daemon, DBMS, cloud technologies, PostgreSQL, serverless

The Bachelor's thesis provides a review of the subject area, analyzes database architectures. The capabilities of PostgreSQL and similar database management systems are compared.

The features of implementing serverless databases are described, the possibility of optimizing PostgreSQL operation is assessed, methods and software solutions for database administration are investigated (traditional, serverless, Open Source, horizontal scaling, information security).

The architecture of a software solution for managing lightweight databases using cloud technologies is developed. The architecture provides for easy implementation of new functions while ensuring high reliability and performance. The features of developing a management daemon and administrative panel are presented.

To assess the effectiveness of the system, an experiment was conducted on a virtual machine. The results of the testing show significant acceleration in performing administration tasks, which allows us to conclude about the effectiveness and prospects of the developed system.

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

Конфігурація ролей - налаштування прав користувачів або груп користувачів у системі керування базою даних для контролю доступу до даних та ресурсів.

Хмарні технології – технології розподіленої обробки цифрових даних, за допомогою яких комп'ютерні ресурси надаються інтернет-користувачу як онлайн-сервіс.

Парсинг – синтаксичний аналіз контенту в мережі по певній математичній моделі, створеній на одній з мов програмування, зазвичай на Python, PHP або Java.

СУБД - система управління базами даних.

Тарифікація хмарних рішень - модель ціноутворення для хмарних послуг, коли користувачі оплачують використання ресурсів у міру їх використання.

AWS (Amazon Web Services) – платформа сервісів хмарних обчислень від компанії Amazon.

Bare metal - підхід, у якому використовується фізичний комп'ютерний сервер, використовуваний лише одним орендарем. На відміну від хмарної моделі сервери без операційної системи належать одному користувачеві.

DSL (Domain-Specific Language) - предметно-орієнтована мова програмування.

MySQL – вільна система управління реляційними базами даних.

PostgreSQL - система управління базами даних з відкритим вихідним кодом.

PyCharm CE - інтегроване середовище розробки мови програмування Python, що надає аналізатор коду, графічний відлагоджувач та інструменти для ефективної роботи розробника.

SQL (Structured Query Language) – декларативна мова програмування для взаємодії користувача з базами даних.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. З початку розвитку інформаційних технологій, питання зберігання та управління даними були основними. Еволюція систем СУБД розпочалася з реляційних моделей, запропонованих Едгаром Коддом у 1970-х роках, та прогресувала до сучасних NoSQL та хмарних рішень. Кожен етап еволюції баз даних пов'язаний з зростаючою потребою в оптимізації, забезпеченні надійності зберігання даних і зручності масштабування.

В останні роки ринок хмарних обчислень показує значний ріст. Це зв'язано із зростанням потреби у гнучких, масштабованих та економічно ефективних рішеннях для управління даними та обчислювальними ресурсами.

Правильне адміністрування баз даних є комплексний і ресурсомісткий процес, що вимагає від організацій значних вкладень у підготовку фахівців та підтримку інфраструктури. Важливість ефективного управління базами даних обумовлена не лише їхньою центральною роллю в сучасних інформаційних системах, а й високими вимогами до безпеки, доступності та інтегрованості даних.

Таким чином простежується явна потреба у розробці програмних рішень, які можуть надійно та ефективно здійснювати управління базами даних як у традиційних, так і у bare-metal хмарних середовищах. Особливо важливо, щоб такі рішення забезпечували гнучкість у налаштуванні та інтеграції, можливості розширеного налаштування безпеки та підтримки різних типів даних та операційних систем.

Мета роботи - створення програмного рішення для забезпечення легкокерованості базами даних в хмарній інфраструктурі.

Завдання, які потрібно вирішити, задля досягнення мети:

– здійснити аналіз трендів та найбільш затребуваних функцій у системах автоматизації управління базами даних;

– оцінити ключові можливості та недоліки існуючих рішень;

– вивчити потенційні складності інтеграції та масштабування наявних

систем;

– розробити архітектуру з урахуванням усіх зібраних даних, що передбачає легкість впровадження нових функцій та забезпечення високої продуктивності та надійності;

– виконати тестування системи.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Введення у дослідження

За даними досліджень, проведених великими аналітичними компаніями, як Gärtner і Forrester, ринок хмарних сервісів продовжить зростати на 15-20% щорічно протягом п'яти років [1]. Розподіл прогнозованого зростання областям відбито у табл. 1.1.

Таблиця 1.1 – Зростання ринку (2025 рік - прогноз)

	2023	2024	2025
Cloud Application Infrastructure Services (PaaS)	145,320	175,565	208,644
Cloud Application Services (SaaS)	205,221	250,804	299,071
Cloud Business Process Services (BPaaS)	66,339	72,923	79,364
Cloud Desktop-as-a-Service (DaaS)	2,784	3,466	3,849
Cloud System Infrastructure Services (IaaS)	143,927	169,818	211,856
Total Market	563,592	672,576	802,784

Тим не менш, попри швидкий розвиток хмарних механізмів, усталені bare-metal сервери, що надаються у хмарі, зберігають свою популярність. Прикладом компанії, яка успішно використовує цей підхід, є Selectel, яка пропонує клієнтам можливість оренди фізичних серверів з повним контролем апаратного забезпечення. Це надає клієнтам переваги, пов'язані з високою продуктивністю та безпекою, які нерідко потрібні для спеціалізованих обчислювальних завдань.

Крім того, значна кількість бізнесів продовжує використовувати власні дата-центри та невеликі серверні потужності, що підтверджується даними галузевих

аналітиків [2] (рис. 1.1).

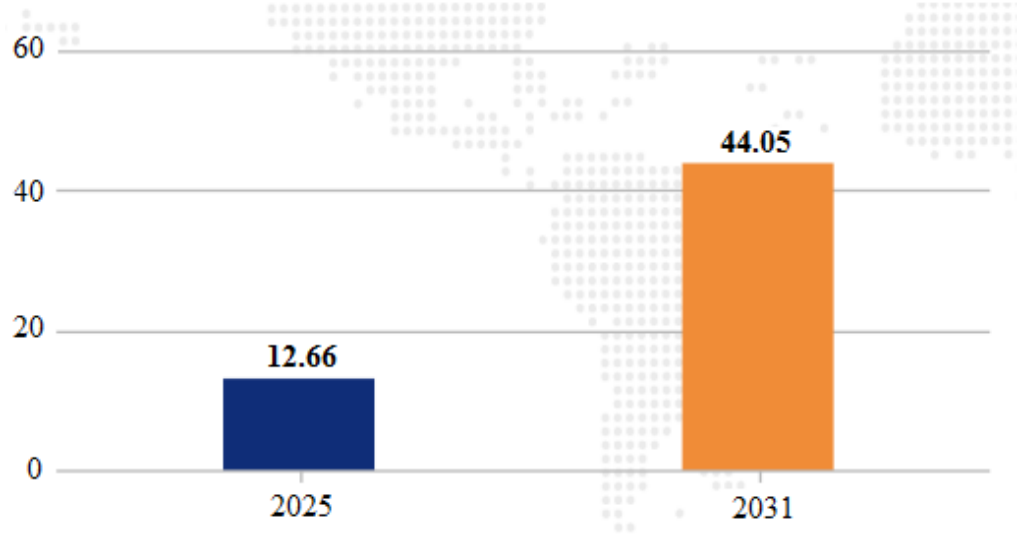

Рисунок 1.1 – Прогноз зростання ринку bare-metal

Цей стійкий тренд обумовлюється кількома ключовими факторами:

– економічна вигода. Незважаючи на видиму вартість володіння та обслуговування власних серверів, для великомасштабних операцій це може бути більш економічним рішенням у довгостроковій перспективі. Привабливість зниження операційних витрат рахунок використання власних ресурсів залишається важливим чинником багатьом компаній;

– контроль та безпека. Управління власними серверами дозволяє організаціям забезпечувати високий рівень безпеки та контролю, що є критичним для обробки чутливої інформації та виконання регульованих завдань;

– незалежність від постачальників хмарних послуг. Наявність власної інфраструктури усуває залежність від сторонніх хмарних провайдерів, що знижує ризики, пов'язані з потенційними простоями та порушеннями доступності сервісів та санкційною політикою.

На фоні цих викликів провайдери хмарних обчислень пропонують рішення, засновані на моделі SaaS (Software as a Service), які обіцяють зниження витрат на управління базами даних за рахунок централізованого адміністрування та обслуговування. Приклад таких рішень є AWS, які надають широкий спектр сервісів для автоматизації та оптимізації роботи з даними. Однак, незважаючи на

переваги, що надаються SaaS-моделлю, її застосування не завжди можливе чи бажане для всіх типів організацій. Значна частка компаній вважає за краще підтримувати управління базами даних всередині власної або bare-metal інфраструктури через вимоги до індивідуалізації, специфічні вимоги безпеки або корпоративну політику.

У тих завдань, автоматизація управління базами даних як спрощує процес адміністрування, а й сприяє підвищенню загальної ефективності і відпопостійкості систем. Підходи до автоматизації повинні враховувати як технічні аспекти, так і бізнес-потреби, адаптуючи технології до умов ринку, що змінюються, і вимогам законодавства.

Таким чином, розробка та впровадження просунутих програмних рішень для управління базами даних є важливим кроком у бік підвищення рівня автоматизації та оптимізації процесів у компаніях, що працюють як у приватній, так і у суспільній сфері. Це забезпечує як економічну вигоду, а й значне поліпшення у сфері управління та захисту критично важливих даних.

1.2 Аналіз та порівняння архітектур баз даних

1.2.1 Загальні відомості

Для повноцінного розуміння процесів оптимізації та адміністрування баз даних потрібне глибоке знання їхнього поточного функціонування та будови. У цьому контексті першорядне значення має аналіз архітектури різних СУБД, що дає змогу виявити ключові закономірності їх роботи, та визначити напрямки для створення гнучких та модульних систем, здатних з рівною ефективністю працювати з різними СУБД.

Концепція баз даних та їх застосування мають глибоке коріння в історії інформаційних технологій [3]. Бази даних є структурованими наборами даних, що зберігаються і керуються за допомогою спеціального програмного забезпечення –

СУБД, які дозволяють користувачам та додаткам взаємодіяти з даними: виконувати їхнє додавання, оновлення, вилучення та видалення. З часом СУБД зазнали значних змін — від простих файлових систем до реляційних і нереляційних (NoSQL) моделей, кожна з яких пропонує різні підходи до організації та управління даними.

