Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп'ютерних наук і технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеню вищої освіти – магістр
за освітньо-професійною програмою
«Комп'ютерні науки»

зі спеціальності
122 – Комп'ютерні науки

тема роботи:
«***Моделювання системи аналізу даних на основі технології OLAP за допомогою програмного забезпечення з відкритим вихідним кодом***»

Виконав студент гр. КН-24м. _____ Шаблій Я. Е.

Керівник _____ Гриценко А. М.

Нормоконтроль _____ Маринич І. А.

Завідувач кафедри _____ Рубан С. А.

Кривий Ріг – 2025

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет: *інформаційних технологій*
Кафедра: *автоматизації, комп'ютерних наук і технологій*
Ступінь вищої освіти: *Магістр*
Спеціальність: *122 – Комп'ютерні науки*

Затверджую
Зав. кафедри: *к.т.н. Рубан С.А.*
_____
« *15* » *травня* 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу магістра

студентові групи КН-24м *Шаблію Ярославу Едуардовичу*

1. Тема кваліфікаційної роботи: *«Моделювання системи аналізу даних на основі технології OLAP за допомогою програмного забезпечення з відкритим вихідним кодом»*

затверджено наказом по університету № *257с* від *13.05.2025 р.*

2. Термін здачі кваліфікаційної роботи: *01.12.2025 р.*

3. Склад кваліфікаційної роботи: *Пояснювальна записка обсягом 80с., додатки, презентація у Microsoft PowerPoint (15 слайдів) в електронному та друкованому вигляді*

4. Консультанти кваліфікаційної роботи:
Розділ 1-3 доц. Гриценко А. М.
_____
Нормоконтроль доц. Маринич І. А.

5. Календарний план:

№	Етапи роботи	Термін виконання
1	*Вступ*	*10.02.25*
2	*Розділ 1*	*15.04.25*
3	*Розділ 2*	*18.07.25*
4	*Розділ 3*	*19.11.25*
5	*Висновки*	*20.11.25*
6	*Оформлення кваліфікаційної роботи*	*25.11.25*
7	*Підготовка презентації та графічного матеріалу*	*28.11.25*
8	*Підготовка доповіді до захисту*	*01.12.25*

6. Дата видачі завдання: *24.12.2024р.*

Керівник _____ /Гриценко А. М./

7. Запевнення: *Я, Іванов Іван Іванович, запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.*
Із чинним Положенням про академічну доброчесність Криворізького національного університету ознайомлений.
Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі умисних порушень робота не допускається до захисту або оцінюється незадовільно.

Здобувач _____ / Шаблій Я. Е./

АНОТАЦІЯ

Шаблій Я. Е. Моделювання системи аналізу даних на основі технології OLAP за допомогою програмного забезпечення з відкритим вихідним кодом : кваліфікаційна робота магістра : 122 – Комп'ютерні науки. Кривий Ріг. Криворізький національний університет, 2025. 75 с.

Протягом останніх кількох років у сфері аналітичної обробки даних спостерігається тенденція розглядати технологію OLAP як застарілу та таку, що поступово відходить у минуле. Однак у контексті цього дослідження передбачається, що за допомогою сучасного програмного забезпечення можна ефективно здійснювати аналіз даних на основі технології багатовимірного аналізу OLAP.

Метою роботи є комплексне порівняння ефективності OLAP-систем, що підтримують технологію багатовимірного аналізу даних OLAP-продуктів з відкритим вихідним кодом.

У першому розділі проаналізовано роботи різних авторів і визначено, що аналітична робота передбачає виконання запитів для отримання змістовної інформації, і це зумовлює популярність технології OLAP, яка дозволяє забезпечити максимальну швидкість обробки запитів.

Другий розділ присвячений оптимізації процесу роботи з даними компанії з метою зробити процес обробки даних більш простим і стандартизованим. Продукт Atoti In-memory OLAP є кращим варіантом, оскільки він відповідає вимогам до продуктивності, гнучкої моделі даних і відкритого вихідного коду.

У третьому розділі здійснено реалізацію системи OLAP, розробку методики оцінки ефективності OLAP-систем, оцінку ефективності та функціональності розробленої системи з точки зору задоволення вимог бізнесу та можливості проведення аналітичних досліджень

Ключові слова:

АНАЛІЗ ДАНИХ, БГАТОВИМІРНИЙ АНАЛІЗ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ З ВІДКРИТИМ КОДОМ, ATOTI, OLAP, PYTHON

ANNOTATION

Shablii Ya. E. Modeling of a Data Analysis System Based on OLAP Technology Using Open-Source Software : Master's Qualification Thesis : 122 – Computer Science. Kryvyi Rih. Kryvyi Rih National University, 2025. 75 p.

Over the past few years, there has been a growing tendency in the field of data analytics to consider OLAP technology as outdated and gradually becoming obsolete. However, within the context of this study, it is assumed that with the help of modern software, data analysis based on multidimensional OLAP technology can still be effectively implemented.

The relevance of the research is determined by the increasing need for a comprehensive evaluation of open-source OLAP systems for data analysis, especially under conditions of cost constraints on the use of foreign software.

The aim of the study is to provide a comprehensive comparison of the efficiency of OLAP systems that support multidimensional data analysis technology among open-source OLAP products.

The first chapter analyzes the works of various authors and establishes that analytical work involves executing queries to obtain meaningful information, which explains the popularity of OLAP technology that enables maximum query processing speed.

The second chapter focuses on optimizing the company’s data processing workflow to make the process simpler and more standardized. The Atoti In-memory OLAP product is identified as the best option, as it meets the requirements for performance, a flexible data model, and open-source accessibility.

In the third chapter, the implementation of the OLAP system is carried out, along with the development of a methodology for evaluating the efficiency of OLAP systems, and an assessment of the effectiveness and functionality of the developed system in terms of meeting business requirements and enabling analytical research.

DATA ANALYSIS, MULTIDIMENSIONAL ANALYSIS, OPEN-SOURCE SOFTWARE, ATOTI, OLAP, PYTHON

ЗМІСТ

ВСТУП

Сьогодні компанії накопичують велику кількість даних. Ці дані необхідно аналізувати, щоб не відставати від конкурентів в ефективності управлінських рішень. Аналіз даних дозволяє ухвалювати ефективні рішення, оскільки вони базуються на інформації, отриманій у минулому. Це дає змогу зменшити непередбачуваність у майбутньому.

Однак накопичення великої кількості даних призводить до того, що для їх обробки необхідно залучати більше ресурсів. Щоб користь від накопичення даних перевищувала витрати на їх обслуговування, були розроблені інформаційно-аналітичні системи, які отримали назву «Системи підтримки ухвалення рішень». Архітектура такої системи загалом включає транзакційну базу даних, «підсистему зберігання та підсистему аналізу». Система аналізу в класичній системі підтримки ухвалення рішень будується на основі технології багатовимірного аналізу даних OLAP (Online Analytical Processing).

Протягом останніх кількох років у сфері аналітичної обробки даних спостерігається тенденція розглядати технологію OLAP як застарілу та таку, що поступово відходить у минуле. Однак у контексті цього дослідження передбачається, що за допомогою сучасного програмного забезпечення можна ефективно здійснювати аналіз даних на основі технології багатовимірного аналізу OLAP.

У нинішній ситуації компанії, які використовують системи аналізу даних на основі технології багатовимірного аналізу OLAP, замислюються над способами проведення багатовимірного аналізу без застосування популярних іноземних платних рішень. Оскільки сервіси для аналізу даних на основі технології багатовимірного аналізу, представлені на вітчизняному ринку, часто є дорогими, рішення з відкритим вихідним кодом (open-source) могли б стати альтернативою для багатьох компаній.

Актуальність теми. Актуальність дослідження визначається зростанням потреби у комплексній оцінці OLAP-систем для аналізу даних з відкритим

вихідним кодом в умовах вартісних обмежень на використання іноземного програмного забезпечення.

Актуальність проявляється у відсутності розроблених методик для всебічної оцінки продуктивності та бізнес-ефективності OLAP-систем, що враховують оцінку стійкості системи, а також у необхідності проведення порівняння ефективності та продуктивності інструментів аналізу даних на основі технології багатовимірного аналізу даних OLAP з відкритим вихідним кодом. В умовах нових ринкових реалій пошук оптимального технологічного рішення стає актуальною проблемою для багатьох підприємств.

Мета та завдання дослідження. Метою роботи є комплексне порівняння ефективності OLAP-систем, що підтримують технологію багатовимірного аналізу даних OLAP-продуктів з відкритим вихідним кодом.

Для досягнення мети роботи необхідно вирішити такі завдання:

- розглянути теоретичні аспекти систем інтерактивної аналітичної обробки даних;
- здійснити огляд існуючих методів і інструментів багатовимірного аналізу даних OLAP, включаючи як комерційні, так і відкриті рішення, з оцінкою їх функціональності та продуктивності;
- дослідити актуальність багатовимірного аналізу даних OLAP;
- застосувати технологію аналізу діяльності підприємства для виявлення проблемних процесів, що потребують аналітичного втручання;
- виконати моделювання та реалізацію OLAP-системи;
- провести оцінку ефективності та функціональності розробленої системи з точки зору задоволення вимог бізнесу та можливостей проведення аналітичних досліджень.

Методи дослідження: в роботі застосовані аналітичні методи та метод експертної оцінки, метод моделювання.

РОЗДІЛ 1
ТЕОРЕТИЧНІ АСПЕКТИ СИСТЕМ ОНЛАЙН-АНАЛІТИЧНОЇ ОБРОБКИ ДАНИХ

1.1 Форми зберігання даних

Глобальне зростання обсягу даних призвело до того, що компанії зіткнулися з необхідністю їх зберігати таким чином, щоб у будь-який момент часу можна було здійснити запит і наочно представити інформацію, якщо це знадобиться.

Транзакційні бази даних (OLTP) використовуються для зберігання транзакцій у реляційній формі. Реляційна форма передбачає наявність у даних реляцій (відношень, зв'язків). Такий спосіб зберігання інтуїтивно зрозумілий користувачам. Відношення дозволяють чітко визначати, як суб'єкти аналізу пов'язані між собою. Наочна таблична форма допомагає швидко перевірити правильність або помилковість виконаних дій.

Запити в таких базах даних оформлюються мовою SQL (мова структурованих запитів), зручність і простота якої також сприяють популярності реляційних баз даних [5].

На рисунку 1.1 наведено процес обробки запиту в реляційній базі даних.

Рисунок 1.1 – Процес обробки запиту до реляційної бази даних

Коли дані запитуються з бази, запит спочатку аналізується (перевірка відповідність синтаксичним правилам SQL). Після цього база даних знаходить спосіб отримати дані найбільш ефективно і складає план виконання, згідно з яким обробляє дані і повертає їх користувачу в тому вигляді, в якому він запитував засобами SQL.

У транзакційній базі даних дані зберігаються таким чином, щоб відповідати концепції нормалізації. Концепція нормалізації у межах реляційної моделі даних було сформульовано Еге. Ф. Коддом: «У нормалізованому відношенні все неключові атрибути функціонально залежить від ключа отношения» [9]. Це означає, що дані, що описують характеристики основних даних, в основній таблиці повинні бути закодовані як деякий унікальний для даного атрибута ключа.

На рисунку 1.2 представлені таблиці, що містять один і той самий набір інформації, зберігання якої реалізовано в різних формах: нормалізованій і ненормалізованій. Як можна побачити з рисунка 1.2, в основній таблиці нормалізованої форми зберігання немає жодної змістовної інформації. Вона містить у собі зовнішні ключі, і тільки їхнє поєднання дозволяє визначити, який саме набір виробів кодує кожен окремий рядок у таблиці.

Нормалізована форма не є наочною та інтуїтивно зрозумілою, оскільки, щоб отримати інформацію, яка має певний зміст для користувача, йому доведеться за ключами шукати у інших таблицях змістовні дані, які вони кодують. У прикладі на рисунку 1.2 за ключами ми можемо визначити колір і матеріал виробу. У ненормалізованій формі це наочно представлено в одній таблиці.

Незважаючи на те, що на перший погляд нормалізована форма зберігання незручна для користувача, це дозволяє не зберігати дані в базі в надлишковому вигляді, оскільки ключ має малу розмірність і потребує мало ресурсів для зберігання.

До плюсів нормальної форми зберігання також належить те, що така форма дозволяє зменшити кількість помилок з урахуванням людського чинника.

Основна таблиця		
РК	Колір	Матеріал
1	2	1
2	1	2
3	2	2

Основна таблиця		
РК	Колір	Матеріал
1	Червоний	Вовна
2	Чорний	Поліамід
3	Червоний	Поліамід

Колір	
РК	Опис
1	Чорний
2	Червоний
3	Білий

Матеріал	
РК	Опис
1	Вовна
2	Поліамід
3	Бавовна

Рисунок 1.2 – Різниця у структурі зберігання у нормальній та ненормалізованій формі

Користувачеві, який додає дані до такої бази, дозволено внести дані, які обмежені розробленою раніше структурою. Тобто набір можливих значень обмежений списком зумовлених ключів. У такому разі помилка при введенні набагато менш ймовірна, тому це дозволяє зберегти цілісність даних.

При цьому така складна структура зберігання, а саме розподіл даних за деякою кількістю таблиць, позначається негативно на швидкості виконання запитів.

Це зумовлено тим, що для виконання запиту на отримання будь-якої змістовної інформації, що містить не ключі, а текстові характеристики, необхідно здійснити об'єднання пов'язаних таблиць.

Операція об'єднання у реляційної базі даних – це дія, якого аналітики намагаються уникати, оскільки є однією з найбільш витратних по ресурсам.

Очевидно, що реляційний формат зберігання не пристосований для аналізу даних, тому що такі бази є занадто повільними з точки зору отримання інформації. При цьому вони є найкращим вибором для зберігання даних, які постійно додаються або оновлюються, оскільки операції запису та зміни в цих

таблицях не вимагають багато ресурсів.

1.2 Технологія OLAP

Вперше термін «OLAP» було введено Е. Ф. Коддом. За визначенням Кодда: «OLAP – це технологія оперативної аналітичної обробки даних, що полягає у підготовці агрегованої інформації на основі великих масивів даних».

Як видно з цього визначення, OLAP дозволяє зберігати та запитувати попередньо розраховані дані. Це вирішує проблему реляційної форми зберігання, яка була зазначена вище: низька швидкість запитів за наявності великої кількості відношень у базі даних.

Технологія OLAP має базову вимогу щодо зберігання даних у багатовимірній формі, у вигляді так званого куба даних. Приклад такої кубічної структури зберігання представлений на рисунку 1.3.

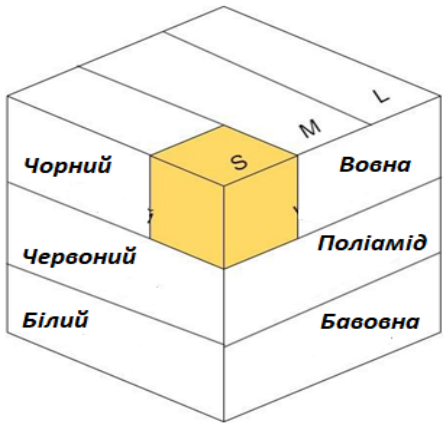

Рисунок 1.3 – Приклад багатовимірного представлення даних

У кубі дані агрегуються за всіма розрізами, які задаються під час його побудови. Наприклад, у кубі на рисунку були визначені виміри: розмір, матеріал і колір. Для будь-якого поєднання цих вимірів у кубі зберігаються попередньо розраховані агреговані кількісні дані. Форма агрегації даних також може бути задана користувачем. Так, можна отримати дані про середню ціну виробів певного кольору та розміру, про суму витрат на вироби з певного матеріалу і кольору тощо.

На рисунку жовтим виділено перетин вимірів, для якого можна швидко отримати агреговані кількісні дані про вироби з вовни чорного кольору та розміру «S». Наприклад, можна порівняти середній прибуток від таких виробів із прибутком від виробів з іншими параметрами, і цей аналіз дозволить ухвалити обґрунтоване рішення щодо того, які вироби виробляти в майбутньому.

При цьому швидкість доступу до даних буде значно вищою, ніж при отриманні цих даних із реляційної форми зберігання. Куб зазвичай потребує рідкісного оновлення, що сприяє тому, що аналітику не доведеться стикатися з перевантаженням сервера бази даних.

Висока швидкість запитів і відсутність перебоїв у роботі сервера бази даних пояснюють популярність багатовимірних кубів даних для компаній, які працюють із великими обсягами даних.

При цьому технологія OLAP має низку слабких сторін [4]:

- Кожне оновлення даних у кубі призводить до необхідності його оновлення. Оскільки перебудова куба даних — це часто дуже ресурсомістке завдання, це може стати значним недоліком. Як буде показано далі в цій роботі, деякі постачальники програмного забезпечення вбудували у свої продукти функції, що дозволяють не оновлювати куб повністю при внесенні певних змін у його структуру.
- Для проведення аналізу зазвичай необхідно виділяти загальні тенденції та тренди. Водночас точність даних може постраждати, оскільки дані спрощуються. При некоректному використанні технології таке спрощення може призвести до ситуації, коли деякі важливі характеристики даних будуть представлені невірно.

1.3 Сховище даних

Концепція сховища даних виникла в середині 1980-х років, коли з’явилася необхідність аналізувати великі обсяги інформації та готувати управлінську

звітність.

На рисунку 1.4 показана типова архітектура сховища даних. Типове сховище включає в себе OLAP-сервер, як це зображено на рисунку 3 [44].

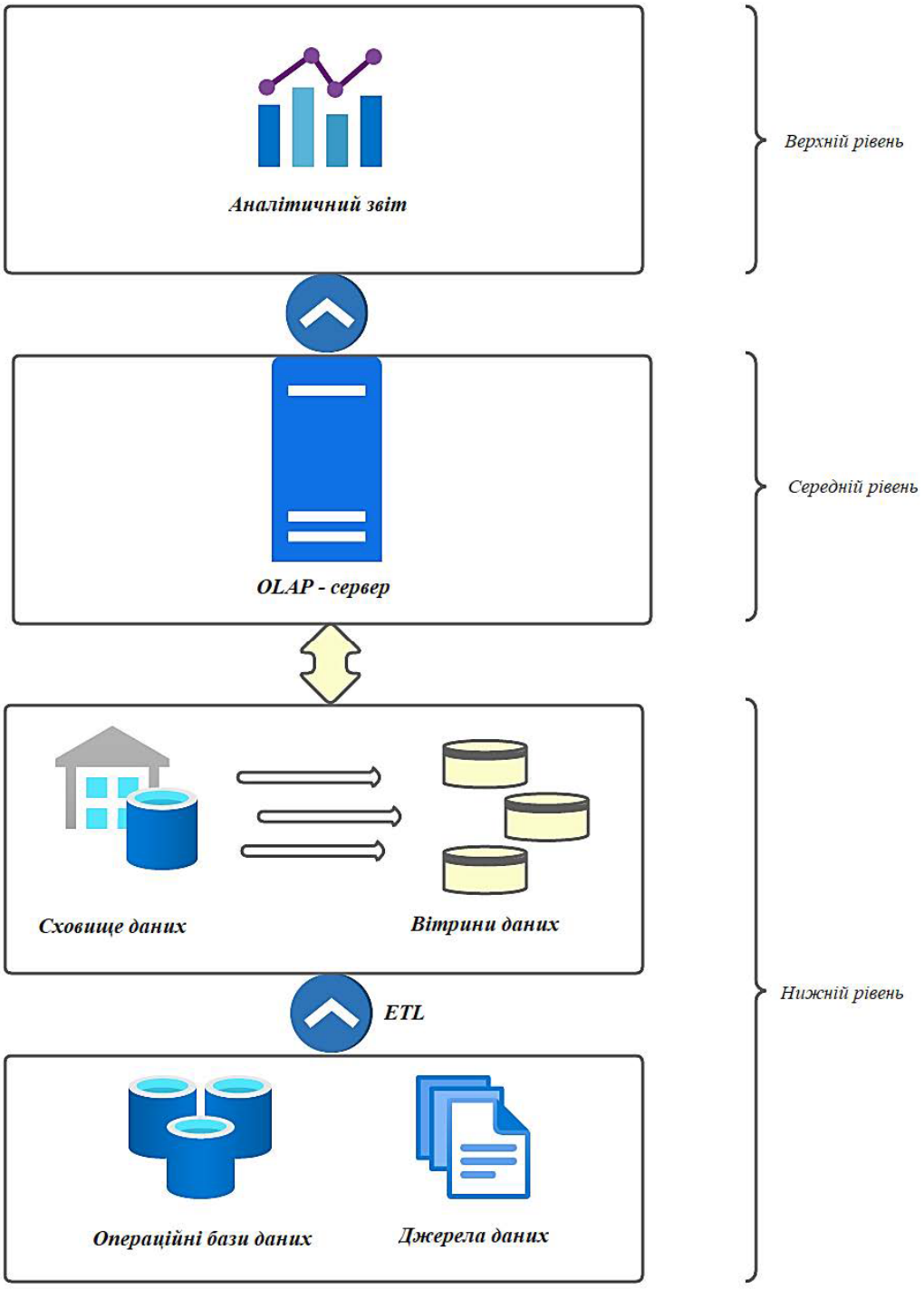

Рисунок 1.4 – Архітектура сховища даних із включенням OLAP-сервера

Поняття сховища даних було вперше визначене Б. Інмоном: «Сховище даних - це предметно-орієнтована, інтегрована, незмінна та така, що підтримує

хронологію, електронна колекція даних для забезпечення процесу прийняття рішень» [49].

