Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

Розробка методів і засобів для зберігання, аналізу та візуалізації багатовимірних даних

Текстова частина
магістерської роботи
за спеціальністю „Інженерія програмного забезпечення” 121

Керівник магістерської роботи
к.н., доц. Горборуков В.В.

_____
(підпис)
“____” _____ 2024 р.

Виконав студент
Рибак В.Я.
“____” _____ 2024 р.

Київ 2024

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав. кафедри інформатики
к.ф-м.н., доц. Гороховський С.С
_____
(підпис)
“_____” _____2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на магістерську роботу

студенту 2 р.н. магістерської програми Інженерія Програмного Забезпечення Рибаку Володимиру Ярославовичу
Розробити Систему для зберігання, аналізу та візуалізації багатовимірних даних

Зміст текстової частини до магістерської роботи:

Зміст
Анотація
Вступ
1 Теоретичні відомості зберігання, аналізу та візуалізації багатовимірних даних
2 Огляд підходів в наявних програмних рішеннях
3 Опис реалізації програмного продукту
Висновки по роботі та рекомендації для подальших досліджень
Список літератури
Додатки

Дата видачі “_____” _____ 2023 р.

Керівник
В.В. Горборуков, кандидат наук, доцент _____
(підпис)

Завдання отримав
В.Я. Рибак _____
(підпис)

Тема: Розробка методів і засобів для зберігання, аналізу та візуалізації багатовимірних даних

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на дипломну роботу	15.10.2023	
2.	Огляд технічної літератури за темою роботи	07.12.2023	
3.	Пошук наявних програмних рішень	12.02.2024	
4.	Аналіз наявних програмних рішень OLAP	24.02.2024	
5.	Аналіз наявних програмних рішень для зберігання, аналізу та візуалізації даних	21.03.2024	
6.	Постановка задачі	10.05.2024	
7.	Реалізація застосунку	15.05.2024	
8.	Написання теоретичної частини	17.05.2024	
9.	Оформлення теоретичної частини	19.05.2024	
10.	Створення слайдів для доповіді та написання доповіді	28.05.2024	
11.	Остаточне оформлення презентації	03.06.2024	
12.	Захист магістерської роботи	11.06.2024	

Студент Рибак В.Я.

Керівник Горборуков В.В. “____”_____ 2024 р.

ЗМІСТ

Анотація **5**
ВСТУП **6**
РОЗДІЛ 1: Теоретичні відомості зберігання, аналізу та візуалізації багатовимірних даних **8**
1.1 OLAP 8
1.1.1 Історія OLAP 8
1.1.2 Основні поняття OLAP 9
1.1.3 Архітектура OLAP систем 11
1.2 Колонкові бази даних 13
1.3 Схема зірки та сніжинки 13
1.4 Реляційні бази даних 14
1.5 Засоби візуалізації даних в веб застосунках 14
Розділ 2: Огляд підходів в наявних програмних рішеннях **16**
2.1 Microsoft Analysis Services 16
2.2 Snowflake 16
2.3 BigQuery 17
2.4 Tableau 18
2.5 Looker studio 19
2.6 Висновки та постановка задачі 20
Розділ 3: Опис реалізації програмного продукту **21**
3.1 Функціональні вимоги 21
3.2 Вибір та обґрунтування вибору інструментів розробки 22
3.2.1 React 22
3.2.1 Node.js та Express.js 22
3.2.1 PostgreSQL 23
3.3 Реалізація застосунку 23
3.3.1 Клієнтська частина застосунку 23
3.3.2 Серверна частина застосунку 25
3.3.2.1 Цілі серверного застосунку 25
3.3.2.2 Аналіз запиту і матеріалізація агрегатів 25
3.3.2.3 API для виконання запитів багатомірного аналізу 26
3.3.2.3 Реалізовані кінцеві точки API 26
3.3.3 Тестування отриманого застосунку 28
Висновки та рекомендації **32**
Список літератури **33**
Глосарій **35**
Додатки **36**
Додаток А 36
Додаток Б 37

Анотація

В рамках даної роботи були представленні теоретичні відомості зберігання, аналізу та візуалізації багатовимірних даних та огляд підходів в існуючих програмних рішеннях. В результаті був запропонований підхід зі збереженням агрегацій для підвищення ефективності виконання складних аналітичних запитів.

Ключові слова: Аналіз багатовимірних даних, онлайн-аналітична обробка, багатовимірні куби даних, агрегування даних.

ВСТУП

Актуальність. У сучасному світі дані набули значної цінності, а їх об’єми зростають з кожним роком. Так, загальний об’єм створених, спожитих та збережених даних у 2010 році становив 2 зетабайти, у 2020 році дане значення досягло 64.2 зетабайти, а прогнозоване значення на 2025 рік становить 180 зетабайтів [1]. Кожна взаємодія, транзакція чи цифрова діяльність створюють нові дані, які організації можуть використати в цілях розвитку себе або ж світу в загальному. Таке швидке зростання об’єму даних одночасно створює і можливості, і виклики, а ефективність підходів до збереження та аналізу даних набула вирішального значення для знаходження цінних тенденцій для прийняття стратегічних рішень компаній.

Аналіз багатовимірних даних відіграє одну з важливих ролей у цьому контексті, надаючи можливість ефективно аналізувати дані з різних сторін та з різним рівнем деталізації, завдяки чому компанії можуть швидше знаходити закономірності, кореляції та тенденції, в порівнянні з аналізом даних у традиційному табличному форматі.

На сьогоднішній день вже реалізовано велика кількість програмних рішень, як спеціалізованих на роботу з багатовимірними даними, так і направлених на різні види даних, включаючи багатовимірні, але враховуючи різні потреби та можливості користувачів, питання швидкості виконання аналітичних запитів до різноманітних джерел даних все ще є актуальним.

Мета дослідження. Розробити систему для аналізу багатовимірних даних направлену на підвищення швидкості виконання складних аналітичних запитів до не спеціалізованих сховищ даних.

Завдання дослідження. Проаналізувати існуючі підходи та програмні рішення до зберігання, аналізу та візуалізації багатовимірних даних. Виявити шляхи покращення ефективності виконання складних аналітичних запитів та реалізувати їх. Випробувати розроблене програмне рішення та описати результати.

Об´єкт дослідження. Підходи та програмні рішення зберігання, аналізу та візуалізації багатовимірних даних.

Предмет дослідження. Підходи для забезпечення ефективності виконання складних аналітичних запитів.

Джерела дослідження. Офіційна документація програмних рішень Microsoft Analysis Services, Snowflake, Looker Studio, Tableau. Електронні ресурси включаючи спеціалізовані веб-статті, наукові роботи, форуми, відео-записи конференцій.

РОЗДІЛ 1: Теоретичні відомості зберігання, аналізу та візуалізації багатовимірних даних

1.1 OLAP

1.1.1 Історія OLAP

Початком історії онлайн-аналітичної обробки даних (OLAP) може слугувати перший продукт, що міг здійснювати OLAP-запити, випущений у 1970 році під назвою Express, але сам термін ввів Едгар Франк «Тед» Кодд тільки у 1993 році разом з дванадцятьма правилами OLAP [2]. Правила, запропоновані Коддом, заклали основу для розробки систем, призначених для забезпечення основної ідеї OLAP - забезпечення швидкого, послідовного та інтерактивного доступу до даних з різних точок зору [3]. Ці принципи були спрямовані на підтримку складних запитів в контексті багатовимірного аналізу, які на той час не могли бути забезпеченими традиційними реляційними базами даних.

