Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Модель оптимального збереження великих обсягів інформації у гібридних сховищах

(тема)

Виконав:
здобувач 2 року навчання,
групи СПм-23-5
Данило АБРАМОВИЧ
(власне ім'я, прізвище)

Спеціальність
123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма
Системне програмування
(повна назва освітньої програми)

Керівник: доц. Олексій ПІСКАРЬОВ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Абрамовичу Данило Олеговичу
(прізвище, ім’я, по батькові)

1. Тема роботи Модель оптимального збереження великих обсягів інформації у гібридних сховищах

затверджена наказом по університету від “ 21 ” квітня 2025 р. № 296 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 16 червня 2025 р.

3. Вхідні дані до роботи 1) Об’єм даних; 2) Частота доступу; 3) Затримка завантаження; 4) Затримка трансформації; 5) Затримка запиту; 6) Канал передачі інформації.

4. Перелік питань, що потрібно опрацювати у роботі

1) Огляд сучасних архітектур і підходів до зберігання великих обсягів даних.

2) Дослідження структури та моделі даних у системах DWH та Lakehouse.

3) Розробка адаптивної математичної моделі гібридного зберігання даних

4) Моделювання ефективності запропонованої моделі.

5) Висновки

.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд презентація – 24 слайди

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Огляд сучасних архітектур і підходів	22.04.25-1.05.25	
	до зберігання великих обсягів даних.		
2	Дослідження структури та моделі даних у	02.05.25-10.05.25	
	системах DWH та Lakehouse.		
3	Розробка адаптивної математичної моделі	11.05.25-15.05.25	
	гібридного зберігання даних.		
4	Моделювання ефективності запропонованої	16.05.25-19.05.24	
	моделі.		
5	Оформлення матеріалів кваліфікаційної роботи	20.05.25-31.05.25	
6	Подання кваліфікаційної роботи керівникові та	01.06.25-07.06.25	
	її попередній захист		
7	Подання кваліфікаційної роботи на	08.06.25-12.06.25	
	рецензування		

Дата видачі завдання " 22 " квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц., Олексій ПІСКАРЬОВ
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 76 с., 19 рис., 3 дод., 19 джерел.

ГІБРИДНІ СХОВИЩА ДАНИХ, DATA LAKEHOUSE, DATA VAULT, ОПТИМІЗАЦІЯ РОЗПОДІЛУ ДАНИХ, ЛОКАЛЬНЕ ЗБЕРІГАННЯ, СТРІМІНГ ДАНИХ, АРХІТЕКТУРА МЕДАЛЬЙОНУ, ЧАСТОТА ДОСТУПУ ДО ДАНИХ.

Метою кваліфікаційної роботи є розробка та дослідження адаптивної модель оптимального збереження великих обсягів інформації у гібридних сховищах, що забезпечує динамічний розподіл даних між локальними та хмарними середовищами з урахуванням чутливості, частоти доступу та продуктивності системи.

У роботі представлено комплексне дослідження сучасних архітектур та методів зберігання великих обсягів інформації у гібридних сховищах даних. Проаналізовано підходи до організації систем збереження: від традиційних Data Warehouse і Data Lakes до сучасних моделей Data Lakehouse, Data Vault 2.0 та гібридних архітектур. Запропоновано адаптивну математичну модель на базі ADH-DVL, що дозволяє оптимально розподіляти дані між локальними та хмарними середовищами. Проведено моделювання роботи моделі при різних сценаріях навантаження. Результати дослідження підтверджують доцільність застосування розробленої моделі у складних інформаційних системах, де важливо забезпечити баланс між продуктивністю, безпекою та вартістю зберігання даних. Робота створює науково-практичну базу для подальших досліджень та впроваджень інтелектуальних систем оптимізації розподілу даних у гібридних середовищах.

ABSTRACT

Master’s thesis: 76 pages, 19 figures, 3 appendices, 19 sources.

HYBRID DATA WAREHOUSES, DATA LAKEHOUSE, DATA VAULT, DATA DISTRIBUTION OPTIMIZATION, LOCAL STORAGE, DATA STREAMING, MEDALLION ARCHITECTURE, DATA ACCESS FREQUENCY.

The major goal of this thesis is to develop and study an adaptive model for optimal storage of large amounts of information in hybrid storages that provides dynamic data distribution between local and cloud environments, taking into account sensitivity, access frequency, and system performance.

The paper presents a comprehensive study of modern architectures and methods for storing large amounts of information in hybrid data warehouses. The approaches to the organization of storage systems are analyzed: from traditional Data Warehouse and Data Lakes to modern models of Data Lakehouse, Data Vault 2.0 and hybrid architectures. An adaptive mathematical model based on ADH-DVL is proposed, which allows for optimal data distribution between local and cloud environments. The model is simulated under various load scenarios. The results of the study confirm the feasibility of applying the developed model in complex information systems, where it is important to ensure a balance between performance, security and cost of data storage. The work creates a scientific and practical basis for further research and implementation of intelligent systems for optimizing data distribution in hybrid environments.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

КСД – корпоративне сховище даних

МНС – мережа нейронних систем

ПЗ – програмне забезпечення

СЗ – система зберігання

СЗД – система зберігання даних

ХО – хмарні обчислення

AI – штучний інтелект (англ., Artificial Intelligence)

ANN – штучна нейронна мережа (англ., Artificial Neural Network)

API – інтерфейс прикладного програмування (англ., Application Programming Interface)

CDC – захоплення змін даних (англ., Change Data Capture)

DaaS – дані як послуга (англ., Data as a Service)

DWH – сховище даних (англ., Data Warehouse)

OLTP – онлайн транзакційна обробка (англ., Online Transaction Processing)

SCD – повільно-змінювані вимірювання (англ., Slowly Changing Dimensions)

ВСТУП

У сучасну цифрову епоху дані стали основним ресурсом, що визначає ефективність функціонування бізнесу, науки та промисловості. Обсяги інформації стрімко зростають завдяки розвитку Інтернету речей, стрімінгових сервісів, автоматизованих систем, аналітичних платформ та глобальних інформаційних мереж. Однак самі по собі дані, без належної структури, узгодженості та механізмів обробки, не мають практичної цінності. Лише після перетворення їх у релевантну інформацію вони стають основою для прийняття обґрунтованих рішень та оптимізації ресурсів.

Управління великими обсягами інформації потребує ефективних технологічних рішень, що здатні забезпечити цілісність, актуальність, безпеку та доступність даних в умовах змінних навантажень та багатокомпонентних архітектур. Традиційні централізовані сховища даних, що спиралися переважно на реляційні моделі, більше не відповідають сучасним вимогам до масштабованості, гнучкості та продуктивності систем збереження інформації. Розвиток аналітики великих даних, застосування штучного інтелекту та машинного навчання формують нові запити до побудови сучасних архітектурних рішень для роботи з даними. У цьому контексті особливу роль відіграють гібридні архітектури сховищ даних, що поєднують переваги різних підходів: реляційних, масштабованих, адаптивних моделей, стрімінгових технологій обробки змін даних та сучасних платформ [1]. Такі гібридні системи дозволяють ефективно інтегрувати структуровані та неструктуровані дані, забезпечуючи їх узгодженість, оперативний доступ до аналітичних інструментів і гнучку адаптацію. Особливої актуальності набуває питання оптимального розподілу даних між локальними та хмарними середовищами для досягнення балансу між продуктивністю, безпекою, економічною доцільністю та відповідністю нормативним вимогам.

1 АНАЛІТИЧНИЙ ОГЛЯД ПІДХОДІВ ДО СТВОРЕННЯ СХОВИЩ ВЕЛИКИХ ОБСЯГІВ ДАНИХ

1.1 Сховище даних

Сховище даних (Data Warehouse, DWH) — це програмна система для централізованого збереження та управління великими обсягами структурованих даних, які збираються з різних джерел (реляційних баз даних - RDBMS, сховищ даних - Data Lakes, зовнішніх джерел чи файлів) для їх подальшого використання в аналітиці, звітності, прогнозуванні та ухваленні бізнес-рішень [1].

Якщо DWH базується на реляційній моделі, де дані організовані у вигляді таблиць, що складаються зі стовпців та рядків, його називають реляційним сховищем даних [2]. Однак існують і нереляційні сховища, які включають NoSQL-системи, такі як колоночні або графові бази даних. Попри розвиток NoSQL-технологій, реляційні сховища залишаються домінуючими у корпоративному середовищі завдяки багаторічному використанню реляційного підходу у моделюванні даних. Концепція DWH виникла як еволюційний розвиток реляційних баз даних, зберігаючи їх модель. Вона ідеально підходить для структурованих даних і підтримує широко розповсюджену мову запитів SQL, що забезпечує високу сумісність і зручність інтеграції в аналітичні системи.

1.2 Корпоративне сховище даних

Якщо сховище даних використовується всією компанією, його називають корпоративним сховищем даних (Enterprise Data Warehouse, EDW). EDW є розширеною версією DWH, яка охоплює ширший спектр джерел та типів даних для підтримки всіх бізнес-підрозділів організації. Таким чином, EDW виконує функцію єдиного джерела правдивої інформації

(Single Version of Truth, SVOT) для компанії, забезпечуючи узгоджене, централізоване представлення даних (рисунок 1.1).