Згідно з визначенням Р. Кімбалла: Сховище даних - це система, яка отримує, очищає, зіставляє та передає вихідні дані у сховище вимірювальних даних, а потім підтримує та реалізує запити й аналіз із метою прийняття рішень» [53].

1.4 Сховище даних із використанням OLAP-сервера

При розробці сховища даних важливим завданням стає визначення форми зберігання в цьому сховищі.

Згідно з [10]: «Для моделювання сховища даних були введені концепції таблиці фактів і таблиці вимірювань. У таблиці фактів зберігаються накопичені кількісні величини, тоді як у таблиці вимірювань - якісні, описові дані. Ключовим правилом поділу даних є таке: у таблиці фактів міститься те, що аналізується, а в таблиці вимірювань - те, відносно чого виконується аналіз».

З цього випливає, що в кубі даних на рис. 1.3 розмір, колір, матеріал - це вимірювання. Ця інформація повинна зберігатися в таблицях вимірювань. Посилання на цю інформацію містяться в таблиці фактів, яка, окрім цього, містить кількісні дані (наприклад, про ціни, виручку, витрати).

Існує дві основні моделі сховища даних: схеми зберігання «зірка» та «сніжинка». Ці схеми отримали свої назви завдяки їхній формі, яка схематично представлена на рисунках 1.5-1.6.

У схемі «зірка» таблиця фактів повинна містити всі зовнішні ключі, що є в базі даних. Усі ці зовнішні ключі є первинними ключами для таблиць вимірювань. Важливою особливістю цієї схеми є те, що таблиці вимірювань не мають самостійної цінності для аналітика, оскільки не містять жодної змістовної інформації. Приблизна структура схеми «зірка» представлена на рисунку 1.5.

На відміну від схеми «зірка», схема «сніжинка» складається з трьох типів

таблиць: таблиці фактів, таблиць вимірювань і таблиць вкладених вимірювань (субвимірювань). Таблиці вимірювань (усі або деякі) містять посилання на таблиці вкладених вимірювань.

Як зазначає [7]: «через наявність таблиць вкладених вимірювань у схемі „сніжинка“ вводиться концепція ієрархії». На рисунку 1.6 показана приблизна структура схеми «сніжинка»

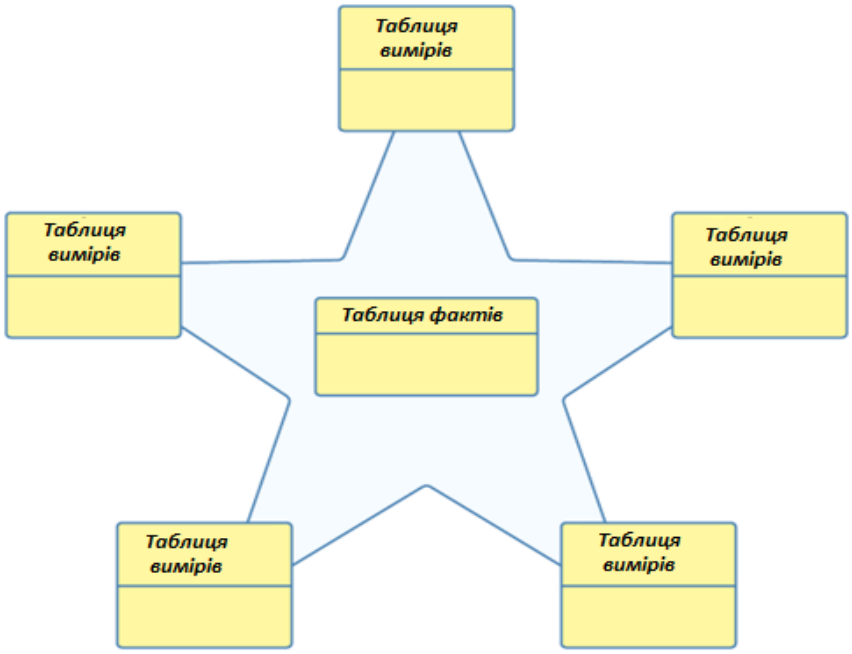

Рисунок 1.5 – Приклад структури сховища даних, збудованого за схемою «зірка»

Продуктивність запитів у схемі «зірка» нижча, ніж у ненормалізованій формі зберігання, але вища, ніж у схемі «сніжинка», оскільки в схемі «зірка» потрібно менше об’єднань [8].

Складна структура зв’язків у схемі «сузір'я» обумовлює низьку продуктивність запитів при такій схемі зберігання. При цьому поглиблення ієрархії вимірювань знижує швидкість запитів значно більше, ніж горизонтальне розширення вимірювань [11].

Однак, якщо дані мають потенціал для подальшої нормалізації, схема «зірка» є більш витратною за необхідними ресурсами зберігання, оскільки дані в такій схемі знаходяться в ненормалізованій формі.

При виборі схеми слід враховувати розмір таблиць вимірів. Якщо вимірювання є широкими таблицями, то краще буде схема «сніжинка». Але слід враховувати, що схема «сніжинка» буде вимогливішою до ресурсів великих обсягів даних [11].

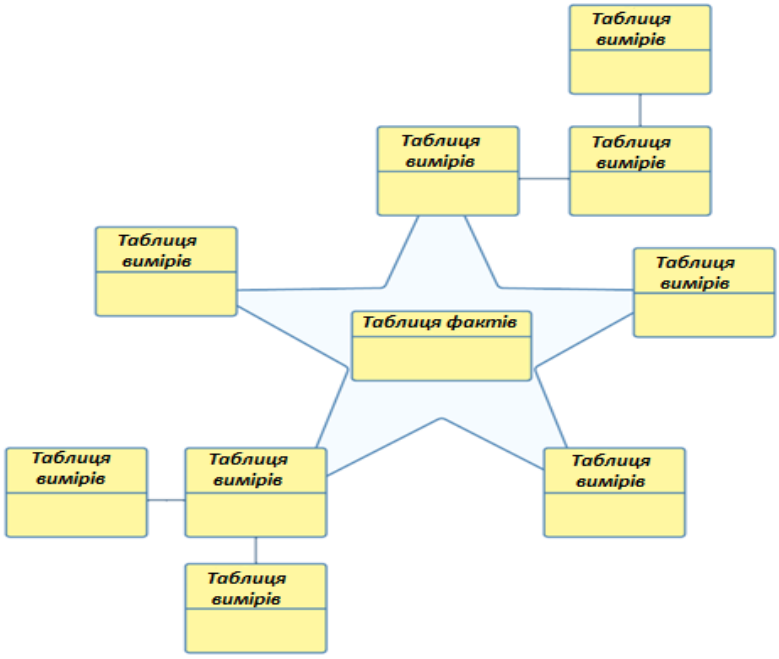

Рисунок 1.6 – Приклад структури сховища даних, побудованого за схемою «сніжинка»

Схема "сузір'я" ("Galaxy scheme", галактика) є розширенням схеми зірки. Таблиць фактів у такій схемі більше однієї, і ці таблиці пов'язані між собою загальними вимірами. Назва такої схеми обумовлена тим, що її можна визначити як сусідство кількох зірок (сузір'я) або сузір'їв (галактика) [12]. На рис. 1.7 представлено приблизну структуру схеми «сузір'я».

Ця схема є нестандартним вибором, однак у низці випадків вона застосовна. Іноді бізнес накопичує різнопланові кількісні дані, які зберігаються в різних таблицях, щоб їх було зручніше аналізувати, розмежовувати доступ до таблиць, а також щоб окремі таблиці не були занадто громіздкими.

Часто ці таблиці фактів є таблицями з великою кількістю рядків. Об'єднання цих таблиць в одну широку таблицю, як цього вимагають схеми «зірка» або «сніжинка», є нераціональним.

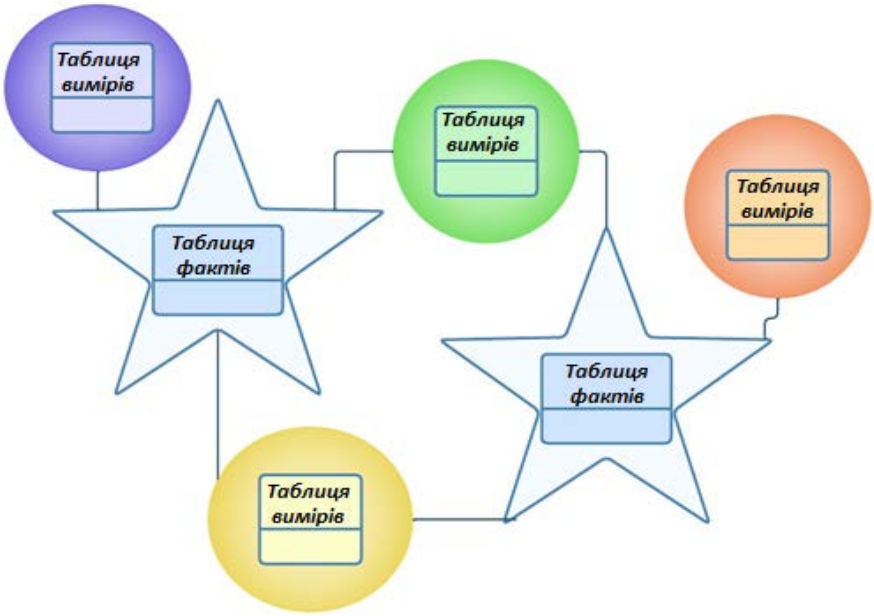

Рисунок 1.7 – Приклад структури сховища даних, побудованого за схемою «сузір'я»

У таблиці 1.1 представлено порівняння схеми «сузір'я» зі схемами «зірка» і «сніжинка».

Таблиця 1.1 – Класифікація моделей даних для OLAP-сховища [45]

Характеристики для порівняння	Схема «зірка».	Схема "сніжинка"	Схема " сузір'я "
Зв'язки у схемі	Одна таблиця фактів поєднана з кількома таблицями вимірів	Одна таблиця фактів з'єднана з кількома таблицями вимірювань, які з'єднані з таблицями субвимірювань	Декілька таблиць фактів пов'язані з таблицями вимірювань
Нормалізація даних	Денормалізована форма даних	Нормалізована форма даних	Нормалізована форма даних
Кількість вимірів	Декілька таблиць вимірювань відносяться до однієї таблиці фактів	Декілька таблиць вимірювань відносяться до однієї таблиці фактів і до субвимірювань	Декілька таблиць вимірювань відносяться до кількох таблиць фактів
Надмірність даних	Є	Ні	Ні

Продуктивність	Мала кількість зовнішніх ключів за рахунок відсутності зовнішніх зв'язків у вимірюваннях призводить до високої продуктивності	Велика кількість зовнішніх ключів призводить до низької продуктивності.	Декілька таблиць фактів і складні зв'язки між ними і вимірами призводять до низької продуктивності
Складність	Проста для розуміння та інтерпретації	Більш складна, ніж схема "зірка"	Використовується для складних структур даних, складна для інтерпретації
Обмеження	Можна використовувати лише одну таблицю фактів, не можна використовувати субвимірювання	Можна використовувати лише одну таблицю фактів	Не можна використовувати субвимірювання
Використання пам'яті	Через надмірність даних вимагає велику кількість пам'яті	Через відсутність надмірності даних вимагає менше місця на диску	Через відсутність надмірності даних вимагає менше місця на диску

1.5 Алгебраїчні операції OLAP

Алгебраїчні операції OLAP складаються з атомарних операторів. Атомарність у даному випадку означає, що ці базові оператори не можуть бути виражені через будь-яку комбінацію інших операторів. Аналіз досліджень із цієї теми показав, що п’ять категорій атомарних операторів вважаються основними для алгебри багатовимірних структур [66].

– SELECT - оператор, який дозволяє користувачу вибирати певні значення куба.

– DRILLING - це тип операцій, який включає в себе оператори ROLL UP та DRILL DOWN. Вони дозволяють переміщатися вглиб ієрархій куба та назад.

- Вибір вищого рівня деталізації здійснюється за допомогою оператора ROLL UP. Наприклад, за допомогою цієї команди можна перейти від виміру «Округа» до виміру «Район».
- Вибір глибшого рівня деталізації по ієрархії виконується оператором DRILL DOWN, що дозволяє, на відміну від попереднього прикладу, перейти від «Району» до «Округа».

– ROTATE - оператор обертання куба. Завдяки обертанню куба можна обирати інші виміри куба для аналізу. FROTATE - оператор для обертання кількісних даних (фактів).

– Оператор ADD дозволяє змінювати користувацькі міри (кастомізовані обчислення, задані користувачем).

- Додавання мір здійснюється командою ADDM.
- Видалення мір здійснюється командою DELM.

– Оператор агрегування AGGREGATE використовується для визначення того, як агреговані значення будуть виводитися у вигляді підсумкового значення за рядками та стовпцями зведених таблиць.

Схематичне зображення цих операцій представлено на рисунку 1.8.

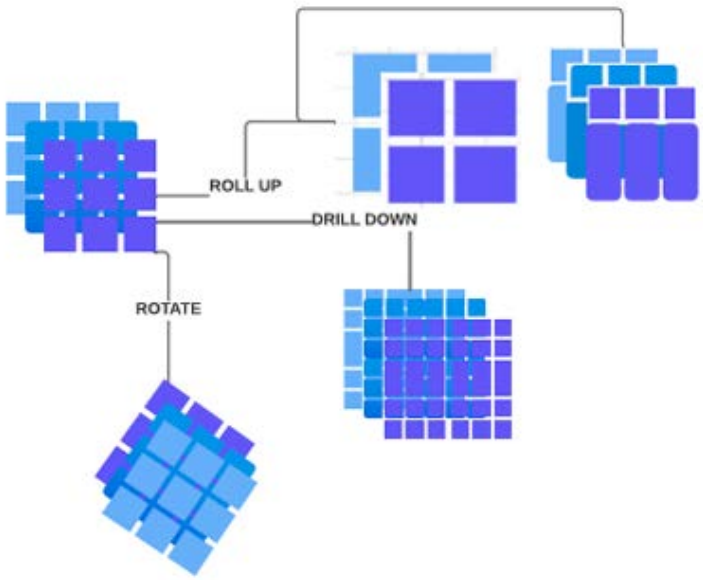

Рисунок 1.8 – Основні алгебраїчні операції OLAP

Алгебра для схеми «сузір'я» може відрізнятися від класичних схем.

Ф. Рават та Р. Турньє виділяють чотири алгебраїчні операції для цієї схеми, які відрізняються від традиційних операцій OLAP:

- Операція фокусування використовується для зосередження даних на кількох осях аналізу.
- Операція вибору підмножини застосовується для зменшення області аналізованих даних.
- Щоб скористатися перевагами ієрархічно впорядкованих параметрів, користувачу необхідні операції зміни рівня деталізації аналізованих даних.
- Для зміни критеріїв аналізу використовується операція обертання аналізованого об'єкта навколо інших осей аналізу або обертання самих осей аналізу навколо різних об'єктів [63].

У деяких моделях автори також зазначають можливість узгодженої обробки параметрів [35, 39, 48].

1.6 Вимоги до OLAP-систем

У 1993 році Е. Ф. Кодд і його колеги розробили ряд правил-рекомендацій для технології OLAP. Очевидно, що ключовою властивістю OLAP-системи є багатовимірність. Це також важлива характеристика для аналізу даних, оскільки розгляд даних з точки зору різних вимірів дозволяє знаходити приховані залежності.

Така система повинна бути інтегрована в загальну архітектуру системи підтримки прийняття рішень і надавати зрозумілий інтерфейс. Вона має оцінюватися користувачами як проста у використанні й доступна навіть для новачків. Клієнт-серверна архітектура повинна забезпечувати легке підключення різних клієнтів до сервера. Маніпулювання даними має бути необмеженим за кількістю вимірів.

Система повинна забезпечувати цілісність даних, водночас надаючи

можливість користувачам спільно працювати з кубом. Вона має бути стабільною та не перевантажуватися при збільшенні числа вимірів. Усі виміри в системі повинні бути рівноправними за значущістю та можливостями. Також необхідна проста інтеграція з різними джерелами даних.

У 1995 році на основі вимог, викладених Е. Ф. Коддом, Н. Пендсом і Р. Критом, було сформульовано тест FASMI, що містить вимоги до додатків для багатовимірного аналізу. На рисунку 1.9 представлені ці вимоги з роботи Н. Пендса [19].

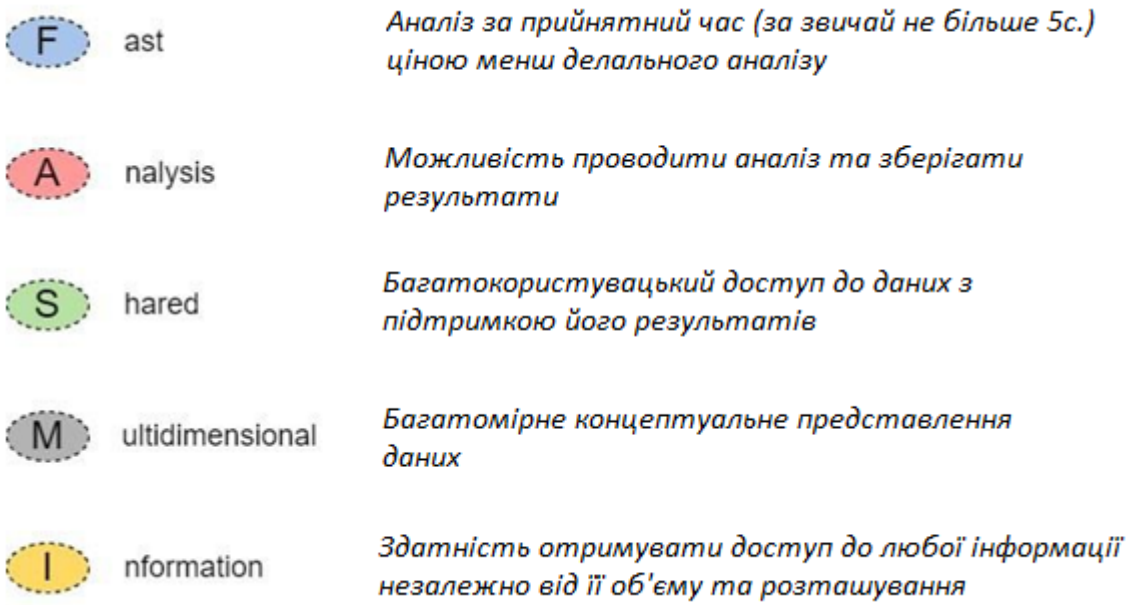

Рисунок 1.9 – Вимоги FASMI

Розмірність та рівні агрегації повинні бути необмеженими. Система повинна підтримувати практично будь-яку кількість вимірів даних, і кожен з цих загальних вимірів має надавати користувачу можливість визначати майже необмежену кількість рівнів агрегації [12].