На сьогоднішній день SQL залишається превалюючим стандартом у галузі реляційних баз даних. Ці системи, такі як PostgreSQL, MySQL, Oracle і Microsoft SQL Server, орієнтовані працювати з даними, організованими у вигляді таблиць із суворо певними зв'язками. Не менш значущим є сегмент NoSQL баз даних, який пропонує гнучкість у структуризації даних, з огляду на поточний стан ринку [4] стає зрозуміло, що стандарт SQL ще довго залишатиметься основним у питанні зберігання даних.

PostgreSQL - це чудовий приклад високопродуктивної SQL бази даних, котра втілює більшість стандартів SQL і надає розширені можливості, такі як підтримка JSON, GIN індексування та багато іншого. Крім того, ця СУБД з відкритим вихідним кодом, що дозволяє безперешкодно оцінити внутрішню реалізацію. Популярність PostgreSQL зростає з кожним роком [5], що робить цей вибір ідеальним для детального аналізу та дозволяє не тільки зрозуміти особливості роботи реляційних баз даних, але й оцінити потенціал підвищення продуктивності та оптимізації, а також проаналізувати основні аспекти адміністрування СУБД.

Вивчаючи та порівнюючи архітектуру PostgreSQL з іншими СУБД, можна виявити універсальні та унікальні аспекти, що необхідно для розуміння загальних принципів роботи баз даних та проектування гнучкої модульної СУБД, здатної адаптуватися до різних вимог та обставин, а також забезпечити покриття всіх потреб адміністрування.

1.2.2 Аналіз архітектури PostgreSQL

Вона є однією з найрозвиненіших відкритих реляційних СУБД, враховуючи

її підтримку широкого спектру типів даних, транзакцій, багатоверсійності та розширюваності. Архітектура PostgreSQL дозволяє їй обробляти великі обсяги даних із високою продуктивністю і надійністю, що робить її ідеальним вибором для складних і масштабованих додатків.

Архітектурний огляд. PostgreSQL слідує за класичною багатопроцесною моделлю клієнт-сервер, де кожне клієнтське з'єднання обслуговується окремим процесом на сервері. Ця модель забезпечує високу ізоляцію та стабільність, оскільки помилки або витоку пам'яті в одному процесі не впливають на інші процеси.

Система зберігання. Файлова структура PostgreSQL використовує складну структуру каталогів і файлів, де кожна база даних представлена своїм каталогом всередині каталогу даних кластера. Таблиці та індекси складаються з одного або більше файлів, які можуть зростати в міру додавання даних.

Таблиці та індекси: PostgreSQL підтримує різні типи індексів: B-tree, Hash, GiST, SP-GiST, GIN та BRIN [6]. Кожен тип індексу оптимізовано для різних видів запитів та типів даних, що дозволяє користувачам налаштувати індексацію відповідно до потреб програми.

Транзакційна модель та MVCC. PostgreSQL використовує механізм управління паралельними версіями даних (MVCC) [6], який дозволяє системі забезпечувати високий рівень паралельного доступу та одночасно підтримувати узгодженість даних. MVCC запобігає "брудному" читанню та дозволяє читання даних під час їх зміни без блокування операцій запису:

Транзакції: PostgreSQL підтримує повноцінні транзакції зі стандартами ACID, що гарантує атомарність, узгодженість, ізоляцію та довговічність операцій.

Журнал транзакцій (WAL): Для підтримки транзакцій PostgreSQL використовує журнал запису попереднього логування (WAL), який допомагає відновлювати базу даних після збоїв та забезпечує можливість реплікації даних.

Продуктивність та оптимізація. PostgreSQL включає просунутий планувальник запитів, який аналізує та оптимізує запити для досягнення найкращої продуктивності. Планувальник використовує статистику за таблицями та

індексами для вибору найбільш ефективного плану виконання запиту.

Крім того, PostgreSQL відомий своєю високою розширюваністю, що дає змогу розробникам збільшувати кількість нових функцій, типів даних і навіть мов обробки запитів через інтерфейс розширень.

Таким чином проведений аналіз архітектури PostgreSQL підкреслює її значущість як однієї з відкритих провідних реляційних СУБД, яка забезпечує високу продуктивність, надійність і масштабованість. Система відрізняється багатопроцесною моделлю клієнт-сервер, просунутими механізмами індексації, підтримкою різних типів даних та транзакцій, що відповідають стандартам ACID.

Особливості, такі як MVCC, журналювання через WAL та можливості оптимізації запитів роблять PostgreSQL ідеальним рішенням для обробки великих навантажень та паралельних доступів. Ці аспекти, у поєднанні з високою розширюваністю системи, надають цінні напрямки розробки нових модульних СУБД, здатних адаптуватися до різноманітних вимог.

1.2.3 Порівняльний аналіз PostgreSQL та альтернативних СУБД

У цьому підрозділі здійснюється детальне порівняння PostgreSQL із трьома іншими популярними СУБД: MySQL, MongoDB та Oracle. Аналіз буде сфокусований на основних технічних характеристиках, можливостях та особливостях, критично важливих для вибору СУБД у різних додатках.

PostgreSQL: Детальний аналіз був наведений у попередній частині.

MySQL – це дуже популярна реляційна СУДБ , яка відома своєю простотою встановлення та використання, а також надійною продуктивністю на відносно невеликих та середніх навантаженнях. MySQL добре підходить для веб-застосунків і часто використовується в поєднанні з PHP і Apache. На відміну від PostgreSQL, MySQL використовує традиційніші методи управління блокуваннями, що може обмежувати її продуктивність при високих навантаженнях [7]. Однак, MySQL

пропонує широкий спектр плагінів та інструментів, які можуть значно розширити її функціональність та керованість, роблячи її кращим вибором для багатьох кодерів та адмінів баз даних.

MongoDB чудова своєю документно-орієнтованою структурою, що використовує формат BJSON, аналогічний JSON. Це дозволяє досягти значної гнучкості в управлінні схемами даних та полегшує масштабування. Все це робить MongoDB оптимальним вибором для додатків, які потребують високої адаптивності до змін схеми даних. MongoDB ідеально підходить для опрацювання значних об’ємів даних. MongoDB володіє просунутими функціями шардингу та автоматичного балансування навантаження, що значно полегшує горизонтальне масштабування.

Oracle Database - це одне з найбільш просунутих та функціональних комерційних рішень для управління бзами даних, орієнтоване на потреби великих підприємств та складних застосунків, які потребують рівня високої безпеки, масштабованості та надійності. Підтримка широкого спектру технологій, включаючи машинне навчання та автоматизоване керування даними, робить Oracle чудовим інструментом, незважаючи на високу вартість та складність підтримки. Oracle забезпечує надійну підтримку транзакцій, багатоверсійне управління даними та ефективні засоби для оптимізації запитів. [8].

У табл. 1.2 наведено порівняння за основними характеристиками кожної бази даних.

При аналізі різних СУБД, таких як PostgreSQL, MySQL, MongoDB і Oracle стає очевидно, що вони всі значною мірою застосовують подібні підходи до управління даними. Це включає підтримку транзакційності, різні техніки, індексації та можливості масштабування. Кожна з цих систем має унікальні характеристики, які можуть бути більш відповідними або менш кращими в залежності від специфічних вимог проекту та завдань, які ставить бізнес.

Таблиця 1.2 – Порівняльний аналіз популярних баз даних

Параметр/СУБД	PostgreSQL	MySQL	MongoDB	Oracle
Тип даних	Широка підтримка, включаючи JSON, XML, BJSON	Широкий діапазон, включаючи JSON	Документо-орієнтовані JSON -подібні структури	Широкий діапазон, включаючи JSON, XML
Модель транзакцій	Повна підтримка ACID	Повна підтримка ACID	Повна підтримка ACID із версії 4.0	Повна підтримка ACID
Індексація	B-tree, Hash, GiST, SP-GiST, GIN, BRIN	B-tree, Hash, Full-text, Spatial	B-tree, Full-text, Geo	B-tree, Bitmap, Spatial, Full-text
Журналювання	Підтримка WAL	Підтримка бінарного лога	Підтримка журналу операцій	Підтримка REDO logs, Archive logs
Масштабованість	Вертикальна і горизонтальна через розширення	Вертикальна і горизонтальна через реплікацію	Горизонтальна, сильна підтримка шардінга	Висока, підтримка різних опцій розподілу даних
Ліцензування	Відкритий вихідний код (PostgreSQL License)	Відкритий вихідний код (GPL) для Community Edition, комерційний для Enterprise	Відкритий вихідний код (SSPL), комерційний	Комерційний з опціями відкритого вихідного коду для деяких компонентів

PostgreSQL виділяється завдяки своїй технічній зрілості, надійності та можливості розширення. PostgreSQL пропонує одну з найбільш досконалих реалізацій реляційної моделі даних, що підтримує виконання складних запитів, роботу з розширеними типами даних та використання сучасних механізмів індексації. MVCC підвищує ефективність обробки конкурентних операцій читання

та запису. PostgreSQL забезпечує значну гнучкість та контроль, що особливо важливо для систем, що потребують високого ступеня адаптації та надійності.

Отже, хоча вибір конкретної СУБД повинен базуватися на ретельному аналізі специфічних технічних вимог, бюджету та стратегічних цілей організації, PostgreSQL є основним базовим вибором для багатьох організацій завдяки своїй технічній вигідності та технічній досконалості. PostgreSQL не тільки відповідає широкому спектру бізнес-застосунків, але й забезпечує технологічну гнучкість та оптимізацію витрат. Водночас, всі розглянуті СУБД, так чи інше, пропонують схожі базові концепції та функціонал, що підкреслює універсальність і взаємозамінність технологій у сучасних інформаційних системах.

2 ТЕОРЕТИЧНА ЧАСТИНА

2.1 Особливості реалізації serverless баз даних

З розвитком хмарних технологій, serverless архітектури стали основними складовими сучасних інформаційних систем У контексті баз даних, serverless рішення пропонують значні переваги в порівнянні з класичними підходами до розгортання та управління базами даних. У цьому підрозділі будуть розглянуті особливості serverless баз даних, на прикладі рішень від AWS та Selectel, а також проведено порівняння з класичними підходами до розгортання.

Serverless бази даних надають динамічне масштабування та оптимізацію ресурсів без необхідності ручного керування інфраструктурою. Ці системи автоматично адаптуються до змін у навантаженні, забезпечуючи високу продуктивність та доступність за мінімальних витрат на управління.

AWS пропонує кілька serverless рішень, таких як Amazon Aurora Serverless та Amazon DynamoDB [9]. Aurora Serverless автоматично масштабує обчислювальні потужності залежно від активності бази даних, дозволяючи користувачам платити лише за ресурси, які вони використовують. DynamoDB надає повністю керовану базу даних NoSQL з можливістю масштабування пропускної здатності та зберігання даних без попереднього планування.