Усі дані в EDW зберігаються у стандартизованому структурованому форматі, а всі користувачі отримують доступ до однакової інформації, що усуває можливі розбіжності та вирішує проблему розрізнених сховищ даних (data silos).

Основною причиною, через яку компанії впроваджують DWH, є необхідність інтеграції численних джерел даних, тобто консолідації інформації з різних систем. Це дозволяє автоматизувати звітність без втрати продуктивності та усунути хаос у запитах до даних з різних джерел [1].

Фактично, DWH містить аналітику та ключові метрики щодо всіх напрямів діяльності організації, забезпечуючи комплексну інформаційну картину. Це сприяє прийняттю обґрунтованих управлінських рішень у реальному часі. Завдяки EDW користувачі можуть отримувати готові дані без необхідності їх очищення, об'єднання чи застосування складної бізнес-логіки для розрахунків – ці процеси вже реалізовані в DWH.

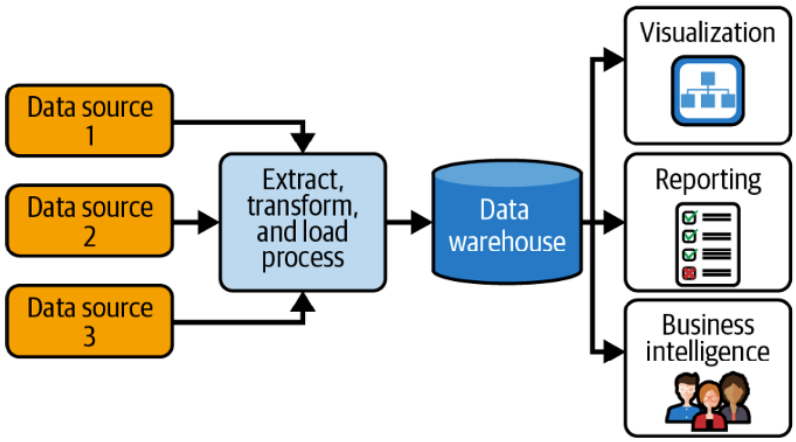

Рисунок 1.1 – Структура корпоративного сховища даних

Крім того, при роботі з корпоративним сховищем даних немає потреби запитувати дозволи на доступ до різних джерел інформації, оскільки всі дані зібрані в єдиній системі з уніфікованими політиками доступу. У сучасних умовах, коли компанії працюють з великими обсягами інформації, гібридні

підходи до побудови EDW набувають все більшої популярності, забезпечуючи гнучкість, масштабованість та ефективне використання ресурсів.

Таке сховище даних дозволяє бізнесу значно підвищити ефективність управління інформацією, її аналізу та використання. Однією з ключових переваг є оптимізація операцій читання. Джерельні бази даних зазвичай розроблені для рівномірного виконання всіх CRUD-операцій (створення, читання, оновлення, видалення), що може впливати на швидкість доступу до інформації. Натомість DWH використовує підхід «write-once, read-many», що робить його ідеальним для аналітичних запитів.

Ще однією важливою перевагою є збереження історичних даних. Залежно від початкових вимог, DWH може містити повну історію змін, що дозволяє аналітикам отримувати звіти за будь-який минулий період без ризику втрати інформації.

Крім того, DWH забезпечує узгодженість у найменуванні таблиць і полів, що особливо важливо для великих організацій, де в різних системах можуть використовуватися нелогічні чи неочевидні назви. Завдяки стандартизованому підходу до іменування, користувачі отримують чітке розуміння структури даних і можуть швидше знаходити необхідну інформацію.

Важливим аспектом є управління майстер-даними (MDM). При консолідації інформації з різних джерел нерідко виникають дублікати, зокрема серед записів про клієнтів, продукти чи постачальників. DWH дозволяє централізовано вирішувати цю проблему, створюючи єдиний набір коректних записів.

Якість даних також покращується завдяки виявленню та виправленню помилок у вихідних системах. DWH не лише очищує інформацію перед її використанням, а й може автоматично сповіщати відповідальних фахівців про знайдені проблеми, допомагаючи усунути їх на рівні джерел.

Ще одним значним плюсом є мінімізація залучення IT-фахівців до

створення звітності. Якщо DWH реалізоване правильно, воно підтримує принцип Business Intelligence (BI) self-service, що дозволяє кінцевим користувачам самостійно отримувати, аналізувати та візуалізувати дані без потреби в додатковій технічній підтримці. Це скорочує час на підготовку звітів і робить бізнес-аналітику більш гнучкою та доступною.

1.3 Еволюція архітектур даних

Щоб зрозуміти, як сховище даних (DWH) взаємодіє з іншими сучасними архітектурами, важливо простежити історичний розвиток систем управління базами даних та методів моделювання інформації.

У період з 1961 по 1970 роки з’явилися перші комерційні системи управління базами даних (СУБД), такі як IMS від IBM та IDS від General Electric. Саме в цей час почалося використання даних у СУБД для аналітики та ухвалення бізнес-рішень, а також зародилося поняття структурованих даних. В основному застосовувалися ієрархічні та мережеві моделі збереження.

З 1971 по 1980 роки відбулося становлення реляційних баз даних (RDBMS) та мови SQL як стандарту для роботи з інформацією. У цей час були закладені основи концепцій ETL (витяг, трансформація, завантаження) та OLTP (операційна аналітика). Запроваджений підхід schema-on-write, який передбачає структурування даних під час запису.

У 1981–1990 роках реляційна модель зазнала подальшого розвитку, з’явилися ER-моделювання та широко поширилися такі RDBMS, як Oracle, IBM DB2 та Microsoft SQL Server [1]. Саме в цей період зародилася концепція DWH, а перші продукти почали реалізовувати багатовимірний аналіз даних. Виникли перші BI-інструменти, зокрема MicroStrategy.

Між 1991 і 2000 роками концепція DWH, запропонована Біллом Інмоном, отримала широке поширення. Бізнес-аналітика стала одним із ключових напрямів в IT-індустрії. У цей період активно розвивалися

технології OLAP, з’явилася мова UML, а в практику впроваджувалися розмірні (dimensional) та об’єктно-орієнтовані моделі даних. Значний розвиток отримали реляційні СУБД, такі як PostgreSQL, що реалізовували об'єктно-реляційну модель з підтримкою наслідування.

У період 2001–2010 років відбувся стрімкий ріст обсягів даних, що сприяло появі концепції Big Data. Запровадження технологій для обробки великих даних, таких як Hadoop, відкрило можливості для розподілених систем збереження. У цей час виникли методи моделювання Anchor Modeling та Data Vault.

З 2011 по 2020 роки спостерігалася інтеграція штучного інтелекту (AI) та машинного навчання в аналітику. Обчислювальні потужності почали переходити в хмарні середовища, а завдяки розробкам Apache (Spark, Kafka, Flink, Airflow) стали можливими масштабовані аналітичні системи. Впровадження технології MPP (масово-паралельної обробки) значно прискорило роботу комерційних СУБД. З’явилися оптимізовані рішення для горизонтального масштабування, як-от Cassandra, MongoDB та Google BigQuery. У цей період виникли концепції Data Lake та Data Fabric. На зміну schema-on-write прийшов підхід schema-on-read, що дозволив гнучкіше працювати з неструктурованими даними.

Починаючи з 2021 року і до сьогодні спостерігається подальший розвиток хмарних технологій та концепції Data as a Service (DaaS), яка передбачає надання даних у вигляді сервісу, при цьому процеси аналітики та машинного навчання (MLOps) стають все більш автоматизованими. У сучасних інформаційних системах набули популярності архітектури Data Lakehouse та Data Mesh [3].

Data Lakehouse поєднує в собі переваги класичного DWH та Data Lake, дозволяючи обробляти як структуровані, так і неструктуровані дані в єдиній платформі. Водночас Data Mesh пропонує новий підхід до управління великими масивами інформації, орієнтуючись на децентралізоване управління доменними даними та їхню незалежну обробку різними

командами.

Таким чином, сховище даних (DWH) залишається фундаментальним компонентом у світі аналітичних систем, але його функціональність значно розширюється завдяки новим технологічним підходам, які дозволяють працювати з даними гнучкіше та ефективніше.

Концепція сховища даних (DWH) виникла як наступний крок у розвитку реляційних баз даних, спрямований на подолання їхніх обмежень [2]. Основні проблеми, з якими зіткнулися RDBMS, включали недостатню продуктивність при виконанні аналітичних запитів та обмеження щодо обсягу даних, що могли ефективно оброблятися.

1.4 Data Lakes

DWH було розроблено частково для вирішення цих проблем. Замість того, щоб виконувати аналітичні запити безпосередньо на реляційних базах, дані копіюються до DWH, де вони оптимізуються для швидкого читання та аналітики. Це дозволяє користувачам створювати звіти та аналізувати дані без навантаження основних операційних баз, які продовжують виконувати свої стандартні функції збереження та обробки транзакцій.