Протягом 1990-х і 2000-х років технологія OLAP зазнала значного розвитку. Розробка спеціалізованих баз даних і програмних рішень OLAP, таких як Microsoft Analysis Services, надала організаціям спеціалізовані інструменти для аналізу даних. Ці системи були розроблені для обробки великих обсягів даних, реалізуючи куби даних, багатовимірні запити та аналіз майже у реальному часі. Постійний розвиток обчислювальних потужностей і технологій зберігання даних ще більше розширив можливості OLAP-систем, зробивши їх невід'ємною частиною сучасної бізнес-аналітики та аналізу даних.

Подальше зростання обсягів великих даних і потреба в аналітиці в режимі справді реального часу змусили рухатись в напрямку розробки нових підходів. Сучасні програмні рішення для збереження, аналізу та візуалізації даних, такі як наприклад Snowflake та Tableau, змінили спосіб

роботи організацій зі своїми даними, відійшовши від вже звичних принципів роботи OLAP систем. Ці інструменти не сфокусовані саме на багатовимірних даних, натомість надають можливість ще більш ефективно працювати з даними, включаючи багатовимірні.

1.1.2 Основні поняття OLAP

В основі OLAP лежить декілька ключових концепцій, які вирізняють його з-поміж інших методів аналізу даних. Ці концепції є важливими для розуміння того, як працюють OLAP системи й які аналітичні можливості вони надають користувачу.

Традиційно, основною особливістю OLAP систем є багатовимірна структура даних в якій інформація формує багатовимірні куби, які також називають OLAP кубами або гіперкубами. В такій структурі кожен вимір кубу представляє собою окрему характеристику даних по якій організація хоче проводити аналіз, наприклад дату, географічне місце або категорії товару, а кожна комірка куба містить агреговані значення числових показників, наприклад загальну суму продажів чи кількість проданих товарів, див. Рисунок 1.1.2. Оскільки деякі характеристики показників можуть мати різний рівень деталізації, виміри можуть формувати ієрархічну структуру, наприклад при наявності дати, можна ввести вимір року, кварталу чи місяця, що дозволить аналізувати дані, як глобально, так і більш детально.

Рисунок 1.1.2. Схематичне зображення OLAP куба

Таким чином, гіперкуби дозволяють користувачам швидко переглядати та аналізувати дані з різних точок зору та різним рівнем деталізації завдяки завчасно підрахованим агрегованим значенням.

Для роботи з багатовимірними кубами, OLAP системи реалізують чотири основні аналітичні операції, які дозволяють користувачам ефективно та гнучко взаємодіяти з даними:

1. **Розгортання та згортання** (Drill-Down and Roll-Up). Операції розгортання та згортання є протилежними, вони дозволяють користувачам переходити між різними рівнями деталізації даних у межах одного виміру. Для забезпечення такої операції має бути визначена ієрархічна структура виміру, наприклад, при розгортанні років, користувач отримає інформацію по кварталам, а при згортанні днів - по тижням.
2. **Зріз** (Slice). Операція зрізу полягає у виборі одного виміру куба, в результаті чого створюється підкуб, в якому вибраний вимір фіксується на певному значенні. Наприклад, зріз за типом товару в

кубі даних продажів в результаті надасть підкуб в якому містяться дані лише за про відповідний тип товару.

3. **Нарізка на кубики** (Dice). Операція нарізки на кубики схожа до зрізу, але дозволяє користувачам вибрати два або більше виміри для отримання підкуба. Ця операція може бути використана, щоб зосередитись на певних частинах куба даних. Наприклад, нарізка за типами товару та роками надасть підкуб з даними по вказаним типам товару за вибрані роки.
4. **Зведення** (Pivot). Операція зведення, або ж обертання, дозволяє користувачам змінити орієнтацію куба. Прикладом обертання куба даних може слугувати зміна місцями рядків і стовпців у звіті для отримання нової точки зору на дані.

1.1.3 Архітектура OLAP систем

Типова архітектура OLAP системи складаються з таких компонентів:

1. **Сховище даних**. Загальне сховище даних організації;
2. **OLAP сервер**. OLAP сервер є головним компонентом системи який відповідає за підготовку даних до виконання аналітичних запитів;
3. **База даних OLAP**. База даних в якій зберігається підготовлені до аналітичних запитів дані, наприклад OLAP куби;
4. **Аналітичні інструменти OLAP**. Інструменти, що дозволяють користувачу виконувати аналітичні операції, зазначені вище, над OLAP кубом;

OLAP-системи поділяються на три основні види за архітектурою, основною відмінністю яких є різний підхід для збереження й обробки даних: MOLAP, ROLAP та HOLAP [4].

Багатовимірний OLAP (MOLAP)

MOLAP є найближчим до класичного підходу OLAP, де збереження даних відбувається у багатовимірних базах даних, що реалізують OLAP куби. Це забезпечує високу швидкість виконання запитів, оскільки дані попередньо агрегуються і зберігаються в спосіб, оптимізований для багатовимірного аналізу. З іншого боку, через обмеження багатовимірних баз даних такі системи погано масштабуються, а для оновлення даних потрібно перебудовувати OLAP куби, що є важкою операцією при великих обсягах даних.

Реляційний OLAP (ROLAP)

ROLAP використовує реляційні бази даних для зберігання та управління даними. В таких системах аналіз відбувається за допомогою SQL-запитів, а їх ефективність забезпечують додатковими налаштуваннями бази даних, наприклад встановленням bitmap індексу. Системи ROLAP є більш масштабованими, ніж MOLAP, однак продуктивність запитів є нижчою через накладні витрати на динамічну генерацію та виконання SQL-запитів.

Гібридний OLAP (HOLAP)

HOLAP поєднує в собі переваги MOLAP і ROLAP, використовуючи гібридний підхід. У таких системах дані розподіляються між багатовимірними та реляційними базами даних, немає чіткого правила, як дані мають бути розподілені, але основною ідеєю може слугувати збереження агрегованих даних в багатовимірній базі даних, а розгорнутих - в реляційній. Підхід HOLAP націлений на надання збалансованого рішення, що об’єднує продуктивність MOLAP та масштабованість ROLAP.

1.2 Колонкові бази даних

Колонкові сховища даних з’явились задовго до появи OLAP, так у 1969 році вже існувала перша колонкова база даних під назвою TAXIR, але саме поява OLAP дала новий поштовх для розвитку цієї технології [5]. Основною причиною цього явища можна назвати високу поєднуваність цих вдох технологій.

Так, в OLAP найпроблемнішим місцем є підрахунок агрегацій, оскільки побудова гіперкубу виконується довго, а реляційні бази даних не можуть забезпечити високу швидкість при виконанні складних аналітичних запитів. Натомість колонкові бази даних, завдяки організації даних на диску в колонковому форматі, дозволяли набагато швидше вичитувати дані по стовпчиках в порівнянні з реляційними базами даних, таким чином значно пришвидшуючи виконання агрегацій [6].

З подальшим розвитком колонкові бази даних отримати нові можливості, для забезпечення ще кращої ефективності при підрахунках агрегацій, наприклад завчасний підрахунок частини агрегацій для типових запитів або розподілення роботи по підрахункам на кілька вузлів.

1.3 Схема зірки та сніжинки

Схеми зірки та сніжинки, це варіанти багатовимірного представлення даних, що використовуються в реляційних сховищах даних. Схема зірки складається з таблиці фактів, та таблиць вимірів, що з’єднані з нею, див. рисонок 1.3. Як і з багатовимірним кубом даних, таблиці вимірів відповідають за характеристики, в даному випадку фактів. Таблиця ж фактів є основною і зберігає дані подій, наприклад покупку товару. Схема сніжинки розширює схему зірки, додаючи таблиці підвимірів, що зв’язуються з вимірами.