1.7 Класифікація OLAP-систем

Сервер OLAP може функціонувати різними методами:

- як система управління реляційними базами даних, яка перетворює

операції з багатовимірними даними на стандартні операції реляційної моделі;

- як система, яка використовує багатовимірну модель зберігання, що дозволяє працювати безпосередньо з багатовимірними даними без використання реляційних таблиць [14].

У роботі [9] вказують, що існують три основні типи OLAP-систем:

- реляційні (ROLAP): дані у вигляді сукупності відносин. ROLAP представлений у більшості BI-систем. Тут немає попередньо визначеної схеми куба, тому ROLAP асоціюється з великими обчислювальними навантаженнями;
- багатовимірні (MOLAP): дані попередньо агреговані, і запити до них є заздалегідь визначеними. Такі дані організовані у вигляді впорядкованих багатовимірних масивів, які представляють собою гіперкуби даних;
- змішані (HOLAP): поєднання інструментів, що реалізують реляційну та багатовимірну модель даних. У цьому випадку агреговані дані зберігаються в структурі MOLAP, але тільки до певного рівня деталізації [13].

На сьогоднішній день, крім традиційних форм OLAP, на ринку присутні альтернативні підходи. Багато компаній замість традиційного підходу вибирають технологію In-memory OLAP. Такий підхід може бути корисним компаніям, які хочуть знайти компроміс між традиційними підходами, оскільки принципова новизна In-memory OLAP полягає в тому, що дані агрегуються і зберігаються в оперативній пам'яті. При цьому система оптимізована для зберігання в оперативній пам'яті, що дозволяє максимально ефективно використовувати її.

Для реалізації цієї технології використовується проміжна система баз даних, яка обробляє запити та "зберігається в оперативній пам'яті комп'ютера, що дозволяє уникнути затримок, пов'язаних з доступом до диска" [10].

Основні відмінності між типами OLAP-систем представлені на рис. 1.10 і таблиці 1.2.

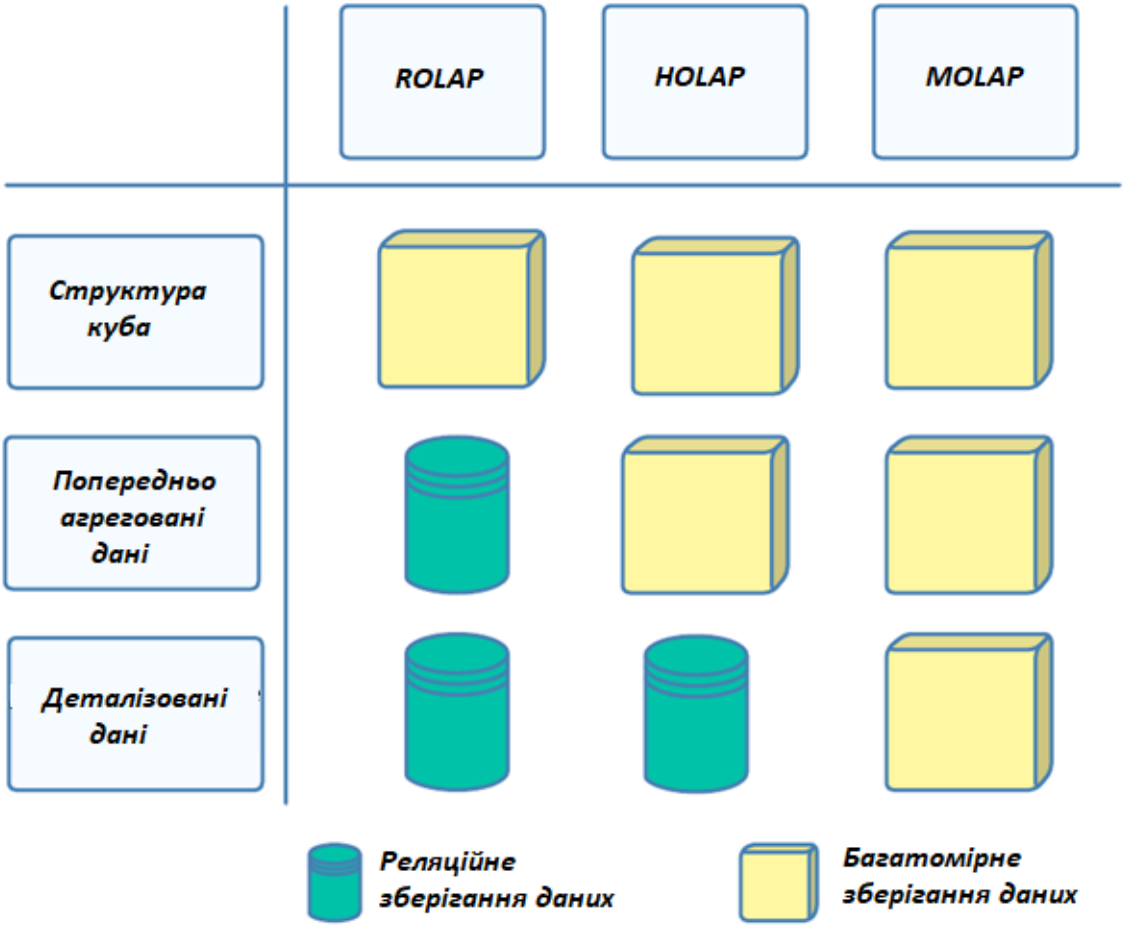

Рисунок 1.10 – Організація зберігання даних у різних типах OLAP-систем

Таблиця 1.2 – Класифікація OLAP-систем [17]

Характеристика	ROLAP	MOLAP	HOLAP
Масштабованість	Високий ступінь масштабованості	Низький ступінь масштабованості	Середній ступінь масштабованості
Використання пам'яті	Найменший можливий обсяг пам'яті	Найбільший можливий обсяг пам'яті	Середній обсяг пам'яті
Швидкість обробки запитів	Низька швидкість	Висока швидкість	Середня швидкість
Рівень захисту інформації та розмежування прав доступу	Високий рівень захисту інформації	Низький рівень захисту інформації	Середній рівень захисту інформації

Продуктивність	Система не оптимізована до виконання складних багатотабличних запитів. Наслідком цього є низька продуктивність	Система оптимізована для швидкого розрахунку агрегатних показників.	Система забезпечує більшу швидкість при отриманні відповідей на складні багатотабличні запити, ніж ROLAP-система, але нижче, ніж MOLAP-система

Як зазначають у [14], до мінусів In-memory OLAP відноситься «обмеженість зберігання та обробки куба даних обсягом основної пам'яті»;

Основні відмінності у архітектурі OLAP-систем представлені на рис. 1.11.

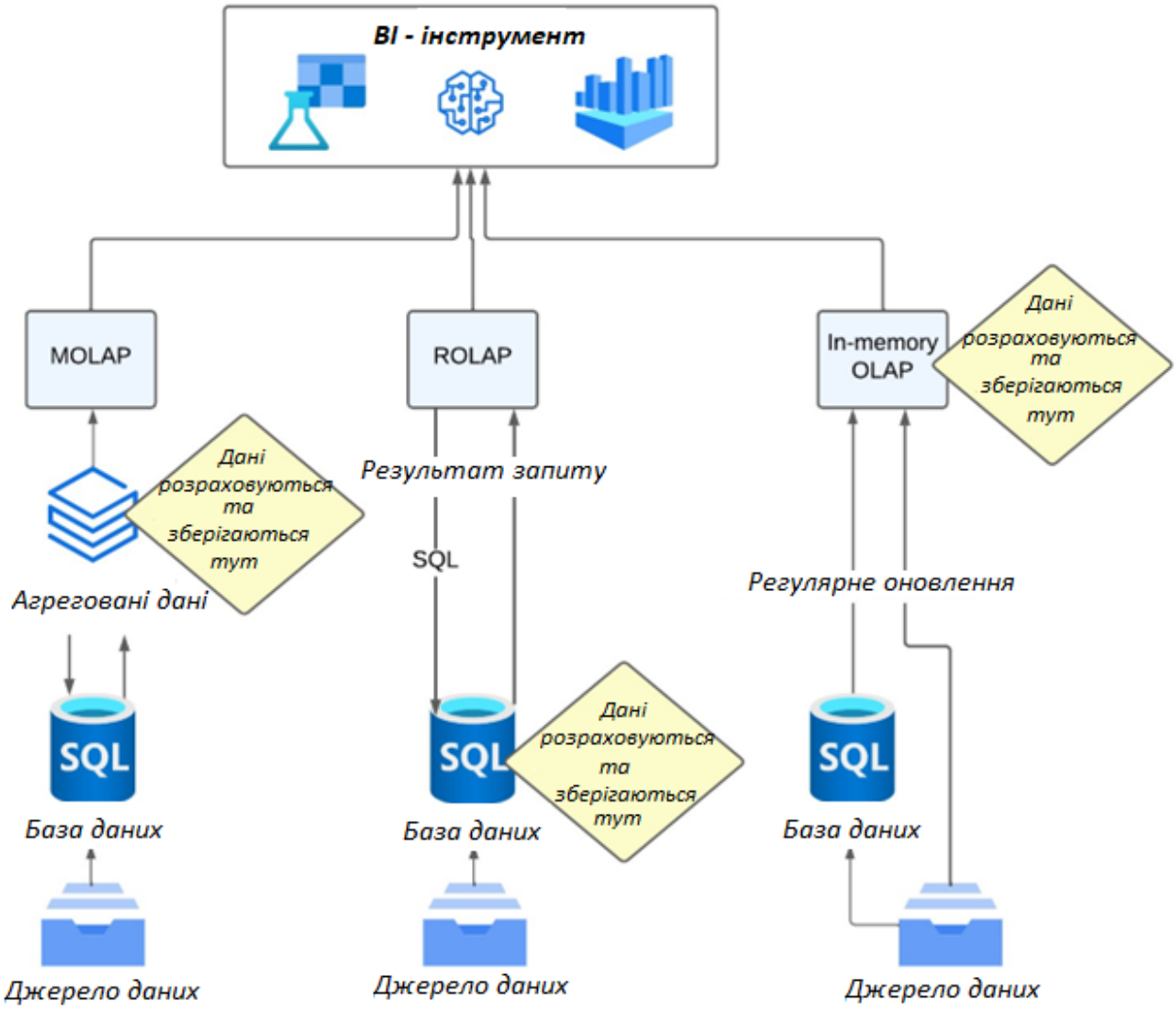

Рисунок 1.11 – Відмінності в архітектурі OLAP-систем

У [11] відносять до плюсів підходу In-memory OLAP те, що «не існує загрози «вибуху» даних, оскільки в кубі не зберігаються попередньо обчислені значення».

Висновки до розділу:

У ході аналізу робіт вітчизняних та зарубіжних авторів були сформульовані такі висновки.

Аналітична робота передбачає виконання запитів для отримання змістовної інформації, і це зумовлює популярність технології OLAP, яка дозволяє забезпечити максимальну швидкість обробки запитів.

OLAP-куб потребує рідкісного оновлення, що сприяє тому, що аналітику не доводиться стикатися з перевантаженням сервера бази даних. Однак, якщо компанії необхідно постійно оновлювати дані для аналітики, технологія OLAP може виявитися непридатною, оскільки перебудова куба даних часто є дуже ресурсоємним завданням.

Серед недоліків технології OLAP можна виділити значні часові витрати на реалізацію системи, необхідність великих обсягів пам’яті та спрощення структури даних.

РОЗДІЛ 2
ОПИС МОЖЛИВОСТЕЙ ТЕХНОЛОГІЇ OLAP ТА АНАЛІЗ ПРОЦЕСІВ

2.1 Огляд функціональних можливостей найпопулярніших програмних продуктів для аналізу даних на основі технології OLAP

На ринку існує безліч різних виробників OLAP-систем. Згідно з доповіддю компанії Gartner на січень 2024 року, найпопулярнішими закордонними продуктами OLAP з закритим вихідним кодом були: Microsoft Analysis Services, Qlik, Oracle Essbase, IBM Cognos TM1

Таблиця 2.1 - Характеристики іноземних OLAP-продуктів із закритим вихідним кодом [12]

OLAP-продукт	Операційна система		Тип зберігання				Інструменти візуалізації
	Windows	Linux	MOLAP	ROLAP	HOLAP	In-Memory OLAP	
Microsoft Analysis Services	Так	Ні	Так	Так	Так	Так	Microsoft Excel, Microsoft Power BI
Essbase	Так	Так	Так	Так	Так	Ні	Oracle Analytics Cloud, Narrative Reporting, Tableau
IBM Cognos TM1	Так	Так	Так	Ні	Ні	Так	IBM Cognos Insight, IBM Performance Modeler, IBMCognos Cafe Cognos BI, TM1 Perspectives for Excel
Qli	Так	Так	Ні	Ні	Ні	Так	QlikView

Microsoft Analysis Services інтегрований у Microsoft SQL Server і використовує різні архітектури зберігання даних, однак основний акцент робиться на MOLAP. Однією з ключових функцій цього сервісу є можливість роботи з кількома пов'язаними таблицями фактів без необхідності їх об'єднання в одну таблицю (схема «сузір'я»). Як зазначає С.М. Сарсимбаєва, ця система

дозволяє «проектувати, створювати та керувати багатовимірними структурами, які містять детальні статистичні дані з кількох джерел даних».

Oracle Essbase є високопродуктивним двигуном MOLAP. Створення кубів Essbase може відбуватися автоматично на основі бізнес-моделі BI, і їх інтеграція в семантичний рівень сприяє підвищенню продуктивності робочих навантажень BI. Сервер Essbase забезпечує розширені можливості багатокористувацького читання та запису, включаючи оновлення даних та перерахунок. Бізнес-користувачі можуть передавати дані назад на сервер і виконувати їх перерахунок з використанням сценаріїв обчислень.

IBM Cognos TM1 використовує багатовимірну базу даних у формі кубів з можливістю зворотного запису в початкові бази даних. Механізм TM1 дозволяє проводити складні обчислення безпосередньо в пам'яті на стороні клієнта, що забезпечує швидку обробку даних і масштабованість.

Серед основних функцій платформи Qlik можна виокремити підтримку спільної роботи в онлайн-режимі в захищеному середовищі та потужні можливості візуалізації даних, включаючи створення дашбордів.

Основною рисою QlikView є застосування асоціативної архітектури з обробкою даних в оперативній пам'яті (In-memory OLAP). Асоціативний аналіз передбачає об'єднання таблиць з різних джерел даних, автоматичне виявлення співпадаючих полів і встановлення асоціативних зв'язків між ними.

2.2 Огляд функціональних можливостей програмних продуктів для аналізу даних на основі технології OLAP з відкритим вихідним кодом

Оскільки «доступність продуктів з відкритим вихідним кодом не залежить від економічної та політичної ситуації, їх впровадження може бути найбільш раціональним рішенням для багатьох компаній» [12]. У таблиці 2.2 представлені характеристики найбільш відомих open-source OLAP-продуктів [15].

Apache Kylin – це MOLAP сервер, який дозволяє швидко обробляти

великі масиви даних шляхом побудови схеми «зірка», створення OLAP-куба, запуску запитів і отримання результатів через API [18]. Jedox Palo – це MOLAP сервер [19].

Таблиця 2.2 – Характеристики OLAP-продуктів з відкритим кодом [32]

OLAP-продукт	Операційна система		Тип зберігання				Інструменти візуалізації
	Windows	Linux	MOLAP	ROLAP	HOLAP	In-Memory OLAP	
Apache Kylin	Ні	Так	Так	Ні	Ні	Так	Superset, Zeppelin, Tableau, Qlik, Redash, Microsoft Excel
Jedox Palo	Так	Так	Так	Ні	Ні	Так	Microsoft Excel, Qlik, Tableau, Jedox Web, Power BI
Mondrian	Так	Так	Ні	Так	Ні	Ні	Jpivot
Atoti	Так	Так	Ні	Нет	Ні	Так	Jupyter Lab, Jupyter Notebook, Microsoft Excel
Cubes	Так	Так	Ні	Так	Ні	Так	CubesViewer
Clickhouse	Ні	Так	Так	Нет	Ні	Ні	Microsoft Excel, DataLens, Tableau

Palo розрахований перш за все на взаємодію з Excel. Більшість завдань, таких як створення вимірів і кубів, виконуються за допомогою плагіна для Excel.

Palo Eclipse Client дозволяє проектувати та переглядати виміри і куби в платформонезалежному середовищі Eclipse Framework [14].

Pentaho від Mondrian – це ROLAP-сервіс, архітектура якого складається з кількох рівнів. Рівень, який у Pentaho називається базовим, необхідний для зберігання агрегованих даних. На рівні вимірів виконуються запити до цих даних. Рівень зберігання – реляційна база даних. Те, як відображаються дані, визначається рівнем представлення [16, 22].

Atoti – це сервіс-бібліотека Python. Вона дозволяє створювати багатовимірні куби за технологією In-memory OLAP. Куб моделюється і відображається в середовищі Python, там же дані можна аналізувати і візуалізувати [16].

Cubes – це фреймворк для Python, який надає можливості для побудови багатовимірних кубів за технологією ROLAP. При цьому Cubes не надає можливість аналізувати і візуалізувати дані, для цих завдань поставляється утиліта CubesViewer. Це веб-інструмент, який може використовуватися для побудови різних графіків [13].

ClickHouse – це NoSQL база даних, яка належить до типу колонкових баз даних. Це високопродуктивна база даних, яка дозволяє без побудови багатовимірних кубів здійснювати аналіз за технологією OLAP. Цьому сприяє колонковий формат зберігання даних, який є дуже ефективним, якщо немає необхідності часто оновлювати дані.

Багато OLAP-продуктів, які поставляються безкоштовно, також надають широкий набір функціональних можливостей для аналізу даних. Для більшості невеликих компаній функцій цих продуктів буде достатньо. Серед них є продукти, які підтримують різні типи зберігання даних, включаючи MOLAP, ROLAP і In-memory OLAP.

Це дозволить вибрати підходящий продукт залежно від вимог проєкту. Зважаючи на той факт, що відкритий вихідний код робить ці продукти доступними незалежно від політичної та економічної ситуації, вони могли б стати підходящим рішенням для багатьох компаній.

2.3 Роль OLAP у сучасному світі

У останнє десятиліття технологія OLAP часто піддається критиці. Цю технологію багато експертів вважають «застарілою і такою, що йде в минуле» [17]. Однак занепад OLAP - це спірне твердження, адже це, по суті, питання правильного застосування термінології.

Приблизно 30 років тому аналітики даних задумались, що використання реляційних баз даних для великих робочих навантажень - це неефективний підхід. Ситуація погіршувалась тим, що в ті часи комп'ютери не були потужними. Це призводило до того, що переважна більшість бізнес-користувачів повинна була працювати з відносно невеликими обсягами пам'яті для виконання робочих навантажень BI. Таким чином, перші практики BI обрали загальний підхід - витягувати з реляційної бази даних необхідні дані і поміщати їх у ефективну структуру даних у пам'яті для маніпулювання. Але в ситуаціях, коли обсяг даних перевищує доступну пам'ять комп'ютера, цей підхід вже не підходить. Тому ранні BI-системи вирішували проблему агрегуванням даних і кешуванням агрегованих даних у вкладеному масиві. Так з'явилася технологія багатовимірних структур, яка згодом стала домінуючим методом зберігання даних для їх аналітики.

Недоліками класичного OLAP-куба є [21]:

- Неможливість роботи в реальному часі;
- Оновлення куба займає багато часу та потребує участі фахівців з різних областей;
- Необхідність великих апаратних ресурсах;
- Чутливість до порушень цілісності даних.

Обробка даних для їх вилучення з OLTP-систем і завантаження в куби є складним процесом, що вимагає спеціальних знань. Наприклад, використовується моделювання вимірів Кімбалла [18], концепція розробки сховища даних Інмона [20], гібридний підхід Data Vault, який поєднує переваги схеми «зірка» та 3-ї нормальної форми, запропонований Деном Лінстедтом у

2000 р. [21, 22].

В останнє десятиліття вибуховий ріст обсягу даних і зростання потужностей обчислювальних засобів призвели до розвитку ряду рішень, які можуть заміняти класичний OLAP-підхід.