Згодом з’явилися Data Lakes як відповідь на обмеження реляційних сховищ даних (DWH), що включали високі витрати, обмежену масштабованість, низьку продуктивність при обробці великих обсягів інформації, складність підготовки звітів та обмежену підтримку неструктурованих даних [1].

Data Lake — це сховище, яке дозволяє зберігати всі можливі типи даних, включаючи структуровані, напівструктуровані та неструктуровані, без необхідності попереднього моделювання або перетворення (рисунок 1.2). На відміну від традиційних DWH, Data Lake не має власного нативного обчислювального механізму і виступає лише як гнучке сховище даних у їхньому початковому форматі. Цей підхід базується на концепції schema-on-

read, що означає, що структура даних визначається лише під час читання та аналізу, а не при записі.

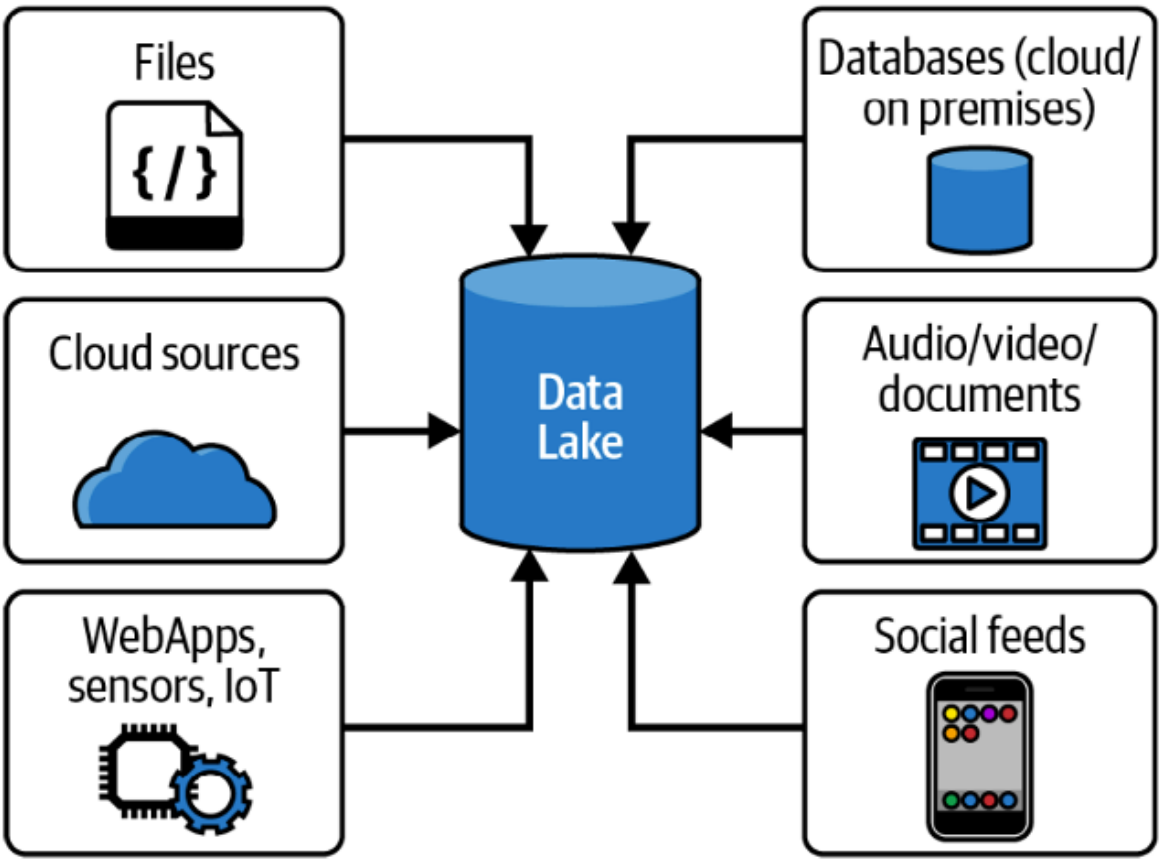

Рисунок 1.2 – Структура сховища Data Lakes

Такий підхід забезпечує високу гнучкість та можливість зберігання величезних обсягів інформації без попередньої оптимізації, що особливо важливо для роботи з потоковими, машинозчитуваними та мультимедійними даними. Однак, відсутність суворої схеми може створювати труднощі у швидкому витягу й обробці інформації, що призвело до появи гібридних рішень, таких як Data Lakehouse.

Компанії, що просували Hadoop і Data Lakes, такі як Cloudera, Hortonworks і MapR, позиціонували ці технології як універсальне рішення для зберігання, обробки та аналізу даних. Вони стверджували, що Data Lakes здатні замінити реляційні DWH, оскільки можуть копіювати, очищувати дані та надавати їх у зручному форматі кінцевим користувачам.

Безумовно, Data Lakes мали кілька важливих переваг. Це був відносно дешевий спосіб збереження необмежених обсягів інформації, що працював як онлайн-архів. Крім того, Data Lakes стали ідеальним місцем для завантаження потокових даних у необробленому вигляді, що особливо важливо для великих організацій, які працюють із логами, телеметрією та іншими великими потоками інформації.

Однак незабаром стало зрозуміло, що повністю замінити DWH не вийде. Основна проблема полягала у складності доступу до інформації: виконання запитів до Data Lakes вимагало технічних навичок, які були далеко не у всіх аналітиків. Для роботи з Data Lakes часто потрібно було знати такі інструменти, як Jupyter, Hive або мови програмування на кшталт Python. Крім того, Data Lakes не забезпечували важливі функції, необхідні для бізнес-аналітики, такі як підтримка транзакцій, управління схемою даних і аудит операцій [4].

Ці недоліки призвели до подальшого розвитку гібридних підходів, таких як Data Lakehouse, які намагаються поєднати переваги сховищ даних (DWH) та гнучкість Data Lakes, надаючи бізнесу як зручність доступу до аналітичних даних, так і ефективне збереження великих обсягів інформації.

1.5 Сучасне сховище даних

Попри всі переваги, Data Lakes не змогли повністю замінити реляційні DWH. Вони виявилися зручними для збереження великих обсягів інформації, але не мали необхідних функцій для якісного аналітичного опрацювання. Проте вони залишилися ефективними для етапів підготовки та зберігання необроблених даних.

Це привело компанії до ідеї поєднати можливості обох рішень: гнучкість і масштабованість Data Lakes з продуктивністю і структурованістю DWH. Таким чином, почали розвиватися гібридні архітектури, у яких Data Lake співіснує поряд із реляційним сховищем, створюючи те, що сьогодні

називають сучасним сховищем даних (Modern Data Warehouse, MDW).

MDW будується на принципах розподіленого зберігання, інтеграції з хмарними технологіями та підтримки різних форматів даних (рисунок 1.3). У цій архітектурі Data Lake використовується як проміжне середовище для збору, зберігання та первинної обробки інформації, а DWH виконує роль системи для глибокого аналізу та бізнес-звітності.

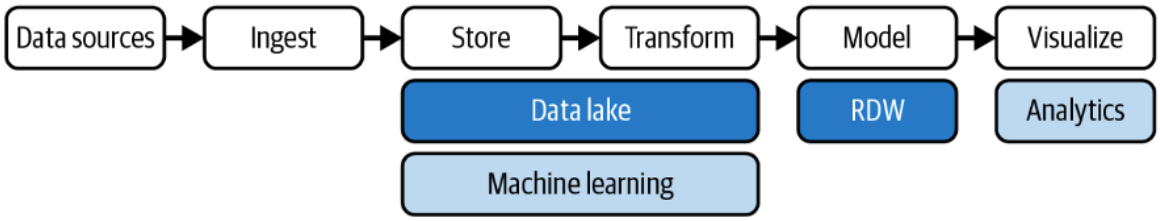

Рисунок 1.3 – Структура сховища Modern Data Warehouse

Завдяки такому підходу компанії отримують найкраще з обох світів: збереження будь-яких типів даних у Data Lake, гнучкість при їхній обробці та ефективне аналітичне використання через DWH. Це дає змогу організаціям швидше отримувати інсайти, підвищувати продуктивність бізнес-аналітики та оптимізувати витрати на зберігання та обробку інформації [2].

Архітектура Modern Data Warehouse (MDW) поєднує в собі переваги Data Lakes та DWH, створюючи ефективну систему управління даними. У цій архітектурі Data Lake використовується для збору, зберігання та підготовки даних, а також для побудови моделей машинного навчання, тоді як DWH слугує основним середовищем для аналітичних запитів, звітності та прийняття управлінських рішень.

1.5 Data Fabric

Однак із зростанням складності даних і потреби в автоматизації процесів управління інформацією почали з’являтися нові підходи. Починаючи з 2016 року, була запропонована архітектура Data Fabric

(рисунок 1.4), яку можна розглядати як еволюцію MDW, доповнену розширеними технологічними можливостями.