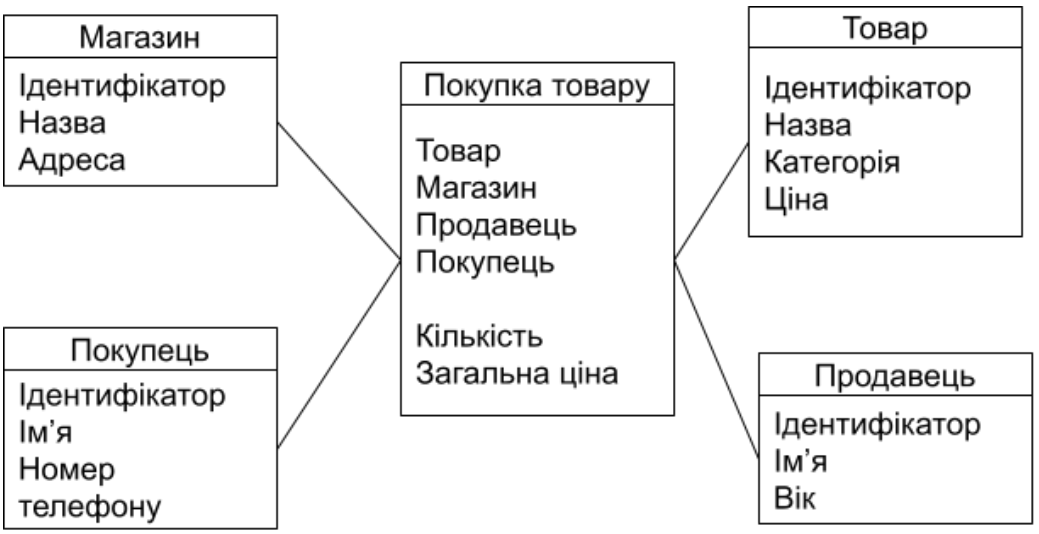


Рисунок 1.3. Приклад схеми зірки

1.4 Реляційні бази даних

Незважаючи на те, що реляційні бази даних не дуже підходять для багатомірних даних, вони все ще дуже широко використовуються. Фактично, в більшості випадків дані спочатку готуються в реляційній базі даних, а потім завантажуються у багатомірну базу. Дуже часто інструменти візуалізації даних працюють напряму з реляційними базами і виконують різноманітні запити, щоб побудувати необхідні агрегати. Тільки коли продуктивність стає незадоволеною, тоді переходять до багатомірних баз даних. При цьому дорогі СУБД мають в своєму арсеналі засоби для часткового вирішення проблем продуктивності. Наприклад, СУБД Oracle з функцією Query Rewrite може автоматично підставляти матеріалізовані view замість складного запиту з агрегатами.

1.5 Засоби візуалізації даних в веб застосунках

В той час як потужні засоби аналізу даних в більшості залишаються окремими рішеннями, існує дуже багато засобів, які дозволяють легко візуалізувати багатомірні дані на вебсторінках того чи іншого застосунку.

Це різноманітні реалізації Pivot Table і бібліотеки для відображення діаграм. Ці бібліотеки часто мають розвинутий інтерфейс, який дозволяє користувачам дивитись різноманітні зрізи даних. Проте вони базуються на статичному наборі даних, який має бути повністю вивантажений на клієнт, а відповідно, має дуже суттєве обмеження по розміру.

Розділ 2: Огляд підходів в наявних програмних рішеннях

2.1 Microsoft Analysis Services

Microsoft SQL Server Analytical Services (SSAS) - це потужний інструмент онлайн-аналітичної обробки, що дозволяє користувачам створювати багатовимірні та табличні моделі даних та виконувати складні аналітичні запити над ними. В загальній архітектурі систем онлайн-аналітичної обробки SSAS виступає у ролі OLAP серверу, що для повноцінної роботи потребує сховище даних та застосунок для візуалізації даних, наприклад Microsoft Excel.

Початково Analytical Services підтримував тільки багатовимірну модель, що реалізує традиційне OLAP рішення, тобто побудова багатовимірних кубів, що дає можливість швидко виконувати аналітичні запити, але з виходом табличної моделі розвиток багатовимірного рішення закінчився [7]. Так, нові рішення, такі як Azure Analysis Services, не підтримують і не будуть підтримувати багатовимірну модель, на відміну від табличної, основною ідеєю якої є збереження інформації в пам’яті в колонковому форматі [8]. Така реалізація дозволяє пришвидшити динамічний підрахунок агрегацій, але потребує значної кількості оперативної пам’яті та потужності процесора.

2.2 Snowflake

Snowflake, на відміну від SSAS, не підпадає під традиційне розуміння OLAP, натомість він є реалізацією хмарного сховища даних розраховано на виконання складних аналітичних запитів, таким чином він може виступати одночасно, як сховище даних та OLAP сервер в загальній архітектурі систем онлайн-аналітичної обробки.

Архітектура Snowflake поділяється на три частини: сховище даних, обробка запитів, хмарні сервіси [9]. На рівні сховища даних реалізована

архітектура спільного диску (shared-disk), тобто дані зберігаються в єдиному сховищі, доступному всім вузлам системи. Для оптимізації сховища, при завантаженні даних в Snowflake, дані стискаються та форматуються в колонковий вигляд. Також дані зберігаються на рівні обробки запитів, реалізуючи архітектуру спільного нічого (shared-nothing). При виконанні запитів кожен вузол у кластері зберігає у собі частину даних, що використовується для масивної паралельної обробки, що у свою чергу забезпечує високу ефективність виконання складних аналітичних запитів. Останній, рівень хмарних сервісів, зв’язує всі компоненти Snowflake між собою задля обробки клієнтських запитів, забезпечуючи аутентифікацію, контроль доступу, розбір та оптимізація запитів, управління інфраструктурою та метаданими. Завдяки такому розділенню, кожен рівень може бути масштабований окремо від інших, що забезпечує більш ефективному використанню ресурсів.

Однією з важливих характеристик Snowflake також є його самокерованість, тобто користувачу достатньо мінімально налаштовувати систему, а далі оновлення, обслуговування та управління буде виконувати сама система. Також варто зазначити, що Snowflake є хмарним рішенням і його можна розгорнуту на трьох основних хмарних платформах Amazon Web Services (AWS), Google Cloud Platform та Microsoft Azure.

Для доступу до даних Snowflake підтримує низку варіантів підключення для різних цілей, але для візуалізації багатовимірних даних, аналогічно як і для SSAS, варто використовувати окремий застосунок, наприклад Tableau.

2.3 BigQuery

BigQuery, як і Snowflake, є реалізацією хмарного сховища даних, але на відміну він Snowflake, BigQuery реалізує безсерверну архітектуру для виконання запитів. Таким чином, платформа виділить ресурси саме тоді, коли вони будуть потрібні, що може допомогти користувачам не витрачати

гроші на надлишкові ресурси. Натомість, на відміну від Snowflake, BigQuery є продуктом компації Google і не може бути розгорнутий на інших хмарних платформах.

2.4 Tableau

На відміну він вище згаданих систем, Tableau є програмним рішенням для візуалізації даних, включаючи багатовимірні куби даних. За рейтингом представленим на сайті Forbes, Tableau займає друге місце серед інструментів для візуалізації даних, як найкращий інструмент для інтерактивних графіків [10]. Після підключення застосуноку до сховища даних, Tableau дозволяє створювати різноманітні інтерактивні інформаційні панелі (dashboards). Як джерело даних можуть виступати, як файли, так і велика кількість різноманітних систем, див. рисунок 2.4, наприклад Microsoft SQL Server, MongoDB BI Connector або Microsoft Analysis Services. В загальному, переважна більшість цих систем підтримують SQL запити.

