МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) _Автоматизації і комп'ютерних систем_____
Кафедра _Інформаційних технологій, штучного інтелекту і кібербезпеки_

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту(декан факультету)	Завідувач кафедри
___ Андрій Форсюк__	_____ Сергій Грибков__
(підпис) (ім'я та прізвище)	(підпис) (ім'я та прізвище)
«12» __лютого_____ 2024р.	«12» ____лютого _____ 2024р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

зі спеціальності 122 «Комп'ютерні науки»
(код та назва спеціальності)

освітньо-професійної програми Інформаційні управляючі системи та технології

На тему: ___Дослідження та обґрунтування оптимального типу бази даних для побудови OLAP репозиторію для сайту-агрегатора пошуку роботи____

Виконав: здобувач_2_ курсу, групи ІС-2-3М

___Вакуленко Сергій Сергійович _____ _____
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник __Андріюк Олена Петрівна _____ _____
(прізвище , ім'я та по батькові повністю) (підпис)

Консультанти_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

Рецензент Володимир Тонупон _____
(ім'я та прізвище) (підпис)

Я як здобувач(ка) Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач_____
(підпис)

Київ - 2024р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) __Автоматизації і комп'ютерних систем_____
Кафедра __Інформаційних технологій, штучного інтелекту і кібербезпеки___
Освітній ступінь__магістр_____
Спеціальність__122 «Комп'ютерні науки»_____
(код і назва)
Освітньо-професійна програма __Інформаційні управляючі системи та технології_____
(назва)

ЗАТВЕРДЖУЮ
Завідувач
кафедри Інформаційних технологій, штучного інтелекту і кібербезпеки
_____Грибков С.В.
" 19 " грудня 2023року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

_____Вакуленко Сергій Сергійович_____
(прізвище, ім'я, по батькові)

1. Тема роботи ___Дослідження та обґрунтування оптимального типу бази даних для побудови OLAP репозиторію для сайту-агрегатора пошуку роботи_____
керівник роботи_____Андріюк Олена Петрівна_доцент, кандидат фізико-математичних наук_____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від 19 грудня 2023року №1006-кс
2. Строк подання здобувачем роботи 22.01.2024 р
3. Вихідні дані до роботи __дані про сайт пошуку роботи, готові рішення OLAP репозиторіїв_____
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

1.)Вибір критеріїв оцінки OLAP
2.)Порівняння різник СКБД
3.)Розробка формули для визначення suitability
4.)Розробка застосунку

5. Перелік графічного матеріалу 9 таблиць, 6 ілюстрацій_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Доц. Андріюк О.П	12.11.23	29.12.23
2	Доц. Андріюк О.П	12.11.23	29.01.24
3	Доц. Андріюк О.П	12.11.23	30.01.24

7. Дата видачі завдання 19 грудня 2023 року

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
	Підготовка та збір інформації	12.11.23	виконано
	Аналіз особливостей OLAP-репозитаріїв	22.12.23	виконано
	Огляд та порівняння типів баз даних	29.12.23	виконано
	Вибір критеріїв оцінки та досліджень	02.01.24	виконано
	Проведення експериментальних досліджень	10.01.24	виконано
	Виведення формули OLAP Overall Suitability	19.01.24	виконано
	Розробка програмного продукту для визначення оптимального БД для OLAP під різні потреби	30.01.24	виконано

Здобувач _____ (підпис) Вакуленко С.С (прізвище та ініціали)

Керівник роботи _____ (підпис) Андріюк О.П (прізвище та ініціали)

АНОТАЦІЯ

Тема кваліфікаційної роботи: "Дослідження та обґрунтування оптимального типу бази даних для побудови OLAP репозиторію для сайту-агрегатора пошуку роботи"

Обсяг: 9 таблиць, 6 ілюстрацій, 8 джерел

Перелік ключових слів:
1. OLAP
2. База Даних
3. Репозиторій
4. Агрегатор Пошуку Роботи
5. Тип Бази Даних

Стислий опис роботи:
Досліджено та обґрунтовано оптимальний тип бази даних для створення OLAP репозиторію на сайті-агрегаторі пошуку роботи. Робота містить 9 таблиць, 6 ілюстрацій та використовує 8 джерел для підтримки обґрунтувань та висновків.

ANNOTATION

Title: "Exploring and Justifying the Optimal Database Type for Building an OLAP Repository for a Job Search Aggregator Website"

Scope: 9 tables, 6 illustrations, 8 sources

Keywords:
1. OLAP
2. Database
3. Repository
4. Job Search Aggregator
5. Database Type

Brief Description:
This work investigates and rationalizes the optimal database type for establishing an OLAP repository on a job search aggregator website. The research encompasses 9 tables, 6 illustrations, and draws from 8 sources to support the justifications and conclusion.

ЗМІСТ

Перелік скорочень, умовних позначень та термінів

1. БД (База даних): Це структурована колекція даних, організованих таким чином, щоб забезпечити ефективний доступ, управління та оновлення цих даних.

2. СКБД (Система керування базами даних): Програмне забезпечення, що забезпечує створення та управління базами даних. Приклади включають PostgreSQL, MySQL, Microsoft SQL Server.

3. DWH (Data Warehouse): Система для зберігання та обробки великих обсягів даних з різних джерел з метою подальшого аналізу та прийняття рішень.

4. OLAP (Online Analytical Processing): Технологія обробки даних, яка дозволяє користувачам аналізувати та отримувати доступ до інформації з різних поглядів та різними способами.

5. OLTP (Online Transaction Processing): Система обробки транзакцій в режимі реального часу, яка забезпечує обробку операцій швидко та ефективно.

6. GB (Gigabyte): Один гігабайт дорівнює 1 мільярду байтів і використовується для вимірювання обсягу даних.

7. SQL (Structured Query Language): Мова запитів, яка використовується для взаємодії з базами даних. SQL включає команди для створення, читання, оновлення та видалення даних в базі даних.

ВСТУП

Актуальність теми

У сучасному інформаційному суспільстві, де обсяги даних зростають експоненційно, ефективне управління та аналіз інформації стає стратегічно важливим завданням для підприємств та організацій. Однією з ключових складових успіху в цьому контексті є використання OLAP-репозиторіїв для аналізу великих обсягів даних та отримання цінних інсайтів.

Один із ключових інструментів для оптимізації аналітичних операцій та забезпечення ефективності є OLAP-репозитарії. Вони дозволяють виконувати швидкий та гнучкий аналіз великих обсягів даних, надаючи користувачам можливість отримати цінні інсайти та приймати обґрунтовані рішення.

З урахуванням специфіки веб-сайтів-агрегаторів роботи, де швидкість та реальний час грають ключову роль у задоволенні потреб користувачів, вибір оптимальної бази даних для побудови OLAP-репозитарію стає завданням важливим та відповідальним. Це дослідження спрямоване на розгляд та обґрунтування оптимального типу бази даних для ефективної реалізації OLAP-репозитарію на веб-сайті-агрегаторі пошуку роботи.

Актуальність теми дослідження визначається:

Зростання конкуренції на ринку праці та швидка динаміка змін у вимогах до кандидатів та вакансій, підкреслюють актуальність ефективного використання OLAP-репозиторіїв для веб-сайтів-агрегаторів пошуку роботи.

Зростання обсягів даних: З кожним роком обсяги даних, пов'язаних із вакансіями, кандидатами та динамікою ринку праці, зростають експоненційно. Висока інформаційна наповненість веб-сайтів-агрегаторів пошуку роботи

вимагає відповідних інструментів для швидкого та ефективного аналізу цих даних.

Зміни в вимогах до робочих місць та кандидатів: Динаміка ринку праці передбачає постійні зміни в обліку нових тенденцій та вимог до кандидатів. Для адаптації до цих змін та надання точної та актуальної інформації користувачам, необхідні потужні OLAP-інструменти.

Потреба в оперативному прийнятті рішень: В умовах швидкого розвитку технологій та конкуренції важливо оперативно реагувати на зміни у вимогах ринку праці. OLAP-репозитарії, оптимізовані для швидкого аналітичного опрацювання даних, надають можливість здійснювати оперативне прийняття рішень для підтримки стратегічного розвитку.

Підвищення конкурентоспроможності веб-сайтів-агрегаторів роботи: Оптимізація інструментів аналізу даних через правильно обрану базу даних дозволяє підвищити конкурентоспроможність веб-сайтів-агрегаторів роботи, пропонуючи користувачам більш точні та швидкі результати пошуку.

З урахуванням цих факторів, дослідження оптимального типу бази даних для побудови OLAP-репозитарію для веб-сайтів-агрегаторів пошуку роботи стає насущною задачею для забезпечення ефективного функціонування та конкурентоспроможності в цьому динамічному сегменті.

Зв’язок роботи з науковими програмами, планами, темами кафедри, університету

Наукова робота виконувалась згідно з науково-дослідною роботою на кафедрі інформаційних технологій, штучного інтелекту і кібербезпеки «Дослідження та використання сучасних інформаційних технологій для виконання функцій та

завдань виробничого і організаційного управління підприємств харчової галузі» (0120U105386 2020–2025 рр.) Національного університету харчових технологій.

Переддипломна робота

Переддипломна робота в компанії Jooble стала переломним моментом для досліджень у галузі обробки та аналізу даних. Протягом виробничої та переддипломної практики було активно занурено ся у розробку систем оркестрації даних та оптимізацію процесів налагодження даних в бізнес-середовищі. Цей період надав глибокі технічні знання та важливий інсайт в те, як дані стають ключовим активом для прийняття управлінських рішень та оптимізації бізнес-процесів.

Участь у команді компанії, де використання даних є стратегічним фактором, вклала в атмосферу, де ефективне управління та аналіз інформації є визначальним для успішності. Ця робота стала не лише робочим завданням, але й джерелом натхнення для подальших досліджень у галузі дата інженірінгу та обробки об'ємних даних.

Основний акцент роботи в Jooble був зосереджений на розробці та оптимізації OLAP-репозитаріїв. Цей практичний досвід став ключовим стимулом для обрання теми кваліфікаціної роботи: 'Дослідження та обґрунтування оптимального типу бази даних для побудови OLAP репозиторію для сайту-агрегатора пошуку роботи’. Мета цього дослідження полягає у вивченні впливу правильного вибору технологій та підходів до обробки даних на ефективність, витрати та якість прийняття управлінських рішень в сучасному бізнес-середовищі.

Враховуючи актуальність теми та значущість результатів для бізнесу, переддипломна робота в Jooble стала не лише закінченим етапом підготовки, але

й важливим етапом власного професійного росту, що вплине на подальше вивчення та розвиток галузі обробки та аналізу даних.

Мета дослідження

Мета роботи полягає у проведенні комплексного аналізу особливостей OLAP-репозитаріїв з метою отримання глибокого розуміння їхнього функціоналу та ефективності. Основні завдання включають в себе аналіз та відтворення дата-моделей відповідної бізнес-моделі з урахуванням розрахунку коефіцієнтів, що визначають їхню продуктивність.

Далі, в рамках дослідження, планується огляд та порівняння різних типів баз даних, використовуваних для OLAP-репозитаріїв. Це дозволить визначити переваги та недоліки кожного типу бази даних у контексті використання в OLAP-системах.

Наступним етапом є вибір критеріїв для оцінки та подальших досліджень, включаючи продуктивність, масштабованість, надійність та інші, які визначають ефективність OLAP-репозитаріїв.

Далі будуть проведені експериментальні дослідження, спрямовані на вивчення впливу обраних критеріїв та характеристик на ефективність OLAP-репозитаріїв. Це надасть об'єктивні дані для подальшого аналізу та порівняння різних аспектів досліджуваних систем.

Завершальним етапом є виведення формули OLAP Overall Suitability, що є комплексним показником, охоплюючим ключові аспекти досліджуваних OLAP-репозитаріїв. Крім того, розроблятиметься програмний продукт, який дозволить визначити оптимальну базу даних для OLAP в залежності від конкретних потреб користувача.