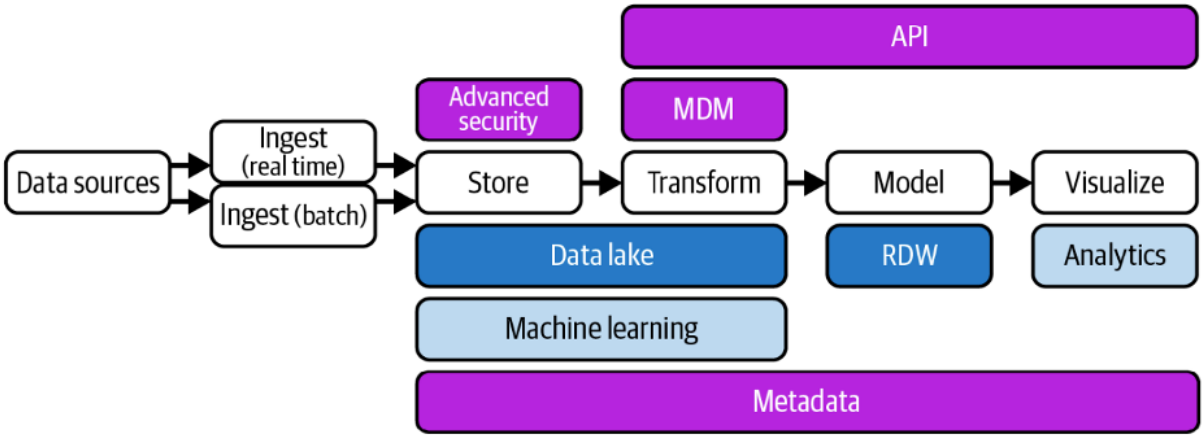

Рисунок 1.4 – Структура сховища Data Fabric

Data Fabric інтегрує обробку даних у реальному часі, вдосконалені політики доступу, централізований каталог метаданих, механізми управління еталонними даними (MDM) та API-інтерфейси для ефективного підключення та використання інформації. Ця архітектура також оптимізує процеси надходження, трансформації, запитів і пошуку даних, роблячи їхню обробку більш швидкою, автоматизованою та масштабованою.

Завдяки Data Fabric компанії отримали можливість більш ефективно працювати з розподіленими джерелами інформації, забезпечуючи безперервний доступ до даних у різних середовищах – локальних, хмарних і гібридних інфраструктурах. Це значно покращило ефективність аналітичних процесів і створило основу для подальшого розвитку автоматизованого управління даними.

1.6 Data Lakehouse

Архітектура Data Lakehouse стала популярною приблизно у 2020 році, коли компанія Databricks представила цю концепцію разом із медальйонною архітектурою (Medallion Architecture). Ідея Data Lakehouse полягає в тому, щоб повністю відмовитися від реляційного DWH та використовувати єдине

сховище даних — Data Lake. У цьому підході всі типи даних, включаючи структуровані, напівструктуровані та неструктуровані, завантажуються безпосередньо в Data Lake, а всі запити та звіти виконуються безпосередньо в ньому ж (рисунок 1.5).

На перший погляд, це схоже на підхід класичних Data Lakes, які намагалися замінити DWH, але зазнали невдачі через низку обмежень.

Ключова інновація Data Lakehouse полягає у додаванні програмного шару транзакційного зберігання, який працює поверх Data Lake і робить його функціонування схожим на реляційну базу даних. Це означає, що система отримує можливість обробляти аналітичні запити з продуктивністю, яка раніше була властива лише DWH, але без необхідності дублювати дані між двома окремими середовищами [3].

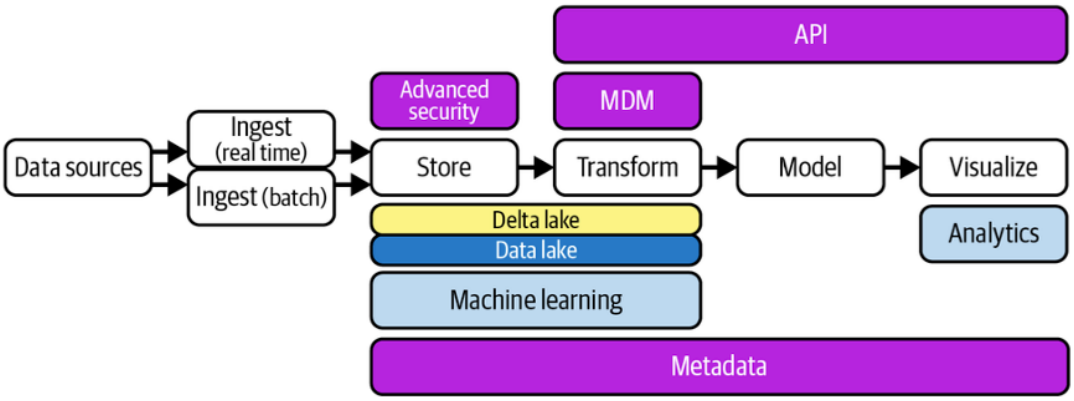

Рисунок 1.5 – Структура сховища Data Lakehouse

Щоб реалізувати цей транзакційний рівень, було розроблено кілька відкритих форматів табличних структур. Найбільш відомі серед них: Delta Lake (від Databricks), Apache Iceberg та Apache Hudi [4].

Головна мета цього рівня — забезпечити ефективну конкуренцію операцій читання та запису без втрати продуктивності та цілісності даних. Завдяки цій технології Data Lakehouse усуває ключові недоліки традиційних Data Lakes, додаючи підтримку транзакцій, керування схемами та покращене управління метаданими.

Таким чином, Data Lakehouse поєднує найкращі риси Data Lake (масштабованість, підтримка різних типів даних, низька вартість зберігання) та DWH (ефективність запитів, контроль цілісності даних, аналітична продуктивність), відкриваючи шлях до нового рівня обробки великих даних.

Крім розглянутих технологій існують й інші. Data Mesh — це концепція децентралізованого управління даними, запропонована Zhamak Dehghani як альтернатива централізованим архітектурам, таким як DWH чи Data Lakehouse. Вона передбачає передачу відповідальності за дані окремим бізнес-доменам, де кожен виступає як незалежний дата-продукт із власною IT-командою. Такий підхід вирішує проблеми власності, якості та масштабування даних [5]. Однак Data Mesh не є універсальним рішенням і потребує радикальної трансформації організаційних процесів, тому підходить лише великим корпораціям із децентралізованими командами. Технології NRT (near real-time) розвиваються у відповідь на зростаючі потреби бізнесу в аналізі даних у реальному часі. Одним із прикладів є Apache Paimon — стримінговий формат, що дозволяє інтегрувати Data Lakehouse із технологіями потокової обробки, такими як Apache Flink та Spark Streaming. Завдяки використанню LSM-дерев, Paimon ефективно підтримує оновлення даних у потоках. З релізом Apache Flink 2.0 можливості цієї інтеграції ще більше зростуть, що зробить Streaming Lakehouse оптимальним рішенням для компаній, яким потрібна автоматизація та аналітика в режимі реального часу.

2 СТРУКТУРА DWH ТА МОДЕЛІ ДАНИХ

2.1 Шарова структура DWH та архітектура «медальйон»

Сучасні Data Lakehouse можуть підтримувати багаторівневу структуру даних, що дозволяє реалізовувати різні стилі моделювання відповідно до функціональних вимог системи. Однією з найпопулярніших архітектур є «медальйонна» (Medallion Architecture), запропонована компанією Databricks.

Концепція медальйонної архітектури (рисунок 2.1) базується на поділі даних на кілька рівнів, які відрізняються ступенем обробки, якістю та структурованістю. Це дозволяє побудувати ефективний конвеєр даних, що проходять поступові етапи очищення, трансформації та оптимізації для аналітичного використання.

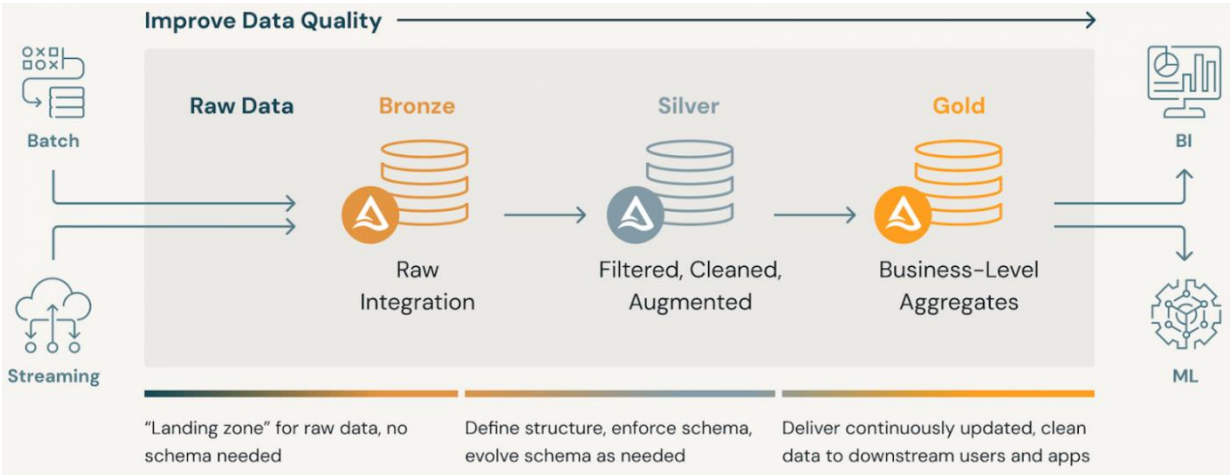

Рисунок 2.1 – Medallion Architecture

Архітектура передбачає три основні рівні [6]:

- Bronze (бронзовий шар) – зберігає сирі, необроблені дані у вихідному форматі. Це може бути потік логів, сенсорні дані, журнали транзакцій або будь-яка інша сировинна інформація.