Також є можливість інтерактивно об’єднувати різні джерела даних, для досягнення більшої повноти даних. Але варто зазначити, що операції над такими даними можуть призвести до сильного погіршення швидкості застосунку.

В загальному випадку, застосунок має бути підключеним до сховища даних, до якого будуть надсилатись динамічно сформовані запити, але не зважаючи на те, що Tableau є рішенням для візуалізації даних, він також має функціонал збереження та ітеративного оновлення даних в оптимізованому локальному колонковому сховищі. Головною перевагою викачування даних в Tableau є швидший відгук на зміни при розробці інтерактивних інформаційних панелей, натомість мінусами виступає час, який для цього потрібен і незахищеність викачаних даних.

Дане програмне рішення добре підходить для роботи з такими системами як Snowflake або з оптимізованими, наприклад індексами, реляційними базами даних. Хоч Tableau і позиціонує себе, як простий в використанні інструмент, що не потребує технічних знань, натомість для ефективної роботи є потреба в виборі та оптимізації джерела даних, що не є простою задачею для пересічної людини.

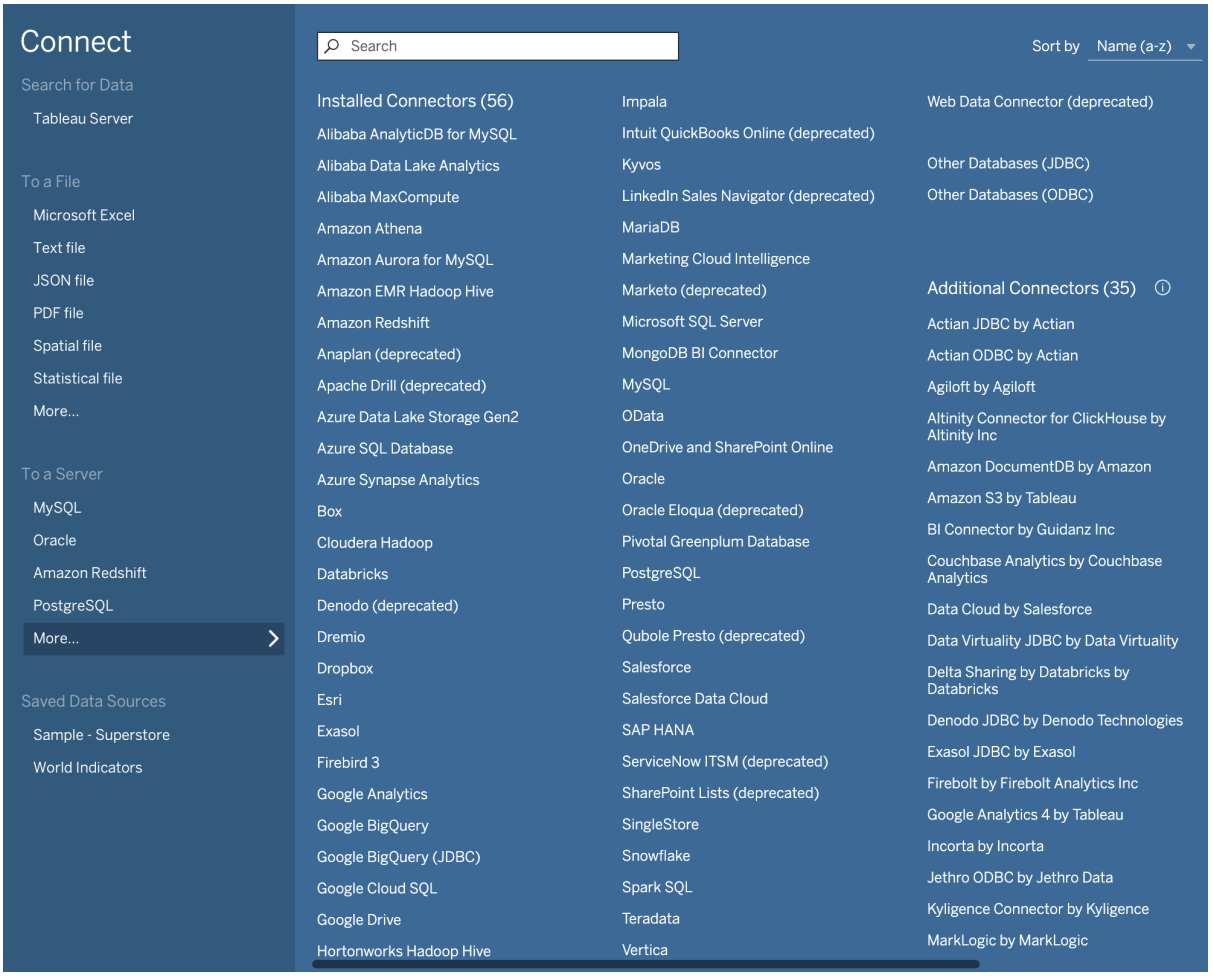


Рисунок 2.4. Додавання джерела даних в застосунку Tableau

2.5 Looker studio

Looker Studio є інструментом для візулізації даних дуже схожим на Tableau, так само є можливість підключення до різноманітних джерел даних та функціонал викачування даних, але на відміну від Tableau, він є

безкоштовним сервісом розробленим компанією Google, тому добре підійде користувачам, що вже зберігаюсь свої дані в BigQuery. Варто зазначити, що як і Tableau, Looker Studio хоч і може використовувати реляційні бази даних, як джерело даних, але ефективність такого рішення є низькою без додаткових оптимізацій джерела даних.

2.6 Висновки та постановка задачі

З наведених прикладів існуючих програмних рішень, можна зробити декілька висновків:

1. Використання традиційної архітектури OLAP, при якій будуються багатовимірні куби, на даний момент не є перспективним рішенням;
2. Новітні програмні рішення використовують колонкову базу даних, для забезпечення ефективності роботи аналітичних запитів;
3. Застосунки для візуалізації даних підтримують велику кількість різних джерел даних, але оптимізація таких джерел даних для роботи з аналітичними запитами лежить на користувачах;

На базі даних висновків, було прийняте рішення реалізувати систему для аналізу та візуалізації багатовимірних даних з інтегрованим підходом матеріалізації та використання агрегатів для оптимізації роботи аналітичних запитів.

Виконання практичного завдання включає такі пункти:

1. На базі існуючих програмних рішень сформулювати функціональні вимоги
2. Вибрати та обґрунтувати вибір інструментів розробки
3. Реалізувати застосунок відповідно до функціональних вимог
4. Перевірити ефективність застосунку
5. Зробити висновки

Розділ 3: Опис реалізації програмного продукту

3.1 Функціональні вимоги

Оскільки для багатовимірних даних основним варіантом візуалізації є зведена таблиця, було прийняте рішення реалізувати візуальну частину за допомогою неї. Застосунок має отримувати дані про потрібну для аналізу схему даних, дозволяти визначати виміри та міри, матеріалізувати агрегації, надавати зведену таблицю, динамічно створювати та виконувати запити до сховища даних зазуючить на наявних матеріалізованих агрегаціях. На рисунку 3.1 можна побачити візуальне представлення потоку застосунку, запит на генерацію виділений кольором та пунктиром, для зображення не обов’язковості даного кроку для користувача.