Наукова новизна одержаних результатів

Однією з ключових наукових новизн даного дослідження є виведення та розробка формули OLAP Overall Suitability, яка враховує різні аспекти ефективності OLAP-репозитаріїв. Ця формула є інтегрованим підходом до оцінки та порівняння різних типів баз даних у контексті їхнього використання для побудови систем аналізу великих обсягів даних.

Розроблена формула враховує три ключові аспекти, що визначають ефективність OLAP-репозитарію: *Insert Performance, Price, та Query Performance.*

- Insert Performance: Враховує ефективність вставки нових даних в систему. Цей фактор важливий для оновлення та розширення обсягів інформації.

- Price: Визначає вартість використання бази даних, включаючи витрати на обслуговування та підтримку.

- Query Performance: Підкреслює ефективність виконання запитів, з особливим акцентом на операції JOIN та GROUP BY, що є ключовими в аналітичних операціях.

Використання цієї формули дозволяє отримати комплексну оцінку та порівняти різні бази даних з точки зору їхньої придатності для побудови OLAP-репозитарію. Такий підхід відкриває нові можливості для раціонального вибору технологій та оптимізації систем обробки великих обсягів даних, сприяючи подальшому розвитку галузі дата інженірінгу та бізнес-аналітики.

Практичне значення одержаних результатів

Отримані результати цього дослідження мають велике практичне значення для сучасних веб-сайтів-агрегаторів пошуку роботи та бізнес-середовищ, що використовують OLAP-репозитарії для аналізу та управління даними. Нижче наведені ключові аспекти, які демонструють практичну вартість отриманих результатів:

1. Оптимізація Вибору Бази Даних: Власний досвід в компанії показав, що правильний вибір бази даних для OLAP-репозитарію суттєво впливає на продуктивність та вартість утримання. Розроблена формула OLAP Overall Suitability активно використовується вже в компанії, сприяючи обґрунтованому вибору нової бази даних для OLAP.

2. Вартість та Продуктивність: Нові результати впроваджені у практику, що дозволяє ефективно балансувати між витратами та продуктивністю OLAP-репозитарію, що стає ключовим елементом стратегії використання даних у компанії.

3. Швидкість запитів: Формула Query Performance враховує швидкість виконання запитів, особливо в умовах великого обсягу даних. Реальний досвід впровадження цього підходу показав покращення у швидкості та реальному часі аналізу.

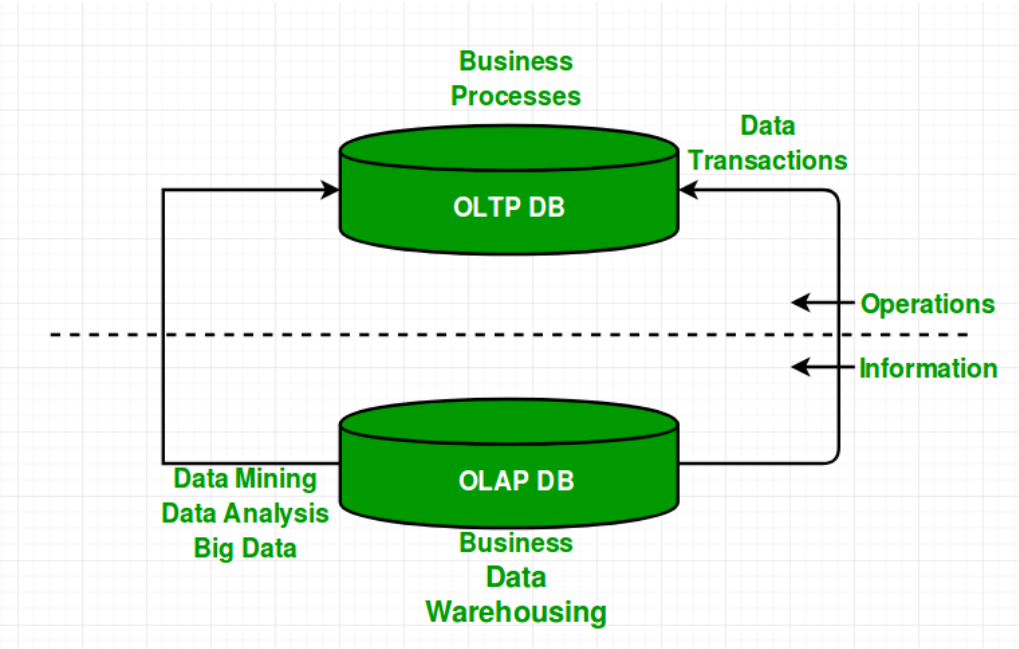
4. Корпоративне Планування та Стратегічні Рішення: Результати дослідження стали стратегічною основою для компанії в області корпоративного планування та прийняття стратегічних рішень, допомагаючи забезпечити синергію між бізнес-потребами та обраною OLAP-технологією.

5. Розвиток Галузі Дата Інженірінгу: Враховуючи активне використання нових підходів та технологій у компанії, результати дослідження допомагають не лише оптимізувати поточні проекти, але й стають підґрунтям для вдосконалення галузі дата інженірінгу в цілому.

Ці практичні аспекти дослідження вже використовуються в реальних умовах бізнесу та допомагають компанії досягати вищих показників ефективності при використанні OLAP-репозитаріїв.

OLAP та OLTP

Онлайн-аналітична обробка даних (OLAP) та о*нлайн-транзакційна*



обробка даних (OLTP) представляють собою дві ключові парадигми обробки інформації.

Рис.1.1 - Дизайн системи OLTP-OLAP

OLAP спрямована на аналіз великих обсягів даних та побудову звітів для прийняття стратегічних рішень, тоді як OLTP фокусується на ефективній обробці транзакцій у режимі реального часу. [1] У контексті сайту-агрегатора пошуку роботи, обидві ці області грають ключову роль у забезпеченні ефективного функціонування та наданні користувачам швидкого та зручного інструменту для отримання інформації щодо доступних вакансій та кандидатів. [2]

Таблиця 1.1 - Порівняльна характеристика OLAP та OLTP

Основні розрізнюючі риси	OLAP	OLTP
Орієнтація на користувачів та систему	Клієнтів і використовується для вирішення потреб у обробці транзакцій та запитів для клієнтів та IT-фахівців	Ринок і спрямована на аналіз даних для користувачів знань, включаючи менеджерів, виконавчих директорів та аналітиків
Зміст даних	Поточні дані в повному, не обробленому та неагрегованому стані	Історичні дані, суми, агрегації
Проектування бази даних	Модель "сутність-зв'язок", орієнтований на застосунок	Модель "зірка" чи "сніжинка", об'єктно-орієнтована
View (Фокус)	Фокусується на поточних даних без посилання на історичні або міжорганізаційні дані	Охоплює кілька версій схеми бази даних, адаптуючись до еволюційного процесу організації
Access Patterns (Шаблони доступу)	Короткочасні, атомарні транзакції, що вимагають механізмів контролю конкурентності та відновлення	Читання та виконання складних запитів

Потреба у сховищі даних (DWH) та OLAP

Розвиток сховища даних та OLAP, незважаючи на наявність операційних баз даних, має свої вагомі причини: [3]

1. Операційні бази даних спроектовані та налаштовані для відомих завдань та навантажень, таких як індексація за допомогою первинних ключів, пошук конкретних записів та оптимізація «заздалегідь визначених» запитів.

Оскільки запити до сховища даних часто є складними, вони передбачають обчислення великої кількості даних на узагальнених рівнях і можуть вимагати використання спеціальних методів організації, доступу та реалізації даних на основі багатовимірних поглядів. Обробка OLAP-запитів в операційних базах даних суттєво погіршила б продуктивність операційних завдань (транзакцій в межах продукту).

2. Операційна база даних підтримує одночасну обробку множини транзакцій. Механізми контролю конкурентності та відновлення, такі як блокування та журналювання, необхідні для забезпечення послідовності та надійності транзакцій. У той час як запити OLAP часто вимагають лише читання записів даних для сумаризації та агрегації. Застосування механізмів контролю конкурентності та відновлення для таких операцій OLAP може піддавати ризику виконання одночасних транзакцій.

3. Прийняття рішень передбачає наявність історичних даних, а операційні бази даних зазвичай не зберігають історичні дані. Таким чином, дані в операційних базах даних, хоча і обширні, завжди далекі від повноти для прийняття рішень.

4. Прийняття рішень потребує консолідації (наприклад, агрегації та сумаризації) даних з різних джерел. Операційні бази даних часто містять лише деталізовані сирі дані, що робить їх непридатними для прийняття рішень.

Розуміння цих аспектів підкреслює необхідність створення надійної системи OLAP та ефективного сховища даних для відповіді на складні вимоги агрегатора пошуку роботи. У наступних розділах будуть розглянуті різні аспекти вибору оптимального типу бази даних для створення OLAP-репозиторію

Online Analytical Processing (OLAP) можна визначити як набір інструментів і підходів до представлення даних з різних вимірів. У більш широкому розумінні

це включає в себе низку практик, спрямованих на моделювання даних/баз даних і створення конкретних аналітичних рішень.

Основні компоненти типової OLAP-системи включають наступне:

1.Data source: Це може бути транзакційна база даних або будь-яке інше сховище, з якого ми беремо дані. Дані в їхньому стандартному форматі не оптимізовані для OLAP-запитів, тому вони потребують трансформації та перетворення перед використанням.

2. OLAP-база даних: Тут ми зберігаємо дані для аналізу. Зазвичай трансформація відбувається перед завантаженням даних до бази даних, але підхід може варіюватися.

3.OLAP-куб: Базовий інструмент для представлення багатовимірних даних для аналізу. Оскільки йдеться про онлайн-аналітичну обробку, куби розгортаються на спеціалізованому сервері. OLAP-куб дозволяє аналітиці групувати або розрізати елементи за різними категоріями, призначений для виконання складних запитів, які не можуть бути оброблені зазвичай базами даних OLTP.

4. Аналітичний інтерфейс: Взаємодія з кубами та іншими аналітичними інструментами для візуалізації даних і створення звітів відбувається через спеціалізований інтерфейс. Більшість інтерфейсів представлені бізнес-інтелект-панелями. Куби можна отримати через ці панелі, що надає користувачу більше контролю.

В цій роботі, основний акцент на другому пункті, оскільки за мету, є дослідження та обгрунтування оптимального типу БД в OLAP репопизиторію для такої бізнес моделі як «сайт пошуку роботи».

Операції з даними в OLAP

OLAP агрегує транзакційні дані з сховища для перетворення їх у придатну форму для аналізу. Як джерело даних, OLAP може використовувати об'єднане сховище, таке як база даних даних, озеро даних або магазин даних, або просто будь-яке місце, де ви зберігаєте історичні дані.

Але для виконання складних користувацьких запитань, ми повинні правильно структурувати дані. Тому в більшості випадків потрібна окрема OLAP-база даних або сховище, яка моделюватиме дані для багатовимірного аналізу.

Запит в OLAP може виглядати приблизно так:
- "показати кількість вакансій для розробників програмного забезпечення в Києві за останні 3 місяці",
- "порівняти кількість вакансій для менеджерів проектів в регіоні Північна Америка за останній рік",
- "групувати вакансії для інженерів-електриків за рівнем досвіду в області Східній США",
- "показати середню зарплату для вакансій в області технічної підтримки за поточний місяць."

Такі аналітичні запити вимагають від бази даних збирати інформацію з кількох таблиць, які категоризують дані за «виміром» (dimensions). Прикладами вимірів можуть бути час, продукт, місцезнаходження, клієнт і т. д.

OLAP моделює базу даних так, що стає можливим швидко збирати дані і представляти їх для аналітиків у багатовимірному режимі, а не у плоскій таблиці. Тому OLTP та OLAP бази даних будуть відрізнятися в численних аспектах.