Щодо OLAP як концепції, то багато з згаданих вище недоліків більше пов’язані з використанням саме важких кубів, а не з самою концепцією OLAP. OLAP, по суті, орієнтований на оптимізацію можливостей вивчення багатовимірних даних, попереднє формування та обчислення відповідних показників і вимірів для швидкого отримання бізнес-інформації [15].

Традиційна система зберігання даних, така як SQL, важко справляється з цим без значної роботи вузьких спеціалістів.

На сьогодні ключовими стратегіями реалізації онлайн-аналітичної обробки даних, окрім традиційного OLAP-куба, заснованого на реляційних базах даних, є застосування In-memory OLAP баз даних і технології NoSQL, зокрема, колонкових баз даних [22].

In-memory OLAP та NoSQL бази даних можуть використовуватися для онлайн аналітичної обробки даних. Вони можуть ефективно взаємодіяти з великими обсягами даних і високою щільністю даних. Зокрема, колонкові бази даних мають архітектуру, яка оптимізована для аналітичних операцій.

In-memory OLAP бази даних дозволяють повністю або частково зберігати дані в оперативній пам'яті. Це призводить до підвищеної продуктивності порівняно з базами даних, оптимізованими для дискового зберігання, оскільки доступ до оперативної пам'яті швидший, а внутрішні алгоритми оптимізації простіші.

Згідно з [19], «зберігання всіх даних (або, принаймні, гарячих даних) в основній пам'яті забезпечує вищу швидкість обробки транзакцій, що частково згладжує негативні ефекти одночасно виконуваних аналітичних запитів»

Відмінності в архітектурі реляційних та колонкових баз даних представлені рисунку 2.1.

Зчитування усіх стовпців при запиті	Реляційна база даних						
Зчитування тільки запитуємих стовпців	Колонкова база даних						

Рисунок 2.1 – Відмінності в архітектурі реляційних та колонкових баз даних

Модель NoSQL виникла у відповідь на потребу в ефективній обробці дуже великих обсягів даних. Технології NoSQL, зокрема колонкові бази даних, орієнтовані на горизонтальне масштабування та роботу з даними, які є або недостатньо структурованими, або постійно змінюються [22]. Це нереляційні бази даних, які не використовують SQL або використовують аналоги SQL.

Тим часом як OLAP-куби вимагають завантаження в куб багатьох цікавих вимірювань, колонкові бази даних дозволяють виконувати аналогічні робочі навантаження OLAP з однаково високим рівнем продуктивності без необхідності створювати нові куби. Іншими словами, це база даних SQL, яка ідеально підходить для робочих навантажень OLAP.

Це також обумовлено тим, що колонкові бази даних стискаються краще, ніж реляційні бази даних. Схожі фрагменти даних зберігаються в пам'яті набагато ефективніше, ніж різні фрагменти інформації. Оскільки колонкові бази даних зберігають стовпці даних, тобто значення з однаковими типами та схожими значеннями, це забезпечує більш ефективне стиснення.

В цілому таке стиснення означає, що при виконанні агрегаційного запиту в пам'ять може бути завантажено більше даних, що, в свою чергу, призводить до більш швидкого виконання загальних запитів [21].

Переваги зберігання даних у колонковій формі полягають у швидкому

пошуку, доступі та агрегації даних. Оскільки колонкові бази даних ефективно сжимають схожі фрагменти даних у пам'яті, операції агрегації виконуються швидше. Цей підхід також дозволяє звільнити додатковий простір для сортування та індексації даних всередині стовпців.

Кінцевим результатом усіх цих властивостей є те, що колонкові бази даних можуть забезпечувати продуктивність, подібну до OLAP-куба, без необхідності явного проектування та збору кубів. Однак продуктивність оновлення в колонковій базі даних дуже низька [17].

Це призводить до того, що багато сучасних колонкових баз даних обмежують можливість оновлення даних після збереження. Також до недоліків NoSQL баз даних можна віднести пом'якшення жорстких вимог до транзакцій, які присутні в реляційних базах даних [19].

Основні принципи ACID гарантують, що цілісність і узгодженість даних у реляційних сховищах буде забезпечена навіть при збої. Оскільки для ряду завдань в суворому дотриманні ACID немає потреби, NoSQL-бази найчастіше пропонують компроміс і принципи BASE, які є менш суворими і дозволяють досягти більшої продуктивності [21, 23].

Принципи BASE, які є менш суворими, дозволяють досягти вищої продуктивності [34, 25]. Відмінності підходів представлені на рисунку 2.2.

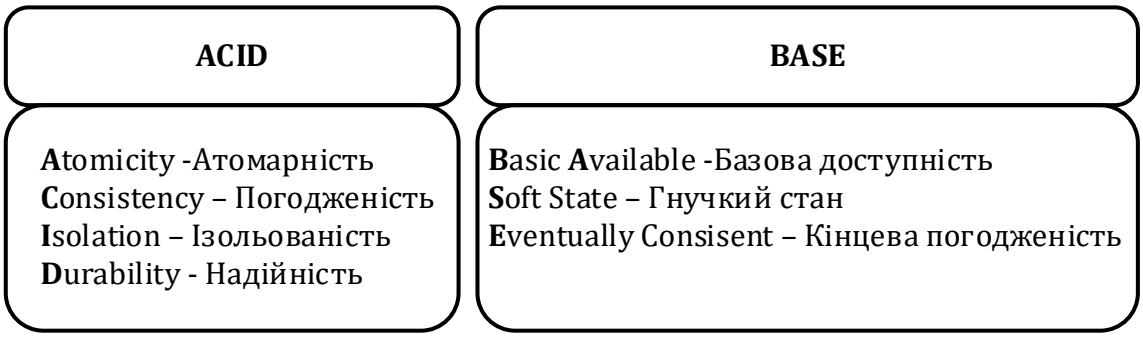

Рисунок 2.2 – Відмінності вимог до SQL та NoSQL баз даних

2.4 Характеристика діяльності підприємства

Підприємство займається розробкою аналітичних звітів на замовлення інших компаній. Команда аналітиків та інженерів розробляє аналітичні звіти,

що відображають ключові метрики та показники ефективності бізнесу клієнтів, а також маркетингові дослідження на основі відкритих даних. Звіти допомагають компаніям приймати обґрунтовані стратегічні рішення на основі даних.

Основні робочі процеси компанії:

- замовлення та збір вимог. Цей процес включає взаємодію з клієнтами для розуміння їхніх потреб та вимог до аналітичних звітів;
- аналіз даних. Цей процес включає збір, очищення, обробку та аналіз даних для створення аналітичних звітів;
- розробка та створення звітів. Цей процес включає розробку аналітичних звітів відповідно до вимог клієнта.

Наразі компанія збирає дані з відкритих джерел до бази даних PostgreSQL. Потім дані проходять первинну обробку мовою SQL. Оброблено дані передаються до середовища розробки зі встановленою мовою програмування Python, де виконується розвідувальний аналіз даних та розробляються аналітичні звіти. Функціональна модель, що відображає структуру та функції системи процесу роботи з даними, зображена на рис. 2.3 - 2.4. На рисунку 2.5 представлена крос-функціональна карта, яка докладно описує обов'язки підпроцесів у процесі.

Для досягнення мети оптимізації процесу збору та зберігання даних і автоматизації завдань при роботі з даними, компанії необхідно зробити процес обробки даних простішим та стандартизованим. Наразі великі масиви даних не можуть бути передані для складання звітів аналітикам у тому вигляді, в якому вони зберігаються в базі даних. Попередньо ці дані необхідно обробляти засобами мов SQL та Python. Ці процеси потребують великих часових ресурсів і спеціальних навичок.

Ключові точки покращення у роботі з великими даними в даній компанії включають:

- оптимізація ресурсів та автоматизація завдань. Оптимізація ресурсів та автоматизація завдань дозволяють скоротити витрати на управління та аналіз даних, що допомагає досягти мети зниження витрат та збільшення прибутку;

Рисунок 2.3 - Контекстна діаграма IDEF0 «ЯК Є» процесу роботи з даними

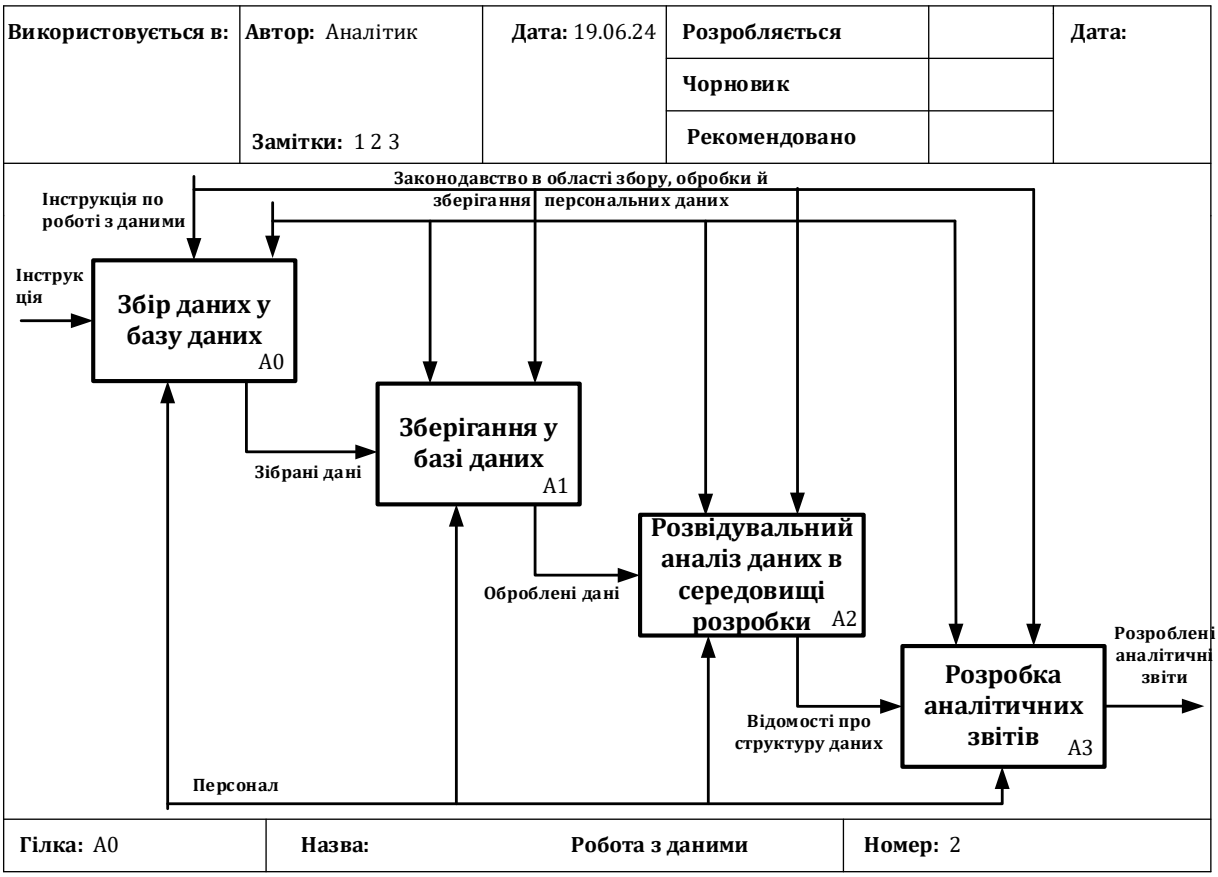

Рисунок 2.4 - Декомпозиція 1-го рівня контекстної діаграми IDEF0 «ЯК Є» процесу роботи з даними

– навчання персоналу. Успішне навчання персоналу в галузі роботи з даними та сучасними технологіями аналізу даних допомагає компанії досягти мети підвищення професійного рівня співробітників.

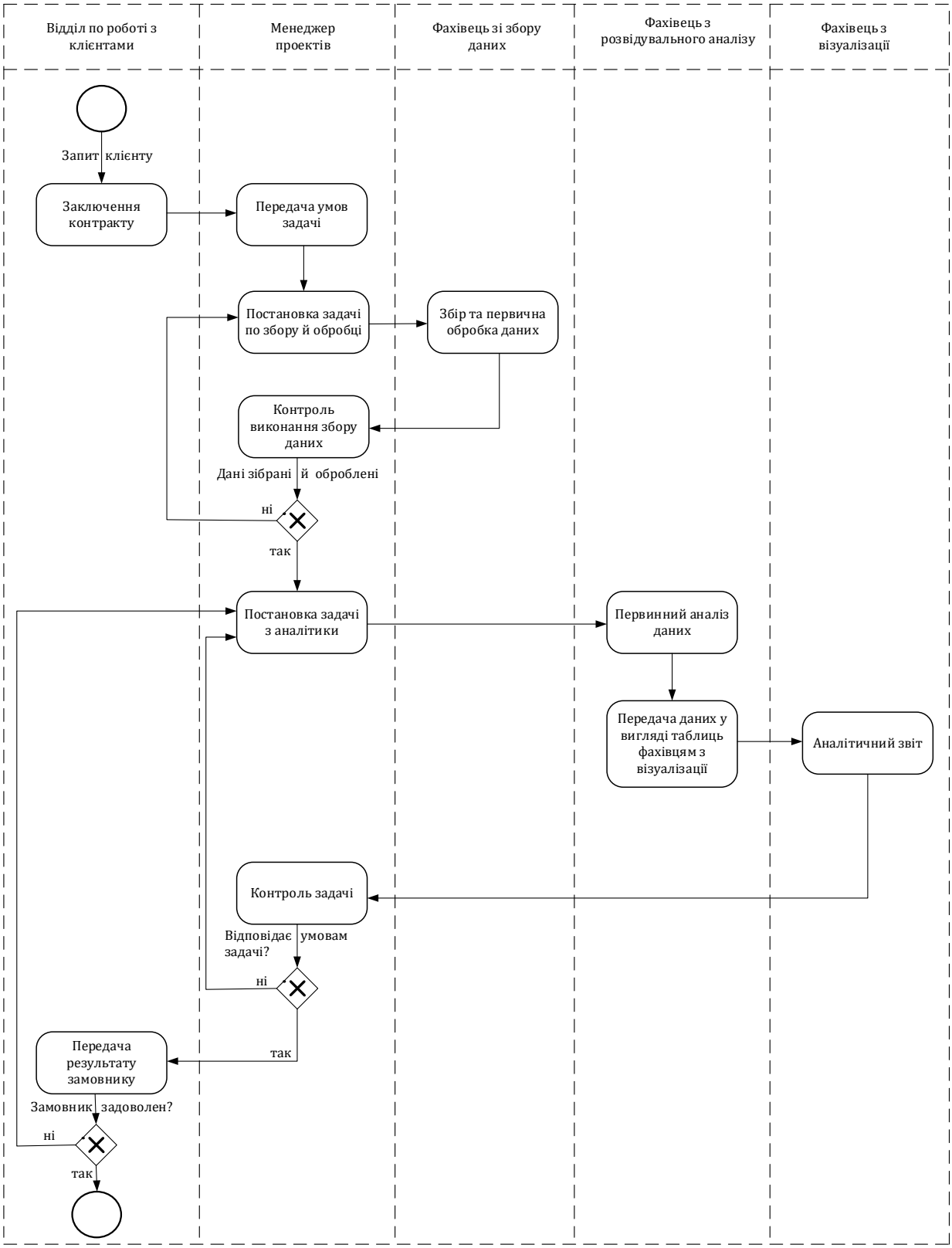

Рисунок 2.5 - Діаграма плавальних доріжок «ЯК Є» процесу роботи з даними

На сьогодні в компанії налагоджені процеси автоматизованого збору даних. Однак методи, які використовуються фахівцями компанії для зберігання та обробки даних, можна розцінювати як застарілі. Враховуючи розвиток Big Data, компаніям необхідно впроваджувати ефективні ETL-процеси, які автоматично обробляють та структурують дані перед передачею в аналітичне середовище. Покращена обробка даних підвищує якість та надійність аналітичних звітів.

Можливою «точкою покращення» можна вважати оптимізацію процесу збору та зберігання даних, автоматизацію завдань при роботі з даними.

2.5 Збір вимог

Основною задачею дослідження є моделювання системи аналізу зазначених даних, які можуть бути використані страховими компаніями для аналізу загальних показників ринку, подальшого прогнозування та використання результатів для прийняття ефективних рішень. Було виявлено такі основні вимоги, яким має відповідати система:

Опис продукту

Система аналізу даних — це система, яка дозволить користувачам:

- Створювати багатовимірні куби даних для аналізу множини вимірів та показників.
- Виконувати аналітичні запити до даних та отримувати відповідні результати.

Доступ до системи аналізу даних може здійснюватися лише авторизованими користувачами, пов'язаними з аналітичними процесами в організації.

Функціональні вимоги:

- Інтерфейс аналізу даних: Користувацький інтерфейс повинен надавати можливості для створення, налаштування та виконання аналітичних запитів до даних.

– Створення кубів даних: Система повинна дозволяти користувачам створювати багатовимірні куби даних для аналізу множини вимірів та показників.

– Агрегування та підсумовування: Система повинна надавати можливість агрегувати та підсумовувати дані для швидкого виконання аналітичних запитів.

– Підтримка різних типів даних: Додаток повинен підтримувати різні типи даних, включаючи числові, текстові, дати та інші.

– Інтеграція з джерелами даних: Система повинна дозволяти інтегрувати дані з різних джерел, таких як бази даних, файли CSV та інші.

– Користувацькі звіти та візуалізація: Користувачам повинні бути доступні гнучкі можливості створення звітів та візуалізації даних, включаючи діаграми, графіки, таблиці та ін.

Нефункціональні вимоги:

– Продуктивність: Система повинна забезпечувати високу продуктивність при виконанні аналітичних запитів навіть із великими обсягами даних.

– Масштабованість: Додаток повинен бути легко масштабованим для обробки зростання обсягів даних та користувацького навантаження.

– Безпека: Система повинна забезпечувати автентифікацію, авторизацію та контроль доступу до даних для забезпечення безпеки інформації.

– Надійність: Додаток повинен бути надійним та мати механізми для виявлення та відновлення від збоїв.

– Зручність використання: Інтерфейс користувача повинен бути інтуїтивно зрозумілим та легким у використанні для широкого кола користувачів.

Інтеграція та взаємодія:

–API та інтеграція: Система повинна надавати API для інтеграції з іншими додатками та інструментами аналізу даних.

–Експорт даних: Користувачі повинні мати можливість експортувати дані

та результати аналізу для подальшого використання.

Вимоги до продуктивності:

–Час виконання запитів: Система повинна забезпечувати швидке виконання аналітичних запитів, включаючи запити до багатовимірних кубів даних.

–Відгук системи: Додаток повинен мати низький часовий відгук для користувацьких запитів та операцій.

Вимоги до підтримки:

– Підтримка та оновлення: Постачальник системи повинен забезпечувати регулярні оновлення, підтримку та обслуговування продукту.