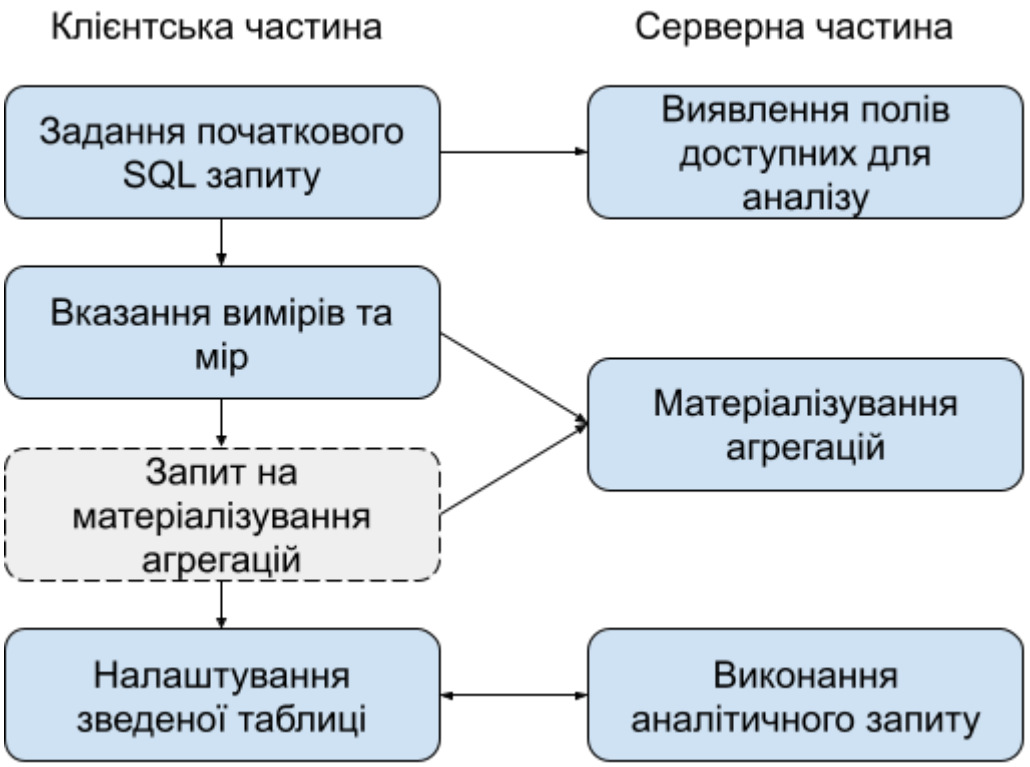


Рисунок 3.1. Схема потоку застосунку

3.2 Вибір та обґрунтування вибору інструментів розробки

3.2.1 React

React - це бібліотека для мови програмування JavaScript з відкритим вихідним кодом, за допомогою якої можна створювати односторінкові користувацькі інтерфейси. Він є найпопулярнішою бібліотекою у 2024 році по даним сайту Medium [11]. Основною перевагою використання таких бібліотек є модульність, тобто написаний один раз код можна використовувати повторноу багатьох місцях, а оскільки React має велику спільноту, є значна кількість вже готових бібліотек, які можна використовувати під час розробки. Так, головним критерієм вибору інструменту для клієнтської частини застосунку слугувала наявність і зручність бібліотеки для реалізації зведеної таблиці. В результаті аналізу наявних рішень, була вибрана бібліотека Syncfusion EJ2 React Pivotview, що реалізує зручний інтерфейс зведеної таблиці [12].

3.2.1 Node.js та Express.js

Для створення серверної частини застосунку головним критерієм у виборі інструментів розробки виступала швидкість розгортання та гнучкість технологій, оскільки від швидкості розгортання напряму залежить швидкість розробки, а гнучкість потрібна для формування динамічних SQL запитів. Таким чином було обрано Node.js в комбінації з Express.js, що задовольняють дані критерії [13].

Node.js - це середовище виконання JavaScript, що дозволяє використовувати JavaScript на для написання серверної частини застосунку.

Express.js - це фреймворк для застосунків Node.js, що надає можливість будувати RESTful API, включаючи маршрутизацію, обробку запитів, відповідей.

3.2.1 PostgreSQL

За основні критерії вибору сховища даних було вирішено взяти популярність бази даних та її відкритість. Таким чином було вибрано PostgreSQL - реляційну систему керування базами даних з відкритим вихідним кодом, що займає четверте місце по популярності за рейтингом сайту DB engines [14].

3.3 Реалізація застосунку

3.3.1 Клієнтська частина застосунку

Задачею клієнтської частини застосунку є надання користувачу можливості взаємодіяти з серверною частиною через візуальні компоненти. В даній роботі клієнтська частина складається з трьох сторінок:

1. **Query**. Дана сторінка відповідає за задання початкового SQL запиту, див. рисунок 3.3.1.1
2. **Data source.** На сторінці data source користувачу надається можливість інтерактивно розподілити поля, що були отримані з основного запиту на виміри та міри, див. рисунок 3.3.1.2. Додатково, на цій же сторінці, користувач може запросити матеріалізувати агрегації для пришвидшення запитів на наступній сторінці.
3. **Pivot.** Сторінка pivot відповідає за візуалізацію даних, тут користувачу надається можливість інтерактивно вибрати виміри та міри для зведеної таблиці та взаємодіяти з самою таблицею, див. рисунок 3.3.1.3. Також, на дану сторінку був доданий перемикач, змінюючиякий, користувач може обрати чи використовувати матеріалізовані агрегації при виклику наступного аналітично запиту.

QUERY DATA SOURCE PIVOT

```
select date_part('year',o.ordertimestamp) order_year,
       date_part('month',o.ordertimestamp) order_month,
           date_part('day',o.ordertimestamp) order_day,
           to_char(o.ordertimestamp, 'Dy') order_weekday,
           c.gender customer_gender,
           case
             when date_part('year',age(c.dateofbirth)) <= 20 then '0-20'
             when date_part('year',age(c.dateofbirth)) between 20 and 40 then '20-40'
             when date_part('year',age(c.dateofbirth)) between 40 and 60 then '40-60'
             when date_part('year',age(c.dateofbirth)) > 60 then '60+'
           end customer_age,
           d.city customer_city,
           pr.name product_name, pr.category product_category, pr.gender product_gender,
           l.name label_name, co.name color_name, s.size,
           p.amount, p.amount*p.price total_cost
  from webshop.order o
  join webshop.order_positions p on p.orderid=o.id
  join webshop.articles a on p.articleid=a.id
  join webshop.products pr on a.productid = pr.id
  join webshop.labels l on pr.labelid=l.id
  join webshop.colors co on a.colorid=co.id
  join webshop.sizes s on a.size=s.id
  join webshop.customer c on o.customer=c.id
  join webshop.address d on o.shippingaddressid=d.id
```