Selectel пропонує керовані бази даних, включаючи PostgreSQL та MySQL, які звільняють користувачів від необхідності управління інфраструктурою, забезпечуючи високу доступність та автоматичне резервне копіювання.

Порівняння з класичними базами даних. Традиційні бази даних, такі як PostgreSQL чи MySQL запущені на віртуальних машинах у хмарі (наприклад, на AWS EC2 [9]) або на власних серверах, вимагають від адміністраторів визначення ємності, налаштування реплікації та резервного копіювання, а також моніторингу продуктивності та доступності.

На відміну від цього, serverless БД спрощують ці завдання, автоматизуючи масштабування, резервне копіювання та інші операції управління [10]. Це дозволяє

розробникам зосередитися на дизайні та оптимізації програм, не турбуючись про підтримку баз даних.

У табл. 2.1 наведено порівняння основних аспектів роботи Serverless та класичних баз даних.

Таблиця 2.1 – Порівняння підходів

Підхід/Параметр	Serverless бази даних	Класичні бази даних
Масштабування	Автоматичне, динамічне	Ручне, вимагає попереднього планування
Управління інфраструктурою	Не потрібно, повністю керується постачальником	Потрібне активне управління та моніторинг
Продуктивність	Оптимізується постачальником, може коливатися при різких стрибках навантаження	Залежить від обраної конфігурації і може бути стабільнішим
Підходить для	Різноманітні програми з змінним навантаженням	Застосунків із передбачуваним навантаженням

Таким чином, можна стверджувати, що серверні бази даних забезпечують поєднання гнучкості, масштабованості та економічної вигоди. Ці якості роблять їх чудовим рішенням для множини сучасних застосунків, особливо в умовах цифрового світу, що розвивається, де потрібна швидка адаптація до змін.

2.2 Оцінка можливості оптимізації роботи PostgreSQL

СУБД PostgreSQL широко відома своєю надійністю та потужними функціональними можливостями, PostgreSQL використовується множиною компаній для обробки даних, що варіюються від фінансових транзакцій до

геопросторової інформації. Популярність PostgreSQL в останні роки зростає завдяки гнучкості та відкритому вихідному коду, що дозволяє значно знизити залежність від вендерських рішень. Проте, враховуючи стрімкий розвиток хмарних технологій та зростаючі вимоги до продуктивності та масштабованості систем, питання оптимізації PostgreSQL стають критично важливими. Виникають дебати про те, чи достатньо існуючих можливостей PostgreSQL для задоволення сучасних потреб, або потрібне додаткове вдосконалення, щоб покращити його продуктивність та керованість в умовах хмарної інтеграції.

Аналіз необхідності оптимізації PostgreSQL. Одним із ключових аргументів на користь оптимізації PostgreSQL є прагнення підвищити загальну продуктивність системи. В умовах хмарних обчислень, де доступні ресурси можуть змінюватися в залежності від поточних потреб, ефективність обробки запитів та швидкість роботи бази даних безпосередньо впливають на досвід користувача та операційні витрати. Оптимізація запитів, покращення індексації та тонке налаштування конфігурацій можуть значно скоротити час відповіді та підвищити продуктивність транзакцій, що особливо важливо для високонавантажених та масштабованих застосунків. Ці покращення сприяють кращому розподілу навантаження та більш ефективному використанню хмарних ресурсів.

Крім того, оптимізація PostgreSQL може значно знизитися витрати завдяки раціональнішому використанню обчислювальних і дискових ресурсів. У хмарному середовищі, де оплата послуг часто залежить від обсягу споживаних ресурсів (наприклад, CPU, пам'яті, дискового простору та пропускної спроможності мережі), ефективне керування даними та оптимізація запитів можуть суттєво зменшити щомісячні витрати.

Проактивна оптимізація, така як поліпшення планів запитів або перехід на ефективніші алгоритми стиснення даних, може зменшити необхідний обсяг зберігання і, відповідно, скоротити витрати на дисковий простір [6].

Масштабованість – критично важливий аспект хмарних додатків. PostgreSQL відомий своєю здатністю масштабуватися [6], проте додаткова оптимізація може покращити цей процес, роблячи його більш гладким та менш ресурсомістким.

Зокрема, оптимізація може поліпшити вертикальне і горизонтальне масштабування бази даних, дозволяючи їй більш ефективно реагувати на обсяги даних і запитів, що змінюються, без шкоди для продуктивності. Гнучкість в управлінні навантаженням і можливість швидкого масштабування вгору або вниз у відповідь на зміни в навантаженні без значних тимчасових витрат або фінансових втрат є значними перевагами в бізнес-оточенні, що динамічно змінюється.

Хмарні технології постійно розвиваються, і бази даних, включаючи PostgreSQL, повинні відповідати новим вимогам та стандартам. Оптимізація PostgreSQL може включати оновлення системи для забезпечення кращої сумісності з новими хмарними сервісами та архітектурами, такими як контейнеризація та мікросервіси.

Це може допомогти у спрощенні процесів розгортання, управління, моніторингу та масштабування баз даних у хмарній інфраструктурі.

Таким чином, існує ряд вагомих причин для початку робіт з оптимізації PostgreSQL, спрямованих на покращення продуктивності, зниження витрат, покращення масштабованості та інтеграції з сучасними хмарними технологіями. Проте слід також розглянути аргументи проти подальшої оптимізації, які будуть обговорені в наступному підрозділі.

Аргументи проти подальшої оптимізації PostgreSQL. PostgreSQL є однією з найбільш зрілих і надійних СУБД на ринку з більш ніж 25-річною історією розвитку. За ці роки PostgreSQL накопичив значний обсяг функціональності та оптимізації, глибоко інтегрованих у його архітектуру. В результаті, база даних має високу стабільність і надійність, що робить її кращим вибором для критично важливих систем. Додаткові спроби "полегшити" або оптимізувати систему можуть порушити цю стабільність, потенційно призводячи до нових помилок та вразливостей.

Крім того, інтенсивні оптимізації можуть спричиняти не лише технічні складності, а й ризики для поточних операцій. Зміни в глибоко укорінених компонентах бази даних можуть призвести до непередбачених наслідків, включаючи зниження продуктивності та втрату даних. Також існує ризик

впровадження нових баґів, які можуть бути не відразу помічені, що особливо критично для систем, які потребують високого рівня доступності та безпеки.

Крім того, приклади існуючих рішень такі як timescaledb [6] показують, що проведення глобальної оптимізації складно реалізовано, і роботи з підвищення продуктивності можливі тільки в окремих крайніх випадках.

Враховуючи зрілість, надійність, складність управління та ризики, пов'язані з додатковою оптимізацією PostgreSQL, можна дійти висновку, що подальша глобальна оптимізація не є необхідною.

PostgreSQL вже досить добре адаптований для роботи в сучасних хмарних середовищах, і фокус повинен бути зміщений на використання та покращення існуючих можливостей системи та адаптацій СУБД під сучасні реалії промислової експлуатації.

2.3 Методи та інструменти адміністрування баз даних

2.3.1 Традиційні підходи до адміністрування баз даних

Традиційні методи містять широкий спектр практик і процедур, починаючи від ручного управління та моніторингу баз даних до використання автоматизованих інструментів, що полегшують щоденну роботу адміністраторів.

Керування даними в традиційних системах часто пов'язане з регулярним обслуговуванням, таким як оновлення схем даних, керування доступом та оптимізація зберігання. Резервне копіювання та відновлення даних виконуються за допомогою заздалегідь запланованих процедур, що часто виконуються вручну, які гарантують безпеку даних у разі збоїв або втрат. Важливою частиною є також підтримка безпеки даних, включаючи регулювання прав та захист від несанкціонованого доступу.

Моніторинг та оцінка продуктивності баз даних – критично важливі завдання, які допомагають виявляти вузькі місця та потенційні проблеми у роботі

систем. Традиційні методи масштабування та оптимізації ресурсів часто вимагають значних зусиль та часу для планування та реалізації.

У контексті PostgreSQL традиційні методи адміністрування включають використання таких інструментів, як pgAdmin для графічного управління базами даних або використання командного рядка для більш тонкого настроювання та управління. Автоматизація завдань, таких як резервне копіювання та моніторинг часто здійснюється за допомогою скриптів на Bash і Python, які можуть бути інтегровані з системними завданнями cron для регулярного виконання [11].

Традиційні підходи до адміністрування можуть бути не так ефективні в умовах сучасних вимог до масштабованості та гнучкості, особливо в контексті великих даних та хмарних обчислень.

Такі підходи вимагають від адміністраторів глибоких знань та умінь, а також можуть спричиняти великі витрати часу та ресурсів на підтримку та оновлення систем. Крім того швидкість реакції адміністратора найчастіше значно повільніша за реакцію повністю автоматизованих систем

Сучасні підходи, такі як хмарні та serverless рішення, пропонують більш гнучке та масштабоване управління базами даних, що може бути кращим для підприємств та застосунків, котрі динамічно розвиваються. Основним недоліком традиційного адміністрування є великі трудовитрати та помилки, що неминуче виникають при ручному управлінні

2.3.2 Управління serverless базами даних у хмарних середовищах

У сучасному світі хмарних технологій, serverless бази даних пропонують значні переваги за рахунок своєї масштабованості та зручності управління. Найбільш популярні платформи, що надають serverless послуги баз даних, включають AWS Lambda з Amazon DynamoDB, Google Cloud Functions з Firebase і Microsoft Azure Functions з Cosmos DB. Кожна з цих платформ пропонує унікальні

особливості та можливості. AWS Lambda у поєднанні з DynamoDB надає потужні інструменти для роботи з NoSQL даними, підтримуючи високу доступність та довготривале зберігання даних. Google Cloud Functions інтегрується з Firebase для надання реального часу оновлень даних та легкої синхронізації з мобільними програмами. Azure Functions з Cosmos DB пропонує глобальне розподілення та мульти-модельне зберігання даних, що робить його ідеальним для комплексних застосунків, що потребують масштабування на міжнародному рівні [12].

Serverless бази даних надають суттєві переваги з погляду економічної ефективності та масштабованості. Клієнти платять лише за фактично використані обчислювальні ресурси, що знижує витрати на підтримку інфраструктури, що не використовується. Також підтримка serverless баз, значно спрощується завдяки розвиненому UI (рис. 2.1).