Дата моделювання в OLAP

Для ефективного впровадження OLAP-аналізу в контексті сайту-агрегатора пошуку роботи важливо ретельно розглянути та вибрати оптимальний підхід до моделювання даних. В зазначеному контексті використання різних схем дата-моделювання стає ключовим етапом для забезпечення ефективності та швидкості аналізу великих обсягів інформації. У цьому розділі дослідження розглядається ряд ключових підходів до дата-моделювання в OLAP, зокрема Star Schema, Snowflake Schema, Galaxy Schema та Data Vault, з метою визначення найбільш відповідного для потреб сайту-агрегатора пошуку роботи.

Star Schema (Зіркова схема)

"Star Schema" визначається як методологія моделювання даних, де центральна таблиця (факт-таблиця) сполучена з різними вимірами через ключі. Основна ідея полягає в тому, що факт-таблиця розташована в центрі, подібно до центральної зірки, і з неї виходять виміри як випромінювання відправницьких променів.

Використання схеми «зірка» спрощує аналіз даних і забезпечує швидкий доступ до великої кількості інформації. Кожен вимір, пов'язаний із факт-таблицею, стає осібною таблицею, що дозволяє зберігати дані у вигляді таблиць, що містять інформацію про конкретний аспект діяльності. Це полегшує аналіз та забезпечує ефективну роботу з великим обсягом даних у багатовимірному середовищі. [4]

SQL-скрипт для створення схеми «зірка» для веб-сайту-агрегатора пошуку роботи:

```
-- Створення таблиці "JobSeekers" (вимір "Кандидат")
CREATE TABLE JobSeekers (
    CandidateID SERIAL PRIMARY KEY,
    FirstName VARCHAR(255),
    LastName VARCHAR(255),
```

```
    Email VARCHAR(255),
    -- Додаткові поля виміру можуть бути додані за необхідності
);

-- Створення таблиці "Jobs" (вимір "Вакансія")
CREATE TABLE Jobs (
    JobID SERIAL PRIMARY KEY,
    JobTitle VARCHAR(255),
    CompanyName VARCHAR(255),
    Location VARCHAR(255),
    -- Додаткові поля виміру можуть бути додані за необхідності
);

-- Створення таблиці "Applications" (факт-таблиця "Заявки на роботу")
CREATE TABLE Applications (
    ApplicationID SERIAL PRIMARY KEY,
    CandidateID INT,
    JobID INT,
    ApplicationDate DATE,
    Status VARCHAR(50),
    -- Додаткові фактичні поля можуть бути додані за необхідності
    FOREIGN KEY (CandidateID) REFERENCES JobSeekers(CandidateID),
    FOREIGN KEY (JobID) REFERENCES Jobs(JobID)
);
```

У цьому SQL-скрипті ми створюємо три таблиці: **JobSeekers** (вимір "Кандидат"), **Jobs** (вимір "Вакансія") та **Applications** (факт-таблиця "Заявки на роботу"). Факт-таблиця містить ключі, які посилаються на відповідні вимірні таблиці, і вона може використовуватися для аналізу стосунків між кандидатами та вакансіями на веб-сайті пошуку роботи.

Snowflake Schema (Схема «Сніжинка»)

Методологія моделювання даних, є розширенням концепції "Star Schema". У цій схемі виміри розгалужуються на додаткові рівні, що створює подібність до гілок сніжинки. Це відбувається через нормалізацію вимірів, що може покращити ефективність у випадках, коли деякі виміри мають багато атрибутів та потребують окремих таблиць.

Основна ідея "Snowflake Schema" полягає в розгалуженні вимірів на декілька рівнів глибини. Наприклад, якщо в "Star Schema" ми маємо таблицю виміру "Продукт" з атрибутами, такими як "Назва продукту", "Категорія" і т.д., то в "Snowflake Schema" ця таблиця може розгалужуватися на окремі таблиці, такі як "Назва продукту", "Деталі категорії" і так далі.

Використання "Snowflake Schema" може бути корисним у випадках, коли потрібно зменшити повторення даних і підтримувати високий рівень нормалізації. Однак це може призвести до збільшення кількості з'єднань між таблицями, що впливає на продуктивність запитів.[4]

Для створення схеми «Сніжинка» здійснимо нормалізацію вимірів у таблиці "Jobs". У цьому випадку ми розгалужимо атрибути "Company Name" та "Location" на окремі таблиці, що представляють додаткові виміри.

```
-- Створення таблиці "JobSeekers" (вимір "Кандидат")
CREATE TABLE JobSeekers (
                CandidateID SERIAL PRIMARY KEY,
                FirstName VARCHAR(255),
                LastName VARCHAR(255),
                Email VARCHAR(255),
    -- Додаткові поля виміру можуть бути додані за необхідності
```

```
);

-- Створення таблиці "Companies" (вимір "Компанія")
CREATE TABLE Companies (
                CompanyID SERIAL PRIMARY KEY,
                CompanyName VARCHAR(255),
    -- Додаткові поля виміру можуть бути додані за необхідності
);

-- Створення таблиці "Locations" (вимір "Локація")
CREATE TABLE Locations (
                LocationID SERIAL PRIMARY KEY,
                LocationName VARCHAR(255),
    -- Додаткові поля виміру можуть бути додані за необхідності
);

-- Створення таблиці "Jobs" (вимір "Вакансія")
CREATE TABLE Jobs (
                JobID SERIAL PRIMARY KEY,
                JobTitle VARCHAR(255),
                CompanyID INT,
                LocationID INT,
    -- Додаткові поля виміру можуть бути додані за необхідності
                FOREIGN KEY (CompanyID) REFERENCES
Companies(CompanyID),
                FOREIGN KEY (LocationID) REFERENCES Locations(LocationID)
);

-- Створення таблиці "Applications" (факт-таблиця "Заявки на роботу")
CREATE TABLE Applications (
```

```
    ApplicationID SERIAL PRIMARY KEY,
    CandidateID INT,
    JobID INT,
    ApplicationDate DATE,
    Status VARCHAR(50),
-- Додаткові фактичні поля можуть бути додані за необхідності
    FOREIGN KEY (CandidateID) REFERENCES JobSeekers(CandidateID),
    FOREIGN KEY (JobID) REFERENCES Jobs(JobID)
);
```

У цьому прикладі атрибути "Company Name" та "Location" були розгалужені на окремі таблиці "Companies" та "Locations". Така нормалізація може зменшити повторення даних, але може також вимагати більше з'єднань при виконанні запитів. Обирайте між "Star Schema" і "Snowflake Schema" в залежності від ваших конкретних вимог і відношень між даними.

Galaxy Schema

Представляє собою розширення концепції "Star Schema" і "Snowflake Schema" у моделюванні даних для побудови дата-вейрхаузів. У цій схемі множина факт-таблиць спільно використовує кілька таблиць вимірів. Однак, на відміну від схеми "Сніжинка", де виміри розгалужуються на додаткові рівні, у галактичній схемі факт-таблиці не обов'язково пов'язані безпосередньо, надаючи більше вільності у розподілі даних.

Основна ідея "Galaxy Schema" полягає в можливості факт-таблиць об'єднуватися через різні виміри. Кожна факт-таблиця може бути пов'язана з власним набором вимірів, і ці набори вимірів можуть перетинатися з іншими факт-таблицями.

Такий підхід дозволяє моделювати складні зв'язки між різними аспектами бізнес-процесів та різними метриками.[5]

Використання галактичної схеми може бути важливим у випадках, коли бізнес-дані мають складні та взаємопов'язані залежності. Однак це також може вимагати більшої уваги до управління взаємозв'язками між факт-таблицями та забезпеченням коректності даних в такій конфігурації.

Для розширення схеми "Snowflake" до "Galaxy" додамо додатковий вимір "Skills" та зв'язок між вакансіями та цим новим виміром. Також додамо додаткові атрибути до існуючих вимірів та факт-таблиці.

```
-- Створення таблиці "Skills" (новий вимір)
CREATE TABLE Skills (
                SkillID SERIAL PRIMARY KEY,
                SkillName VARCHAR(255),
   -- Додаткові поля виміру можуть бути додані за необхідності
);

-- Оновлення таблиці "JobSeekers" (додано поле SkillsID)
ALTER TABLE JobSeekers
   ADD COLUMN SkillsID INT,
ADD FOREIGN KEY (SkillsID) REFERENCES Skills(SkillID);

-- Оновлення таблиці "Jobs" (додано поле SkillsID)
ALTER TABLE Jobs
   ADD COLUMN SkillsID INT,
ADD FOREIGN KEY (SkillsID) REFERENCES Skills(SkillID);

-- Створення таблиці "Applications" (факт-таблиця "Заявки на роботу")
```

```
CREATE TABLE Applications (
                ApplicationID SERIAL PRIMARY KEY,
                CandidateID INT,
                JobID INT,
                SkillsID INT, -- Додано поле SkillsID
                ApplicationDate DATE,
                Status VARCHAR(50),
    -- Додаткові фактичні поля можуть бути додані за необхідності
                FOREIGN KEY (CandidateID) REFERENCES
JobSeekers(CandidateID),
                FOREIGN KEY (JobID) REFERENCES Jobs(JobID),
                FOREIGN KEY (SkillsID) REFERENCES Skills(SkillID) --
Додано зв'язок з Skills
);
```

Data Vault

Цей підхід спрямований на побудову адаптивної системи зберігання даних, яка легко масштабується та адаптується до змін у даних та бізнес-правилах. Забезпечує гнучкість та швидкість реакції на зміни.

На відміну від традиційних методологій, Data Vault не тільки визначає спосіб моделювання даних, але й пропонує конкретну структуру для зберігання та обробки інформації. Основні компоненти Data Vault Schema включають Hub (Центральний Зберіг), Link (Зв'язок) і Satellite (Супутник), які утворюють основу для побудови гнучких та масштабованих хранилищ даних.[6]

1. Hub: Це ядро Data Vault Schema, де зберігаються унікальні ключі для ідентифікації даних. Кожен Hub представляє сутність або об'єкт, який має свій унікальний ідентифікатор.

2. Link (Зв'язок): Визначає взаємозв'язки між сутностями, використовуючи унікальні ключі з Hub. Link дозволяє встановлювати взаємозв'язки та логічні з'єднання між об'єктами.

3. Satellite (Супутник): Цей компонент дозволяє зберігати контекстні дані та повну історію змін для кожної сутності чи взаємозв'язку. Satellite забезпечує гнучкість та повноту даних, включаючи їхню історію.

Data Vault Schema використовується для створення централізованого історизованого сховища даних, де інформація зберігається в формі, яка підтримує повну історію змін. Ця схема робить архітектуру даних більш гнучкою та пристосованою до швидкої зміни вимог бізнесу. Data Vault Schema рекомендується для компаній, які мають потребу в ефективному управлінні та аналізі великих обсягів даних, зокрема в областях, де історична інформація грає ключову роль у прийнятті стратегічних рішень.

```
-- Створення центрального Hub "Skills" для Data Vault Schema
CREATE TABLE Hub_Skills (
                    SkillsID SERIAL PRIMARY KEY,
                    SkillName VARCHAR(255) NOT NULL,
);

-- Створення супутника "Skills" для збереження контекстних даних та історії
змін
CREATE TABLE Satellite_Skills (
                        SatelliteID SERIAL PRIMARY KEY,
                        SkillsID INT NOT NULL,
                        LoadDate DATE NOT NULL,
                        SkillName VARCHAR(255) NOT NULL,
                        IsActive BOOLEAN NOT NULL,
```

```
                    FOREIGN KEY (SkillsID) REFERENCES
Hub_Skills(SkillsID)
);

-- Оновлення Hub "JobSeekers" (додано поле SkillsID)
ALTER TABLE Hub_JobSeekers
    ADD COLUMN SkillsID INT,
    ADD FOREIGN KEY (SkillsID) REFERENCES Hub_Skills(SkillsID);

-- Оновлення Hub "Jobs" (додано поле SkillsID)
ALTER TABLE Hub_Jobs
    ADD COLUMN SkillsID INT,
    ADD FOREIGN KEY (SkillsID) REFERENCES Hub_Skills(SkillsID);

-- Створення факт-таблиці "Applications" для відстеження зв'язків між
кандидатами, вакансіями та навичками
CREATE TABLE Link_Applications (
                    ApplicationID SERIAL PRIMARY KEY,
                    CandidateID INT NOT NULL,
                    JobID INT NOT NULL,
                    SkillsID INT NOT NULL,
                    ApplicationDate DATE NOT NULL,
                    Status VARCHAR(50) NOT NULL,
                    FOREIGN KEY (CandidateID) REFERENCES
Hub_JobSeekers(CandidateID),
                    FOREIGN KEY (JobID) REFERENCES Hub_Jobs(JobID),
                    FOREIGN KEY (SkillsID) REFERENCES
Hub_Skills(SkillsID)
);
```

Вибір конкретної моделі дата-моделювання в OLAP для сайту-агрегатора пошуку роботи залежить від унікальних вимог проекту, обсягу та специфіки оброблюваних даних. Не завжди є необхідність обирати лише одну схему, але важливо ураховувати схеми, які краще відповідають характеру даних та потребам аналізу.