Дані по ринку страхування містять інформацію про кілька накопичених кількісних величин (про премії, виплати та договори), а також якісні дані, що їх описують (довідники). З цього випливає, що переважно використовувати схему «зірка», як це показано на діаграмі класів, представленій на рисунку 2.6

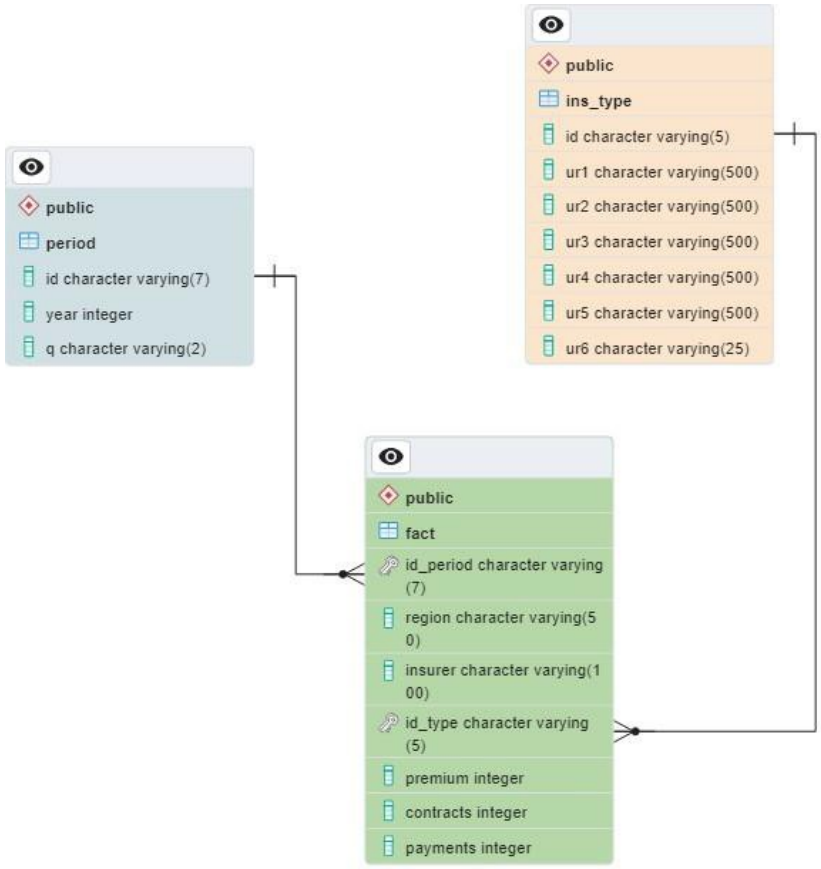

Рисунок 2.6 - Діаграма класів OLAP-сховища

У даній структурі даних таблиця фактів «fact» містить наступну інформацію:

- про страховика (поле «insurer»);
- регіон, де зафіксовано страховий факт (поле «region»);
- період, коли зафіксовано страховий факт (поле «period»);
- вид страхування (поле «id_type»);
- суму премій, зібраних страховиком (поле «premium»);
- число укладених договорів (поле «contracts»);
- виплати, здійснені страхувальникам (поле «payments»).

Таблиця фактів пов'язана з вимірами (таблицями-довідниками) period, в якому дані про період розбиті на роки та квартали, і ins_type, в якому розшифрована інформація про вид страхування. Дані про види страхування мають ієрархічну структуру, як показано на рисунку 2.7. На рисунку 2.7 представлена не повна ієрархія, насправді вона має 5 рівнів.

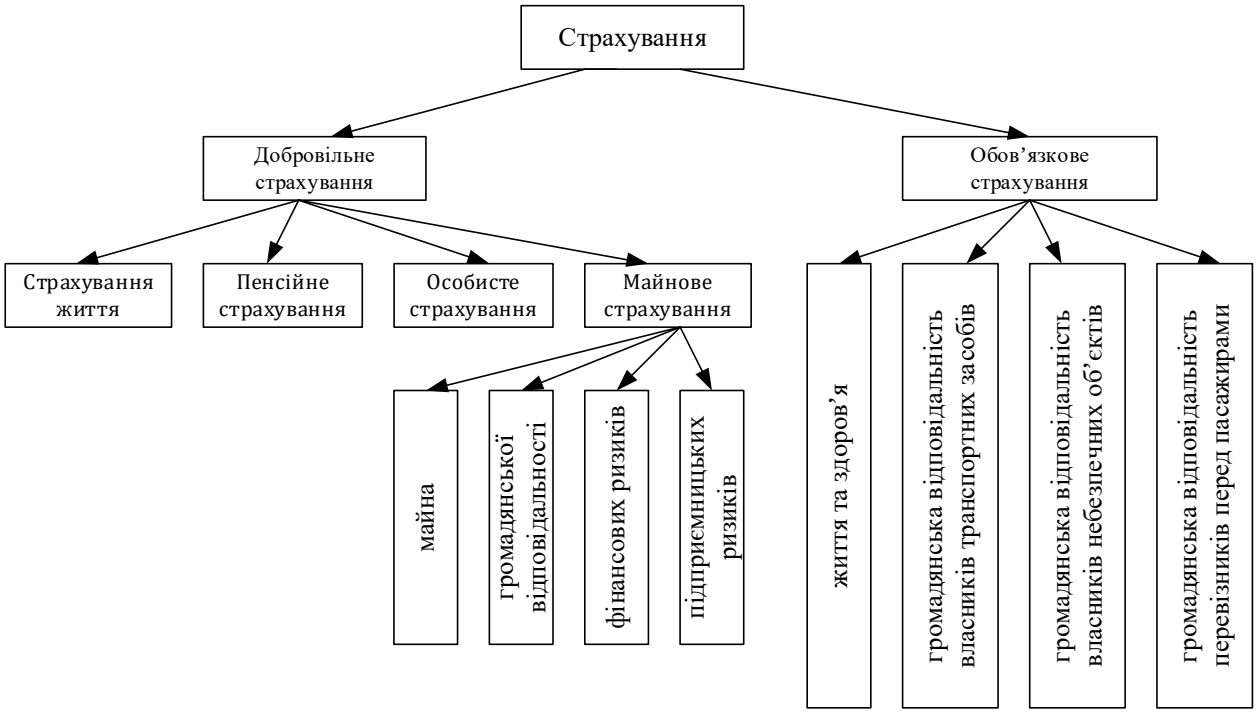

Рисунок 2.7 – Ієрархічна структура видів страхування

Структура даних вимагає, щоб рішення підтримувало створення та обробку ієрархічних структур.

На основі проведених досліджень, аналізу вимог компанії та даних, які компанія планує аналізувати, була вироблена рекомендація розглянути в якості альтернативи впровадженим технологіям використання технології багатовимірного аналізу даних OLAP.

База даних OLTP, яку зараз використовує компанія для аналізу даних, оптимізована для обробки операцій транзакційної обробки, а не для аналітичних запитів. Оскільки пов'язані дані зберігаються в кількох таблицях, аналіз вимагає використання повільних, неефективних та складних для складання SQL-запитів з використанням SQL-команди JOIN, яку використовують, щоб об'єднувати рядки таблиць на основі сполучного стовпця між ними. Проведення аналізу за допомогою SQL-команди JOIN в OLTP-базі може призвести до надмірного використання ресурсів сервера та зниження продуктивності для інших операцій транзакційної обробки.

При цьому, OLAP дозволить аналізувати зведені дані за різними вимірами, такими як регіон, період та вид страхування, та здійснювати зведені розрахунки для отримання цінної інформації. Враховуючи ієрархічну структуру даних про види страхування, OLAP може дозволити швидко обробляти та аналізувати такі дані, дозволяючи користувачам проводити аналіз на різних рівнях деталізації.

Проте класичні OLAP-рішення мають низку недоліків. На сьогоднішній день ключовими стратегіями реалізації онлайн аналітичної обробки даних є застосування In-memory OLAP баз даних та технології NoSQL, зокрема, колонкових баз даних.

Колонкові бази даних повільно працюють на запис та оновлення даних. Оскільки для компанії важливо мати можливість оновлювати дані для аналітики, варіант побудови системи аналітики на основі колонкових баз даних було відхилено.

Найбільш переважним варіантом для компанії видається використання In-memory OLAP-продукту Atoti, оскільки він є сучасним рішенням з відкритим вихідним кодом. Atoti надає гнучку модель даних з можливістю

швидкого створення та зміни структури кубів даних без необхідності перезавантаження даних. Atoti також підтримує створення обчислюваних мір та ієрархій.

Atoti передбачає роботу в Python, але при цьому виключає етап розгортання куба в окремому інструменті. По суті, Atoti дозволяє пройти весь етап аналізу даних, задіюючи лише один інструмент. Це дозволяє обмежити набір вимог до співробітників лише знанням мови програмування Python.

Оскільки зараз співробітникам необхідне знання мови запитів SQL на дуже високому рівні для того, щоб збирати з великого обсягу даних об'єкти, придатні для аналізу, це могло б стати істотним спрощенням роботи і дозволило б скоротити витрати на фахівців з вузькою спеціалізацією. Також, на відміну від класичного OLAP-куба, куб в Atoti не передбачає використання мови багатовимірних виразів MDX.

Висновки до розділу:

Для оптимізації процесу роботи з даними компанії необхідно зробити процес обробки даних простішим та стандартизованим.

Важливі напрямки покращення включають ефективний збір та зберігання даних, оптимізацію ресурсів та автоматизацію завдань, а також навчання персоналу. Компанії слід звернути увагу на сучасні методи обробки та аналізу даних. В цілому, покращення процесу роботи з даними допоможе компанії підвищити ефективність та якість послуг, що надаються, а також досягти конкурентної переваги на ринку аналітичних послуг.

В результаті аналізу вимог компанії до системи аналізу даних та особливостей даних, з якими вона працює, рекомендується розглянути впровадження технології багатовимірного аналізу даних OLAP. Це зумовлено необхідністю забезпечити високу продуктивність при виконанні аналітичних запитів навіть із великими обсягами даних, а також гнучкими можливостями створення звітів та візуалізації даних. Найбільш кращим варіантом для компанії видається використання In-memory OLAP-продукту Atoti, оскільки

він відповідає вимогам щодо продуктивності, гнучкої моделі даних та відкритого вихідного коду.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ СИСТЕМИ OLAP В ATOTI

3.1 Моделювання системи OLAP в Atoti

На основі проведених досліджень було вироблено рекомендацію розглянути як альтернативу впровадженим технологіям використання технології багатовимірного аналізу даних, що надається Atoti.

Платформа Atoti - це бібліотека з відкритим вихідним кодом, у якій реалізована можливість інтегрувати повноцінний OLAP-куб у середовище Python. Це технологія, що дозволяє безпосередньо запитувати дані, створювати звіти й отримувати аналітичні дані «на льоту». Фахівці з візуалізації даних за допомогою Excel або застосунку Atoti можуть звертатися до OLAP-кубів та отримувати багатовимірні перетини у вигляді зведених таблиць.

На рисунку 3.1 представлено порівняння архітектури класичного сховища даних Microsoft Analysis Services та архітектури Atoti [36].

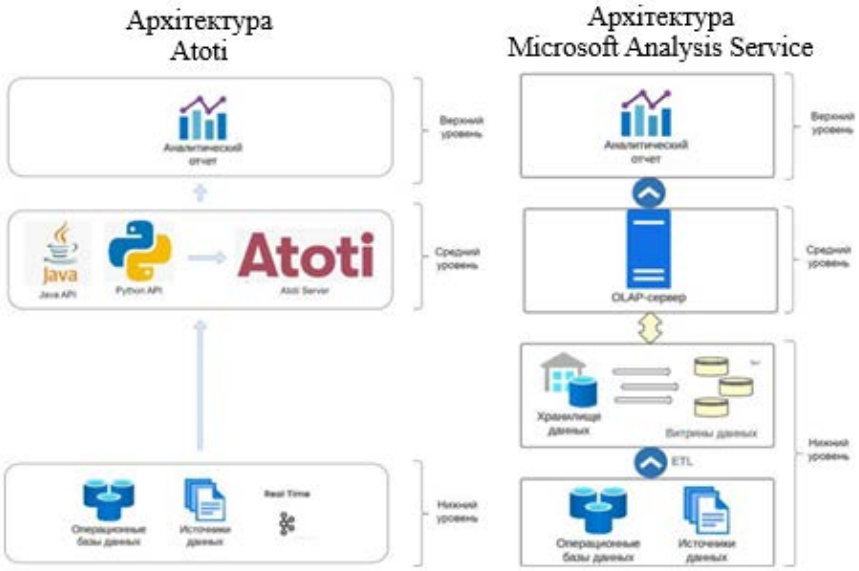

Рисунок 3.1 - Порівняння архітектури Microsoft Analysis Services та архітектури Atoti

Після впровадження рішення з'являється можливість усунути етап

попереднього аналізу та обробки даних, що зробить процес роботи з даними менш тривалим і дозволить значно зменшити кількість помилок на основі людського фактору. На рисунках 3.2–3.2 показано, що система аналітики з включенням класичного OLAP-куба вимагає участі більшої кількості фахівців.

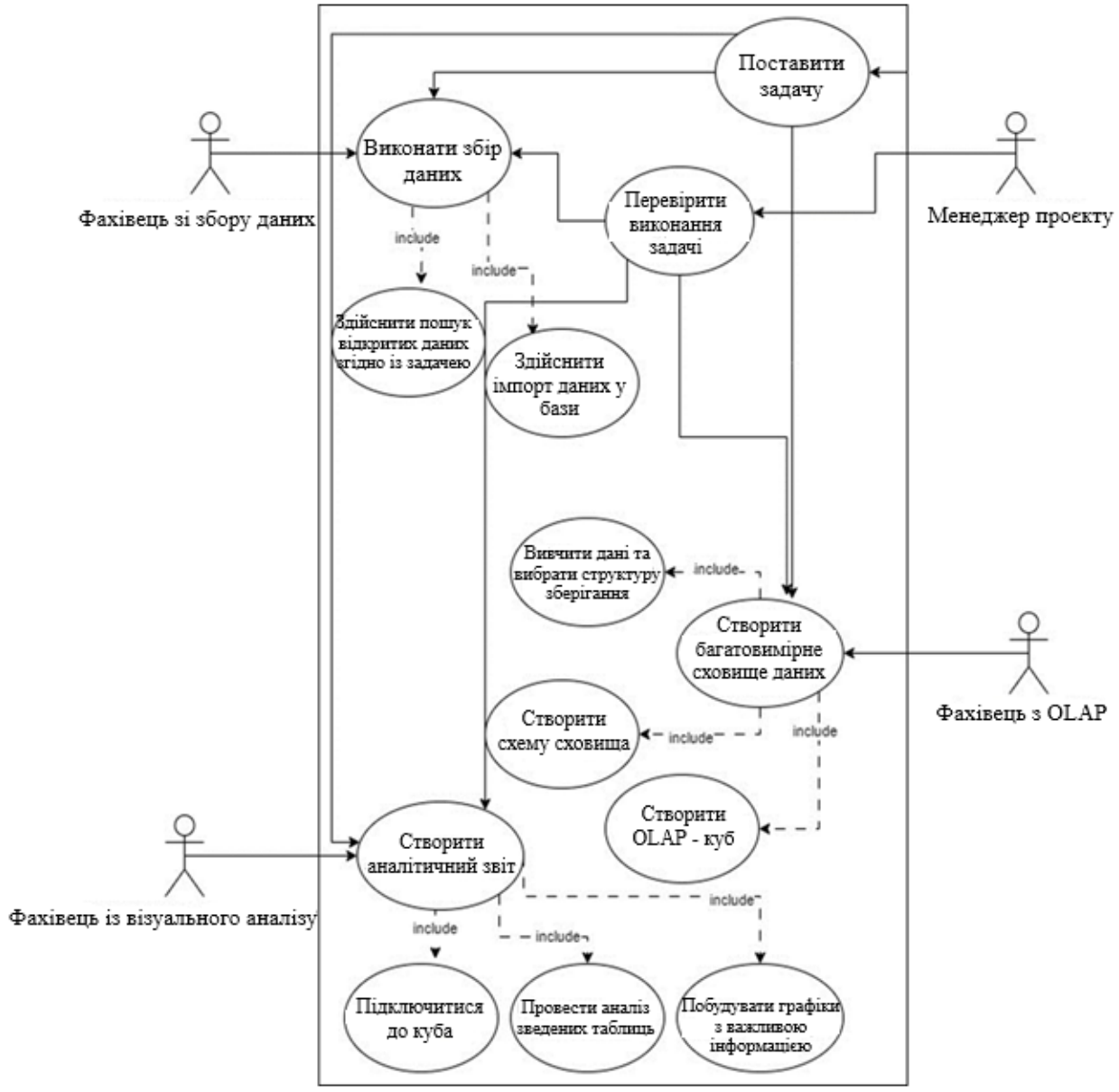

Рисунок 3.2 – Діаграма варіантів використання системи із включенням Microsoft Analysis Services

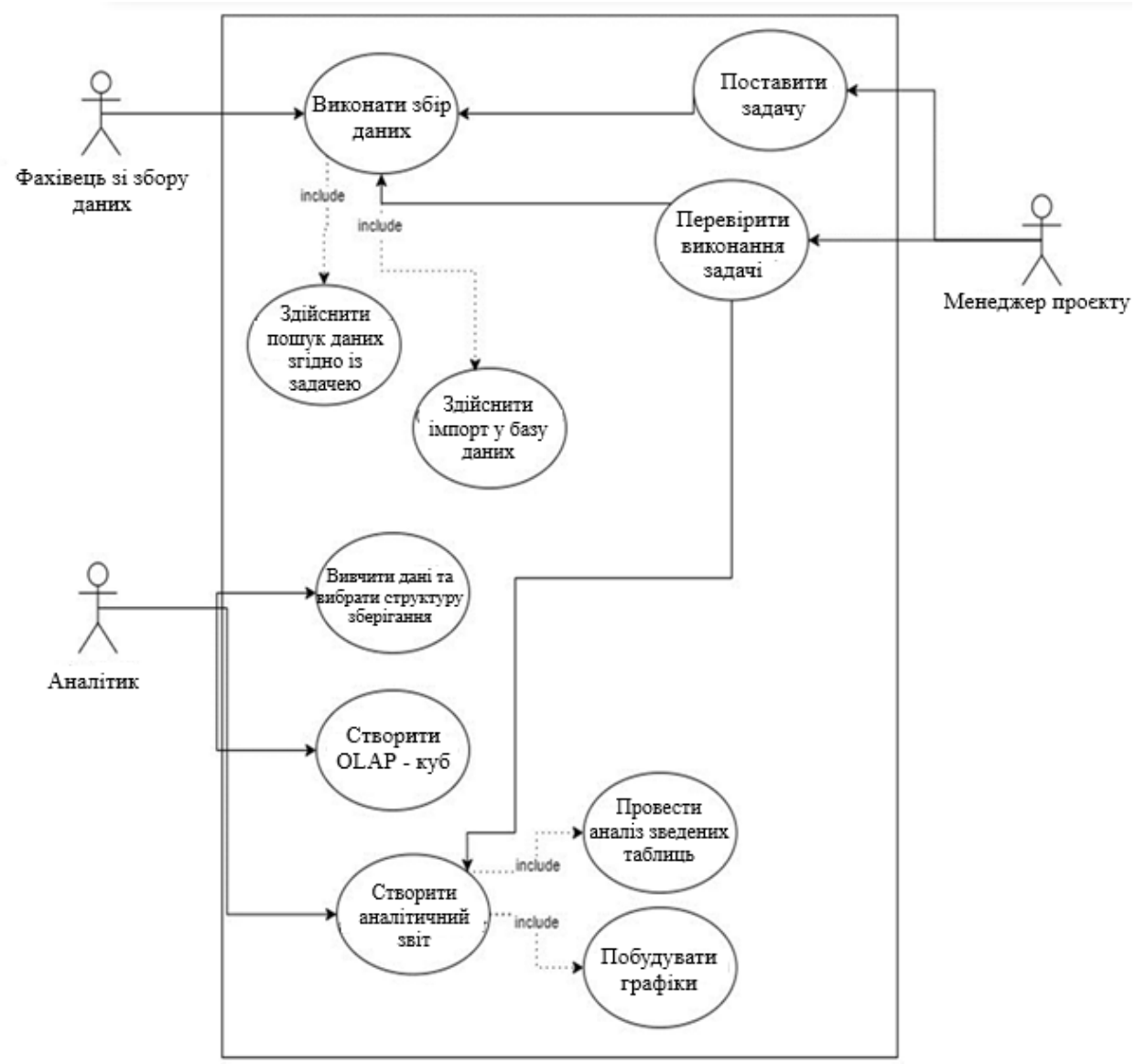

Рисунок 3.3 – Діаграма варіантів використання системи із включенням Atoti

Таким чином, запропонований підхід надає можливість скоротити витрати на роботу вузьких фахівців.

3.2 Реалізация системи

Сервіс Atoti дозволяє збирати, аналізувати та візуалізувати дані в середовищі програмування Python, а саме в ноутбуках Jupyter. Jupyter - це інтерактивне середовище для роботи з кодом. У ньому можна створювати блоки

коду, які можна виконувати окремо та візуалізувати результати.

Jupyter підтримує кілька мов програмування, включно з Python, R та Julia. Ноутбуки можна інтегрувати з різними інструментами та джерелами даних.

Також у Jupyter можна відображати не тільки код, але й текстові вставки, картинки, візуалізувати графіки за допомогою спеціалізованих бібліотек.