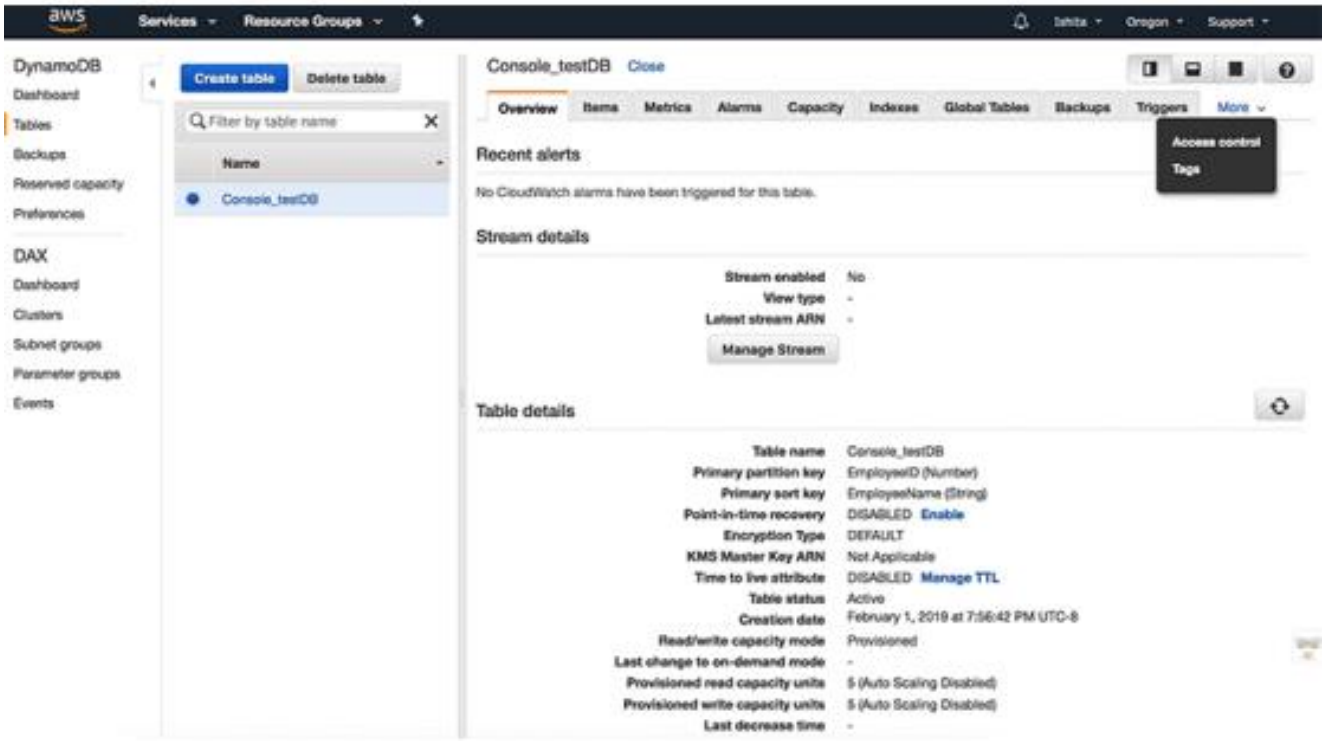

Рисунок 2.1 – Консоль керування DynamoDB

Проте, існують і значні виклики. Проблеми з продуктивністю, особливо з холодним стартом функцій, можуть впливати на досвід користувача, викликаючи затримки при першому запуску застоунків після періоду неактивності [13]. Безпека

та управління доступом також є складними завданнями, оскільки управління доступами та захист даних у хмарному середовищі потребують ретельного налаштування та постійного моніторингу

Застосування serverless технологій для керування базами даних показує безліч успішних проектів у різних галузях. Не тільки для невеликих компаній-початківців, але і для великих і бізнесів. Компанія DropBox застосовує AWS DynamoDB для обробки метаданих документів, що зберігаються в реальному часі, забезпечуючи високу доступність і масштабованість своїх сервісів [14].

2.3.3 Аналіз Open Source рішень

Для адміністрування баз даних широко використовуються Open Source інструменти, які пропонують потужні функціональності за низьких витрат. Наприклад, phpMyAdmin є одним із найпопулярніших веб-інтерфейсів для керування базами даних MySQL. Цей інструмент дозволяє користувачам виконувати широкий спектр завдань, від базових операцій вставки та видалення до складніших запитів та адміністративних функцій. Аналогічно, pgAdmin - це найбільш популярний інструмент, котрий застосовується для управління базами даних PostgreSQL, що надає графічний інтерфейс для налаштування, управління та розгортання баз даних [15].

Обидва ці інструменти підтримують автоматизацію рутинних завдань, що є критично важливим для забезпечення ефективності та оптимізації роботи адміністраторів баз даних. Автоматизація є ключовим фактором, що дозволяє прискорити операції та скоротити можливість людської помилки в управлінні базами даних [16].

На додачу до управлінських інструментів, існує безліч Open Source фреймворків та бібліотек, які використовуються для автоматизації адміністративних завдань. Наприклад, Ansible надає модулі для керування базами

даних, які дозволяють автоматизувати завдання, такі як розгортання, оновлення та резервне копіювання. Інструменти, такі як Puppet та Chef [17], також пропонують потужні можливості для керування конфігураціями баз даних, дозволяючи адміністраторам оптимізувати та підтримувати середовища баз даних з високим ступенем контролю та точності. Однак такі системи не мають доброзичливого UI і вимагають навичок адміністратора.

Використання Open Source рішень для управління базами даних пропонує ряд переваг, включаючи значне зниження витрат. Крім того, гнучкість Open Source рішень дозволяє адміністраторам налаштовувати системи відповідно до унікальних вимог бізнесу, покращуючи масштабованість і адаптованість до потреб, що змінюються.

Незважаючи на численні переваги, використання Open Source рішень пов'язане з деякими викликами. Однією з основних проблем є підтримка: оскільки ці інструменти часто розробляються спільнотами, відсутність централізованої підтримки може створити труднощі для організацій, які потребують негайного вирішення проблем чи спеціалізованої допомоги. Крім того, питання безпеки є значним бар'єром, оскільки відкритий вихідний код може бути схильний до вразливостей, якщо не забезпечується належний рівень оновлення та захисту.

2.3.4 Горизонтальне масштабування

Наприклад розглянемо MongoDB і PostgreSQL. MongoDB використовує шардинг для розподілу даних по різних серверах, що дозволяє лінійно масштабувати програми зі збільшенням навантаження. У PostgreSQL горизонтальне масштабування часто досягається через партиціонування, яке управляє поділом великих таблиць на більш дрібні та керовані частини [11].

Однак, незважаючи на свої переваги, горизонтальне масштабування спричиняє значні складнощі в адмініструванні. Ручне керування багатьма шардами

або партиціями може бути надзвичайно складним, вимагаючи від адміністраторів баз даних потужного рівня спеціальних знань, умінь і уваги до найменших деталей. Кожен шард або партиція повинна бути належним чином налаштована для забезпечення балансу навантаження, а також підтримки ефективності та продуктивності на всіх рівнях системи.

Проблеми виникають через необхідність постійного моніторингу та налаштування для оптимізації продуктивності та доступності. Наприклад, неправильно налаштовані шарди в MongoDB можуть призвести до нерівномірного розподілу даних, що зменшує переваги масштабування і може спричинити перевантаження окремих вузлів. Аналогічно, у PostgreSQL, неефективне партиціонування може утруднити виконання запитів та збільшити час відповіді системи [18].

Враховуючи складність процесів, більшість організацій прагне автоматизувати управління горизонтальним масштабуванням. Використання сучасних інструментів автоматизації та оркестрації, таких як Kubernetes для управління контейнеризованими базами даних або Terraform для інфраструктурного коду, може значно спростити адміністрування. Ці інструменти дозволяють адміністраторам налаштовувати та масштабувати бази даних з мінімальним втручанням, автоматизуючи рутинні завдання та покращуючи надійність системи.

Горизонтальне масштабування має ключове значення для управління сучасними базами даних, забезпечуючи необхідний рівень масштабованості та продуктивності. Однак складнощі, що виникають при його адмініструванні, потребують детального підходу і часто змушують використовувати автоматизовані рішення для ефективного управління інфраструктурою.

2.3.5 Інформаційна безпека в адмініструванні

У сучасному світі всі системи зберігання даних стикаються з великою кількістю кіберзагроз, ці загрози можуть нести ризики як для самої інформації, так і для цілісності системи.

Однією з найбільш поширених загроз є SQL-ін'єкції, при яких зловмисники вводять шкідливий SQL-код у запити, що дозволяє читати, змінювати або видаляти дані, до яких вони не повинні мати доступ.

Іншою поширеною проблемою є кібератаки на рівні доступу. Користувачі отримують доступ до даних або функцій бази даних, до яких у них доступу не повинно бути, що порушує принципи конфіденційності та цілісності даних, таким чином наражаючи на небезпеку.

Додаткові загрози можуть виникати від працівників, котрі володіють доступом до баз даних. Власне вони і можуть навмисне або через недогляд викликати витік або втрату даних. Усі озвучені вразливості вимагають серйозної уваги та розробки стратегій щодо їх запобігання. Наслідки можуть бути катастрофічними, аж до повної втрати даних та юридичних наслідків для організації.

Для забезпечення захисту даних та протидії можливим атакам існує низка інструментів та технологій. Шифрування даних, як на етапі передачі, так і зберігання, є критично важливим шаром захисту, що забезпечує збереження даних. При правильній конфігурації навіть при втраті даних вони залишаться нечитаними без відповідних ключів шифрування.

Додатково, СУБД часто містять інструменти для моніторингу та логування дій користувачів, що дозволяє адміністраторам відстежувати всі дії та оперативно реагувати на незвичайні чи неавторизовані запити.

Важливо мати ретельно протестовані плани відновлення після збоїв. Ці плани повинні включати регулярне створення резервних копій даних, котрі повинні зберігатися в безпечному місці і регулярно оновлюватися. У разі кібератаки чи

технічного збою ці заходи можуть бути вирішальними для швидкого відновлення операцій без втрат даних.

Розробка та підтримання ефективної політики безпеки потребує постійної уваги та адаптації до мінливого ландшафту загроз. Це важливий аспект адміністрування баз даних, що потребує інтеграції як технологічних, так і організаційних заходів.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО РІШЕННЯ

3.1 Розробка архітектури системи

Вибір моделі взаємодії між клієнтом та сервером у системах керування конфігураціями значно впливає на продуктивність, масштабованість та безпеку. Після аналізу існуючих на ринку рішень було обрано модель, де клієнт опитує сервер. У такій моделі керуючий сервер опрацьовує запити клієнтів (керованих серверів), надаючи централізоване керування конфігураціями. Цей підхід дозволяє значно спростити розробку та підтримку, досить надійний та дозволяє швидко розгорнути складну систему з множиною серверів. Однак, він також має недоліки, включно із потенційні затримки через велику завантаженість сервера та ризики безпеки, пов'язані з централізацією управління.