Розглядання різних моделей, таких як Star Schema, Snowflake Schema, Galaxy Schema, чи навіть Data Vault, може виявитися корисним при врахуванні конкретних вимог і виборі оптимального підходу до дата-моделювання. Враховуючи це при виборі бази даних та застосуванні відповідних схем у OLAP,

можна забезпечити ефективний аналіз та обробку даних для забезпечення успішної роботи сайту та задоволення потреб користувачів у пошуку роботи.

Аналіз дата моделювання в OLAP

Аналіз моделей визначає ключові параметри, які впливають на дослідження та обґрунтування оптимального типу бази даних для побудови OLAP репозиторію для сайту-агрегатора пошуку роботи. Важливо провести дослідження та тестування за такими пунктами:

1. Час імплементації:

- *Складність моделі:* Більш складні схеми, наприклад, Snowflake або Data Vault, можуть вимагати більше часу для розробки та імплементації порівняно з простішими схемами, такими як Star Schema.

Star Schema (Базовий час - 4 одиниці):
Обгрунтування: Star Schema зазвичай включає основну факт-таблицю та декілька вимірів. Це вимагає створення приблизно 4 таблиць.

Snowflake Schema (Коефіцієнт 1.5):
Обгрунтування: У Snowflake Schema зазвичай кожна таблиця розгалужується на декілька додаткових таблиць. Таким чином, для того, щоб врахувати основні дані, може знадобитися більше часу, наприклад, 6 таблиць.

Data Vault (Коефіцієнт 2.0):
Обгрунтування: Data Vault має більш високий рівень абстракції, що призводить до створення додаткових таблиць для зберігання метаданих та історічних даних. Таким чином, може знадобитися більше часу, наприклад, 8 таблиць.

Таблиця 2.1 - Таблиця коефіцієнтів складності дата моделі

Складність моделі	PostreSQL	ClickHouse
Star Schema	1.0	1.0
Snowflake Schema	1.5	1.5
Data Vault	2.0	2.0

- *Інструменти та технології:* Використання спеціалізованих інструментів та технологій може сприяти швидкішій імплементації. Наприклад, використання інструментів, призначених для оптимізації OLAP.

Star Schema (Базовий час - 4 одиниці):
Обгрунтування: Star Schema користується стандартними інструментами для баз даних та OLAP, забезпечуючи ефективність розробки на базі відомих технологій.

Snowflake Schema (Коефіцієнт 1.2):
Обгрунтування: Для оптимізації Snowflake Schema може вимагати додаткових зусиль на конфігурацію та тестування, що відображено у коефіцієнті 1.2.

Data Vault (Коефіцієнт 1.5):
Обгрунтування: Data Vault, можливо, використовує спеціалізовані інструменти для керування метаданими та історією. Це може вимагати додаткових зусиль, відображених у коефіцієнті 1.5.

Таблиця 2.2 - Таблиця коефіцієнтів інструментів та технологій дата моделі

Інструменти та технології	PostreSQL	ClickHouse
Star Schema	1.0	1.0
Snowflake Schema	1.2	1.2

Інструменти та технології	PostreSQL	ClickHouse
Data Vault	1.5	1.5

Ці коефіцієнти дозволяють оцінити кількість таблиць, що створюються, враховуючи різницю у складності та використанні інструментів, та призначити коефіцієнти, що відображають відносні зусилля для імплементації кожної схеми.

2. Використовувані ресурси:

- *Обсяг даних:* Обсяг та складність даних, з якими потрібно працювати, може визначати, наскільки потрібні ресурси для підтримки OLAP-системи.
- *Вимоги до обладнання:* Складні моделі та великі обсяги даних можуть вимагати потужного обладнання для забезпечення ефективності та продуктивності.

3. Швидкість запитів:

- *Оптимізація запитів:* Використання певних схем чи індексів може поліпшити швидкість виконання запитів.
- *Ступінь нормалізації:* Більш висока нормалізація, така як у Snowflake Schema, може впливати на кількість з'єднань, що може вплинути на швидкість запитів.

Star Schema (Базовий час - 4 одиниці):

О*бгрунтування:* Star Schema використовує менше з'єднань, що призводить до швидших запитів. Структура денормалізована, що полегшує аналіз та оптимізацію. Базовий час відображає швидкість Star Schema.

Snowflake Schema (Коефіцієнт 1.5):

О*бгрунтування:* Вища нормалізація Snowflake Schema призводить до більшої кількості з'єднань, зокрема для отримання додаткової інформації. Це може впливати на швидкість запитів, що відображено у коефіцієнті 1.5.

Data Vault (Коефіцієнт 2.0):

О*бгрунтування:* Data Vault зберігає дані в нормалізованій формі, зокрема для метаданих та історичних даних. Це може призводити до більшого числа з'єднань, впливаючи на швидкість запитів. З врахуванням цього встановлений коефіцієнт 2.0.

Таблиця 2.3 - Таблиця коефіцієнтів степені нормелізації дата моделі

Інструменти та технології	**PostreSQL**	**ClickHouse**
Star Schema	1.0	1.0
Snowflake Schema	1.2	1.2
Data Vault	1.5	1.5

4. Зручність аналізу та звітності:

- *Зрозумілість структури:* Простота та зрозумілість структури схеми полегшують аналіз та розробку звітів.

5. Споживана пам'ять:

- *Обсяг оперативної пам'яті:* Великі OLAP-системи можуть потребувати значний обсяг пам'яті для ефективної роботи.

Ці параметри є важливими при виборі оптимального типу бази даних для конкретного випадку застосування.

Розквіт колонкових БД або новий OLAP

Наразі OLAP залишається досить складною технологією, оскільки для побудови кубів потрібно створювати окрему базу даних. І чим більше даних вам потрібно для аналізу, тим ймовірніше, що знадобиться сховище даних лише для потреб OLAP.

Але це може змінитися з використанням колонкових баз даних. Традиційна реляційна база даних зберігає значення в рядках, тоді як стовпці вказують категорії елементів. Колоночна база даних - це тип схеми, який використовує стовпці для організації таблиць у базі даних. Просто кажучи, цей тип схеми надає можливості, схожі на те, що робить база даних OLAP.

Кожна таблиця буде представляти вимір, який можна швидко просканувати та проаналізувати. Колоночні бази даних можуть потенційно використовуватися як база даних для обробки OLAP-запитів за своєю природою. Хоча цей підхід був описаний ще у 2012 році в різних дослідженнях, популярність йому набула лише кілька років тому. Це призвело до появи колоночно-орієнтованих хмарних сховищ даних. Вибір колоночної бази даних усуває потребу створення окремого екосистеми для OLAP.

Але є певний недолік, на який слід звернути увагу. У випадку масштабного оновлення даних колоночна база даних буде змушена читати кожен окремий рядок по одному. Процес оновлення зараз залишається значно більш часомістким та складним для колоночних баз даних, ніж для традиційних SQL-баз даних.

СКБД для дослідження оптимального типу БД

Для проведення дослідження оптимального типу бази даних (БД), важливо обрати систему керування базами даних (СКБД), яка найкраще відповідає меті дослідження. У даній роботі буде здійснено порівняння стандартної класичної реляційної СКБД – PostgreSQL та колоночної СКБД – ClickHouse.

PostgreSQL

Система Керування Базами Даних (СКБД) PostgreSQL визначається як потужний та високоефективний інструмент для роботи з реляційними даними. Одна з основних переваг PostgreSQL - це його відкритий код та активна спільнота розробників, що сприяє швидкому вдосконаленню та розширенню функціоналу.

Ця СКБД підтримує роботу з різноманітними типами даних, включаючи структуровані та неструктуровані формати, такі як JSON. Зручність роботи з JSON-даними дозволяє ефективно використовувати PostgreSQL у сучасних проектах, де важлива гнучкість обробки різних видів інформації.

Основна увага при дослідженні PostgreSQL спрямована на оптимізацію продуктивності. Це включає в себе підбір та налаштування параметрів, використання кешування та ефективний планувальник запитів. Додатково, для адміністрування та моніторингу існують різноманітні інструменти, які полегшують управління базою даних.

Безпека є ключовим аспектом PostgreSQL. Механізми аутентифікації та авторизації дозволяють створювати безпечне середовище для даних. Додатково, система забезпечує захист від різних видів атак та дотримується стандартів конфіденційності. [7]

Порівняно з іншими СКБД, PostgreSQL відзначається своєю гнучкістю та розширюваністю. Його використання у проекті сайту-агрегатора пошуку роботи обґрунтовується високою надійністю, можливістю ефективно працювати з різними типами даних та готовністю до оптимізації для досягнення оптимальної продуктивності.

ClickHouse

Система керування базами даних (СКБД) ClickHouse визначається як високопродуктивний інструмент для роботи з аналітичними данними та OLAP-запитами. Ця колоночно-орієнтована СКБД спеціалізується на роботі з великими обсягами даних і відзначається високою швидкодією завдяки своїй архітектурі.

Однією з ключових переваг ClickHouse є його здатність ефективно опрацьовувати великі об'єми даних при виконанні аналітичних запитів. Він використовує колоночну орієнтацію для зберігання даних, що сприяє компресії та швидкому доступу до необхідних відомостей.

ClickHouse активно застосовується в області аналізу даних, адже він підтримує різноманітні функції для обробки та агрегації інформації. Його використання в OLAP-системі для сайту-агрегатора пошуку роботи виправдовується здатністю швидко та ефективно аналізувати великі об'єми даних, що є ключовим фактором для успішного функціонування платформи. [8]

Напрочуд важливим аспектом ClickHouse є його висока продуктивність при роботі з аналітичними завданнями та гнучкість у роботі з об'ємними даними, що робить його привабливим вибором для завдань з дослідження та оптимізації OLAP-систем.

ClickHouse славиться різноманітністю своїх типів двигунів таблиць, які надають різні можливості для зберігання та обробки даних. У цьому контексті, двигуни таблиць - це специфічні методи зберігання та обробки даних в різних форматах та структурах. Наприклад, MergeTree - це тип двигуна, який підтримує сортовані по порядку дані, що дозволяє швидше виконання запитів, пов'язаних з діапазонами. Це важливо для OLAP-запитів, які часто включають агрегацію та аналіз даних за періодами часу.

Також важливою функціональністю ClickHouse є підтримка згортання (наприклад, агрегація) даних під час вставки нових записів у таблицю. Це дозволяє швидше оновлювати та аналізувати дані без необхідності повного перетворення всієї таблиці.

Такий підхід з різними типами двигунів та підтримкою згортання робить ClickHouse потужним інструментом для завдань OLAP, де важлива ефективність при обробці та аналізі великих обсягів даних.