У бібліотеку Atoti вбудовано модуль, що дозволяє інтегруватися з популярною бібліотекою Pandas. Зокрема, дані Atoti можуть зберігатися у структурі Pandas DataFrame, яка є загальноприйнятою для роботи з табличними та багатовимірними даними в середовищі програмування Python.

Для побудови системи аналізу даних на основі Atoti в Jupyter необхідно встановити відповідні бібліотеки.

Код на рисунку 3.4 читає дані з бази даних PostgreSQL (три різні таблиці: таблиця фактів та дві таблиці вимірів) в Atoti, використовуючи метод read_sql() сесії Atoti.

```
fact = session.read_sql(
    """SELECT id_period, region, insurer, id_type, premium, contracts, payments
    FROM public.fact""",
    url="jdbc:postgresql://host:5432/db?user=user&password=password",
    table_name="fact"
)

ins_type = session.read_sql(
    """SELECT * FROM public.ins_type""",
    url="jdbc:postgresql://host:5432/db?user=user&password=password",
    table_name="ins_type"
)

period = session.read_sql(
    """SELECT * FROM public.period""",
    url="jdbc:postgresql://host:5432/db?user=user&password=password",
    table_name="period"
)
```

Рисунок 3.4 – Отримання даних з бази даних PostgreSQL в Atoti

Оскільки дані компанії зберігаються в базі даних PostgreSQL, для їх завантаження в ноутбук необхідно використовувати JDBC-драйвер, що дозволяє

отримувати дані з бази даних. У методі atoti.session.read_sql() бібліотеки Atoti необхідно прописати рядок підключення до бази даних. Цей метод повертає дані у вигляді Pandas DataFrame.

У цих запитах як аргумент методу передається SQL-запит для вибірки даних з таблиці. Зокрема, запит для отримання даних для таблиці фактів вибирає стовпці id_period, region, insurer, id_type, premium, contracts та payments з таблиці public.fact у базі даних PostgreSQL. table_name - це ім'я таблиці, яке буде використовуватися для створення тимчасової таблиці в Atoti для зберігання даних.

Ключова особливість Atoti як BI-сервісу - це можливість створення багатовимірного куба даних. При цьому, числові стовпці будуть визначені як міри, категоріальні стовпці - як виміри куба. На рисунку 3.5 показано процес приєднання вимірів до таблиці фактів (.join()), після чого куб можна візуалізувати за допомогою методу cube.schema.

```
fact.join(ins_type, fact['id'] == ins_type['id'])
fact.join(period, fact["period"] == period["id"])
```

```
cube = session.create_cube(fact)
```

```
cube.schema
```

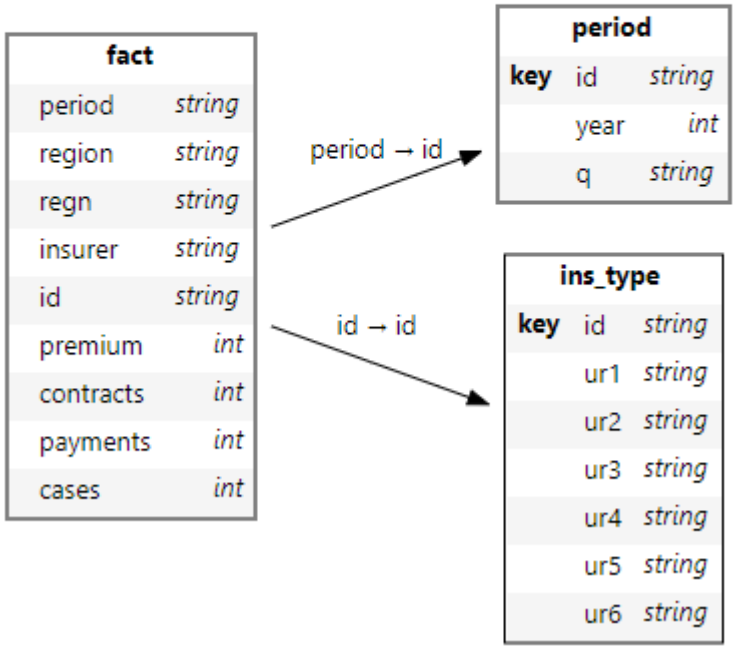

Рисунок 3.5 – Створення куба даних в Atoti

Як видно з рисунка 3.5, створення куба в Atoti не вимагає написання складного коду. Творці продукту дотримуються концепції створення прототипу, тобто, куб швидко створюється та збирається, а потім збагачується додатковими обчисленнями й агрегатами. Це дозволяє протестувати велику кількість варіацій куба. Внесення змін у куб не є складною та трудомісткою задачею.

Перевага Atoti перед класичним OLAP-підходом полягає в тому, що параметри куба можна змінювати після збирання куба даних: додавати обчислювані міри та змінювати елементи. Цей підхід дозволяє змінювати куб у реальному часі. Так, у середовищі JupyterLab можна налаштувати ієрархії для вимірів вже після того, як куб було зібрано, як це показано на рисунку 3.6. Після цієї дії куб можна не перезбирати, зміни вже внесені в його структуру.

```
h["ins_type"] = {
    "ur1": ins_type["ur1"],
    "ur2": ins_type["ur2"],
    "ur3": ins_type["ur3"],
    "ur4": ins_type["ur4"],
    "ur5": ins_type["ur5"],
    "ur6": ins_type["ur6"],
}
```

```
h["period_hier"] = {
    "year": period["year"],
    "quater": period["q"]
}
```

Рисунок 3.6 – Змін вимірювань в Atoti

В Atoti також доступне обчислення користувацьких мір, зокрема складних мір із використанням вбудованих функцій Atoti, таких як ранжування (atoti.rank()) та агрегування (atoti.total()), як це показано на рисунку 3.7. Агрегатні функції, вбудовані в Atoti, дозволяють виконати складні обчислення без необхідності використовувати складні конструкції. При цьому, налаштування користувацьких мір також не вимагає ручного оновлення куба

даних.

```
m["  Сума премій тис.грн   ."] = m["premium.SUM"]/1000
m["  Сума виплат тис.грн   ."] = m["payments.SUM"]/1000
m[' Кількість полісів шт.'] = m['contracts.SUM']

m["  Ранг по преміям "] = tt.rank(m["premium.SUM"], h['insurer'], ascending=False)

m["  Середня премія, грн.  "] = m['premium.SUM']/m['contracts.SUM']*1000

m['   Доля страхової компанії   '] = m["premium.SUM"]/tt.total(m["premium.SUM"], h['insurer'])
```

Рисунок 3.7 – Приклад обчислюваних заходів в Atoti

Явною перевагою Atoti є відсутність необхідності використовувати в роботі мову запитів і багатовимірні вирази. На рисунку 26 показано, як багатовимірні вирази формуються автоматично під час роботи в інтерактивному середовищі Python.

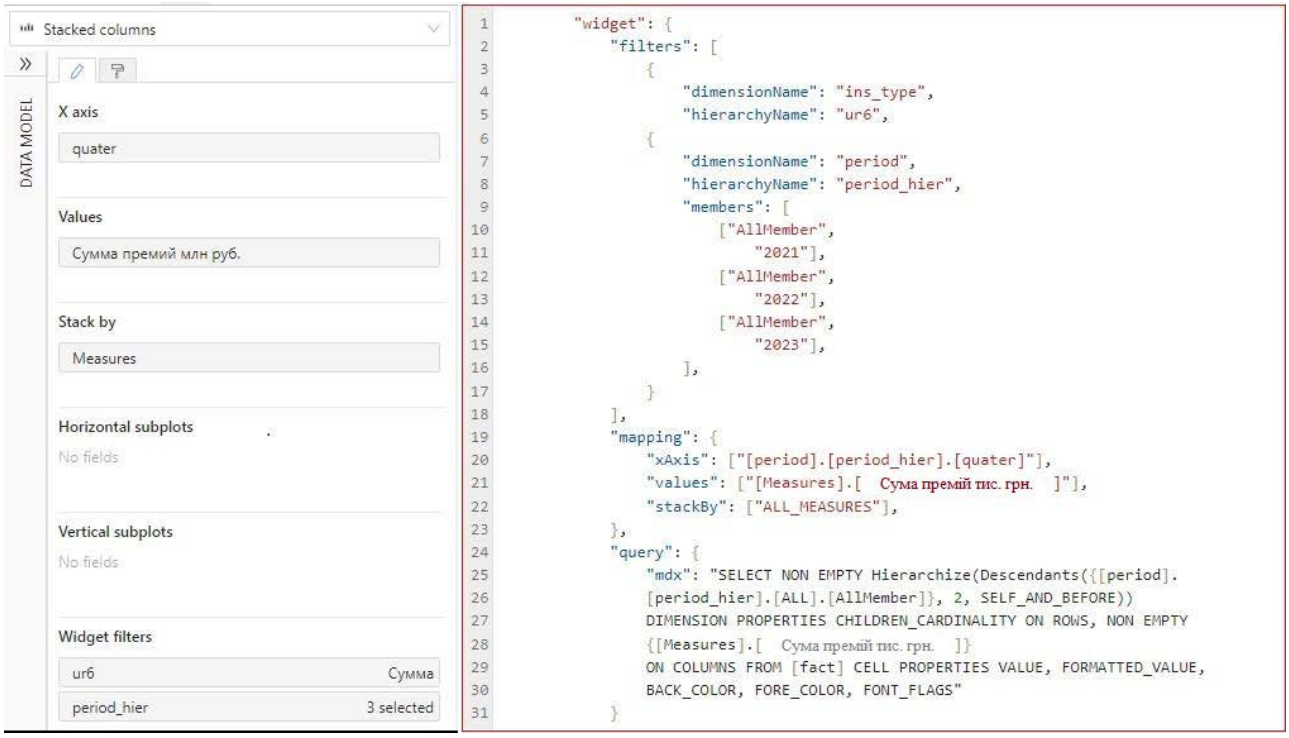

Рисунок 3.8 – Автоматичне формування MDX-запиту в Atoti

Atoti надає різні можливості візуалізації даних безпосередньо в JupyterLab, що дозволяє аналітикам і розробникам проводити аналіз даних прямо в

інтерактивному середовищі. На рисунку 3.8 представлено графік, сформований на основі запиту drag-and-drop, тобто шляхом перетягування та відпускання елементів інтерфейсу для створення запиту даних. Такий спосіб роботи дозволяє користувачам інтуїтивно вибирати та аналізувати дані, не використовуючи мову запитів безпосередньо.

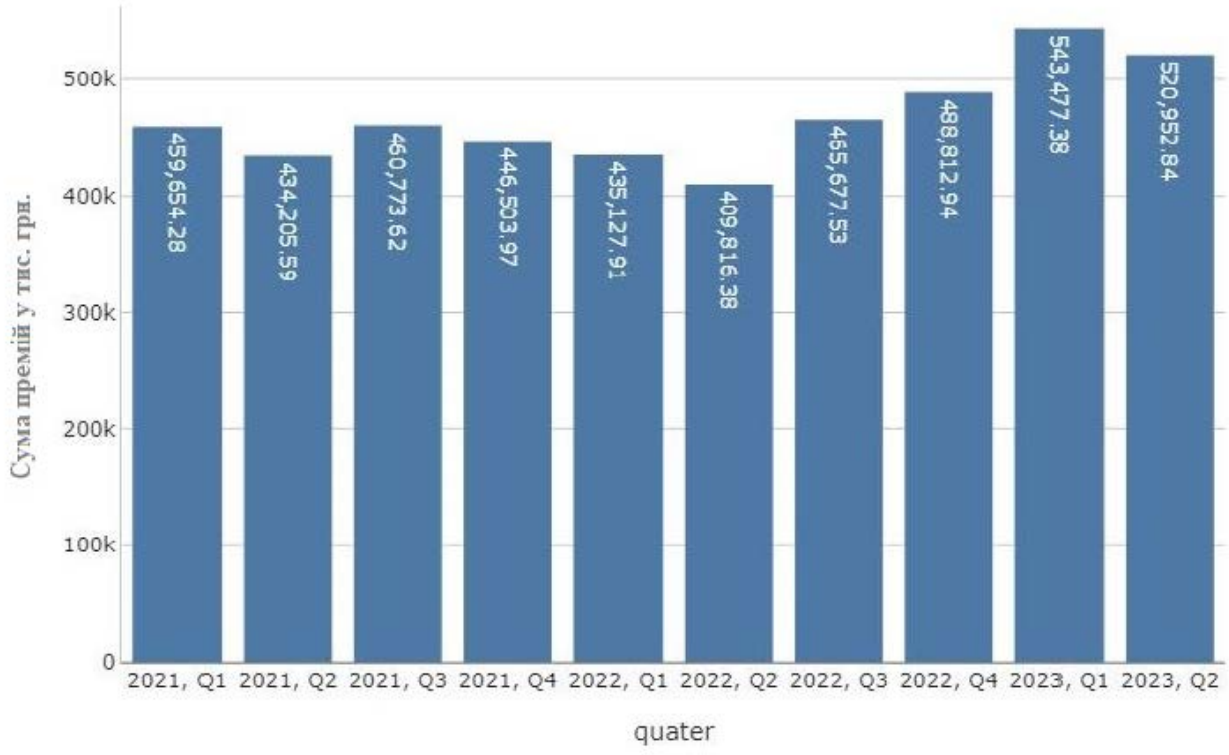

Рисунок 3.9 – Візуалізація запиту в Atoti

Застосунок для візуалізації Atoti є зручним інструментом із великою кількістю функцій, що дозволяє будувати графіки без використання коду. На рисунках нижче представлені основні функції, які є затребуваними аналітиками. На рисунку 3.10 можна побачити, що в Atoti є підтримка складних обчислюваних фільтрів. На графіку будуть відфільтровані компанії, які не входять у топ-10 компаній за сумою виторгу у зазначений період.

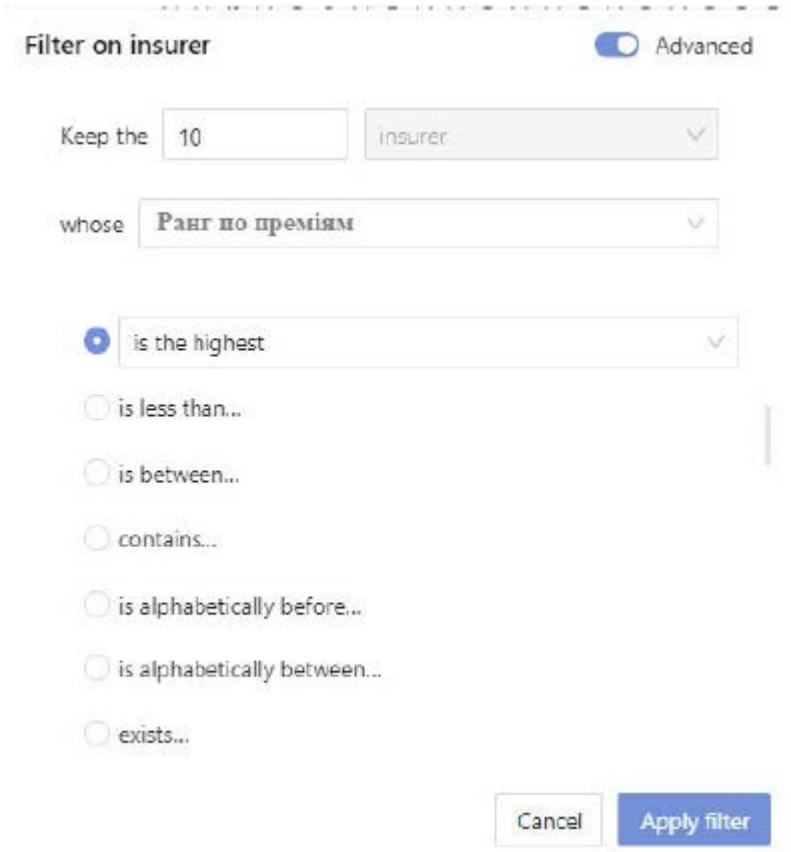

Рисунок 3.10 – Використання складних фільтрів

На рисунку 3.11 показано можливість умовного форматування осередків: більш прибуткові компанії будуть пофарбовані зеленим, інші червоним.

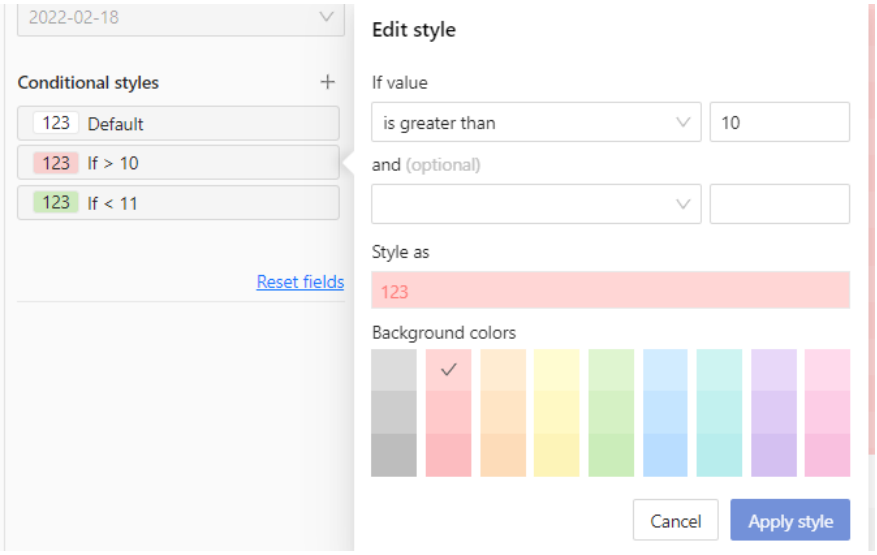

Рисунок 3.11 – Можливість налаштування умовного форматування осередків

На рисунку 3.12 наведені можливості кастомізації графіка: форматування назв осей, легенди, підписи даних, кольорів елементів графіка.

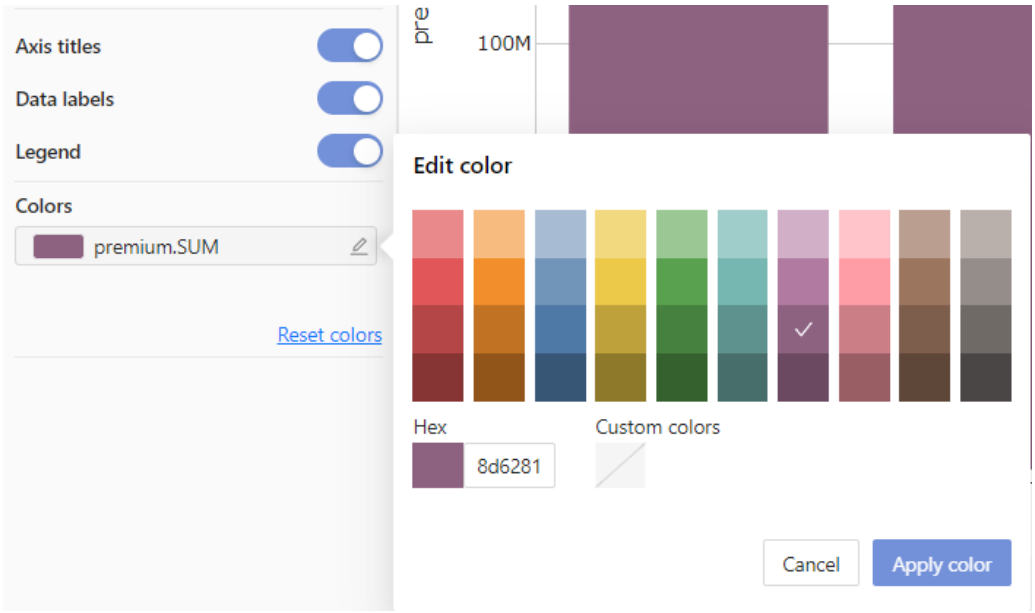

Рисунок 3.12 – Можливості кастомізації графіків

3.4 Оцінка ефективності рішення

3.4.1 Методика оцінки ефективності

Для проведення оцінки ефективності впровадженого рішення було розроблено методику всебічної оцінки продуктивності та бізнес-ефективності систем аналізу даних. Вона враховує такі аспекти:

– Критерії оцінки продуктивності:
– порівняння часу відгуку систем на стандартний набір запитів;
– оцінка надійності системи за тривалого навантаження.
– Критерії оцінки функціональності:
– порівняння часу відгуку систем на стандартний набір запитів;
– аналіз здібностей інтеграції системи з іншими інструментами;
– оцінка інтуїтивності інтерфейсу системи для користувача.