SAVE

Рисунок 3.3.1.1. Сторінка Query

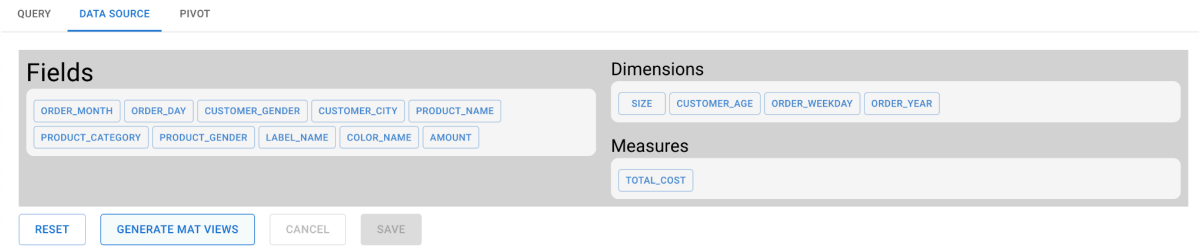


Рисунок 3.3.1.2. Сторінка Data source

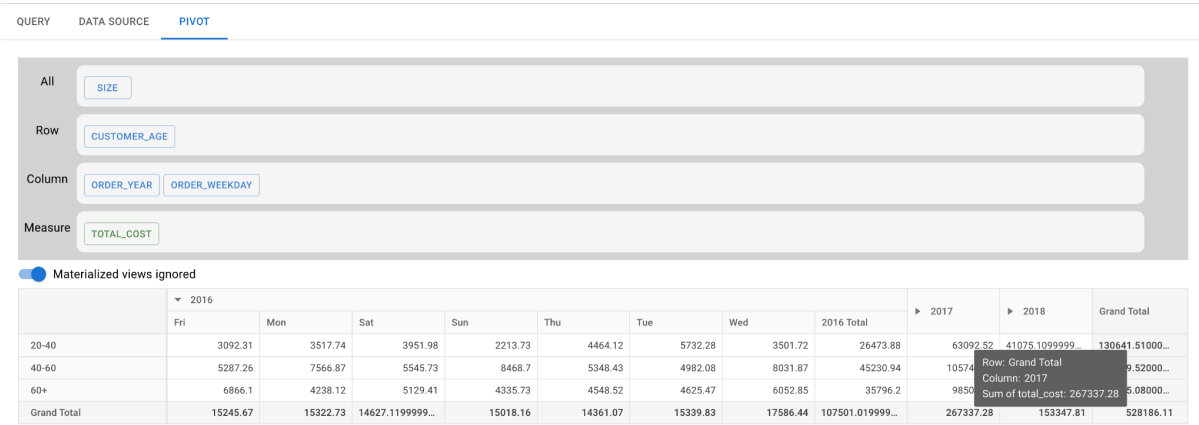


Рисунок 3.3.1.3. Сторінка Pivot

3.3.2 Серверна частина застосунку

3.3.2.1 Цілі серверного застосунку

- Імплементація API для використання UI застосунком;
- Збереження запитів для виконання багатовимірного аналізу;
- Аналіз запитів і виконання додаткової матеріалізації агрегацій задля покращення ефективності багатовимірних аналітичних запитів;
- Реалізація універсального API для виконання багатовимірних аналітичних запитів, що при необхідності будуть автоматично виконані використовуючи матеріалізовані агрегації;

3.3.2.2 Аналіз запиту і матеріалізація агрегатів

На вхід застосунку подається SQL запит, який є результатом об’єдання таблиць в схему зірка чи сніжинка. Результатом цього запиту є набір даних, що складається з багатьох стрічок і колонок. Кожна колонка являє собою або вимір, або міру. Після надання користувачу можливості визначити виміри та міри, ця інформація також передається на вхід серверному застосунку.

При роботі аналітики візуалізують дані на основі різноманітних вимірів, які невідомі наперед. Конкретні багатомірні аналітичні запити використовують тільки підмножину всіх вимірів, таким чином, згрупований по цих вимірах результат також включає набагато менше стрічок даних. Проте, щоб отримати правильний результат, потрібно виконати повний запит, який потім групувати по потрібним вимірам, що займає суттєвий час. Щоб оптимізувати кількість даних, що обробляються, а відповідно і час виконання запиту, застосунок вміє формувати і зберігати дані, агреговані по певних вимірах.

В даному випадку важливо забезпечити баланс між кількістю збережених агрегацій, тобто розміром сховища, що буде використаних для

збереження агрегацій, і можливістю ефективно виконувати різноманітні запити. Таким чином, якщо агрегат включає вимір з малим кардинальним числом, тобто кількість унікальних значень даного виміру в агрегаті є малим, то агрегат також буде ефективним для виконання запитів без цього виміру, при умові що всі інші виміри присутні в агрегаті.

Застосунок реалізує наступний алгоритм для побудови агрегатів:

- Формується список можливих наборів вимірів, групується і сортується по спаданню кількості вимірів.
- Для кожної групи наборів вимірів, ітерується по наборам.
- Для кожного набору вимірів, знаходиться «батьківський» агрегат, який уже сформований і включає всі необхідні виміри, а також має найменшу кількість записів. Якщо агрегований запит для поточного набору вимірів вертає суттєво меншу кількість записів (наприклад, в 10 разів) порівняно з «батьківським» агрегатом, то будується агрегат для поточного набору вимірів.
- Маленькі агрегати, наприклад менше 10 000 записів, не будуються, так як не дають суттєвого виграшу в часі виконання запитів.

3.3.2.3 API для виконання запитів багатомірного аналізу

Застосунок реалізує API для виконання запитів багатомірного аналізу. На вхід передається список вимірів і мір, а на вихід набір даних, агрегованих по переданих вимірах. При цьому застосунок автоматично підбирає найкращий збережений агрегат або ж будує запит на основі початкового. Найкращий агрегат підбирається з виконанням умови, що він включає всі необхідні виміри, і має найменшу кількість записів.

3.3.2.3 Реалізовані кінцеві точки API

- **POST /queries**

Створює новий запит. Приймає на вхід SQL запит. Повертає запит із доданим списком полів.

- **PUT /queries/{id}/fields**

  Оновлює поля запиту із вказанням, яке поле є виміром, а яке - мірою.

- **POST /queries/{id}/materialize**

  Аналізує запит і матеріалізує необхідні агрегати.

- **GET /queries/{id}**

  Повертає дані запиту, що включає ідентифікатор, SQL, список полів, список створених агрегатів.

- **POST /queries/{id}/data**

  На основі переданих вимірів і мір виконує запит. Реалізує гібридний підхід, автоматично використовуючи збережені агрегати або ж будує запит на основі вхідного запиту.

- **GET /queries**

  Повертає усі запити.

3.3.3 Тестування отриманого застосунку

Для тестування був підготовлений набір даних з 6 128 640 записів, сформований як запит з 9 таблиць, приєднаних за схемою «сніжинка».

В якості вимірів було вибрано 8 колонок: customer_age, order_year, order_month, order_day, order_weekday, product_category, product_gender, size. В якості мір(фактів): amount та total_cost.

```
select date_part('year',o.ordertimestamp) order_year,
       date_part('month',o.ordertimestamp) order_month,
       date_part('day',o.ordertimestamp) order_day,
       to_char(o.ordertimestamp, 'Dy') order_weekday,
       c.gender customer_gender,
       case
           when date_part('year',age(c.dateofbirth)) <= 20 then '0-20'
           when date_part('year',age(c.dateofbirth)) between 20 and 40 then '20-40'
           when date_part('year',age(c.dateofbirth)) between 40 and 60 then '40-60'
           when date_part('year',age(c.dateofbirth)) > 60 then '60+'
           end customer_age,
       d.city customer_city,
       pr.name product_name, pr.category product_category, pr.gender product_gender,
       l.name label_name, co.name color_name, s.size,
       p.amount, p.amount*p.price total_cost
from webshop.order o
         join webshop.order_positions p on p.orderid=o.id
         join webshop.articles a on p.articleid=a.id
         join webshop.products pr on a.productid = pr.id
         join webshop.labels l on pr.labelid=l.id
         join webshop.colors co on a.colorid=co.id
         join webshop.sizes s on a.size=s.id
         join webshop.customer c on o.customer=c.id
         join webshop.address d on o.shippingaddressid=d.id
```