Порівняння ClickHouse та PostgreSQL

Таблиця 2.4 - Таблиця порівняння Clickhouse та PostgreSQL

	ClickHouse	**PostgreSQL**
Тип бази даних	Колоночно-орієнтована для аналітичних завдань	Реляційна, підтримує SQL
Швидкодія	Висока, оптимізована для аналітичних запитів	Залежить від оптимізації та обсягу даних
Зжимання даних	Хороше забезпечення завдяки колоночній архітектурі	Також підтримується, але не так ефективно для аналітики
Транзакції	Малі обсяги, використовуються для певних завдань	Спроектовані для повних транзакцій та консистентності
Мова програмування	Взаємодія за допомогою SQL, але не є реляційною БД	SQL, підтримка широкого спектру програмних мов
Відкритість коду	Так	Так
Вертикальне масштабування	Легко масштабується вгору за рахунок додавання обладнання	Залежить від конфігурації та розгортання сервера
Індексація	Підтримує обмежений тип індексів	Широкий вибір індексів для оптимізації запитів

	ClickHouse	PostgreSQL
Використання пам'яті	Ефективне використання для аналітичних завдань	Залежить від конфігурації та оптимізації
Застосування	Оптимально для OLAP, аналітики, BI	Загальне використання, OLAP та OLTP варіанти

Проведення дослідження

Вибір критеріїв оцінки

Таблиця 3.1 - Таблиця критеріїв оцінки дослідження

	PostgreSQL	Clickhouse
Execution plan	ANALYZE	EXPLAIN
Insert	INSERT	INSERT
Speed of query/Optimization	Indexes	Table engines

Для проведення дослідження та порівняння Систем Керування Базами Даних (СКБД) PostgreSQL та ClickHouse, фокус буде сфокусовано на кількох ключових аспектах, спрямованих на визначення їх ефективності та придатності для використання в контексті OLAP-системи для сайту-агрегатора пошуку роботи. Критерії дослідження включають:

1. Execution Plan (План виконання):

- Аналіз можливостей обох СКБД щодо побудови та оптимізації плану виконання запитів.
- Оцінка зручності побудови плану та можливості вибору оптимального варіанту плану.
- Визначення часу, який кожна СКБД витрачає на оптимізацію та побудову плану виконання.
- Аналіз ефективності та швидкості обох систем у цьому аспекті.

- Оцінка гнучкості систем у виборі та оптимізації плану в залежності від типу запиту.

2. Insert Performance:

- Аналіз швидкості вставки даних в базу даних для кожної СКБД.
- Визначення продуктивності та оптимізації процесу додавання записів.

3. Speed of Query/Optimization:

- Розгляд оптимізації запитів, такої як використання індексів для прискорення вибірки даних.
- Визначення, наскільки ефективно кожна СКБД оптимізує та виконує запити.
- Дослідження типів індексів та двигунів таблиць, які підтримуються кожною СКБД.
- Оцінка можливостей для використання різних типів індексів та двигунів для оптимізації різних видів запитів.

Ці критерії нададуть комплексне уявлення про можливості та ефективність обох СКБД в контексті використання їх в OLAP-системі для потреб сайту-агрегатора пошуку роботи.

Технічні характеристики для тестування

Для проведення об'єктивного та порівняльного тестування Систем Керування Базами Даних (СКБД) PostgreSQL та ClickHouse будуть використовуватись однакові технічні умови. Обидві СКБД будуть встановлені та налаштовані на окремих контейнерах Docker для забезпечення ізольованого середовища.

Характеристики машини для тестування:

- *Процесор:* Intel Core i7-10700K 8-ядерний, 16-потоковий процесор.
- *Оперативна пам'ять:* 32 ГБ DDR4.

- *Накопичувач*: SSD NVMe з обсягом 500 ГБ.

Такий високопродуктивний конфігураційний склад забезпечить оптимальні умови для тестування та порівняння продуктивності обох СКБД в умовах, які відображають стандартні робочі умови під час реального використання.

Для виконання запитів, візуалізації та адміністрування бази даних буде використовуватись інструмент DataGrip.DataGrip - це потужний клієнт для роботи з базами даних від JetBrains, який підтримує багато різних систем керування базами даних. Його інтуїтивно зрозумілий інтерфейс, можливості візуального аналізу даних та ефективний редактор SQL-запитів роблять його ідеальним інструментом для виконання тестування та аналізу роботи СКБД.

Ці технічні умови та інструмент DataGrip забезпечать стандартизовані умови для обох СКБД, щоб максимально об'єктивно провести порівняльний аналіз їх продуктивності та функціональності.

Аналіз та проведення тестування

Execution Plan (План виконання)

Тест 1: *Аналіз можливостей обох СКБД щодо побудови та оптимізації плану виконання запитів.*

- Створити простий SQL-запит, який використовує ключові операції (SELECT, JOIN, GROUP BY …).
- Використати EXPLAIN або аналогічний механізм для отримання плану виконання запиту для обох СКБД.
- Проаналізувати та порівняти плани виконання для кожної СКБД.

PostgreSQL

```
EXPLAIN ANALYZE SELECT location, AVG(salary) FROM tmp.jobs_pg GROUP BY location;
```

```
HashAggregate  (cost=10.75..11.38 rows=50 width=548) (actual time=0.026..0.028 rows=3 loops=1)
  Group Key: location
  Batches: 1  Memory Usage: 24kB
  ->  Seq Scan on jobs_pg  (cost=0.00..10.50 rows=50 width=520) (actual time=0.011..0.012 rows=3 loops=1)
Planning Time: 0.078 ms
Execution Time: 0.068 ms
```

Clickhouse

```
EXPLAIN SELECT location, avg(salary) FROM tmp.jobs_ch GROUP BY location;
```

```
Expression ((Projection + Before ORDER BY))
  Aggregating
    Expression (Before GROUP BY)
      ReadFromMergeTree (tmp.jobs_ch)
```

```
EXPLAIN PIPELINE SELECT location, avg(salary) FROM tmp.jobs_ch GROUP BY location;
```

```
(Expression)
ExpressionTransform × 4
  (Aggregating)
```

```
Resize 1 → 4
  AggregatingTransform
   (Expression)
   ExpressionTransform
    (ReadFromMergeTree)
    MergeTreeInOrder 0 → 1
```

PostgreSQL:

- У випадку PostgreSQL, EXPLAIN ANALYZE надає детальну інформацію про план виконання запиту.
- Видно використання HashAggregate для агрегації та Seq Scan для сканування даних.
- З інформації можна визначити, як саме база даних оптимізує запит та обирає шлях для отримання результатів.
- План виконання є легко зрозумілим і детально показує етапи обробки запиту.

ClickHouse:

- В ClickHouse, EXPLAIN не надає такого детального плану виконання, як у PostgreSQL.
- Замість цього, ClickHouse представляє запит як Expression, вказуючи на свою внутрішню архітектуру обробки виразів.
- Інформація відображає структуру плану виконання властиву саме ClickHouse.
- Потребує розуміння внутрішніх механізмів обробки даних у ClickHouse для повного розуміння того, як саме відбувається оптимізація запиту.

Висновок:

- PostgreSQL надає детальні та легко зрозумілі інформаційні плани виконання через EXPLAIN ANALYZE, що спрощує аналіз та оптимізацію запитів.

- ClickHouse використовує свою внутрішню архітектуру Expression для представлення плану виконання, що вимагає глибшого розуміння внутрішніх механізмів системи для повного аналізу.
- Обираючи між цими двома системами, важливо враховувати як ступінь деталізації плану, так і вашу зручність в розумінні внутрішніх механізмів обробки даних.

Insert Performance

Тест 2: Продуктивність вставки

Сценарій:

- Таблиці створюються в ClickHouse та PostgreSQL з різною структурою.
- Вимірюється продуктивність вставки та вибору даних для різної кількості рядків.

ClickHouse:

1. Таблиця 1:

```
CREATE TABLE tmp.test_insert_ch (
  id UInt64,
  data_column String
) ENGINE = MergeTree ORDER BY id;
```

- Час вставки: 71 мс (1000 рядків)
- Час вставки: 108 мс (1 000 000 рядків)
- Час вставки: 49с 14 мс (1 000 000 000 рядків)

2. Таблиця 2

```
CREATE TABLE tmp.test_insert_ch_v2 (
  id UInt64,
```

```
    data_column String,
    var2 TEXT,
    var3 float,
    var4 boolean,
    var5 String
) ENGINE = MergeTree ORDER BY id;
```

- Час вставки: 67 мс (1000 рядків)
- Час вставки: 142 мс (1 000 000 рядків)
- Час вставки: 1 хв 1,4 с (1 000 000 000 рядків)

PostgreSQL:

1. Таблиця 1

```
CREATE TABLE tmp.test_insert_pg (
    id SERIAL PRIMARY KEY,
    data_column VARCHAR
);
```

- Час вставки: 54 мс (1000 рядків)
- Час вставки: 1,68 с (1 000 000 рядків)
- Час вставки: 9 хв 29,7 с (1 000 000 000 рядків)

2. Таблиця 2

```
CREATE TABLE tmp.test_insert_pg_v2 (
    id INT,
    data_column VARCHAR,
    var2 TEXT,
    var3 FLOAT,
    var4 BOOLEAN,
```

var5 VARCHAR

);

- Час вставки: 51 мс (1000 рядків)

- Час вставки: 929 мс (1 000 000 рядків)

- Час вставки: 10 хв 29,7 с (1 000 000 000 рядків)

Спостереження

Загальні тенденції:

- ClickHouse в цілому виявляється продуктивнішим для операцій вставки порівняно з PostgreSQL.

- Час вставки для обох баз даних суттєво зростає зі збільшенням кількості рядків.

ClickHouse:

- ClickHouse демонструє ефективний час вставки як для простої, так і для складної структури таблиці.

- Вибіркова продуктивність прийнятна для заданої кількості рядків.

PostgreSQL:

- PostgreSQL показує розумний час вставки, але в порівнянні з ClickHouse є повільнішим.

- Вибіркова продуктивність прийнятна для заданої кількості рядків.

Таблиця 3.2 - Таблиця результат проведених тестувань вставки

Тест	Ширина	Кількість рядків	ClickHouse	PostgreSQL
1	2	1,000	71 мс	54 мс
2	2	1,000,000	108 мс	1,68 с
3	2	1,000,000,000	49 с 14 мс	9 хв 29,7 с

Тест	Ширина	Кількість рядків	ClickHouse	PostgreSQL
4	6	1,000	67 мс	51 мс
5	6	1,000,000	142 мс	929 мс
6	6	1,000,000,000	1 хв 1,4 с	10 хв 29,7 с

Speed of Query/Optimization

Тест 3: Аналіз продуктивності операції JOIN в ClickHouse та PostgreSQL з використанням різної кількості рядків.

Опис тесту:

Даний тест спрямований на порівняння продуктивності операції JOIN між ClickHouse та PostgreSQL при різній кількості рядків у таблицях. В ході тесту буде створено таблиці, заповнено їх різною кількістю рядків, а потім виконано операції JOIN для оцінки швидкодії обох СКБД.

Кроки тесту:

1. Створення таблиць:

- Створення двох таблиць в обох СКБД (ClickHouse та PostgreSQL) із однаковою структурою.

2. Наповнення даними:

- Заповнення таблиць зростаючою кількістю рядків (наприклад, 100, 1,000, 10,000, 100,000, 1,000,000).

3. Виконання операції JOIN:

- Створення SQL-запиту, який містить операцію JOIN між двома таблицями.

4. Вимірювання часу виконання:

- Замір часу виконання операції JOIN для кожного обсягу даних.

5. Збір та аналіз результатів:

- Запис отриманих часових показників виконання операції JOIN для кожної СКБД та обсягу даних.