- Silver (срібний шар) – містить очищені та узгоджені дані. На цьому

рівні усуваються дублікати, виправляються помилки, а також застосовуються базові трансформації, щоб структурувати інформацію для подальшого використання.

- Gold (золотий шар) – включає агреговані, оптимізовані та готові до аналітики дані, які використовуються для звітності, BI-аналітики та машинного навчання.

Цей підхід забезпечує гнучкість у роботі з великими масивами інформації та дозволяє поєднувати Data Lakehouse з класичними методами моделювання даних.

Медальйонна архітектура усуває обмеження традиційних Data Lakes і DWH, забезпечуючи ефективну організацію даних для потокової обробки та бізнес-аналітики. У межах Data Lakehouse використовуються різні підходи до побудови шарів даних залежно від завдань і типу даних. Крім поділу на бронзовий, срібний і золотий рівні, також застосовуються моделі, як-от Data Vault, що дозволяє масштабувати систему без втрати даних, або Anchor Modeling для зручного відстеження змін [7].

Dimensional Modeling залишається актуальним для аналітики та звітності. У рамках Data Lakehouse можливе комбінування підходів: на рівні DDS часто застосовують Data Vault для збереження історії змін, а для Data Mart — розмірне моделювання для швидкого доступу до аналітичної інформації. Такий гібрид забезпечує баланс між гнучкістю, масштабованістю й продуктивністю, дозволяючи системі адаптуватися до бізнес-вимог і ефективно працювати з різними типами навантаження.

2.2 Проєктування реляційного DWH у рамках Data Lakehouse

Щоб зрозуміти, як побудувати DWH як концепцію або частину Data Lakehouse, важливо розглянути ключові технології проєктування, його основні етапи та методи реалізації.

При створенні DWH з нуля зазвичай застосовується top-down підхід (не

слід плутати з top-down методом за Біллом Інмоном). Цей підхід добре підходить для звітності та історичної аналітики, коли кінцеві користувачі потребують відповідей на питання: що сталося? (описова аналітика) та чому це сталося? (діагностична аналітика). У цьому випадку спочатку формується загальне бачення та стратегія розробки, визначаються цілі системи, після чого створюються її конкретні компоненти.

На відміну від цього, при побудові Data Lake часто застосовується протилежний bottom-up підхід, де спочатку накопичуються необроблені дані, а потім формується їхня структура відповідно до бізнес-потреб.

Проєктування DWH починається з аналізу бізнес-вимог до звітності та KPI (ключових показників ефективності). На основі цих вимог формується загальна архітектура DWH, визначаються його рівні, моделі даних та механізми інтеграції з іншими системами. Це є основою для розробки технічних вимог.

Наступним етапом є моделювання даних, яке є ключовим у процесі проєктування DWH. Вибір відповідної моделі визначає, як будуть виглядати каталоги, схеми, таблиці та поля в системі. Правильно розроблена модель даних фактично є технічним завданням на структурування інформації в DWH.

Після створення моделі формується ETL-процес — механізм вилучення даних з різних джерел, їх перетворення у потрібний формат та завантаження в DWH. Це критично важливий етап, оскільки саме він забезпечує коректність, цілісність і узгодженість інформації.

На базі готової структури впроваджуються BI-інструменти, що дозволяють кінцевим користувачам ефективно працювати з даними, створювати звіти, аналітичні панелі та прогнозні моделі [7].

Після запуску система проходить тестування, оптимізацію та масштабування відповідно до змін бізнес-вимог та технічних обмежень.

Оскільки моделювання даних є одним із найважливіших етапів, необхідно ретельно підходити до вибору методології проєктування.

Структура повинна бути логічною, ефективною, забезпечувати високу продуктивність та спрощувати процеси витягу й аналізу даних. Грамотно побудоване DWH дозволяє компаніям отримувати корисну бізнес-аналітику, приймати обґрунтовані рішення та підвищувати ефективність управління даними.

2.3 Процес розробки моделі даних для DWH

Розробка моделі даних є критично важливим етапом проєктування DWH, оскільки вона визначає, як дані будуть організовані, інтегровані та використовувані в системі. Загалом цей процес включає кілька послідовних етапів.

Першим етапом є збір вимог. Найефективнішим методом у цьому контексті є report-driven підхід, орієнтований на звітність. Він передбачає аналіз потреб кінцевих користувачів у даних через дослідження вимог до звітності. Для цього проводяться індивідуальні або групові інтерв'ю із зацікавленими сторонами – бізнес-аналітиками, користувачами BI-систем, технічними експертами. Зібрані вимоги узагальнюються та перевіряються на відповідність існуючим джерелам даних. Після затвердження цієї інформації вона стає основою для процесу моделювання та подальшого розвитку BI-системи [3, 8].

Другим кроком є концептуальне моделювання, яке передбачає створення високорівневої моделі даних. Вона відображає ключові об'єкти та їхні взаємозв'язки. На цьому етапі обирається схема представлення даних або методологія нотації, будуються ER-діаграми (Entity-Relationship), визначаються сутності та їхні основні атрибути, встановлюються взаємозв'язки між ними.

Далі слідує логічне моделювання, яке передбачає деталізацію концептуальної моделі відповідно до вимог системи. Визначаються атрибути сутностей, первинні та зовнішні ключі, обмеження цілісності даних, рівень

нормалізації. Відбувається зіставлення атрибутів із сутностями та встановлення правил зв’язків.

На етапі фізичного моделювання логічна модель трансформується у фізичну структуру бази даних з урахуванням особливостей архітектури DWH. Визначаються таблиці, їхні поля, зв’язки, індекси, тригери, процедури та інші об’єкти бази даних. Якщо система побудована на Big Data-технологіях, фізичне моделювання може бути менш критичним або навіть не використовуватися, оскільки такі системи часто працюють з гнучкими schema-on-read підходами.

Окремо важливим етапом є документування моделі, яке включає опис структури даних, таблиць, відносин, атрибутів та інших компонентів DWH. Це забезпечує зрозумілість системи для всіх учасників процесу – від розробників до бізнес-аналітиків.

Останнім етапом є обслуговування та оновлення моделі. Зміни у бізнес-вимогах можуть вимагати адаптації моделі даних, тому необхідно регулярно проводити моніторинг її ефективності та вносити відповідні коригування.

Таким чином, модель даних є основою продуктивного, масштабованого та ефективного DWH, яке забезпечує швидкий та точний доступ до аналітичної інформації.

2.4 Основні підходи до моделювання даних у DWH

Проєктування DWH базується на різних методах моделювання даних, серед яких найчастіше застосовуються реляційне, розмірне та Data Vault 2.0.

Реляційне моделювання спирається на принципи реляційних баз даних, де інформація організована у вигляді нормалізованих таблиць. Такий підхід забезпечує збереження цілісності даних і мінімізує дублювання, що особливо важливо для транзакційних систем (OLTP). Проте, у контексті аналітичних навантажень (OLAP), виконання запитів у нормалізованій структурі може бути менш ефективним через велику кількість об'єднань між таблицями.

Розмірне моделювання використовується для оптимізації аналітичних запитів. Воно передбачає побудову структури за схемою «зірка» або «сніжинка». У схемі «зірка» всі вимірювальні таблиці напряму пов’язані з фактами, що спрощує запити та підвищує продуктивність. Схема «сніжинка» є більш нормалізованою версією, у якій вимірювання розбиваються на додаткові таблиці, що зменшує дублювання, але ускладнює запити. Цей підхід широко застосовується у BI-системах та DWH через його зручність у побудові аналітичних звітів [9].

Метод Data Vault 2.0 поєднує найкращі риси реляційного та розмірного моделювання, забезпечуючи гнучкість, масштабованість та можливість простого відстеження історії змін даних. Він базується на розподілі інформації між трьома основними структурами: хабами, які містять унікальні ключі бізнес-об'єктів, лінками, що визначають зв’язки між об'єктами, і сателітами, які зберігають історичні дані та атрибути. Завдяки такій структурі цей метод добре підходить для складних розподілених систем, які постійно розширюються.

Кожен із підходів має свої переваги, і вибір залежить від конкретних вимог проєкту. В аналітичних системах нерідко комбінуються різні методи моделювання, щоб досягти оптимальної продуктивності та зручності роботи з даними.