Рисунок 3.3.3.1. SQL запит для тестування

Для вказаних вимірів і мір було автоматично згенеровано необхідні агрегати.

```
Total row count: 1532160
Total cardinality: 49896 , effect: 3070.71
Excluded: ; cardinality: 1
Dimensions: customer_age,customer_gender,order_month,order_weekday,order_year,product_category; count: 45228(-97.05%); took: 6680ms
          !!! materialize data to table mt_olap_8c11848987f4c318532a07a4cd08b2d5d39a90fa
Excluded: customer_age; cardinality: 3
Dimensions: customer_gender,order_month,order_weekday,order_year,product_category; count: 15138(-66.53%); took: 29ms
          !!! materialize data to table mt_olap_a31fdd4cd731f78585fc98bc33f86677c5d50e32
Excluded: customer_gender; cardinality: 2
Dimensions: customer_age,order_month,order_weekday,order_year,product_category; count: 22702(-49.81%); took: 29ms
Excluded: order_month; cardinality: 12
Dimensions: customer_age,customer_gender,order_weekday,order_year,product_category; count: 4158(-90.81%); took: 26ms
          !!! materialize data to table mt_olap_71ea4daf99a3b9ceebf5701c8c763c4ab3357040
Excluded: order_weekday; cardinality: 7
Dimensions: customer_age,customer_gender,order_month,order_year,product_category; count: 6531(-85.56%); took: 27ms
          !!! materialize data to table mt_olap_291d42bf0c8ddc319cd1ded828a8dc24ed1a0d54
Excluded: order_year; cardinality: 11
Dimensions: customer_age,customer_gender,order_month,order_weekday,product_category; count: 4536(-89.97%); took: 27ms
          !!! materialize data to table mt_olap_797ad70ee776ca21aca38e7e50dba0319b24874e
Excluded: product_category; cardinality: 9
Dimensions: customer_age,customer_gender,order_month,order_weekday,order_year; count: 5046(-88.84%); took: 28ms
          !!! materialize data to table mt_olap_9a4b9ccb1beba36a07c382a311796763ab561dfd
Excluded: customer_gender; cardinality: 2
Dimensions: order_month,order_weekday,order_year,product_category; count: 7569(-50.00%); took: 15ms
```

Рисунок 3.3.3.2. Матеріалізування агрегацій

Тестування проводилось у двох режимах:

1. Без згенерованих агрегатів.
2. Зі згенерованими агрегатами.

При цьому реєструвався час виконання запитів і який агрегат використовувався (список вимірів, для яких побудований агрегат).

Результати наведені в таблиці нижче.

Запит (виміри)	Кількість записів	Час без агрегатів	Час з агрегатами	Використані агрегати
customer_age order_year order_month order_day order_weekday product_category	585412	15882мс	1476мс	customer_age order_year order_month order_day order_weekday product_category

product_gender size				product_gender size
customer_age order_year order_month product_category size	16306	11182мс	147мс	customer_age order_year order_month product_category product_gender size
order_year order_month product_category	1089	3874мс	120мс	customer_age order_year order_month product_category product_gender
order_year order_month	121	3149мс	97мс	customer_age order_year order_month order_day product_gender size
size	5	2249mc	73mc	customer_age order_weekday product_category product_gender size
customer_age product_category	27	4987мс	71мс	customer_age order_weekday product_category product_gender size

customer_age order_year order_month order_day size	54788	11751мс	189мс	customer_age order_year order_month order_day order_weekday product_gender size
order_weekday product_gender	14	2359мс	66мс	customer_age order_weekday product_category product_gender size

Висновок: побудовані агрегати надають суттєве покращення у часі виконання запитів, порівняно із запитами на основі початкового запиту. Побудовані агрегати перевикористовуються для різних запитів.

Висновки та рекомендації

В ході даної роботи було розглянуто теоретичні відомості зберігання, аналізу та візуалізації багатовимірних даних, проведений огляд наявних програмних рішень, як спеціалізованих для багатовимірних кубів, так і направлених на роботу з різними форматами даними. На базі описаних даних, були визначенні недоліки наведених систем та запропонований підхід генерації та використання матеріалізованих агрегацій для забезпечення більшої ефективності виконання аналітичних запитів.

Розроблено застосунок, що дозволяє виконувати багатовимірний аналіз даних з реалізацією запропонованого підходу. Описані частини та алгоритм роботи застосунку. Проведене тестування застосунку для перевірки коректності його роботи, яке довело ефективність запропонованого підходу.

Можливі покращення застосунку:

- Щоб оптимізувати час пошуку агрегатів при великій кількості вимірів і великому наборі даних можна враховувати селективність вимірів. Проте при намаганні це реалізувати, виявилось, що селективність одного виміру тут недостатня. Можна аналізувати селективність кількох вимірів порівняно з селективністю кожного виміру окремо для виявлення залежності між вимірами.
- Замість отримання SQL на вхід можна надавати можливість формувати схему зірочка чи сніжинка. Не тільки для зручності користувачів, але також щоб отримати інформацію про потенційні залежності між вимірами.
- На даний момент застосунок не підтримує фільтри по вимірам чи мірам, необхідно додати таку функціональність.

Список літератури

1. Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025. [Електронний ресурс] – Режим доступу: https://www.statista.com/statistics/871513/worldwide-data-created/
2. Online analytical processing [Електронний ресурс] – Режим доступу: https://en.m.wikipedia.org/wiki/Online_analytical_processing
3. Online analytical processing (OLAP) overview [Електронний ресурс] – Режим доступу: https://www.ibm.com/docs/bg/filenet-p8-platform/5.5.x?topic=reports-online-analytical-processing-olap-overview
4. What is OLAP (Online Analytical Processing)? [Електронний ресурс] – Режим доступу: https://aws.amazon.com/what-is/olap/
5. Columnar Database [Електронний ресурс] – Режим доступу: https://www.sciencedirect.com/topics/computer-science/columnar-database
6. An Introduction to Columnar Databases [Електронний ресурс] – Режим доступу: https://www.odbms.org/2023/04/an-introduction-to-columnar-databases/
7. Tabular vs Multidimensional models for SQL Server Analysis Services [Електронний ресурс] – Режим доступу: https://www.mssqltips.com/sqlservertip/4154/tabular-vs-multidimensional-models-for-sql-server-analysis-services/
8. Comparing tabular and multidimensional solutions [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/analysis-services/comparing-tabular-and-multidimensional-solutions-ssas?view=asallproducts-allversions

9. Snowflake - Key Concepts & Architecture [Електронний ресурс] – Режим доступу: https://docs.snowflake.com/en/user-guide/intro-key-concepts
10. The Best Data Visualization Tools Of 2024 [Електронний ресурс] – Режим доступу: https://www.forbes.com/advisor/business/software/best-data-visualization-tools/
11. Top 10 Most Popular JavaScript Frameworks to Choose in 2024 [Електронний ресурс] – Режим доступу: https://medium.com/@evincedevelop/top-10-most-popular-javascript-frameworks-to-choose-in-2024-269453cdaf35
12. EJ2 React Pivotview Getting Started [Електронний ресурс] – Режим доступу: https://ej2.syncfusion.com/react/documentation/pivotview/getting-started
13. Top 5 NodeJS pros and cons [Електронний ресурс] – Режим доступу: https://anywhere.epam.com/en/blog/node-js-pros-and-cons
14. DB-Engines Ranking [Електронний ресурс] – Режим доступу: https://db-engines.com/en/ranking
15.