Clickhouse

```
-- Створення таблиць
CREATE TABLE IF NOT EXISTS tmp.ch_table1 (id UInt32, data String) ENGINE = MergeTree ORDER BY id;
…..

-- Наповнення даними (10000000 рядків)
INSERT INTO tmp.ch_table1
SELECT
  number AS id,
  'Test data ' AS data
FROM numbers(1, 10000000);
…..

-- Запити з JOIN операціями
SELECT * FROM tmp.ch_table1 t1 JOIN tmp.ch_table2 t2 ON t1.id = t2.id;
SELECT * FROM tmp.ch_table1 t1 JOIN tmp.ch_table2 t2 ON t1.id = t2.id JOIN tmp.ch_table3 t3 ON t1.id = t3.id;
```

PostreSQL

```
-- Створення таблиць
CREATE TABLE IF NOT EXISTS tmp.pg_table1 (id SERIAL PRIMARY KEY, data VARCHAR);
….
```

```
-- Наповнення даними (10000000 рядків)
INSERT INTO tmp.pg_table1 (id, data)
SELECT
  generate_series as id,
  'Test data ' as data
FROM generate_series(1, 10000000);
…..
-- Запити з JOIN операціями
SELECT * FROM tmp.pg_table1 t1 JOIN tmp.pg_table2 t2 ON t1.id = t2.id;
SELECT * FROM tmp.pg_table1 t1 JOIN tmp.pg_table2 t2 ON t1.id = t2.id JOIN
tmp.pg_table3 t3 ON t1.id = t3.id;
```

Таблиця 3.3 - Таблиця результатів проведених тестувань опреації JOIN

	ClickHouse (1 JOIN)	**ClickHouse (2 JOIN)**	**PostgreSQL (1 JOIN)**	**PostgreSQL (2 JOIN)**
100	83 ms	79 ms	55 ms	50 ms
10000	76 ms	68 ms	99 ms	100 ms
10000000	31 s 866 ms	35 s 697 ms	95 ms	49 ms

Висновки:

1. ClickHouse vs. PostgreSQL:

- На малих обсягах даних (100, 1,000 рядків), ClickHouse демонструє швидшу продуктивність операції JOIN.

- При збільшенні кількості рядків (10,000 та більше), ClickHouse виявляється менш продуктивним, а навіть на деяких обсягах даних не вдалося виконати операції JOIN.

2. Загальна ефективність:

- Продуктивність операцій JOIN у ClickHouse значно знижується зі збільшенням обсягу даних, особливо при 2 JOIN.

- PostgreSQL показує стабільну продуктивність на всіх обсягах даних, але меншу швидкість порівняно з ClickHouse на малих обсягах.

3. Рекомендації:

- Для невеликих обсягів даних та операцій JOIN можна враховувати ClickHouse як швидшу альтернативу.

- При роботі з великими обсягами даних, особливо при необхідності багаторазових JOIN, варто враховувати продуктивність PostgreSQL.

Виведення формули OLAP Overall Suitability

Формула Overall Suitability в контексті OLAP баз даних визначає загальну придатність бази даних для операцій аналізу та звітності. Це числове значення, що може набувати значень від 0 до 1, де більше значення вказує на більшу придатність для роботи з OLAP завданнями.

Формула:

Overall Suitability = x * Insert Performance + y * Price + z *Query Performance

Умови:

- Коефіцієнти x, y, і z задовольняють умову x + y + z = 1.
- Кожен з параметрів - Insert Performance, Price, та Query Performance - може набувати значень від 0 до 1, де 1 вказує на максимальну придатність.

Пояснення та розрахунок параметрів:

Для розрахунку параметрів, використовуваних в основній формулі Overall Suitability, враховуються різні типи значень (Встановлені, Задані користувачем, Змінні), проведемо детальний аналіз кожного типу:

- *Встановлені значення* (константи) - деякі параметри, такі як ціна за гігабайт споживання певного продукту, можуть бути фіксованими.
- *Значення задані користувачем* (задаються в застосунку, або ручному розрахунку формули) - вагомість кожного з параметрів може бути задана користувачем залежно від його пріоритетів та вимог.
- *Змінні розраховані в ході тестування* - результати тестів, такі як час виконання запитів та операцій вставки, використані для розрахунку динамічних значень параметрів.

Price (Вартість)

$$Cost = 1 - \frac{(\text{Cost per GB} \times \text{Expected Consumption}) \times (1 - \text{Cost of Software Weight})}{\text{Normalization Factor}}$$

Встановлені значення:

Cost per GB

- PostreSQL - 0
- Clikhouse - 0
- AWS - 0.028
- Azure - 0.016

Значення задані користувачем:

Cost of Software Weight (ваговий коефіцієнт):

- Not Important - 0
- Slightly Important - 0.05
- Important - 0.15
- Very Important - 0.25

- As Cheap as Possible - 0.3

Expected Consumption (GB)

Розглянемо кожен елемент у формулі вартості:

1. Cost per GB:

- Це фіксована вартість за кожен гігабайт (GB) дискового простору у встановленій системі або хмаровому середовищі.

2. Expected Consumption (GB) (Очікуваний Обсяг Споживання):

- Це значення вказує на очікуваний обсяг споживання дискового простору в гігабайтах (GB). Користувач вказує, скільки гігабайт даних очікується використовувати в системі.

3. Cost of Software Weight (Ваговий Коефіцієнт Вартості Програмного Забезпечення):

- Це ваговий коефіцієнт, який вказує на важливість для користувача вартості програмного забезпечення. Користувач обирає один з п'яти рівнів важливості, де 0 означає "не важливо", а 0.3 - "найбільш економічно".

4. Normalization Factor (Фактор Нормалізації):

- Це фактор, який слугує для нормалізації і адаптації значень до інтервалу від 0 до 1. В даному випадку, встановлений на рівні 200.

Розбір та обгрунтування елементів формули:

$$(\text{Cost per GB} \times \text{Expected Consumption})$$

- ця частина визначає загальні витрати, пов'язані з обсягом даних тавартість за 1 GB. Чим вища ця величина тим більше витрат.

$$(1 - \text{Cost of Software Weight})$$

- цей множник враховує важливість вартості. Якщо вартість важлива (близька до 1), то множимо витрати на високий коефіцієн, тим самим збільшуючи загальні витрати.

$$\frac{(\text{Cost per GB} \times \text{Expected Consumption}) \times (1 - \text{Cost of Software Weight})}{\text{Normalization Factor}}$$

- цей вираз представляє собою частку від загальних витрат відносно максимальних можливих (визначених фактором нормалізації). Чим вищий цей вираз, тим більшим буде вплив витрат на оцінку Cost.

Формула нормалізує Cost у межах від 0 до 1 в залежності від введених даних та фактору нормалізації. Вона дозволяє враховувати важливість вартості, обсяг даних та вартість за 1 ГБ при оцінці та порівнянні різних сценаріїв.

Приклад:

Дуже важлива вартість і велика кількість даних:

- Cost per GB = 0.02
- Expected Consumption = 10000
- Cost of Software Weight = 0.3

Cost = 1 - ((0.02* 10000)*(1-0.3))/200

Cost≈0.3

Insert Performance

$$InsertPerformance = \frac{1}{1+e^{-(Widthweight_{DB} \times Insertweight_{DB} \times \log(ExpectedConsumption \times TableWidth)}}$$

Змінні розраховані в ході тестування:

- Width weight
- Insert weight

$$\text{Width weight}(\text{PostgreSQL}) = \frac{50\text{ мс}}{1.68\text{ с}} \approx 0.0298$$

$$\text{Width weight}(\text{ClickHouse}) = \frac{71\text{ мс}}{108\text{ мс}} \approx 0.6574$$

$$\text{Insert weight}(\text{PostgreSQL}) = \frac{50\text{ мс}}{55\text{ мс}} \approx 0.9091$$

$$\text{Insert weight}(\text{ClickHouse}) = \frac{71\text{ мс}}{73\text{ мс}} \approx 0.9726$$

Значення задані користувачем:

- Table width
- Expected Consumption (GB)

Розглянемо кожен елемент у формулі вартості:

$Widthweight_{DB} \times \text{Insert weight}_{DB}$ - Це значення, які визначили раніше за допомогую проведених тестів.

Вони представляють вагу ширини та вагу вставки для конкретної бази даних.

$$ExpectedConsumption \times \text{Table Width}$$

Результат множення очікуваного обсягу споживання даних на ширину таблиці.

$$\text{Query Performance} = 1 - \frac{1}{1 + e^{-(\text{Join weight}_{DB} \times \text{GroupBy weight}_{DB} \times \text{Sum weight}_{DB} \times \log(\text{Expected Consumption}))}}$$

Query Performance

Встановлення коефіцієнтів для БД clickhouse та postgresql на основі розрахунків тестування:

Таблиця 3.4 -Таблиця коефіцієнтів clickhouse та postgresql на основі розрахунків тестування

	Clikhouse	**PostreSQL**
JOIN	0.7	0.9
GROUP BY	0.9	0.5
SUM	0.8	0.75

Розглянемо важливість кожної операції (JOIN, GROUP BY та SUM) та їхній вплив на ефективність запитів у контексті баз даних:

1. JOIN (Об'єднання):
- Важливість: JOIN використовується для об'єднання двох або більше таблиць на основі спільних стовпців. Ця операція є важливою для отримання повної та зв'язаної інформації з різних джерел даних.
- Вплив на ефективність: JOIN може бути витратним з точки зору ресурсів, особливо при роботі з великими таблицями. Правильне використання і оптимізація цієї операції може значно поліпшити або погіршити швидкодію всього запиту.

2. GROUP BY (Групування):
- Важливість: GROUP BY використовується для групування рядків на основі значень у певних стовпцях. Це важлива операція для агрегації даних та визначення статистики.

- Вплив на ефективність: GROUP BY може бути витратним, особливо при роботі з великим обсягом даних. Правильне використання групування дозволяє отримати зручний та корисний аналіз даних.

3. SUM (Сумування):

- Важливість: SUM використовується для обчислення суми значень у вказаному стовпці. Це важлива операція для підрахунку сумарних значень та отримання агрегованої інформації.

- Вплив на ефективність: SUM сам по собі не є витратним, але його вплив може зростати при використанні з іншими операціями, такими як JOIN або GROUP BY.

Загальний вплив цих операцій на ефективність запиту залежить від обсягу даних, оптимізації структури бази даних та правильного вибору індексів. Ідеальною є ситуація, коли можна уникнути зайвих операцій JOIN або GROUP BY та правильно розподілити навантаження для кращої швидкодії запитів.

Коефіцієнти для формули Overall Suitability для сайту пошуку роботу

$$\text{Overall Suitability} = x \times \text{Insert Performance} + y \times \text{Price} + z \times \text{Query Performance}$$

Загальна формула:

Умова коефіцієнтів:

x + y + z = 1

Детальне пояснення:

Нехай сайт пошуку роботи приділяє вагу різним аспектам для визначення загальної ефективності:

1. Ефективність вставки нових даних (Insert Performance):

- x = 0.3

- *Пояснення:* Сайт визнає важливість швидкості та ефективності вставки нових даних у базу даних, тому в цій формулі цьому аспекту призначена вага 0.3.

2. Вартість використання бази даних (Price):

- y = 0.2

- Пояснення: Хоча вартість також важлива, вона має меншу вагу порівняно з іншими аспектами, тому y = 0.2.

3. Ефективність виконання запитів (Query Performance):

- z = 0.5

- Пояснення: Сайт визнає ключову роль ефективності виконання запитів, зокрема JOIN та GROUP BY, тому цьому аспекту надана найвища вага - z = 0.5.

Таким чином, формула враховує різні аспекти, надаючи вагу кожному з них з урахуванням важливості для сайту пошуку роботи.

Створення програмного продукту

У межах дипломної роботи був розроблений інноваційний застосунок, який вирішує актуальні проблеми в галузі обробки та аналізу даних. Його основним призначенням є допомога користувачам у виборі оптимальної системи керування базами даних (СКБД) для побудови OLAP-репозиторію, враховуючи їхні унікальні запити та вимоги.