На ринку існує безліч інструментів для адміністрування групи серверів, в включно із такими рішеннями як Puppet. Puppet пропонує автоматизоване керування конфігураціями, що допомагає забезпечити ідентичність налаштувань на всіх серверах системи [19]. Це знижує ймовірність помилок та спрощує процес масштабування та управління. Однак, Puppet та подібні до нього інструменти можуть бути складними в налаштуванні та вимагати значних витрат часу на вивчення та підготовку персоналу.

При розробці системи було прийнято рішення звернути особливу увагу на спрощення інтерфейсу та налаштування.

Основна архітектура нашої системи включає два ключові компоненти:

– демон контролю;

– панель адміністрування.

Демон контролю встановлюється на кожен сервер. Він відповідає за збір метрик з сервера, обробку команд управління та валідацію цих команд через перелік дозволених дій. Це дозволяє забезпечити не тільки моніторинг стану серверів у реальному часу, але і швидке реагування на зміни у вимогах або проблеми, що виникають.

Панель адміністрування надає централізований інтерфейс для керування всіма аспектами системи. За допомогою панелі можна агрегатувати метрики з усіх серверів, керувати доступом користувачів, налаштовувати бекапи та реагувати на системні алерти. Панель спрощує адміністрування великих систем, надаючи інтуїтивно зрозумілі інструменти контролю та управління.

Пропонована архітектура наведена на рис. 3.1.

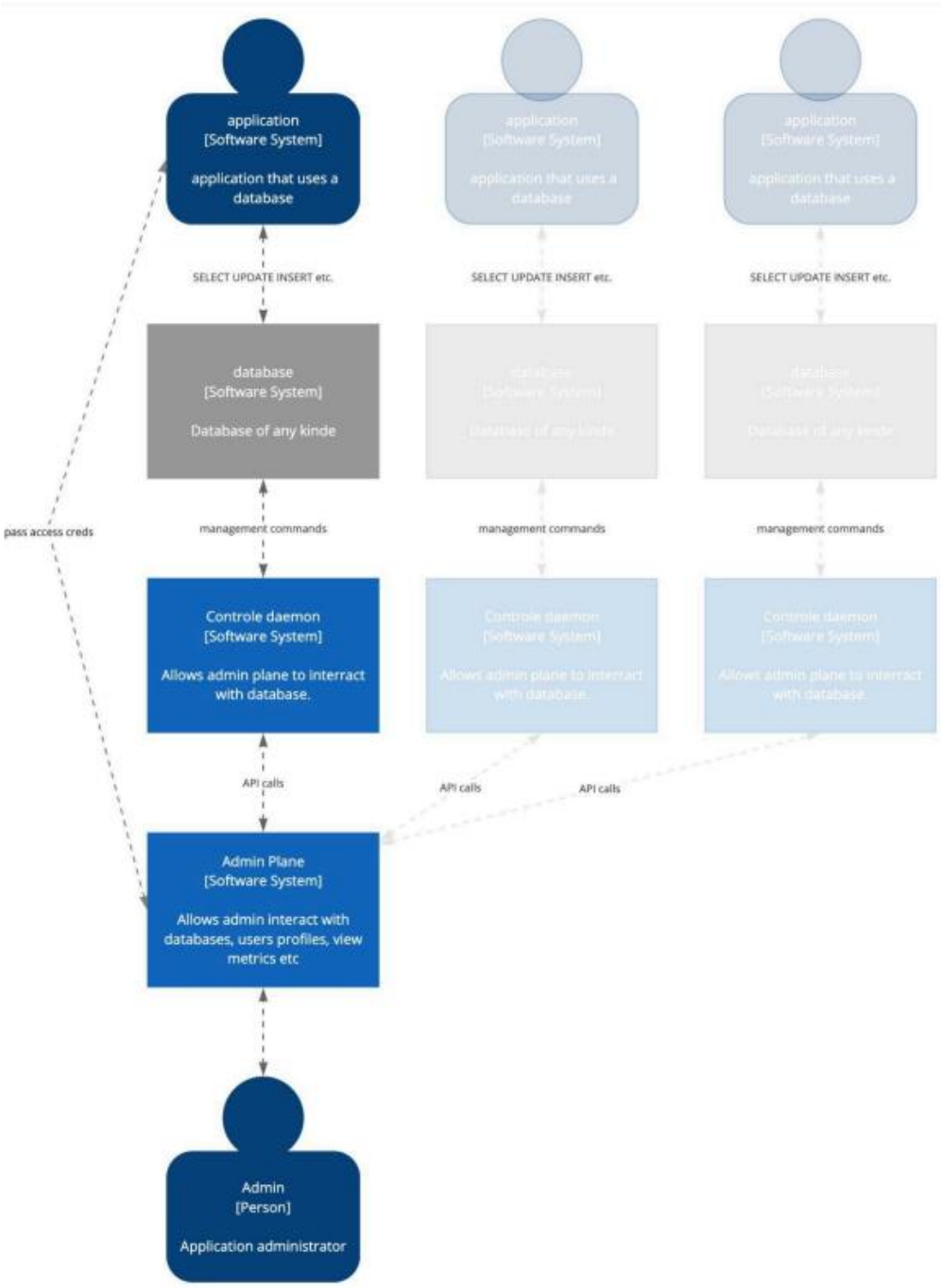

Рисунок 3.1 – Архітектура системи

Вона призначена для усунення складнощів, пов'язаних з масштабуванням та керуванням великими системами, забезпечуючи при цьому високий рівень безпеки та контролю.

3.2 Розробка демона управління

Демон управління є ключовим компонентом системи адміністрування серверів, що виконують функції_моніторингу, управління конфігураціями та обробки команд. Розроблений для роботи на кожному сервері, демон управління забезпечує реальний моніторинг та оперативне реагування на зміни в системі, що критично важливо для підтримки високої доступності та продуктивності.

Демон управління відповідає за кілька основних завдань:

– обробка команд. Демон приймає та виконує команди від панелі адміністрування, забезпечуючи таким чином керування сервером у реальному часі. Команди можуть включати перезавантаження сервісів, оновлення ПЗ або зміну налаштувань безпеки;

– збір метрик. Систематично збираються дані про завантаження процесора, використання пам'яті, стан мережевих підключень, роботу бази даних та інші критично важливі параметри. Ці метрики передаються на панель адміністрування для аналізу та прийняття рішень;

– валідація команд: Усі команди перевіряються на відповідність списку дозволених дій, що запобігає виконанню потенційно шкідливих або помилкових операцій.

Демон управління розробляється мовою програмування Python, що забезпечує гнучкість, оперативність та легкість інтеграції з іншими системними компонентами. Python підтримує різні бібліотеки та фреймворки, які можуть бути застосованими для розробки мережевих з'єднань, паралельної обробки та роботи з базами даних.

Для спрощення управління конфігураціями та підвищення рівня абстракції розроблена спеціалізована DSL. Ця мова необхідна для передачі команд з адміністративної панелі до демона, а також дозволяє адміністраторам валідувати команди та скрипти управління у термінах, що спрощує процес розробки та знижує ймовірність помилок.

Демон управління тісно інтегрований з панеллю адміністрування, яка агрегує зібрані метрики та надає інтерфейс користувача для відправки команд. Інтеграція з базами даних та іншими серверами забезпечується через захищені API та протоколи передачі даних, що гарантує цілісність та конфіденційність інформації, що передається.

Безпека демона управління забезпечується через використання шифрування для всіх вхідних та вихідних з'єднань, автентифікацію користувачів та рольову модель доступу. Застосування Zero Trust архітектури дозволяє забезпечити належний рівень безпеки та контролю [20].

Ця архітектура демона управління забезпечує ефективне та безпечне керування серверами в рамках розподіленої системи, мінімізуючи ручне втручання та підвищуючи автоматизацію процесів адміністрування.

3.3 Розробка адміністративної панелі

Адміністративна панель є центральним елементом системи управління та моніторингу серверів, надаючи адміністраторам потужний інструмент контролю та аналізу стану інфраструктури. Ця панель дозволяє централізовано керувати налаштуваннями, переглядати важливі метрики і реагувати на наявні інциденти в реальному режимі.

Адміністративна панель має низку критично важливих функцій:

– керування доступом. Панель дозволяє налаштовувати права доступу для різних користувачів та груп, забезпечуючи, щоб лише авторизований персонал мав

доступ до управління ресурсами;

– моніторинг метрик. Відображення даних про продуктивність кожного сервера, включаючи CPU, пам'ять, дисковий простір та мережну активність. Це дозволяє оперативно виявляти та вирішувати проблеми з продуктивністю;

– керування бекапами. Інтерфейс для налаштування та керування резервним копіюванням даних, що критично важливо для відновлення після збоїв;

– керування алертами. Конфігурація та керування повідомленнями про події та зміни у стані системи, що дозволяє швидко реагувати на потенційні проблеми.

Дизайн адміністративної панелі орієнтований на максимальну зручність та ефективність використання. Панель пропонує інтуїтивно зрозумілий, чуйний інтерфейс, котрий здатен легко адаптовуватися до різноманітних пристроїв та розмірів дисплеїв. Серйозним аспектом дизайну є забезпечення швидкого доступу до найбільш часто використовуваних функцій, а також можливість налаштування інтерфейсу користувача під індивідуальні уподобання адміністраторів (рис. 3.2).