2.4.1 Реляційне моделювання

Реляційне моделювання, розроблене Едгаром Ф. Коддом у 1970 році, базується на організації даних у вигляді таблиць і визначенні зв’язків між ними. Взаємозв’язки між таблицями, їхніми рядками та стовпцями встановлюються за допомогою первинних і зовнішніх ключів. Первинний ключ є унікальним ідентифікатором кожного запису в таблиці, що гарантує відсутність дублювання значень. Зовнішній ключ — це атрибут у дочірній таблиці, який посилається на первинний ключ батьківської таблиці,

забезпечуючи цілісність даних.

Процес реляційного моделювання зазвичай починається з побудови ER-діаграми (Entity-Relationship), що є високорівневим представленням сутностей та їхніх взаємозв’язків. Це дозволяє систематично описати структуру бази даних перед її фізичною реалізацією [10].

Наступним кроком є застосування правил нормалізації, які використовуються для поділу даних на менші, більш керовані таблиці. Нормалізація мінімізує надлишковість, знижує залежність між даними та підвищує їхню цілісність. Виділяють шість рівнів нормалізації, які послідовно застосовуються до необхідного рівня. Найчастіше в реляційних моделях використовується третя нормальна форма (3NF). Таблиця відповідає 3NF, якщо вона задовольняє умовам 1NF і 2NF, а також якщо всі її атрибути функціонально залежать лише від первинного ключа та не мають транзитивних залежностей, тобто коли один атрибут не залежить від іншого, який не є ключем.

Для ведення історичних змін даних використовуються спеціальні історичні таблиці. Вони зазвичай є копіями вихідних таблиць із додатковими атрибутами для відстеження змін, такими як часові мітки, ідентифікатори користувачів і значення «до» та «після» для кожного стовпця.

Реляційна модель найчастіше реалізується у вигляді OLTP-баз даних із використанням СУБД таких як Oracle, Microsoft SQL Server, MySQL, PostgreSQL. У контексті Data Lakehouse реляційні бази виступають джерелами даних для Data Lake або можуть бути використані в структурованих аналітичних сховищах.

2.4.2 Розмірне моделювання

Розмірне моделювання, запроваджене у 1996 році, оптимізує аналітичні запити шляхом структурування даних у таблиці фактів і вимірювань. Воно базується на реляційній моделі, але спрощує побудову запитів та покращує

продуктивність, особливо при великій кількості таблиць. Факти — це числові показники (наприклад, продажі чи виручка), які можна агрегувати, а вимірювання описують ці факти (час, клієнт, продукт) і зазвичай організовані в ієрархії для глибшого аналізу.

Замість природних ключів використовується сурогатний — унікальний штучний ідентифікатор, що забезпечує стандартизацію, уникає дублювання та захищає конфіденційні дані. Природні ключі можуть бути надто складними або ненадійними. Завдяки розмірному моделюванню BI-системи та DWH можуть швидко генерувати зведену інформацію для прийняття рішень, що робить цей підхід популярним у бізнес-аналітиці.

2.4.3 Повільно-змінювані вимірювання

У розмірному моделюванні для збереження змін у таблицях вимірювань застосовується підхід Slowly Changing Dimensions (SCD). Він дозволяє керувати історичністю даних залежно від потреб бізнесу. SCD Type 1 оновлює значення без збереження історії — підходить для несуттєвих змін, як-от номер телефону. Натомість SCD Type 2 зберігає історію, створюючи новий рядок з оновленими значеннями та атрибутами для фіксації часу дії запису.

Вибір типу SCD залежить від потреб у збереженні історії: для глибокого аналізу використовують Type 2, для простих оновлень — Type 1. На відміну від історичних таблиць у реляційній моделі, SCD зосереджується лише на вимірюваних атрибутах, важливих для аналітики, і зберігає зміни з меншою деталізацією [11].

Основна різниця між моделями 3NF та розмірними моделями полягає у ступені нормалізації. У розмірному моделюванні застосовується денормалізація, що передбачає включення надлишкових копій даних у кілька таблиць. Це дає змогу скоротити кількість необхідних таблиць та прискорити виконання запитів, оскільки мінімізується кількість об'єднань (JOIN). Менша

кількість об’єднань робить роботу з даними більш зручною для кінцевих користувачів при створенні звітів та аналітичних панелей.

Проте використання денормалізації накладає додаткові вимоги щодо підтримки цілісності даних. Надлишкові копії повинні бути синхронізовані між таблицями, що вимагає ретельно налаштованого процесу інтеграції та завантаження інформації (ETL/ELT). Це забезпечує узгодженість даних і гарантує, що зміни в одній частині моделі вірно відображаються у всій системі.

У розмірному моделюванні найчастіше використовуються дві популярні схеми: схема «зірка» (star schema) та схема «сніжинка» (snowflake schema).

Схема «зірка» є найбільш поширеним способом організації даних у DWH. У цій моделі таблиця фактів розташована в центрі структури, а навколо неї знаходяться таблиці вимірювань, які пов’язані з нею через зовнішні ключі. Такий підхід забезпечує простоту та високу продуктивність виконання запитів, оскільки мінімізує кількість необхідних об'єднань (JOIN). У цій схемі таблиця фактів містить числові метрики або агрегати, які можуть бути підраховані або проаналізовані, наприклад, сума продажів чи кількість замовлень. Таблиці вимірювань зберігають описові атрибути, такі як назви продуктів, інформацію про клієнтів, дати операцій та географічні дані.

Основна перевага схеми «зірка» (рисунок 2.2) полягає у швидкості виконання запитів і зручності роботи з даними для бізнес-аналітики. Вона добре підходить для BI-систем, у яких користувачі часто формують звіти та аналітичні панелі, а також дозволяє ефективно оптимізувати аналітичні запити за рахунок використання індексів та агрегованих даних.

На відміну від схеми «зірка», схема «сніжинка» використовує нормалізовану структуру вимірювань, розбиваючи їх на кілька взаємопов’язаних таблиць. Це означає, що атрибути вимірювань поділяються на підкатегорії та зберігаються в окремих таблицях, що зменшує дублювання даних та полегшує їхнє оновлення.

Основна перевага цього підходу полягає у зниженні обсягу збережених даних та спрощенні процесу оновлення окремих атрибутів, оскільки зміни відбуваються в одній таблиці, а не у всіх, які можуть містити дублікати. Однак, чим вищий ступінь нормалізації вимірювань, тим складнішою стає структура даних, що може ускладнити написання аналітичних запитів та знизити продуктивність через збільшення кількості об'єднань (JOIN).

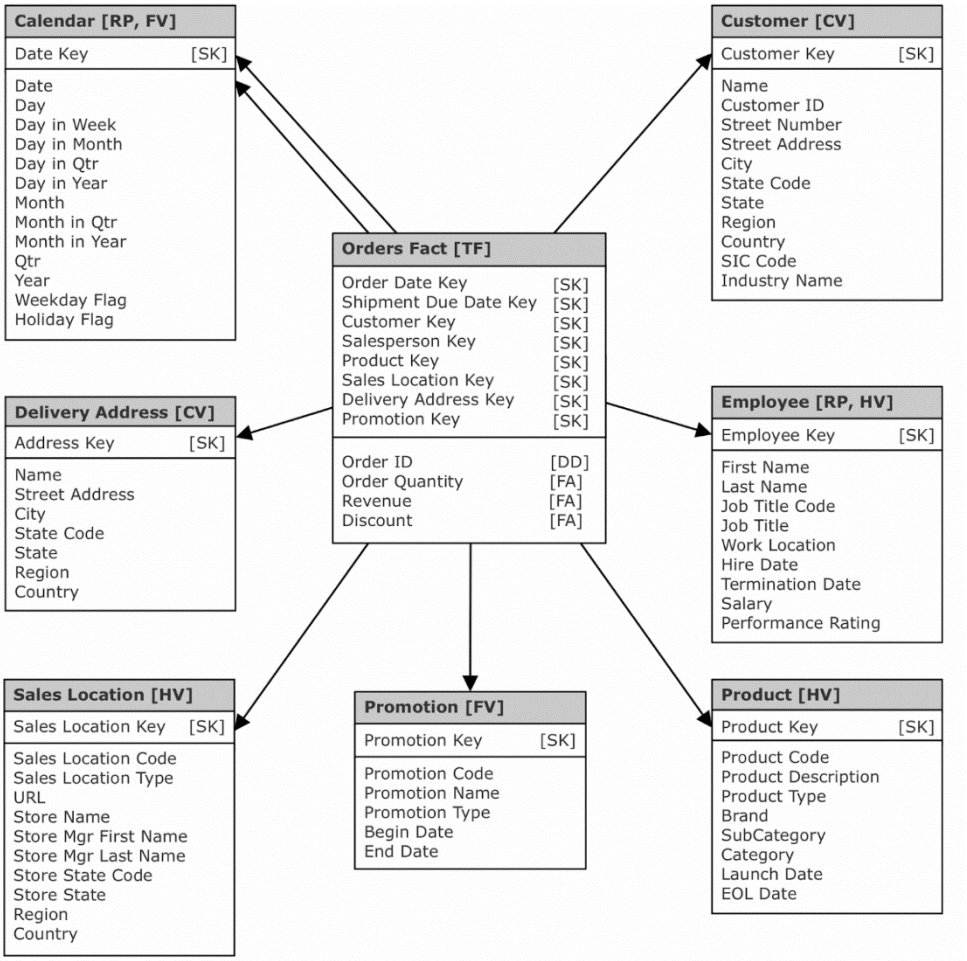

Рисунок 2.2 - Схема «зірка»

Схема «сніжинка» (рисунок 2.3) добре підходить для сценаріїв, де важливіша економія місця та структурованість даних, ніж швидкість запитів. Вона часто застосовується у випадках, коли обсяг вимірювань значний і є необхідність часто оновлювати довідкові дані без ризику їхньої неузгодженості.