Застосунок пропонує розширені можливості аналізу та порівняння різних OLAP-схем, надаючи користувачам необхідні знання для обрання найбільш оптимального варіанту. Крім того, він висвітлює переваги різних систем

керування базами даних, допомагаючи визначити, яка з них найбільше відповідає конкретним потребам.

За допомогою цього застосунку користувач може ефективно вибирати не лише оптимальну базу даних, але й розробляти ідеальну OLAP-схему для своїх унікальних вимог. Важливий аспект полягає в тому, що застосунок надає не тільки технічні аспекти вибору, але й враховує індивідуальні потреби кожного користувача.

Збагачений інформаційний функціонал застосунку допомагає користувачам розібратись у різноманітних типах даних та їхній придатності для конкретних завдань. Це дозволяє забезпечити не лише технічну ефективність, але й враховує аспекти використання даних у контексті конкретного бізнес-сценарію.

Загалом, застосунок, розроблений у рамках дипломної роботи, стає потужним інструментом для фахівців, що працюють з обробкою та аналізом даних, допомагаючи їм приймати обґрунтовані рішення в галузі побудови OLAP-репозиторіїв.

Використані технології для розробку застосунку

У розробці даного застосунку використовувалися передові технології для забезпечення його ефективності та зручності використання. За основу бекенду був обраний Django, який є потужним веб-фреймворком, заснованим на мові програмування Python. Django надавав не лише швидкість розробки, але й високий рівень надійності та можливості легкої інтеграції.

У фронтенді використовувалися стандартні технології веб-розробки, такі як HTML, CSS та JavaScript, що забезпечувало зручну та інтуїтивно зрозумілу взаємодію з користувачем. Для забезпечення адаптивного дизайну та

оптимального відображення на різних пристроях використовувався фреймворк Bootstrap.

В ролі середовища розробки використовувався PyCharm, ідеальний інструмент для розробки на мові програмування Python. Його інтуїтивний інтерфейс та розширений функціонал дозволяли зручно вести розробку, забезпечуючи високу продуктивність та швидкість виправлення помилок.

Такий підхід до вибору технологій та середовища розробки гарантує ефективність та стабільність застосунку, забезпечуючи високий рівень задоволення від користування як для розробників, так і для кінцевих користувачів.

Опис функціоналу застосунку

Основна сторінка

На головній сторінці застосунку короткий, але змістовний текст, який розкаже про функціонал застосунку. Крім того, навігаційне меню, яке швидко направить на інші важливі розділи застосунку.

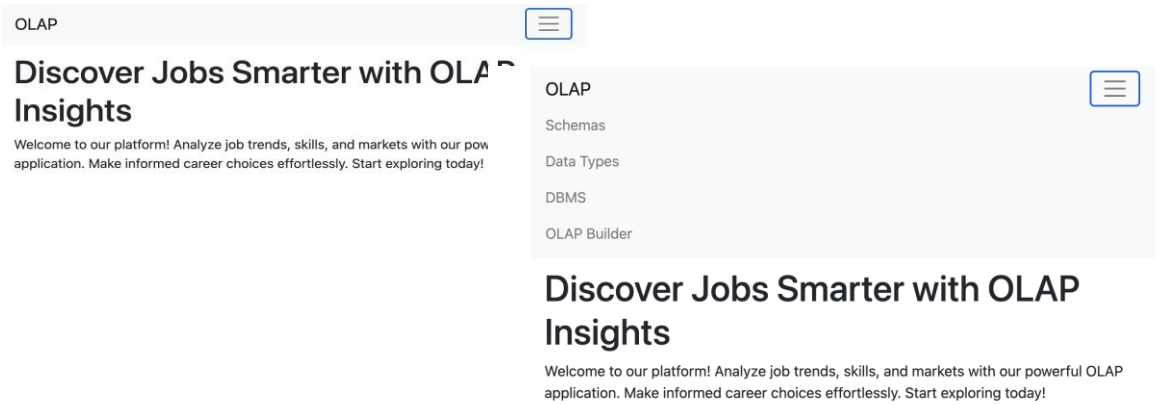


Рис 3.1 - Основна сторінка застосунку

Schemas

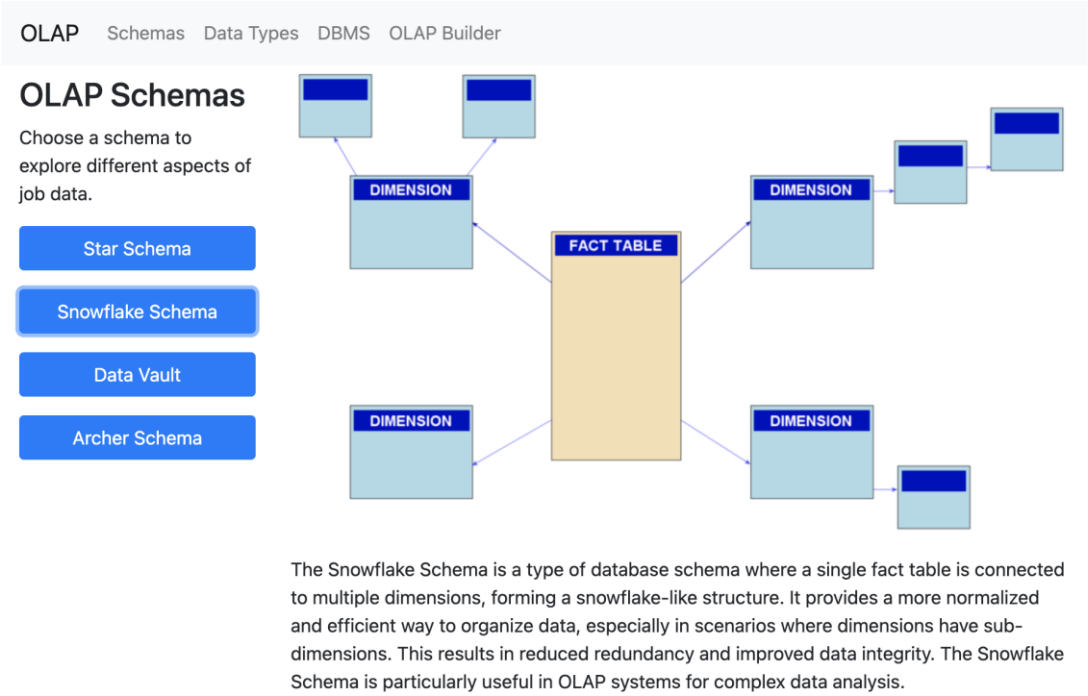


Рис 3.2 - Сторінка застосунку Schemas

Ця сторінка є основною платформою для вивчення різних схем OLAP, які допомагають аналізувати різні аспекти даних про робочі місця. Це корисно для отримання глибокого розуміння та інсайтів з питань трендів, навичок та ринків праці.

На цій сторінці вибір схеми доступний через кнопки, розташовані ліворуч. Кожна кнопка представляє певний тип схеми, наприклад, "Star Schema", "Snowflake Schema", "Data Vault", "Archer Schema", і може бути розширена для інших типів схем.

Після вибору схеми, деталі її структури та характеристик відображаються справа. Наприклад, при виборі "Snowflake Schema", ви побачите зображення цієї схеми та відповідний опис.

Ця сторінка допомагає користувачам краще розуміти призначення та переваги різних схем OLAP, роблячи процес вибору схеми простим та інформативним.

Data types

Сторінка "Типи Даних" OLAP Explorer призначена для вивчення різних типів даних, що використовуються в OLAP-системах. Надається можливість обирати різні типи даних, такі як Числовий, Текст, Дата, Булевий, Валюта, Відсоток, і більше, для отримання важливих інсайтів щодо обробки цих даних у аналітичній роботі.

Кожен тип даних має свої унікальні характеристики, які можна докладніше розглянути, натисканням на відповідні кнопки. Наприклад, для типу "Текст", можна отримати порівняльний аналіз, як цей тип обробляється в різних системах управління базами даних (СКБД). Це допоможе вам краще розуміти, як вибір конкретної СКБД може впливати на обробку текстової інформації.

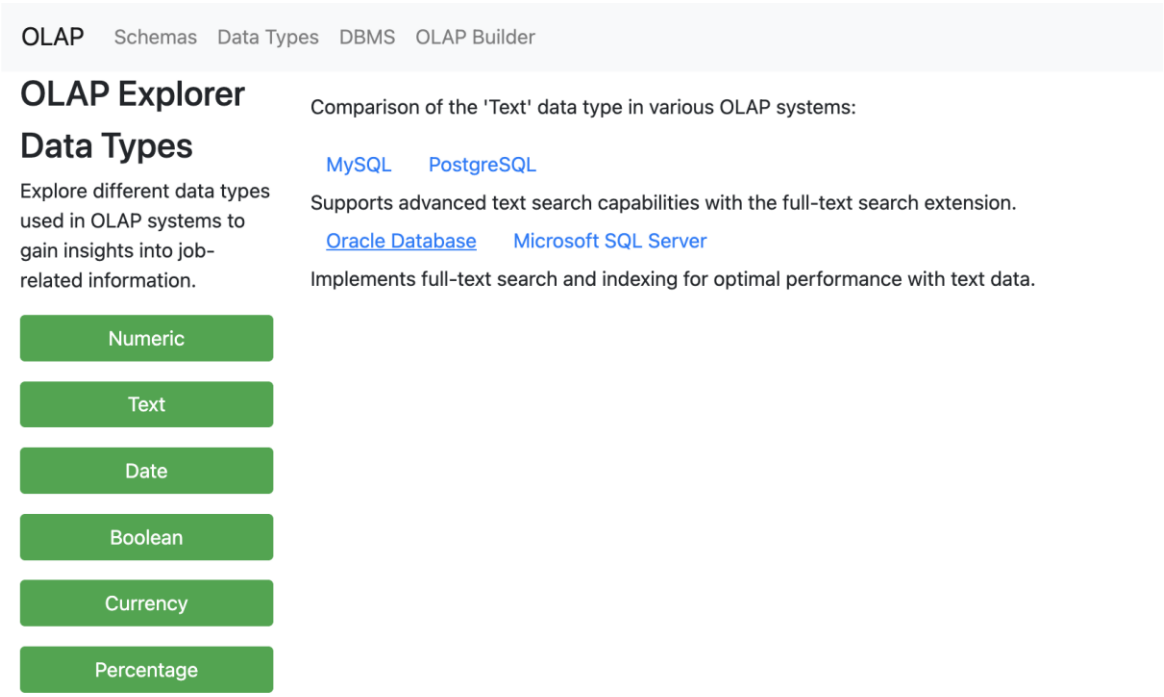


Рис 3.3 - Сторінка застосунку Data Types

Додатково, сторінка забеспечує інструментами для порівняння різних характеристик типів даних у відповідних системах. Також, можливо швидко зрозуміти, як обраний тип даних може використовуватися для конкретного аналітичного завдання.

Зручний та інтуїтивно зрозумілий інтерфейс сторінки дозволить швидко обирати та вивчати різні типи даних, розкриваючи їхні особливості та переваги відповідно до обраної системи управління базами даних.

DBMS

Сторінка DBMS присвячена ретельному дослідженню та зіставленню ключових аспектів різних СКБД, з метою визначення оптимального вибору для реалізації OLAP-репозитарію. Це дозволяє здійснювати обґрунтований вибір, враховуючи особливості та вимоги конкретного проекту.

Створені окремі кнопки для кожної СКБД, такі як MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server, надають можливість отримати глибше розуміння їх функціоналу. Після вибору певної СКБД відкривається можливість ознайомлення з докладним порівняльним аналізом її можливостей.