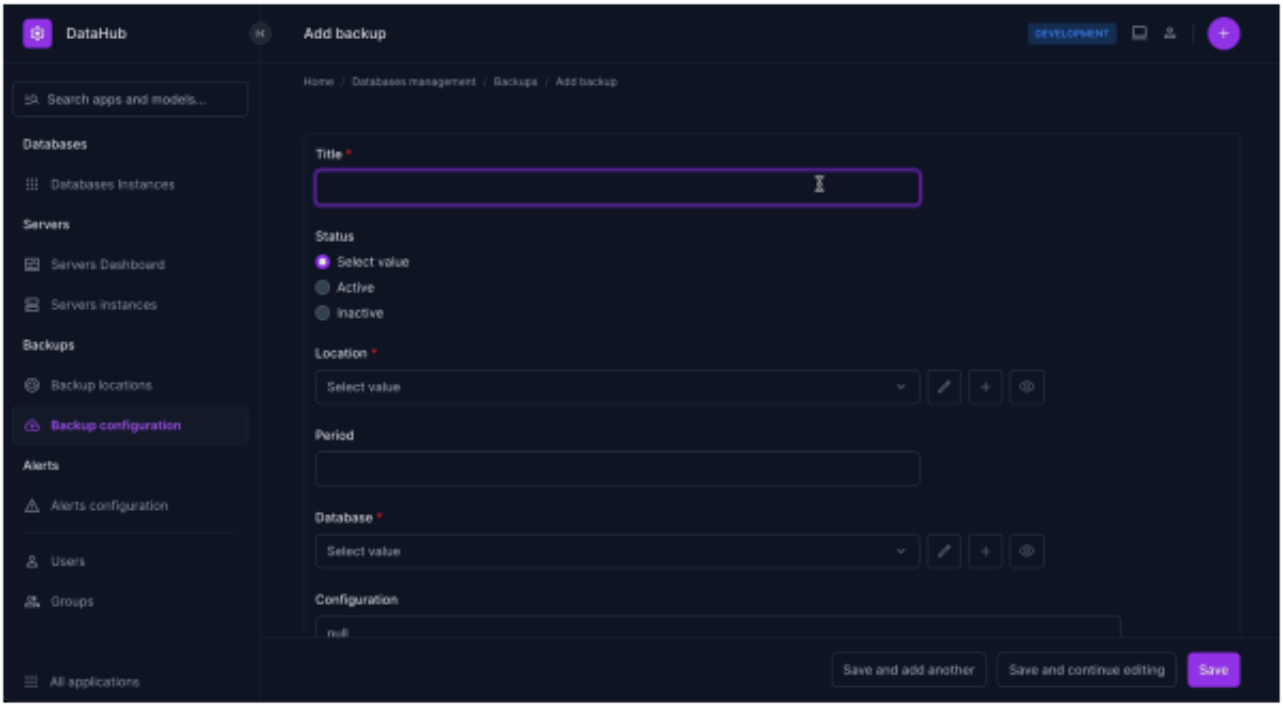

Рисунок 3.2 – Адміністративна панель

Безпека адміністративної панелі є пріоритетом з огляду на її центральну

роль в управлінні критично важливими системами. Усі комунікації між панеллю та серверами захищені із застосуванням новітніх методів шифрування, зокрема і TLS/SSL. Також передбачені заходи захисту від XSS та CSRF атак, що забезпечує захист даних адміністраторів та системи загалом.

Адміністративна панель тісно інтегрована з демоном управління та іншими компонентами системи, забезпечуючи безперебійний обмін даними та командами. Це включає прийом метрик і алертів, і передачу команд від адміністраторів, що направляються серверам через демони. Використовувані механізми API забезпечують надійну та безпечну взаємодію компонентів, підтримуючи цілісність та узгодженість даних на всіх рівнях системи.

На рис. 3.3 представлена архітектура адміністративної панелі.

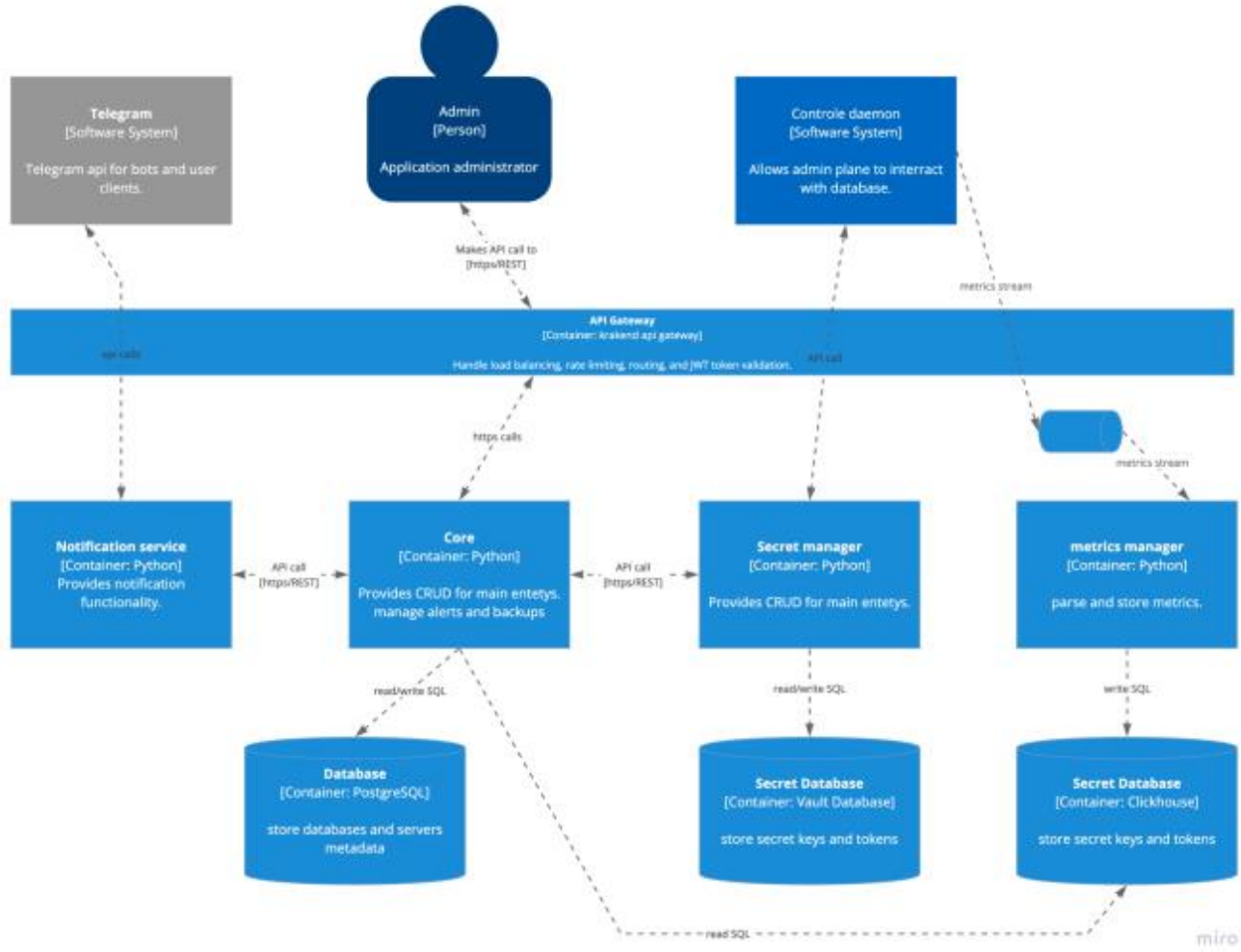

Рисунок 3.3 – Архітектура адміністративної панелі

Така архітектура забезпечує не тільки ефективне управління ресурсами та швидке реагування на події в системі, але й високий рівень безпеки та зручність використання, що робить її незамінним інструментом для сучасних адміністраторів

3.4 Розробка DSL

DSL для управління конфігураціями є спеціалізованим інструментом, що дозволяє адміністраторам ефективно керувати та автоматизувати різні аспекти IT-інфраструктури. Метою створення такої мови є спрощення процесів, зниження можливості помилок та надання потужного, але зрозумілого середовища для виконання адміністративних завдань.

Основна мета розробки DSL – надати адміністраторам інтуїтивно зрозумілий інструмент, здатний адаптуватися під специфічні потреби управління різноманітними системами.

DSL призначена для спрощення складних команд, забезпечуючи модульність та гнучкість, що дозволяє легко інтегрувати нові функції та підтримувати різні типи баз даних. Для підтримки гнучкості розроблено структуру класів, що дозволяє легко інтегрувати нові команди та бази даних.

DSL надає адміністраторам набір наперед визначених команд, котрі володіють широким спектром задач керування, від моніторингу до управління безпекою. Ось деякі зі стандартних команд, включених до DSL:

– configureDatabaseConnection(databaseType, connectionstring) – налаштування підключень до різних типів баз даних;

– scheduleBackup(databaseName, schedule, backupType) – планування регулярних бекапів даних;

– monitorPerformance(metricType, threshold, action) – моніторинг продуктивності з автоматичними діями при досягненні певних порогів.

Технічна реалізація DSL включає використання сучасних інструментів

розробки мов програмування та парсерів, таких як ANTLR, який дозволяє визначати граматику мови та генерувати відповідні парсери. Кожна команда DSL перетворюється на специфічні API виклики, що взаємодіють із системними компонентами. Це забезпечує як високу продуктивність виконання завдань, і легкість інтеграції з існуючими системами управління.

Крім того, DSL підтримує розширюваність через модулі, написані на Python, що дозволяє адміністраторам створювати власні команди та функції, адаптовані до їхнього конкретного середовища. Це робить систему гнучкою та здатною підлаштовуватися під потреби користувачів.

Діаграма класів наведена на рис. 3.4.