Розмірні моделі, так само як і 3NF-моделі, можуть бути представлені у

вигляді ER-діаграм. Вони відображають структуру DWH, показуючи зв’язки між фактами, вимірюваннями та іншими аналітичними об’єктами [4].

Коли говорять про розмірне моделювання у DWH, зазвичай згадують Ральфа Кімбала— популяризатора цього підходу та автора книги «The Data Warehouse Toolkit». Його методологія та концепція розмірного моделювання стали стандартом у проєктуванні DWH, і ці терміни часто використовують як синоніми.

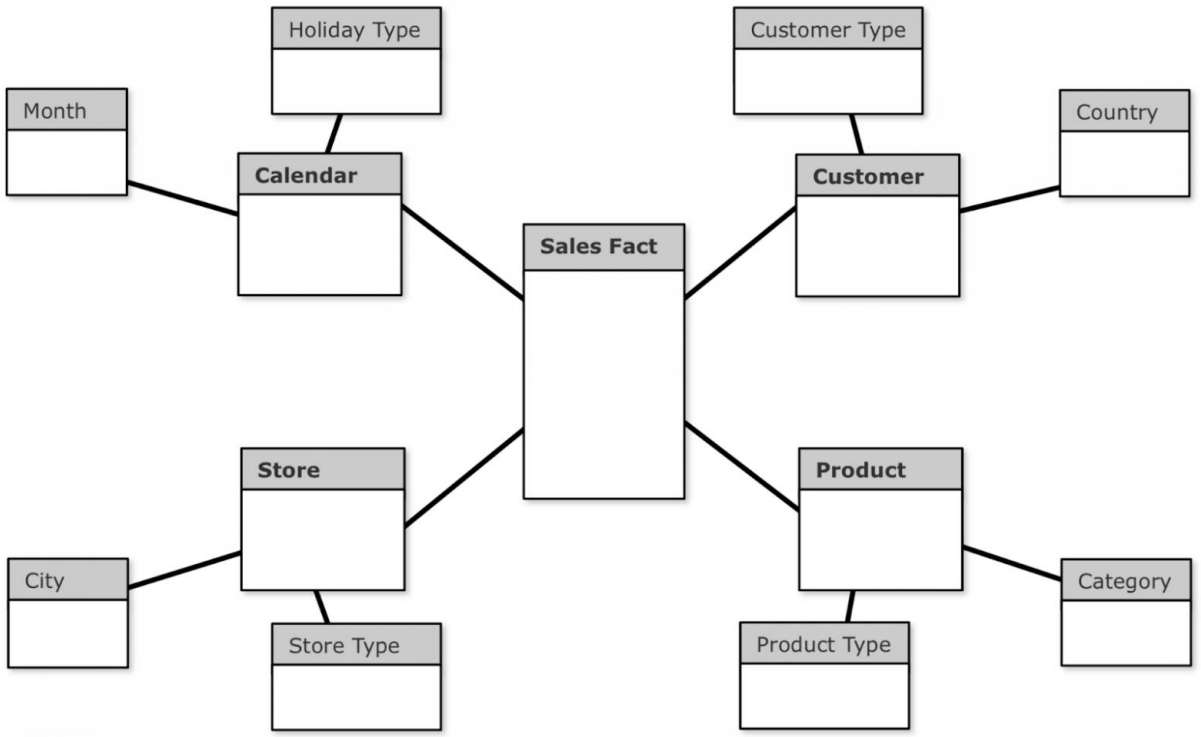

Рисунок 2.3 - Схема «сніжинка»

Кімбалл розробив широкий набір методів і концепцій, які стали основою аналітичних сховищ (рисунок 2.4). Серед них погоджені вимірювання, повільно змінювані вимірювання (SCD), мусорні вимірювання, міні-вимірювання, місткові таблиці (Bridge Tables) та таблиці фактів періодичних знімків (Periodic Snapshot Fact Tables).

Крім того, Кімбалл запропонував власний підхід до побудови DWH, який базується на bottom-up методології. На відміну від top-down підходу Білла Інмона, у якому спочатку формується корпоративне сховище даних, а

потім створюються аналітичні вітрини, методологія Кімбалла починається з витягу сирих даних із OLTP-систем у тимчасові реляційні стейджингові таблиці без попереднього перетворення чи очищення.

Далі йде етап інтеграції, очищення та забезпечення якості даних, після чого вони організовуються у вигляді розмірної моделі з таблицями фактів і вимірювань. Завершальним етапом є створення аналітичної структури, що оптимізує доступ до інформації та підтримує BI-аналітику.

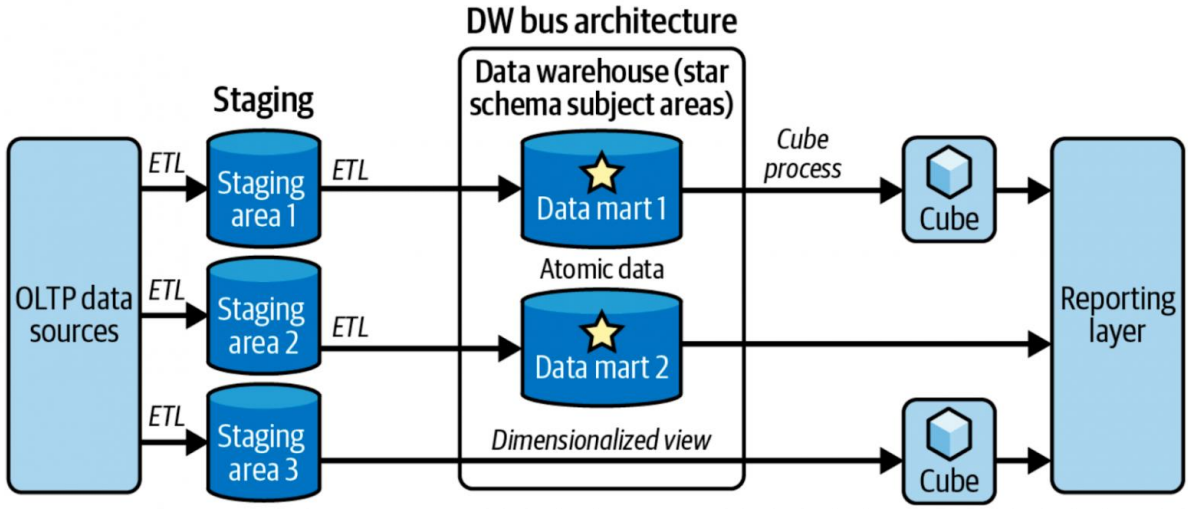

Рисунок 2.4 - Побудова DWH за методом Кімбалла

Методологія Кімбалла дозволяє швидко реалізовувати окремі аналітичні рішення, які можуть поступово інтегруватися в єдине сховище. Завдяки своїй гнучкості цей підхід широко використовується у побудові DWH, особливо у сценаріях, де необхідно швидко впровадити аналітичні звіти та оптимізувати процеси бізнес-аналітики.

Головна відмінність методології Кімбалла від підходу Інмона (рисунок 2.5) полягає у відсутності нормалізованого реляційного DWH як проміжного шару. У моделі Кімбалла дані після завантаження у стейджингові таблиці одразу копіюються в незалежні Data Marts – аналітичні сховища, які обслуговують конкретні потреби бізнес-підрозділів.

Data Marts у цьому підході є автономними, оскільки кожен з них фокусується на певній предметній області або бізнес-процесі, а не управляє

всіма даними в централізованому репозиторії. На відміну від архітектури Інмона, де всі дані спочатку нормалізуються та інтегруються в єдине корпоративне DWH, у Кімбалла кожен Data Mart побудований за розмірною моделлю та безпосередньо підтримує аналітичні запити бізнес-користувачів.