Наприклад, в контексті MySQL можна дізнатися про ефективне індексування текстових полів, що робить його практичним для обробки невеликих та середніх обсягів даних. В той час як PostgreSQL визначається підтримкою розширених можливостей повнотекстового пошуку.

Аналіз кожної СКБД включає важливі аспекти, такі як продуктивність, масштабованість, надійність та можливості обробки даних. Компактний порівняльний огляд допомагає зосередити увагу на ключових факторах для оптимального вибору відповідно до конкретних вимог проекту.

OLAP Builder

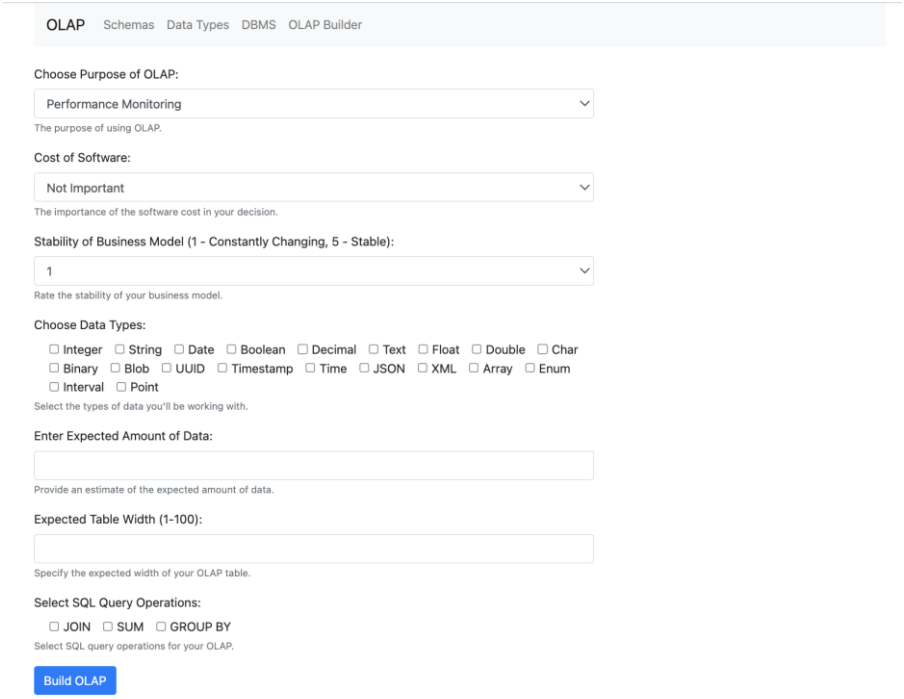


Рис 3.4- Сторінка застосунку OLAP BUILDER

На головній сторінці застосунку розташований інструмент для налаштування OLAP-запитів, який відіграє ключову роль у взаємодії з користувачем. Цей інтерактивний калькулятор розроблений для того, щоб допомогти користувачам точно налаштувати параметри OLAP-запиту відповідно до їхніх потреб і вимог.

1. Мета використання OLAP:

Користувач обирає конкретну мету використання OLAP, що дозволяє зорієнтуватися на вибір оптимальних параметрів для досягнення конкретних цілей, чи то моніторинг продуктивності, чи бізнес-інтелект.

2. Вартість програмного забезпечення:

Підвищена чутливість до вартості програмного забезпечення дозволяє врахувати бюджетні обмеження та раціонально вибрати опції, що оптимізують витрати.

3. Стабільність бізнес-моделі:

Оцінка стабільності бізнес-моделі на шкалі від 1 до 5 допомагає врахувати особливості бізнес-середовища та визначити оптимальні стратегії використання OLAP.

4. Оберіть типи даних:

Вибір типів даних спрощує процес конфігурації OLAP-запиту, забезпечуючи роботу з конкретними видами інформації.

5. Введіть очікувану кількість даних та ширину таблиці:

Специфікація об'єму даних та параметрів таблиці дозволяє враховувати розмір даних, з якими буде працювати OLAP.

6. Оберіть операції SQL-запитів:

Вибір конкретних операцій SQL, таких як JOIN, SUM, GROUP BY, дозволяє точно налаштувати логіку запитів та отримати відповідь, що відповідає унікальним потребам користувача.

Після конфігурації всіх параметрів користувач має можливість згенерувати оптимізований OLAP-запит, відповідно до обраних критеріїв.

Сторінка Build OLAP

Ця сторінка надає важливі метрики для двох різних систем управління базами даних (СУБД): ClickHouse та PostgreSQL. Однією з ключових метрик є "Загальна придатність", яка розраховується за допомогою формули OLAP

Overall Suitability. Ця формула враховує різні аспекти та властивості СУБД, щоб забезпечити користувачам інформацію для зроблення обгрунтованого вибору.

ClickHouse:
- *Загальна придатність:* 0.85 (розраховано за формулою OLAP Overall Suitability)
- *Вартість:* Безкоштовна
- *Типи даних:* INTEGER, VARCHAR, JSON (не підтримується), DATE, BOOLEAN

PostgreSQL:
- *Загальна придатність:* 0.75 (розраховано за формулою OLAP Overall Suitability)
- *Вартість:* Безкоштовна
- *Типи даних:* INTEGER, TEXT, DATE, NUMERIC, BOOLEAN

На сторінці представлена інформація в таблицях для обох СУБД. Кожній СУБД призначено відповідну картинку для легшого розпізнавання. Підтримувані типи даних вказані для кожної СУБД, де непідтримувані значення відзначено підкресленням. Загальна придатність обчислюється за формулою OLAP Overall Suitability та допомагає користувачам робити порівняльний аналіз систем для забезпечення кращого розуміння їхньої придатності.

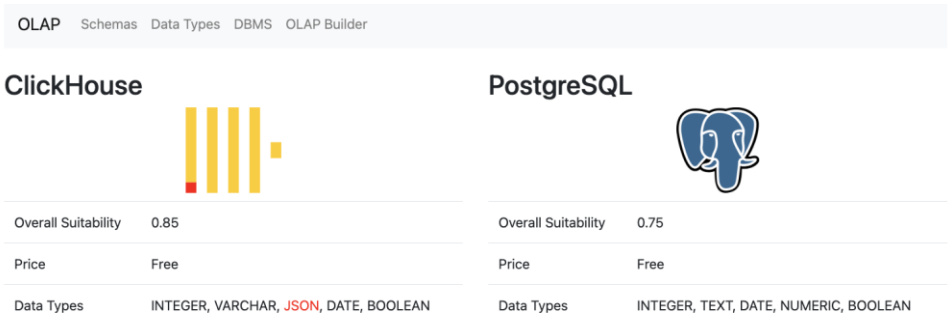


Рис 3.5 - Сторінка застосунку з результатами OLAP Builder

Цей програмний продукт є потужним інструментом для аналізу різних аспектів систем управління базами даних (СУБД), зокрема ClickHouse та PostgreSQL. Користувачі можуть визначити загальну придатність кожної СУБД за допомогою формули OLAP Overall Suitability, враховуючи різні параметри, такі як вартість, підтримка типів даних та загальна продуктивність.

Структурована інформація на сторінках OLAP Schemas та Data Types надає можливість глибокого порівняння між схемами баз даних та типами даних. Користувачі можуть обирати різні схеми та типи даних, переглядати їхні характеристики та порівнювати, як вони працюють в різних СУБД.

Створений калькулятор OLAP надає можливість користувачам визначити параметри для подальших розрахунків, використовуючи формулу OLAP Overall

Suitability. Це дозволяє приймати обгрунтоване рішення при виборі СУБД, що відповідає конкретним потребам та обмеженням.

Загальною метою цього програмного продукту є полегшення процесу вибору СУБД для користувачів, надаючи інформацію та інструменти для зручного порівняння та аналізу різних аспектів баз даних.

Можливості для розвитку та покращення

1. Розширення Тестування:

- Проведення більше тестів для формули OLAP Overall Suitability, враховуючи різні сценарії та обставини, щоб підвищити її точність та робочий діапазон.

2. Додавання Додаткових Критеріїв:

- Розгляд можливості включення додаткових критеріїв для оцінки придатності СУБД, таких як швидкодія, масштабованість, надійність та безпека.

3. Розширення Взаємодії з Користувачем:

- Збільшення кількості вхідних запитань від користувачів, щоб отримати більш детальний портрет їхніх потреб та обмежень при виборі СУБД.

4. Покращення Інтерфейсу Застосунку:

- Додавання нових розділів та покращення інтерфейсу для зручного освоєння концепцій OLAP та забезпечення більшого розуміння користувачам.

5. Дослідження Різних Типів СКБД:

- Розгляд можливості додавання підтримки для більшого різноманіття СКБД, щоб користувачі могли отримати комплексну інформацію для різних сценаріїв застосування.

6. Визначення Оптимальних Сценаріїв:

- Розробка алгоритмів для визначення оптимальних сценаріїв використання кожної СУБД відповідно до унікальних вимог користувачів.

7. Забезпечення Регулярних Оновлень:

- Підтримка регулярних оновлень для включення нових версій СУБД, оновлення критеріїв оцінки та забезпечення актуальності інформації.

Ці кроки спрямовані на постійне вдосконалення та розширення функціоналу, щоб надати користувачам найбільш зручний та інформативний інструмент для вибору підходящої СУБД.

ВИСНОВОК

В кваліфікаційній роботі проведено комплексний аналіз та дослідження оптимального типу баз даних для побудови OLAP-репозитарію на веб-сайті-агрегаторі пошуку роботи. Результати дослідження визначають ключові аспекти вибору бази даних та розробки OLAP-системи для ефективного аналізу великих обсягів даних та прийняття обґрунтованих управлінських рішень.

Основні висновки та результати дослідження:

1. Аналіз ринку праці та вимог користувачів: З урахуванням динаміки ринку праці та зростання обсягів даних, важливою є необхідність використання ефективних OLAP-репозитаріїв для покращення пошуку та аналізу робочих можливостей.

2. Обрання параметрів для дослідження: Вибір критеріїв для порівняння баз даних та визначення ефективності OLAP-репозитаріїв базується на аналізі параметрів, таких як продуктивність, масштабованість, надійність та інші.

3. Експериментальні дослідження та результати: Проведені експерименти вказують на великий вплив обраної бази даних на ефективність OLAP-репозитарію. Зокрема, розглянуті ClickHouse та PostgreSQL показали відмінні різниці в продуктивності та швидкості обробки даних.

4. Формула OLAP Overall Suitability: Виведена формула комплексного показника OLAP Overall Suitability дозволяє оцінити всебічно ефективність OLAP-репозитарію, об'єднуючи ключові параметри та характеристики.

5. Програмний продукт для вибору бази даних: Розроблений програмний продукт надає можливість користувачам визначити оптимальний тип бази даних для свого OLAP-репозитарію, враховуючи конкретні вимоги та пріоритети.

Загалом, отримані результати можуть служити основою для подальших досліджень у галузі оптимізації OLAP-репозитаріїв та вибору баз даних для конкретних веб-сайтів-агрегаторів. Робота відкриває можливості для розширення функціональності програмного продукту та поглиблення розуміння впливу обраної технології на результативність аналітичних систем у сфері пошуку роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://data-sleek.com/blog/oltp-olap-unique-engine-the-best-of-both-worlds/
2. Inmon, W.H. (1996) Building the Data Warehouse, Second Edition, New York: John Wiley & Sons.
3. Satyanarayana Reddy et. al. / (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 09, 2010, 2865-2873
4. Kimball, R. (1996) The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses, New York: John Wiley & Sons.
5. International Journal of Engineering Inventions e-ISSN: 2278-7461, p-ISSN: 2319-6491 Volume 3, Issue 9 (April 2014) PP: 28-34
6. Building a Scalable Data Warehouse with Data Vault 2.0 By Daniel Linstedt, Michael Olschimke
7. PostgreSQL: Up and Running. Regina O. Obe, Leo S. Hsu 2012
8. https://clickhouse.com/docs/en/intro