Критерії оцінки бізнес-ефективності:

– аналіз витрат на впровадження та підтримку системи;

– дослідження впливу системи на швидкість і якість бізнес-процесів.

Методи для проведення оцінювання за цією методикою:

– розробка та застосування стандартних бізнес-сценаріїв для імітації реального робочого навантаження;

– проведення опитувань серед користувачів для оцінки зручності використання та впливу на бізнес.

Ця методика являє собою комплексний підхід до оцінки OLAP-систем. Вона враховує як технічні аспекти, так і їхній вплив на бізнес. Важливість такого підходу особливо відчутна для малих компаній, де кожне рішення має бути обґрунтованим і приносити реальну користь. Ми розглядаємо не лише швидкість і надійність системи, а й те, як вона впливає на роботу компанії в цілому.

Сценарії тестування, які наближені до реальних робочих умов, стануть реальним підтвердженням ефективності рішення. Це дозволить підтвердити практичну застосовність у повсякденних завданнях компанії.

Згідно з методикою, необхідно порівняти продуктивність запитів та провести навантажувальне тестування в Atoti, реляційній таблиці та класичному OLAP-кубі Microsoft Analysis Services.

3.4.2 **Збірка OLAP-куба в Visual Studio**

З метою порівняння ефективності впровадженого рішення з класичними підходами до аналізу багатовимірних даних створимо OLAP-куб на основі наявних даних у Microsoft Analysis Services за допомогою Visual Studio. Visual Studio — це інтегроване середовище розробки від Microsoft, призначене для створення різних типів застосунків, включно з проєктами багатовимірних даних. Структура куба у Visual Studio представлена на рисунку 3.13.

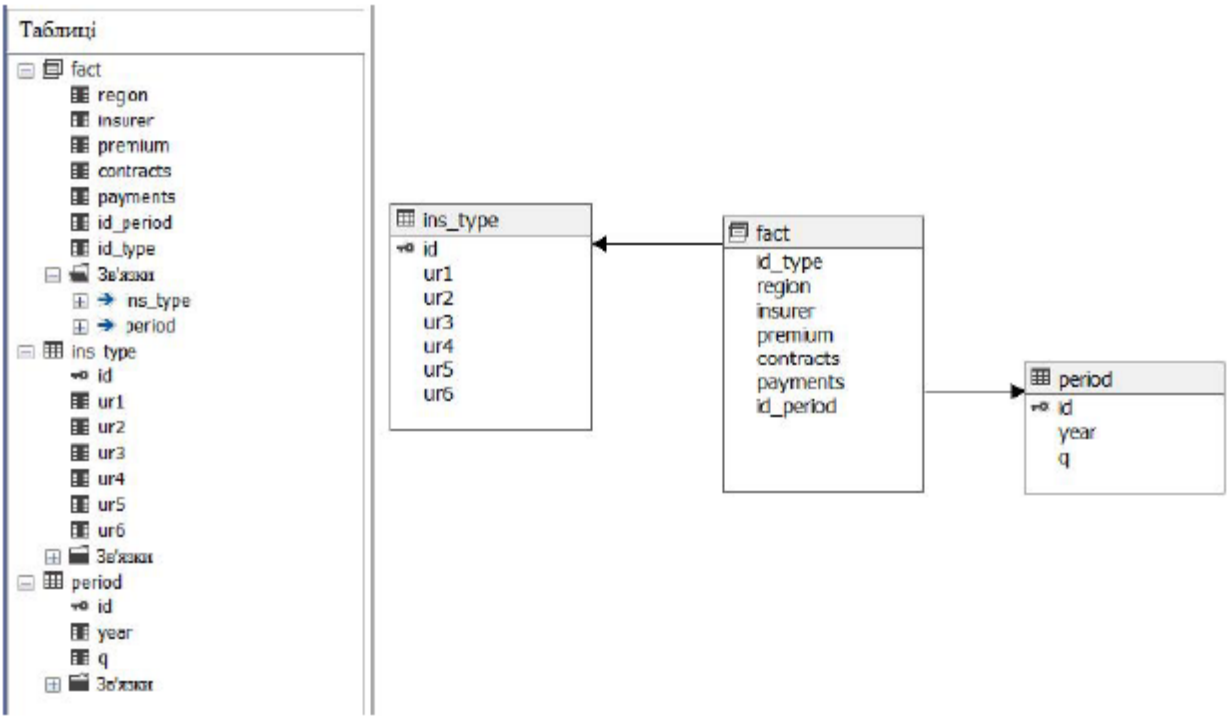

Рисунок 3.13 – Структура OLAP-куба у Visual Studio

Analysis Services також надає можливість визначення ієрархій, як показано на рисунку 3.14, та обчислення заходів користувача, як показано на рисунку 3.15.

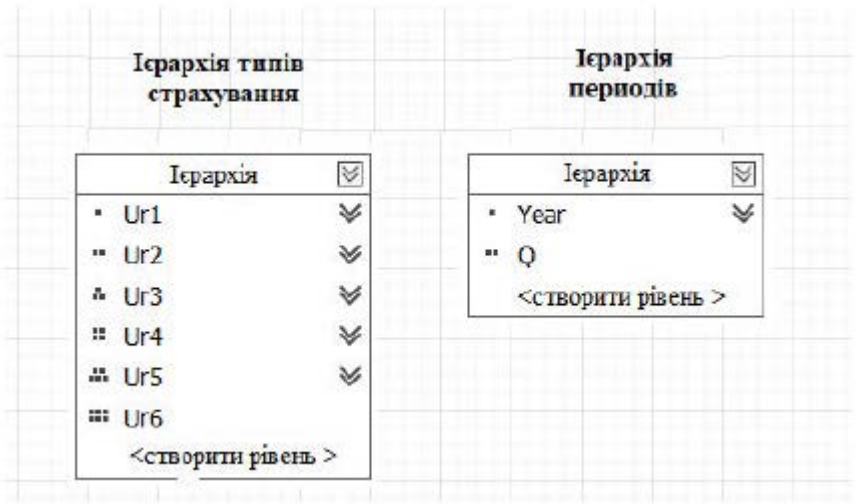

Рисунок 3.14 – Визначення ієрархій у Visual Studio

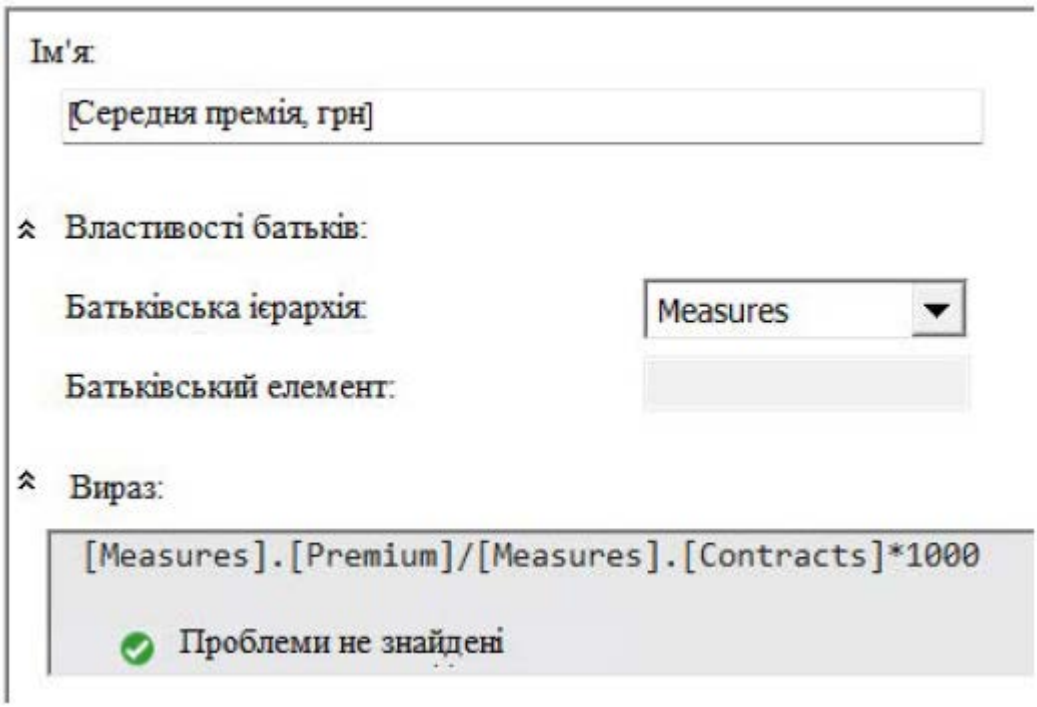

Рисунок 3.15 – Розрахунок обчислюваних заходів у Visual Studio

3.4.3 **Тестування продуктивність запитів**

Запит, який використовується для тестування продуктивності, виконує агрегацію суми зібраних премій по періодах (роках-кварталах) для видів страхування, що належать до категорії "Сума" за певні роки (2021, 2022, 2023). Ця метрика може бути корисною для аналізу динаміки суми премій та ухвалення стратегічних рішень на основі цих даних.

CPU time - це час, який процесор фактично витрачає на виконання коду програми. Він вимірюється в процесорних тактах або циклах і показує загальну кількість часу, протягом якого процесор був зайнятий виконанням коду програми.

На рисунках 3.16 – 3.18 представлено цей запит та його CPU time у досліджуваних продуктах.

```
EXPLAIN ANALYZE
SELECT fact.period, SUM(premium) FROM fact
JOIN period
on fact.period = period.id
JOIN ins_type
on fact.id = ins_type.id
WHERE period.year in (2021, 2022, 2023) and ins_type.Ur6 = 'Сума'
GROUP BY fact.period
```

1	2021-Q1	459654274
2	2021-Q2	434205606
3	2021-Q3	460773626
4	2021-Q4	446503982
5	2022-Q1	435127915
6	2022-Q2	409816397
7	2022-Q3	465677522
8	2022-Q4	488812939
9	2023-Q1	543477395
10	2023-Q2	520952844

Execution Time: 715.875 ms

Рисунок 3.16 – CPU виконання запиту у базі даних PostgreSQL, виражене в мс

ObjectName	Error	ClientProcessID	SPID	CPUTime
			6050	0
			6050	
			6050	0
			6050	
		316	4199	
			4199	0
			4199	0
		316	4199	0

```
SELECT
    { [Measures].[Premium] } ON COLUMNS,
    { [Period].[Id].members } ON ROWS
FROM [CBR2 1 1]
WHERE ( [Ins Type].[Ur6].[Сумма],
          { [Period].[Year].[2021],
            [Period].[Year].[2022],
                [Period].[Year].[2023] } )
```

	Premium
All	370035204
2021-Q1	459654274
2021-Q2	434205606
2021-Q3	460773626
2021-Q4	446503982
2022-Q1	435127915
2022-Q2	409816397
2022-Q3	465677522
2022-Q4	488812939
2023-Q1	543477395
2023-Q2	520952844

Рисунок 3.17 – CPU виконання запиту в Microsoft Analysis Services, виражене в мс

```
from time import process_time

t1_start = process_time()/1000

cube.query(
          cube.measures['Сума премій тис. грн. '],
          levels=[lvl['quater']],
          filter=(lvl['year'].isin('2021','2022','2023')) &
                 (lvl['ins_type', 'ur6', 'ur6'] == 'Сума')
          )

t1_stop = process_time()/1000

print("CPU в мілісекундах:", t1_stop-t1_start)
```

Сума премій тис. грн.

year	quater	
2021	Q1	459,654.28
	Q2	434,205.59
	Q3	460,773.62
	Q4	446,503.97
2022	Q1	435,127.91
	Q2	409,816.38
	Q3	465,677.53
	Q4	488,812.94
2023	Q1	543,477.38
	Q2	520,952.84

```
CPU в миллисекундах: 1.5625000000000014e-05
```

Рисунок 3.18 – CPU виконання запиту Atoti, виражене в мс

Як можна бачити з рисунків 3.16 – 3.18, використання OLAP дозволяє виконати вказаний запит набагато швидше. Це відбувається за рахунок того, що дані в OLAP-кубі попередньо агрегуються і для різних комбінацій вимірювань.

3.4.4 **Навантажувальне тестування рішень**

Бізнес-сценарій для страхової компанії

Метою цього бізнес-сценарію є підвищення ефективності страхових продуктів та покращення задоволеності клієнтів.

Етапи реалізації:

- Виявлення трендів у зібраних преміях за останні три роки. Оцінка того, які періоди є найбільш прибутковими.
- Запит мовою SQL:
- Оцінка того, як змінюється інтерес до добровільного страхування майна протягом року. Визначення кварталів з найбільшим та найменшим попитом.
- Виявлення регіонів з найбільшою та найменшою кількістю укладених договорів для подальшої локалізації маркетингових зусиль.
- Ідентифікація страховиків із найвищими показниками у сумі премій для можливого перерозподілу ресурсів.
- Оцінка того, які види страхування потребують найбільших виплат і можуть мати потребу в перегляді умов або цін.

Результатом виконання сценарію будуть:

- Визначення найбільш і найменш прибуткових сегментів бізнесу.
- Розробка стратегій щодо оптимізації страхових продуктів та управління ризиками.
- Планування бюджету та ресурсів на основі даних про продуктивність продуктів.

Для проведення навантажувального тестування на основі розробленого бізнес-сценарію використано реальні робочі процеси компанії, включно зі створенням тестових сценаріїв, які імітують типові операції, такі як запити на отримання даних, та використання даних, які відображають реальні обсяги операцій страхової компанії, для оцінки здатності системи справлятися з високим навантаженням.

Для проведення тестування використовується спеціалізований застосунок JMeter, який дозволяє перевірити пропускну здатність та стабільність системи. Як зазначено на рисунку 3.19, тест-план у JMeter включає:

- використання SQL-запитів, описаних у бізнес-сценарії, які дозволяють оцінити різні аспекти роботи системи;

- моделювання роботи 60 користувачів, які одночасно відправляють запити до системи;

- моделювання періоду поступового збільшення навантаження. Воно становить 60 с, що означає, що всі 60 користувачів почнуть відправляти запити протягом однієї хвилини. Кожен із користувачів повторює набір запитів 5 разів.

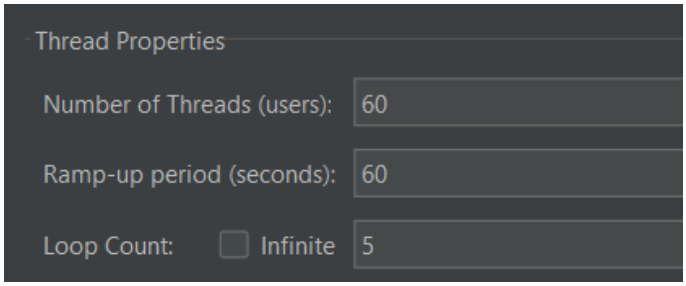

Рисунок 3.19 – Елемент тест-плану в JMeter

Тест-план дозволяє імітувати реальне навантаження на систему та аналізувати її здатність справлятися з високою конкурентністю й інтенсивністю операцій.

На рисунках 3.20 – 3.22 представлені звіти з агрегованими даними про проведення тестування. Пункт «Throughput» — це опис пропускної здатності, вираженої в запитах на секунду, для кожного запиту та загалом для тесту в рядку підсумків «Total».

Label ↑	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
Avg Contracts	70	1198	505	2716	572,60	0,00%	1,2/sec	6,29	0,00	5256,0
Max Premium	48	906	403	1947	383,23	0,00%	48,2/min	0,29	0,00	374,0
Measures by Period	66	1455	613	3572	726,42	0,00%	1,1/sec	0,24	0,00	227,0
Payments by InsType	65	1498	541	3770	827,92	0,00%	1,0/sec	1,77	0,00	1756,0
Property Contracts	51	894	432	2041	391,00	0,00%	51,2/min	0,04	0,00	48,0
TOTAL	300	1221	403	3770	673,38	0,00%	4,8/sec	8,02	0,00	1724,8

Рисунок 3.20 – Оцінка пропускної здатності для навантаження у Postgres

Label ↑	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received K...	Sent KB/...	Avg. Byt...
Avg Contracts	50	5	3	11	1,98	0,00%	52,6/min	19,22	0,79	22443,0
Max Premium	68	5	3	23	2,68	0,00%	1,2/sec	18,54	1,10	16477,0
Measures by Period	60	5	3	13	1,93	0,00%	1,0/sec	10,33	1,03	10408,0
Payments by InsType	54	5	3	15	2,31	0,00%	54,9/min	12,09	0,82	13520,0
Property Contracts	68	5	3	13	1,86	0,00%	1,2/sec	17,64	1,06	15146,0
TOTAL	300	5	3	23	2,20	0,00%	5,1/sec	76,55	4,73	15423,6

Рисунок 3.21 – Оцінка пропускної здатності для навантаження у Microsoft Analysis Services

Label ↑	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughp...	Received K...	Sent KB/s...	Avg. Bytes
Avg Contracts	76	26	3	1175	146,14	0,00%	1,3/sec	17,83	1,10	14093,0
Max Premium	48	4	2	9	1,79	0,00%	50,1/min	0,78	0,66	952,0
Measures by Period	69	4	2	9	1,66	0,00%	1,2/sec	4,34	0,95	3776,0
Payments by InsType	44	4	2	10	1,85	0,00%	46,3/min	2,78	0,59	3692,0
Property Contracts	63	9	2	362	44,75	0,00%	1,1/sec	3,96	0,83	3666,0
TOTAL	300	11	2	1175	76,96	0,00%	5,1/sec	29,29	4,06	5902,4

Рисунок 3.22 – Оцінка пропускної здатності для навантаження в Atoti

На рисунках 3.23 – 3.25 представлені графіки зміни очікування відгуку системи зі збільшенням навантаження кожного запиту.

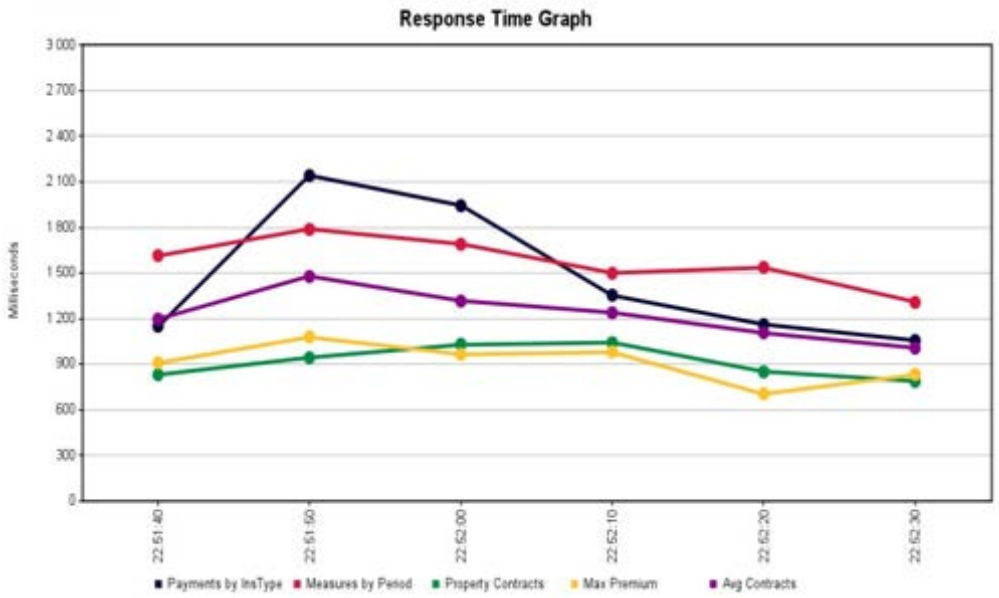

Рисунок 3.23 – Час відгуку системи в Postgres, виражений у мс

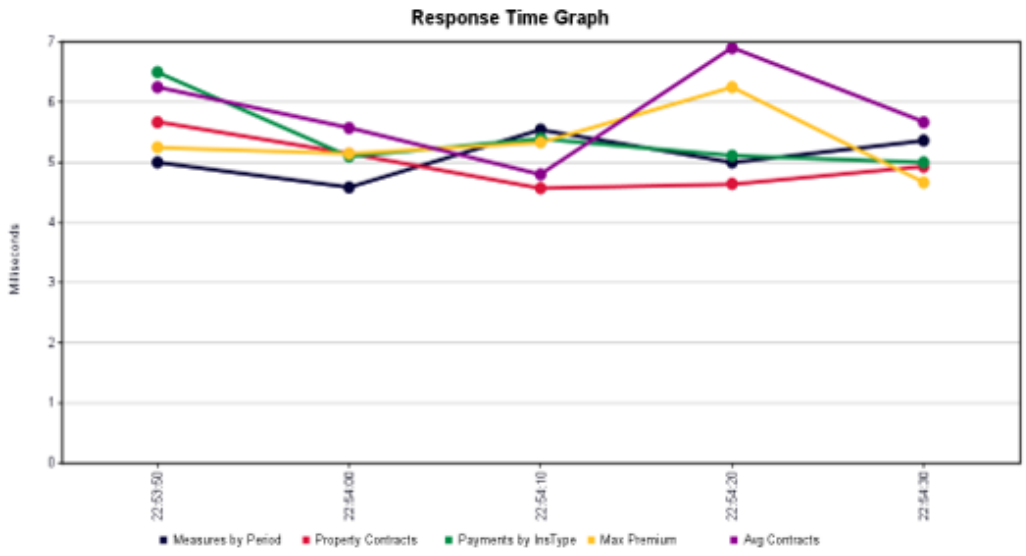

Рисунок 3.24 – Час відгуку системи в Microsoft Analysis Services, виражений у мс

Пропускна здатність Postgres нижча, ніж у продуктах багатовимірного аналізу даних, а очікування відгуку в рази нижче, ніж у конкурентів. Час відгуку в Atoti є досить стабільним, тоді як у Microsoft Analysis Services при збільшенні навантаження спостерігається значне зростання часу отримання відповіді від системи.