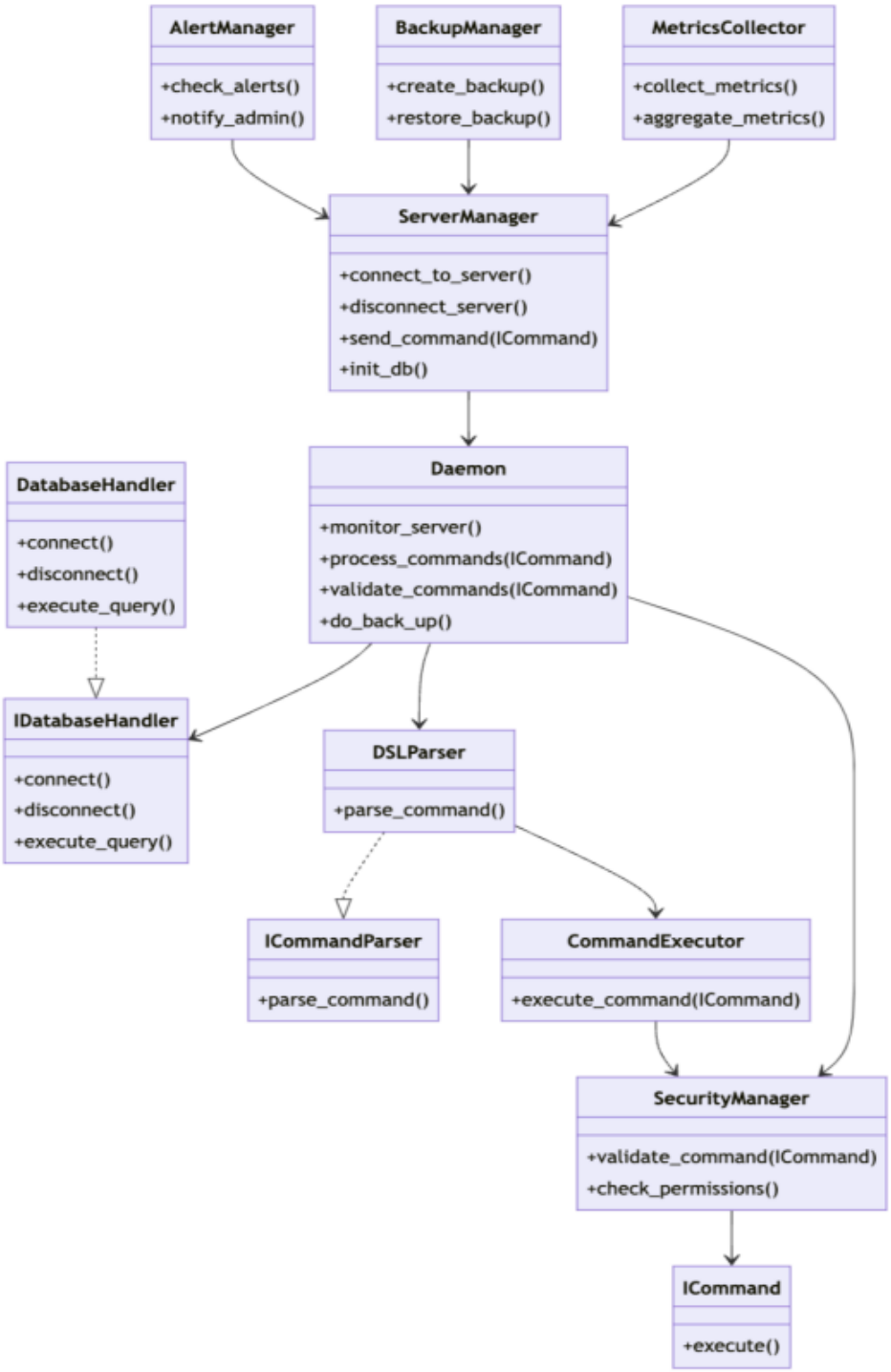

Рисунок 3.4 – Діаграма класів системи

3.5 Обговорення результатів

СУБД, розроблена з використанням DSL, є комплексним рішенням, що

включає демон управління, адміністративну панель і DSL для налаштування та управління. Кожен із цих компонентів був розроблений з урахуванням необхідності забезпечення високої функціональності та гнучкості в управлінні.

Демон управління автоматизує налаштування баз даних та виконує команди, що надходять з адміністративної панелі. Це дозволяє оперативно реагувати на зміни та автоматизувати рутинні процеси. Демон займає мінімальне місце і компілюється в бінарний формат, що дозволяє уникнути багатьох залежностей.

Адміністративна панель надає користувачам графічний інтерфейс для керування всією інфраструктурою, спрощуючи процес конфігурації та моніторингу. Це робить панель доступною навіть для адміністраторів, які не мають глибоких технічних знань в області баз даних.

DSL був розроблений для спрощення управління та конфігурації. Його модульна структура і можливість розширення за допомогою модулів на Python дозволяють системі легко адаптуватися під будь-які вимоги і інтегруватися з новими технологіями.

Для оцінки ефективності системи було проведено експеримент. На віртуальній машині було потрібно:

– здійснити інсталяцію PostgreSQL 14.2;

– підготувати дві ролі: службова та спостерігач;

– налаштувати автоматичне створення резервних копій та надсилання їх до хмарного сховища;

– налаштувати автоматичне повідомлення про використання доступної пам'яті на 80%.

Для виконання всіх дій учасникам експерименту можна було користуватися будь-якими утилітами. Крім того, учасники отримали документацію з використання розробленого DSL. На рис. 3.5 наведено діаграму отриманих результатів.

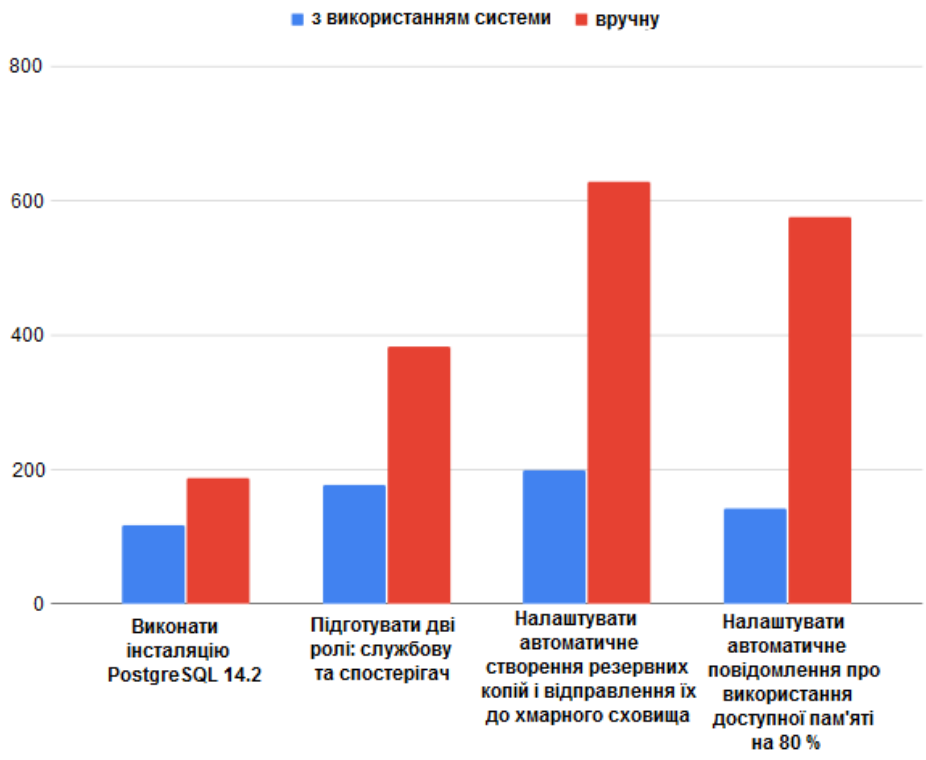

Рисунок 3.5 – Результати експерименту

В результаті експерименту виявлено значне прискорення у виконанні завдань адміністрування, що дозволяє зробити висновок про ефективність та перспективність розробленої системи.

У процесі розробки та впровадження системи було виявлено різні технічні та операційні складності. Ключовими викликами стали інтеграція DSL з існуючими системами та забезпечення безпеки даних.

Модульна структура та гнучкість системи відкривають значні можливості для подальших покращень та адаптації під нові технологічні тренди. У перспективі можлива інтеграція машинного навчання для предиктивного аналізу та автоматизації складних адміністративних завдань, що може покращити ефективність і продуктивність СУБД.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Навчання працюючих і інструктажі з охорони праці

Однією із складових ефективної роботи з профілактики виробничого травматизму є належна підготовка, навчання та підвищення кваліфікації працівників з питань охорони праці. Загальний порядок проведення навчання з питань охорони праці встановлений Законом України «Про охорону праці» (ст. 18. «Навчання з питань охорони праці»).

Виконання вимог Закону України «Про охорону праці» в частині проведення навчання та перевірки знань з питань охорони праці здійснюється відповідно до Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці, затвердженого наказом Держкомітету України з нагляду за охороною праці 26 січня 2005 р. № 15 (далі — Типове положення).

Нагляд за дотриманням вимог Типового положення здійснюють органи державного нагляду за охороною праці, а координацію та методичний супровід — Головний навчально-методичний центр та навчальні підрозділи експертно-технічних центрів Держгірпромнагляду.

Вивчення предмета «Охорона праці» при підготовці, перепідготовці та підвищенні кваліфікації працівників, які залучаються до виконання робіт з підвищеною небезпекою, на підприємстві регламентується п. 2.3 Типового положення. На підприємствах згідно з п. 1.1 Додатку 3 Типового положення навчання та перевірку знань з питань охорони праці повинні проходити керівники, заступники керівників, головні спеціалісти, керівники основних виробничих та технічних служб, безпосередньо пов’язані з організацією безпечного ведення робіт. Крім цього, згідно з п. 5 Додатку 3, навчання та перевірку знань з питань охорони праці мають проходити керівники, спеціалісти служб охорони праці, члени комісій з перевірки знань з питань охорони праці, особи, відповідальні за технічний стан і безпечну експлуатацію об’єктів підвищеної небезпеки підприємств.

Типове положення встановлює порядок та місце проведення навчання та

перевірки знань з питань охорони праці посадових осіб (п. 5.2 та п. 5.3). Посадові особи, перелік яких наведено в п. 5.2, проходять навчання у Головному навчально-методичному центрі Держнаглядохоронпраці. Перевірка знань цієї категорії посадових осіб проводиться комісією, створеною наказом Держгірпромнагляду.

Організацію навчання та перевірки знань з питань охорони праці працівників на підприємстві здійснюють працівники служби кадрів або інші спеціалісти, яким роботодавець доручив організацію цієї роботи. Навчання та перевірка знань з питань охорони праці працівників (виконавців і посадових осіб), які не залучаються до виконання робіт підвищеної небезпеки, проводиться не рідше ніж один раз на три роки. Посадові особи та працівники, які виконують роботи підвищеної небезпеки, проходять спеціальне навчання та перевірку знань відповідних нормативно-правових актів з охорони праці не рідше одного разу на рік.

Посадові особи малих підприємств (п. 5.4), які не мають можливості створити власні комісії з перевірки знань з питань охорони праці та провести навчання з питань охорони праці, проходять навчання та перевірку знань в навчальних закладах, які мають відповідний дозвіл.

Спеціальне навчання з питань охорони праці може проводитись безпосередньо на підприємстві або навчальним закладом, який має відповідний дозвіл. При проведенні такого навчання на підприємстві навчальні плани та програми розробляються з урахуванням конкретних видів робіт, виробничих умов і функціональних обов’язків працівників і затверджуються наказом керівника підприємства.

Періодичність інструктажів, навчання та перевірки знань з питань охорони праці залежить від видів виконуваних робіт та встановлюється Типовим положенням. Перевірка знань з питань охорони праці після проведення спеціального навчання проводиться комісією підприємства.

Якщо на підприємстві неможливо створити комісію з перевірки знань з питань охорони праці (п. 4.4 Типового положення), перевірка знань проводиться комісією спорідненого підприємства або Теруправління Держгірпромнагляду.