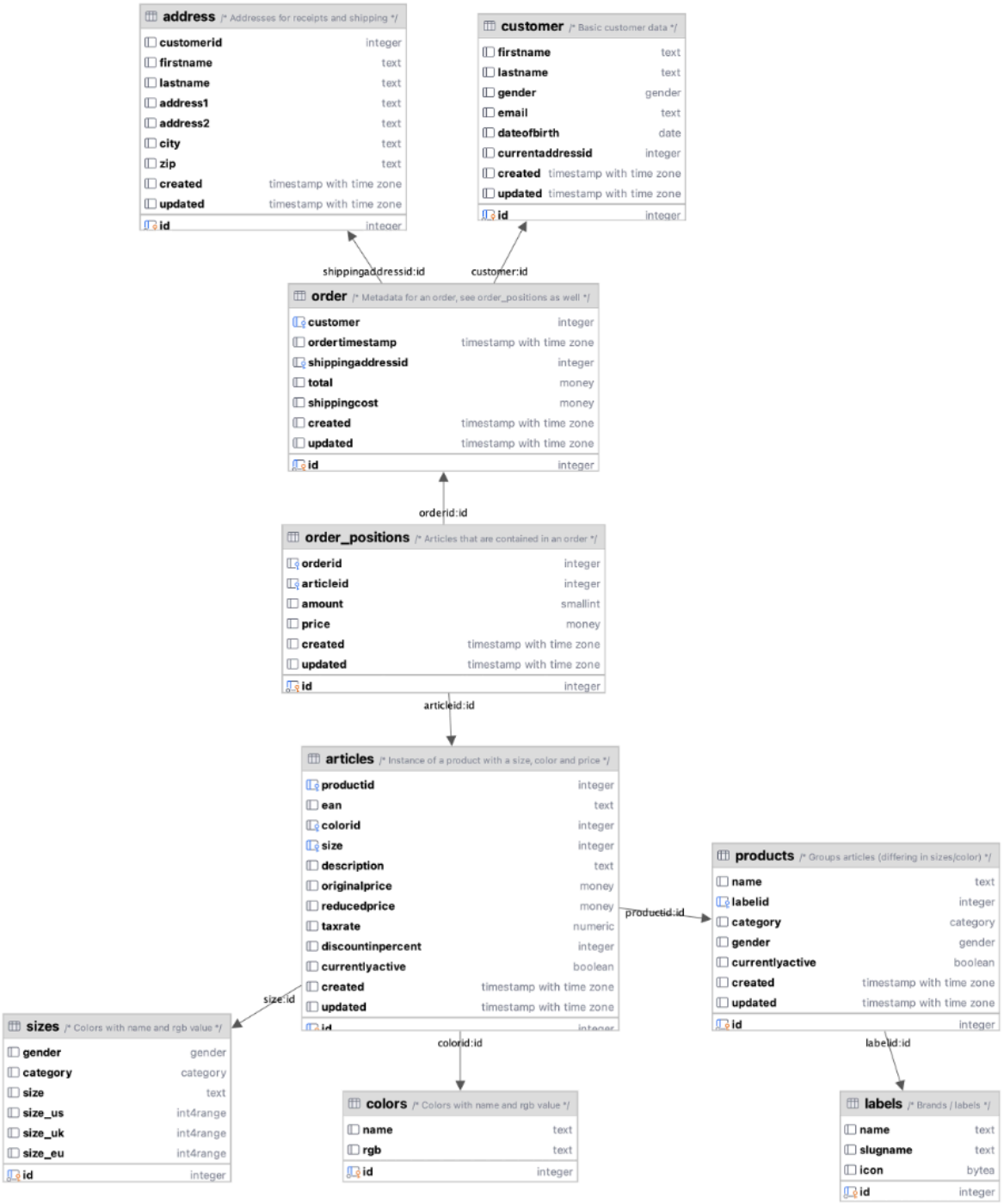
Глосарій

Поняття	Визначення
OLAP	Online Analytical Processing, аналітична обробка у реальному часі
MOLAP	Multidimensional Online Analytical Processing, багатовимірна онлайн аналітична обробка
ROLAP	Relational Online Analytical Processing, реляційна онлайн аналітична обробка
HOLAP	Hybrid Online Analytical Process, гібридна онлайн аналітична обробка
SSAS	Microsoft SQL Server Analytical Services, інструмент аналітичної обробки та аналізу даних у Microsoft SQL Server.
UI	User Interface, інтерфейс користувача
Агрегація	процес обробки великих обсягів інформації з бази даних і організації її в більш зручний для використання формат
Куб даних	Багатовимірний масив даних
Матеріалізація агрегацій	Збереження підрахованих значень в таблицю

Додатки

Додаток А

Використана схема даних



Додаток Б

OpenAPI специфікація серверного додатку

```
openapi: 3.0.0
info:
  title: Multidimentional Query Materializer
  version: 0.1.0
  description: This is a prototype of Multidimentional Query Materializer and Executor
servers:
  - url: 'http://localhost:3000'
paths:
  /queries:
    post:
      summary: Creates a new query
      tags:
        - Queries
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Query'
      responses:
        '200':
          description: The created query.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Query'
  '/queries/{id}':
    get:
      summary: Get query
      tags:
        - Queries
      parameters:
        - in: path
          name: id
          schema:
            type: string
            required: true
            description: Query id
      responses:
        '200':
          description: The created query.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Query'
```

```
'/queries/{id}/fields':
  put:
    summary: Update query fields
    tags:
      - Queries
    parameters:
      - in: path
        name: id
        schema:
          type: string
          required: true
          description: Query id
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: array
            items:
              $ref: '#/components/schemas/Field'
            example:
              - field_name: product_category
                use: dimension
              - field_name: color_name
                use: dimension
              - field_name: product_name
                use: dimension
              - field_name: size
                use: dimension
              - field_name: amount
                use: measure
              - field_name: total_cost
                use: measure
    responses:
      '200':
        description: The created query.
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Query'
'/queries/{id}/materialize':
  post:
    summary: Update query fields
    tags:
      - Queries
    parameters:
      - in: path
        name: id
```

```
        schema:
          type: string
          required: true
          description: Query id
    responses:
      '200':
        description: The created query.
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Query'
'/queries/{id}/data':
  post:
    summary: Query data
    tags:
      - Queries
    parameters:
      - in: path
        name: id
        schema:
          type: string
          required: true
          description: Query id
      - in: query
        name: ignore_materialized
        schema:
          type: boolean
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            required:
              - dimensions
              - measures
            properties:
              dimensions:
                type: array
                items:
                  type: string
                example:
                  - product_category
                  - color_name
                  - size
              measures:
                type: array
                items:
```

```
                type: string
              example:
                - amount
                - total_cost
      responses:
        '200':
          description: The created query.
          content:
            application/json:
              schema:
                type: array
                items:
                  type: array
              example:
                - - product_category
                  - color_name
                  - size
                  - amount
                  - total_cost
                - - Apparel
                  - MEDIUMORCHID
                  - S
                  - '768'
                  - '$99,072.00'
                - - Footwear
                  - PINK
                  - S
                  - '256'
                  - '$71,674.88'
components:
  schemas:
    Field:
      type: object
      required:
        - field_name
      properties:
        field_name:
          type: string
          example: product_category
        use:
          type: string
          enum:
            - dimension
            - measure
          example: dimension
    Query:
      type: object
      required:
```

```
      - id
      - sql
    properties:
      id:
        type: string
        description: Unique id of query.
      sql:
        type: string
        description: SQL query.
      fields:
        type: array
        items:
          $ref: '#/components/schemas/Field'
    example:
      id: webshop_sales
      sql: 'select * from webshop.sales'
  Data:
    type: array
tags:
  - name: Queries
    description: API to manage your queries.
```