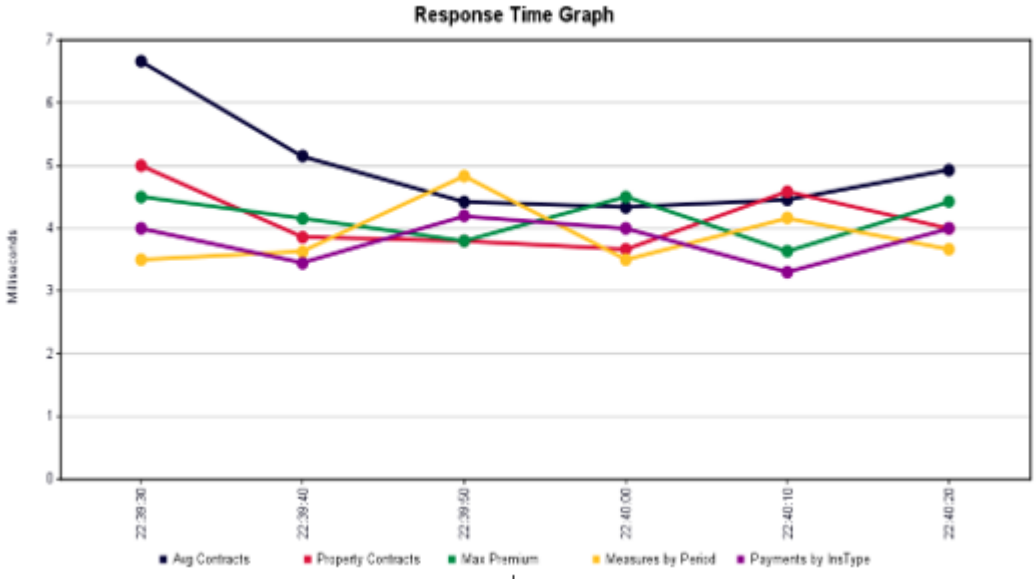

Рисунок 3.25 – Час відгуку системи Atoti, виражене в мс

3.4.5 **Порівняльна оцінка рішень**

У таблиці 3.1 представлено фінальне порівняння продуктів. Оцінка проводилася за такими параметрами:

- *продуктивність запитів.* Те, наскільки швидко система може обробляти запити на аналіз даних, є базовим фактором для можливості проводити аналіз даних ефективно. Це найважливіший параметр для компанії, оскільки низька швидкість запитів, а відповідно й підготовки звіту, робить компанію менш конкурентоспроможною на ринку аналітичних послуг;
- *стійкість системи.* Те, наскільки довго система здатна справлятися з великими навантаженнями, визначає ефективність роботи компанії в частині можливості виконання термінових замовлень;
- *підтримка швидкої зміни вимірів та агрегацій.* Цей параметр також є важливим, оскільки часто потреби бізнесу змінюються дуже швидко. Для

того, щоб ефективно реагувати на зміни, необхідно їх вчасно вловити. Підтримка швидких змін у структурі звіту може суттєво збільшити ефективність аналізу;

- *ієрархічна структура.* Цей параметр також є важливим, оскільки дані компанії містять ієрархії. Проходження рівнями ієрархії може допомогти зробити звіт більш інформативним і допомагає уникнути помилок за рахунок того, що аналітик краще розуміє структуру даних;

- *користувацькі обчислення.* Якщо користувачі можуть створювати свої власні міри, це зробить звіти більш кастомізованими для конкретного бізнесу. Без цього, звіт вийде шаблонним, що не додає йому цінності на ринку аналітичних послуг;

- *інтуїтивний інтерфейс.* Цей параметр не є найбільш пріоритетним, оскільки досвідчені фахівці здатні проводити аналіз без візуальних підказок з боку системи. Однак це може збільшити швидкість підготовки аналізу менш досвідченими фахівцями;

- *візуалізація даних.* Необхідність візуалізації даних звіту в сторонніх системах робить процес аналізу більш тривалим. Швидка візуалізація дозволяє оцінити дані на льоту і, якщо це необхідно, швидко внести зміни в запит;

- *швидкість розгортання та оновлення.* Це один із ключових факторів ефективності системи аналізу даних, особливо, якщо внесення змін у обчислювані елементи можливе тільки з подальшим оновленням куба. Якщо куб розгортається тривалий час, це зробить систему аналізу даних цієї компанії неповороткою машиною. Співробітники компанії будуть змушені вибирати між якістю аналізу та його швидкістю;

- *масштабованість.* Цей параметр дозволяє оцінити, наскільки ефективно компанія зможе обробляти дані, якщо їхня кількість постійно зростатиме;

- *інтеграція з іншими системами* дозволяє спростити роботу з даними, оскільки немає необхідності займатися міграцією даних з однієї системи в іншу. Інтеграція передбачає зручний спосіб отримати функціональність

кількох продуктів в одному сервісі без значних часових витрат.

Таблиця 3.1 - Порівняння ефективності систем аналізу даних

Критерій	Реляційна форма зберігання	Analysis Services	Atoti
Продуктивність запитів	Низька через необхідність виконання безлічі складних SQL-запитів та операцій об'єднання даних	Висока завдяки попередньо обчисленим агрегатам і кешуванню проміжних результатів	Висока завдяки попередньо обчисленим агрегатам і кешуванню проміжних результатів
Стійкість системи	Пропускна здатність низька, час відгуку тривалий, досить стабільний	Пропускна здатність висока, час відгуку низький, нестабільний	Пропускна здатність висока, час відгуку низький, стабільний
Підтримка швидкої зміни вимірів та агрегацій	Можливість обмежена структурою таблиць і стандартними функціями SQL	Середня через обмеження на кількість вимірів та складність додавання нових атрибутів	Висока завдяки гнучкій моделі даних і можливості швидко змінювати структуру кубів
Ієрархічна структура	Не застосовується	Висока, підтримка багатовимірних ієрархій даних	Висока, легко створювати ієрархії з різних вимірів та атрибутів
Користувацькі обчислення	Не застосовується	Висока, користувачі можуть створювати користувацькі міри та показники	Висока, можливість гнучко налаштовувати обчислювані міри та показники
Інтуїтивний інтерфейс	Не застосовується	Висока, інтуїтивний інтерфейс аналізу даних	Висока, простий та інтуїтивно зрозумілий користувацький інтерфейс
Візуалізація даних	Не застосовується	Висока, підтримка різних інструментів візуалізації	Висока, різноманітні інструменти візуалізації даних

Швидкість розгортання та оновлення	Не застосовується	Низька, вимагається перестворення куба при зміні структури або даних	Висока, швидке створення та зміна структури кубів без перезавантаження даних
Масштабованість	Низька, обмежена обсягом і структурою бази даних	Середня, обмежена продуктивністю сервера та пам'яттю	Висока, хороша масштабованість за рахунок оптимізованої роботи з пам'яттю
Підтримка різних типів даних та джерел	Низька, обмежена типами даних і форматами файлів	Середня, підтримка основних типів даних та баз даних	Висока, широка підтримка різних типів даних та джерел

Висновки до розділу:

У контексті вимог, що висуваються до системи аналізу даних компанією, Atoti є найкращим вибором порівняно з реляційною формою зберігання та Microsoft Analysis Services.

Atoti забезпечує високу продуктивність запитів завдяки попередньо розрахованим даним і кешуванню проміжних результатів, має вищу стійкість, ніж конкуренти, та дозволяє створювати ієрархічні структури вимірів.

Користувачі можуть створювати користувацькі обчислення та міри в Atoti, що робить систему гнучкішою та такою, що налаштовується під конкретні потреби бізнесу.

Інтуїтивний інтерфейс та різноманітні інструменти візуалізації даних роблять роботу з Atoti простою та зручною для широкого кола користувачів. Забезпечує високу швидкість розгортання та оновлення кубів, що дозволяє оперативно адаптувати аналітичні рішення до мінливих потреб бізнесу. Цей підхід дозволяє «не лише змінювати куб у реальному часі, а й відійти від необхідності використовувати різні інструменти для створення кубів даних, аналізу даних та візуалізації даних» [5].

Висока масштабованість Atoti забезпечує стабільну продуктивність навіть під час роботи з великими обсягами даних і має широку підтримку джерел, що

дозволяє користувачам аналізувати дані з різних джерел і форматів. Також слід зазначити, що Atoti є продуктом із відкритим вихідним кодом, що дає змогу знизити політичний ризик та забезпечує гнучкість у розвитку та підтримці системи.

ВИСНОВКИ

В ході дослідження була сформульована гіпотеза дослідження: системи аналізу даних на базі продуктів OLAP з відкритим вихідним кодом можуть конкурувати за ефективністю з комерційними продуктами OLAP.

Для підтвердження цієї гіпотези були вирішені наступні задачі:

– Розглянуто теоретичні аспекти роботи інтерактивних аналітичних систем обробки даних OLAP. За результатами виконання даного завдання було встановлено, що однією з основних причин широкого використання технології OLAP в компаніях, що працюють з великими обсягами даних, є висока швидкість обробки запитів, що є ключовим фактором для аналітичних цілей;

– огляд існуючих методів та інструментів багатовимірного аналізу даних OLAP, включаючи як комерційні, так і відкриті рішення, з оцінкою їх функціональності та продуктивності. За результатами виконання цього завдання було з'ясовано, що на ринку представлено безліч продуктів OLAP, які поставляються безкоштовно, і при цьому мають широкий функціонал для аналізу даних. Враховуючи той факт, що відкритий вихідний код робить ці продукти доступними незалежно від політичної та економічної ситуації, вони можуть бути підходящим рішенням для багатьох компаній;

– досліджено актуальність використання технології багатовимірного аналізу даних OLAP. За результатами проведеного аналізу було встановлено, що роль класичного підходу OLAP на сьогоднішній день не така вже й велика. Класичні методи поступаються місцем більш ефективним методам обробки даних, які забезпечують більш гнучкий і масштабований аналіз. Ці технології дозволяють працювати з великими обсягами даних і забезпечують більш високу продуктивність в порівнянні з традиційними методами;

– Проведено аналіз діяльності компанії з метою виявлення проблемних процесів, які потребують аналітичного втручання. В результаті аналізу вимог компанії до системи аналізу даних та особливостей даних, з

якими вона працює, була висловлена рекомендація розглянути можливість впровадження технології багатовимірного аналізу даних OLAP у вигляді інструменту Atoti. Це пов'язано з необхідністю забезпечення високої продуктивності при виконанні аналітичних запитів навіть з великими обсягами даних;

– моделювання та впровадження системи OLAP. Система була успішно змодельована;

– розроблено методику оцінки ефективності систем аналізу даних OLAP з включенням навантажувального тестування. Оцінюється ефективність і функціональність розробленої системи з точки зору відповідності вимогам бізнесу і можливості проведення аналітичних досліджень. Тестування показало, що впроваджена система не поступається за своїми можливостями класичному кубу OLAP і технології, що включає використання OLTP-таблиць для аналізу даних, яка використовувалася в компанії раніше. Крім того, впроваджена система є більш кращою з точки зору ряду параметрів, таких як швидкість змін в системі і інтуїтивність інтерфейсу.

За результатами роботи було визначено такі ключові моменти.

– Технологія OLAP залишається актуальною, незважаючи на те, що сьогодні існують більш високопродуктивні бази даних, що дозволяють не зберігати заздалегідь розраховані дані. Однак такі бази даних застосовні не до всіх структур даних і функцій. Системи OLAP як і раніше користуються попитом, в зв'язку з тим, що їх модифікують, щоб залишатися актуальними;

– Включення навантажувального тестування в методику оцінки ефективності систем OLAP показало його важливість для визначення реальної продуктивності і стабільності систем в умовах інтенсивної експлуатації;

– Використовуючи сучасні інструменти OLAP з відкритим вихідним кодом, такі як Atoti, ви можете ефективно виконувати аналіз даних на основі технології OLAP. Такі послуги можуть дозволити вітчизняним компаніям легше та з меншими витратами трансформувати системи аналітики даних у нинішніх політичних умовах.

Таким чином, дослідження підтвердило гіпотезу про те, що моделювання систем аналізу даних на основі продуктів OLAP з відкритим вихідним кодом може конкурувати за ефективністю з комерційними продуктами.

Отримані експериментальні дані демонструють високу ефективність і продуктивність системи аналізу даних на основі OLAP в порівнянні з альтернативними методами.

Таким чином, результати даного дослідження дають корисні рекомендації щодо використання ефективної та економічно вигідної альтернативи комерційним системам OLAP для компаній, які стикаються з необхідністю заміни раніше використовуваних іноземних рішень в області аналізу даних. Розроблена методика оцінки дозволяє визначити найбільш підходящі системи для конкретних потреб бізнесу з урахуванням стабільності системи. Це може допомогти компаніям скоротити витрати на аналітику, інфраструктуру та підвищити швидкість та ефективність прийняття рішень.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1 Сарсимбаева С. М. Разработка гибридных OLAP систем многомерного анализа данных на основе Microsoft analysis services // Вестник Алматинского университета энергетики и связи. 2019. № 1(44). С. 79-85

2 Agrawal R., Gupta A., Sarawagi S.: Modeling Multidimensional Databases. Int. Conf. on Data Engineering (ICDE), pp.232–243, 1997

3 Atoti. Atoti Documentation: FAQ. 2022 Retrieved from https://docs.atoti.io/latest/getting_started/tutorial/tutorial.html (Дата звернення 27.01.2025)

4 Balke W., Homoceanu S. Data Warehousing & Data Mining - C4 // Institut für Informationssysteme Technische Universität Braunschweig, 2009

5 Benjelloun M. El, Amin E. Impact of using Snowflake Schema and Bitmap Index on Data Warehouse Querying // Int. J. Comput. Appl., vol. 180, no. 15, 2018. pp. 33–35

6 Cabibbo L., Torlone R.: A Systematic Approach to Multidimensional Databases. 5th Italian Symposium on Advanced Database Systems (SEBD), pp.361–377, 1997.

7 Codd E. F., Codd S. B., Salley C. T. Providing OLAP (on-line analytical processing) to user-analysts: An IT mandate. Technical report // Wiley, 1993

8 Codd E. F. Further Normalization of the Data Base Relational Model// Research Report / RJ / IBM / San Jose, California RJ909, 1971

9 Codd E. F. A Relational Model of Data for Large Shared Data Banks// Communications of the ACM Volume 13 Issue, June 1970, pp 377–387.

10 Data Warehouse Architecture: Traditional vs. Cloud: 2019 URL: https://panoply.io/data-warehouse-guide/data-warehouse- architecture-traditional-vs-cloud/ (Дата звернення 27.01.2025)

11 Dageville B. The Snowflake Elastic Data Warehouse // SIGMOD/PODS’ 16 June 26 - July 01, 2016, San Francisco, CA, USA, 2016

12 Gyssen M., Lakshmanan L. V. S.: A Foundation for Multi- Dimensional

Databases. 23rd Int. Conf. on Very Large Data Bases (VLDB), pp.106–115, 1997

13 Jedox Palo Knowlegement Base. 2022 URL: https://knowledgebase.jedox.com/jedox/planning/palo-olap-functions.html (Дата обращения 27.01.2025)

14 Karmani M. Z., Mustafa G., Sarwar N., Wajid S. A Review of Star Schema and Snowflakes Schema. 2020, 13 с.

15 Kimball R., Ross M., The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling // Wiley, 2002

16 Khan W., Khan K., Teerath, Cheng Z. & Raj, Kislay, Roy, Arunabha M., Luo B. SQL and NoSQL Databases Software architectures performance analysis and assessments -- A Systematic Literature review. 10.48550/arXiv.2209.06977.

17 Lapa, Joaquim & Bernardino, Jorge & Figueiredo Ana. A comparative analysis of open source business intelligence platforms // ACM International Conference Proceeding Series. 2014 Linstedt D. (December 2010). Super Charge your Data Warehouse. Dan Linstedt. ISBN 978-0-9866757-1-3

18 Linstedt D. (December 2010). Super Charge your Data Warehouse. Dan Linstedt. ISBN 978-0-9866757-1-3

19 OLAP’s promising path forward. 2021 Retrieved from https://medium.com/atoti/olaps-promising-path-forward- f83cbda9c375 (Дата обращения 27.01.2025)

20 Ouali, Nahla, Tmar, Mohamed, Gargouri, Faiez. A data-centric approach to manage business processes // Computing 98, 2015

21 Pendse N. The OLAP Report: Succeeding with On-line Analytical Processing // Business Intelligence, 1995

22 Ravat F., Teste O., Tournier R., Zurfluh G. Designing and Implementing OLAP Systems from XML Documents. 10.1007/978-0-387-87431- 9_15.

23 Ravat F., Teste O., Tournier R., Zurfluh G. A Conceptual Model for Multidimensional Analysis of Documents. 4801. 550-565. 10.1007/978-3-540-75563-0_37

24 Thomsen C., Pedersen T. A Survey of Open Source Tools for Business

Intelligence // International Journal of Data Warehousing and Mining. 2005

25 Tournier R.: OLAP model, algebra and graphic language for multidimensional databases. Scientific Report n°IRIT/RR2007-6–FR, IRIT, Université Paul Sabatier (Toulouse 3), France

26 Маринич І. А., Тронь В. В. Методичні рекомендації до виконання кваліфікаційної роботи магістра для студентів спеціальності 151 “Автоматизація та комп’ютерно-інтегровані технології”. Кривий Ріг : Видавничий центр КНУ, 2022. 50с.

27 ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура і правила оформлення. Київ, ДП «УкрННЦ», 2015. 26с. (Інформація та документація).

28 ДСТУ 8302:2015. Бібліографічне посилання. Загальні вимоги та правила складання Київ, ДП «УкрННЦ», 2016. 16 с. (Інформація та документація).

29 ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень в українській мові. Загальні вимоги та правила. Київ, ДП «УкрННЦ», 2013. 23 с. (Інформація та документація)

30 ДСТУ 3651.0-97 Метрологія. Одиниці фізичних величин. Основні одиниці фізичних величин Міжнародної системи одиниць. Основні положення, назви та позначення Київ, Держстандарт України, 1998. 27 с. (Інформація та документація).