Всі працівники та посадові особи підприємства, включаючи посадових осіб,

відповідальних за виконання робіт підвищеної небезпеки (крім зазначених в п. 5.2 та п. 5.3 Типового положення), проходять навчання та перевірку знань з питань охорони праці на підприємстві. Типове положення не зобов'язує, але й не забороняє проводити навчання всіх виконавців та посадових осіб (особливо тих, що виконують роботи підвищеної небезпеки) в навчальних закладах. У нашій країні є багато підприємств, де таке навчання проводиться, і це має позитивні наслідки. Ті витрати, які при цьому несуть підприємства, перекриваються створенням більш безпечних умов праці і в результаті збереженням життя та здоров'я працівників.

Також в навчальних закладах проходять навчання та перевірку знань із загальних питань охорони праці всі посадові особи та фахівці, які проводять інструктажі або навчання підлеглих працівників з питань охорони праці, виконують роботи з проектування об'єктів, а також інші працівники, незалежно від того, передбачено таке навчання Типовим положенням чи ні.

4.2 Санітарно-гігієнічні вимоги до умов праці

На робочих місцях працівників, які відповідальні за експлуатацію сервісу управління механізмом авторських прав на мультимедійні файли, необхідно забезпечити дотримання вимог, затверджених Наказом Мінсоцполітики від 14.02.2018 за № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями».

Приміщення для роботи з ЕОМ мають бути обладнані системами опалення, кондиціонування повітря, або припливно-витяжною вентиляцією. У приміщеннях на робочих місцях мають забезпечуватись оптимальні значення параметрів мікроклімату: температури, відносної вологості та рухливості повітря відповідно до норм та правил, а також ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціонування», затверджених наказом Мінрегіону від 25.01.2013 р. № 24.

Відповідно до санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99 в офісних приміщеннях, обладнаних ЕОМ, температура повітря повинна становити 22-25°С, відносна вологість повітря – 40-60 %, швидкість руху повітря – не більше 0,1 м/с [21].

Приміщення, призначені для роботи з ЕОМ, повинні мати природне освітлення. У виробничих приміщеннях, обладнаних ЕОМ, необхідно створити належне освітлення. При експлуатації сервісу управління механізмом авторських прав на мультимедійні файли, важливим, з точки зору охорони праці, є забезпечення достатньої величини природного та штучного освітлення, які визначені у НПАОП 0.00-7.15-18 [22]. Природне світло повинно бути бічним, зорієнтованим, як правило, на північ чи північний схід, і забезпечувати коефіцієнт природної освітленості не нижче 1,5%. При виробничій потребі дозволяється експлуатувати ЕОМ у приміщеннях без природного освітлення за узгодженням з органами Держпромгірнагляду та органами й установами санітарно-епідеміологічної служби. Вікна приміщень повинні мати регулювальні пристрої для відчинення, а також жалюзі, штори тощо. Штучне освітлення приміщення з робочими місцями, обладнаними відеотерміналами ЕОМ загального та персонального користування, має бути всеосяжним і рівномірним. У випадку, коли переважають роботи з документами, допускається комбіноване освітлення (додатково до загального освітлення встановлюється світильники місцевого освітлення). Світильники розміщуються збоку від робочих місць (переважно ліворуч), або локально над робочим місцем (при розташуванні відеотерміналів ЕОМ за периметром приміщення). Як джерело світла при штучному освітленні застосовуються, як правило, люмінесцентні лампи. У світильниках місцевого освітлення допускається застосування ламп розжарювання. Рівень освітленості на робочому місці має становити 300-500 лк. При використанні комбінованого освітлення не допускається відблисків на поверхні екрана та збільшення освітлення екрана вище 300 лк.

Орієнтація вікон повинна бути на північ або північний схід, вікна повинні мати жалюзі, які можна регулювати, або штори; не дозволяється розміщувати

кабінети обчислювальної техніки у підвальних приміщеннях будинків; кабінети, обладнані комп'ютерною технікою, в навчальних закладах повинні розміщуватись в окремих приміщеннях з природним освітленням та організованим обміном повітря; стіни, стеля і підлога та обладнання кабінетів комп'ютерної техніки повинні мати покриття із матеріалів з матовою фактурою з коефіцієнтом відбиття: стін — 40- 50 %, стелі — 70 - 80 %, підлоги — 20-30 %, предметів обладнання — 40-60 % (робочого столу — 40-50 %, корпуса дисплею та клавіатури — 30-50 %, стелажів — 40-60 %); поверхня підлоги повинна мати антистатичне покриття та бути зручною для вологого прибирання; забороняється використовувати для оздоблення інтер'єру приміщень комп'ютерних кабінетів полімерні матеріали (дерев'яно-стружкові плити, шпалери, що придатні для миття, плівкові та рулонні синтетичні матеріали, шаровий паперовий пластик та ін.), що виділяють у повітря шкідливі хімічні речовини, які перевищують гранично допустимі концентрації; вміст шкідливих хімічних речовин в повітрі дошкільних та учбових приміщень з комп'ютерами не повинен перевищувати середньодобові концентрації [21].

Організація робочого місця фахівця із експлуатації сервісу повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним вимогам ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги».

Відстань від екрана до ока фахівців, які працюють за комп’ютером визначається згідно з вимогами ДСанПіН 3.3.2.007-98.

Рівень шуму не повинний перевищувати: на місцях, де працюють програмісти та оператори ЕОМ, 55 дБА, у лабораторіях, де складаються алгоритми та ведеться робота з документацією – 60 дБА, у машинному залі – 65 дБА, у приміщеннях, де розміщені гучні агрегати обчислювальних машин – 75 дБА.

ВИСНОВКИ

В результаті виконаної роботи було отримано такі результати:

– проаналізовано тренди та функції, котрі є найбільш затребуваними у СУБД;

– виконано оцінку ключових можливостей та недоліків існуючих програмних рішень;

– досліджено потенційні труднощі інтеграції і масштабування поточних систем;

– розроблено архітектуру із урахуванням усіх зібраних даних, що передбачає легкість впровадження нових функцій із одночасним забезпеченням високої продуктивності та надійності.

– проведено успішне тестування розробки.

Для подальшої роботи можна виділити перспективи.

Розширення підтримки інших баз даних: для забезпечення ще більшої гнучкості та застосовності можна розглянути можливість додавання підтримки інших популярних баз даних, таких як MySQL, Oracle, MongoDB та інші.

Також можна розширити функціональність системи асинхронної реплікації, включаючи можливості моніторингу, адміністрування та управління реплікацією. Наприклад, можна додати функції моніторингу продуктивності, автоматичного відновлення після збоїв, налаштування реплікації за певними правилами та багато іншого.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gartner Forecasts Worldwide Public Cloud End-User Spending to Total $723 Billion in 2025. [Електронний ресурс] – Режим доступу до ресурсу: https://www.gartner.com/en/newsroom/press-releases/2024-11-19-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-total-723-billion -dollars-in-2025 (дата звернення: 02.04.2025).

2. Global Bare Metal Cloud Market – Industry Trends and Forecast to 2031. [Електронний ресурс] – Режим доступу до ресурсу: https://www.databridgemarketresearch.com/reports/ global-bare-metal-cloud-mar ket (дата звернення: 02.04.2025).

3. Codd E. F. A relational model of data for large shared data banks //Communications of the ACM. 1970. V. 13. №. 6. P. 377-387.

4. Sadalage P. J., Fowler M. NoSQL distilled: a brief guide to the emerging world of polyglot persistence. Pearson Education, 2013.

5. Must-Know Postgresql Statistics. [Електронний ресурс] – Режим доступу до ресурсу: https://gitnux.org/postgresql-statistics/ (дата звернення: 10.04.2025).

6. Obe R. O., Hsu L. S. PostgreSQL: up and running: a practical guide to the advanced open source database. " O'Reilly Media, Inc.", 2017.

7. PERFORMANCE BENCHMARK POSTGRESQL / MONGODB // [Електронний ресурс] – Режим доступу до ресурсу: https://info.enterprisedb.com/rs/069-ALB-339/images/PostgreSQL_MongoDB_ Benchmark-WhitepaperFinal.pdf (дата звернення: 13.04.2025).

8. Kuhn D., Kyte T. Expert Oracle Database Architecture: Techniques and Solutions for High Performance and Productivity. Apress, 2022.

9. Gupta B., Mittal P., Mufti T. A review on amazon web service (aws), microsoft azure & google cloud platform (gcp) services //Proceedings of the 2nd International Conference on ICT for Digital, Smart, and Sustainable Development, ICIDSSD 2020, 27-28 February 2020, Jamia Hamdard, New Delhi, India. 2021.

10. Arora I., Gupta A. Cloud databases: a paradigm shift in databases //International Journal of Computer Science Issues (IJCSI). 2012. V. 9. №. 4. P. 77.

11. Cubukcu U. et al. Citus: Distributed postgresql for data-intensive applications //Proceedings of the 2021 International Conference on Management of Data. 2021. P. 2490-2502.

12. Saraswat M., Tripathi R. C. Cloud computing: Comparison and analysis of cloud service providers-AWs, Microsoft and Google //2020 9th international conference system modeling and advancement in research trends (SMART). IEEE, 2020. P. 281-285.

13. Arora I., Gupta A. Cloud databases: a paradigm shift in databases //International Journal of Computer Science Issues (IJCSI). 2012. V. 9. №. 4. P. 77.

14. Dropbox Saves Millions by Building a Scalable Metadata Store on Amazon DynamoDB and Amazon S3. [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/solutions/case-studies/dropbox-dynamodb-case-study/?p=pm&c=database&pd=ddb&z=5 (дата звернення: 18.04.2025).

15. Shirur R., Joshi V. Analysis of Tools for Measuring PostgreSQL Query Cost //Atomic Spectroscopy-Part C. 2023. V. 44. №. 3. P. 65-81.

16. The Importance of Automating Database Tasks. [Електронний ресурс] – Режим доступу до ресурсу: https://elnion.com/2024/05/10/the-importance-of-automating-database-tasks/ (дата звернення: 21.04.2025).

17. Postgresql. [Електронний ресурс] – Режим доступу до ресурсу: https://forge.puppet.com/modules/puppetlabs/ postgresql/readme (дата звернення: 24.04.2025).

18. Shaik B. PostgreSQL Configuration //Best Practices for Performance and Security. Apress, Berkeley, CA. 2020.

19. Frank F. et al. Puppet: Mastering Infrastructure Automation. Packt Publishing Ltd, 2017.

20. Stafford V. A. Zero trust architecture //NIST special publication. 2020. V. 800. P. 207.

21. Зеркалов Д.В. Безпека життєдіяльності та основи охорони праці. Навчальний посібник. К.: «Основа». 2016. – 267 с.

22. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. [Електронний ресурс] - Режим доступу: https://zakon.rada.gov.ua/laws/show/z0508-18 (дата звернення: 10.05.2024).

ДОДАТКИ

Додаток А. Диск з роботою