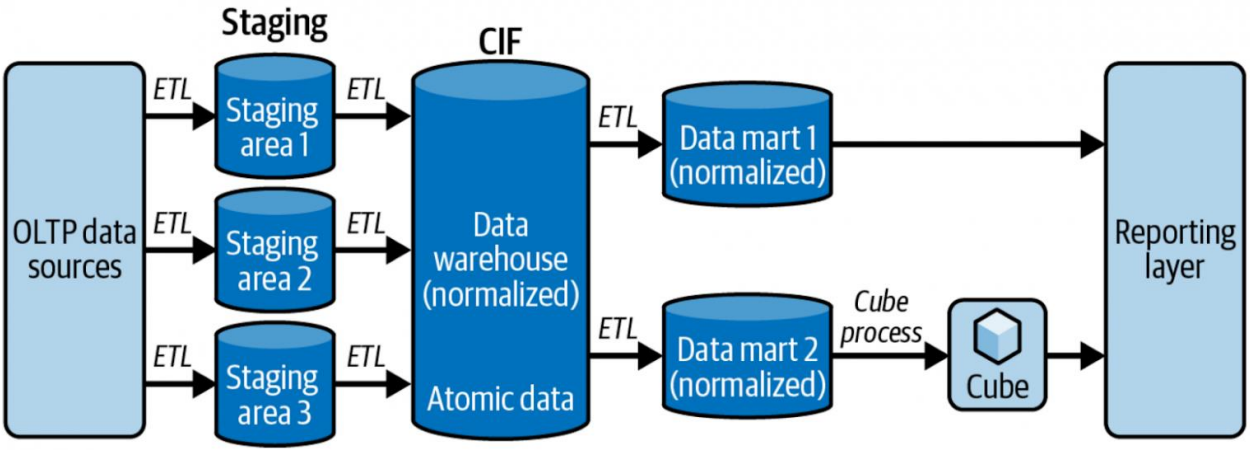

Рисунок 2.5 - Побудова DWH за методом Інмона

Методологія Інмона передбачає побудову нормалізованого DWH як центрального сховища з єдиним форматом даних, а аналітичні вітрини (Data Marts) виступають як другий рівень. Натомість підхід Кімбалла орієнтований на швидке впровадження аналітики, де кожен Data Mart створюється для конкретного бізнес-завдання. Інмон більше підходить для великих корпорацій, Кімбалл — для організацій, що прагнуть швидких результатів.

Обидва підходи — класичні, але мають обмеження: відсутність масштабованості, повільне завантаження даних та високі інфраструктурні витрати. З появою Big Data та хмарних рішень вони почали адаптуватися до сучасних вимог, інтегруючись у гібридні моделі. Сучасні архітектури поєднують нормалізацію, розмірне моделювання та розподілені обчислення для ефективної роботи з великими даними. Таким чином, класичні підходи до побудови DWH трансформувалися, адаптуючись до нових вимог бізнесу та технологічних реалій, що відкриває можливість ефективнішого використання великих даних у сучасних інформаційних системах.

2.4.4 Концепція Data Vault

Концепція Data Vault, створена Деном Лінстедтом у 2000 році, є методологією проєктування DWH, яка забезпечує гнучкість, масштабованість і збереження історії змін (рисунок 2.6). Вона краще пристосована до Big Data, ніж традиційні реляційні чи розмірні моделі. Модель базується на трьох основних компонентах: хабах (Hubs), посиланнях (Links) і сателітах (Satellites).

Хаби представляють бізнес-сутності (клієнти, продукти) з унікальними ключами. Посилання моделюють взаємозв'язки між хабами, відображаючи транзакції або ієрархії. Сателіти зберігають описові атрибути та історію змін, пов'язуючись із хабами або посиланнями. Така структура дозволяє ефективно масштабувати сховище, інтегрувати нові джерела й відстежувати зміни без втрати продуктивності.

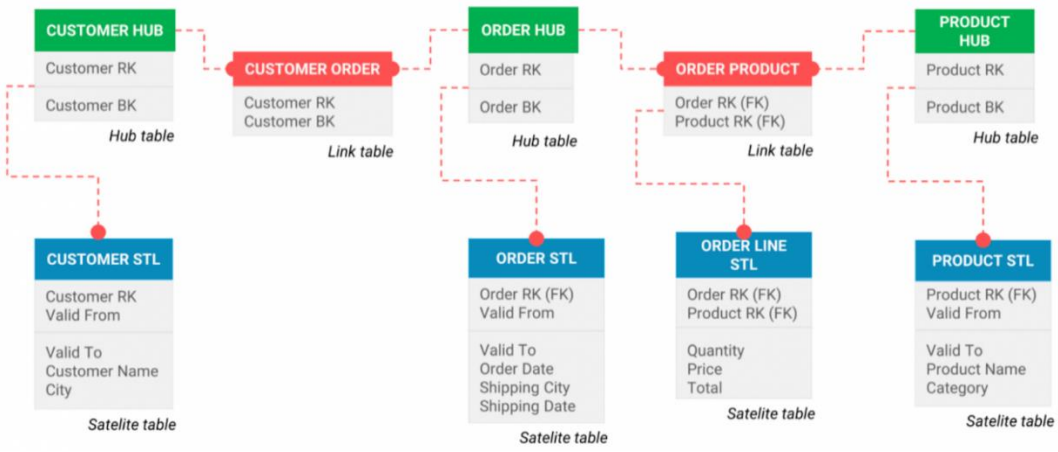

Рисунок 2.6 - Концепція Data Vault

Data Vault ідеально підходить для компаній, що працюють із великими обсягами даних у динамічних умовах, де важливе збереження історії та гнучка адаптація. У межах цієї методології виділяють два рівні: Raw Vault і Business Vault, що дозволяє розділити первинні дані та бізнес-орієнтовані перетворення.

Raw Vault містить необроблені дані з джерел і забезпечує історичність,

незмінність і легку інтеграцію. Business Vault доповнює його бізнес-логікою: обчисленнями, перевірками якості та агрегатами для формування аналітичних структур (Data Marts). Такий поділ дозволяє зберігати стабільність і масштабованість на базовому рівні, водночас надаючи зручні інструменти для бізнес-аналітики [4].

2.4.5 Концепція Anchor Modeling

Anchor Modeling — це гнучкий підхід до проєктування сховищ даних, який добре підходить для роботи з великими, постійно змінними обсягами інформації. Основу моделі складають якорі, що представляють сутності або події, з атрибутами для змінних характеристик та зв’язками для встановлення відносин між ними. Також використовуються вузли для моделювання загальних властивостей.

Переваги цього підходу — масштабованість і легка адаптація до змін без перегляду всієї структури. Проте велика кількість об’єднань (JOIN) може знижувати продуктивність, а складність моделі потребує глибокого розуміння. Незважаючи на це, Anchor Modeling залишається ефективним варіантом для середовищ із високою динамікою даних.

2.5 Інструменти реалізації DWH

ETL (Extract, Transform, Load) та ELT (Extract, Load, Transform) — це підходи до інтеграції даних у DWH або Data Lake. Вибір між ними залежить від типу даних, архітектури та вимог до продуктивності. В ETL дані спершу витягуються, трансформуються, а потім завантажуються у сховище. Цей підхід забезпечує узгодженість формату, але складні трансформації можуть впливати на продуктивність і вимагати повторного завантаження у разі помилок.

У ELT трансформація відбувається вже після завантаження даних у

сховище, що дозволяє зберігати початкову інформацію, зменшує навантаження на джерела і краще підходить для неструктурованих даних. ELT гнучкіший і ефективніший для сучасних архітектур, особливо в хмарних середовищах або при роботі з великими обсягами інформації.

У сучасних DWH та Data Lakehouse частіше використовується ELT, оскільки він краще підходить для великих обсягів даних і хмарних платформ. Водночас ETL залишається актуальним для випадків, де важливий контроль за трансформаціями до завантаження. Оскільки вихідні системи змінюються, важливо визначити частоту оновлень і спосіб виявлення змін у даних.

Існують два основні методи вилучення: повне (Full) і інкрементальне (Incremental). Повне вилучення отримує всі дані й підходить для невеликих таблиць, але створює велике навантаження при роботі з великими обсягами. Інкрементальне витягує лише змінені записи, знижуючи навантаження, проте вимагає точного виявлення змін (наприклад, через часові мітки або CDC). На практиці часто комбінують обидва підходи залежно від типу таблиці, що дозволяє підтримувати актуальність даних і оптимізувати продуктивність.

Окрім повного та інкрементального вилучення, застосовується онлайн-вилучення — отримання даних з вихідної системи в реальному часі або з мінімальною затримкою. Це може реалізовуватись через прямий доступ до таблиць або проміжні системи, що фіксують зміни (наприклад, журнали транзакцій). Такий підхід актуальний для систем, які працюють у режимі real-time. Для інкрементального вилучення найчастіше використовуються мітки часу, які дозволяють вибирати лише змінені записи. Також ефективним є Change Data Capture (CDC), який відстежує вставки, оновлення та видалення без надмірного навантаження. Інші методи включають тригери баз даних, оператор MERGE для комбінованих оновлень та партиціювання таблиць, що дозволяє обробляти лише змінені частини даних. Усе це сприяє підвищенню продуктивності та актуальності інформації в DWH.

2.6 Приклад архітектури ELT у Data Lakehouse

Архітектура ELT (рисунок 2.7) у Data Lakehouse будується на сучасних технологіях, які дозволяють ефективно витягувати, завантажувати та трансформувати дані у великомасштабних аналітичних системах. Для реалізації такого підходу використовуються рішення з відкритим вихідним кодом, які застосовуються у світовій практиці.

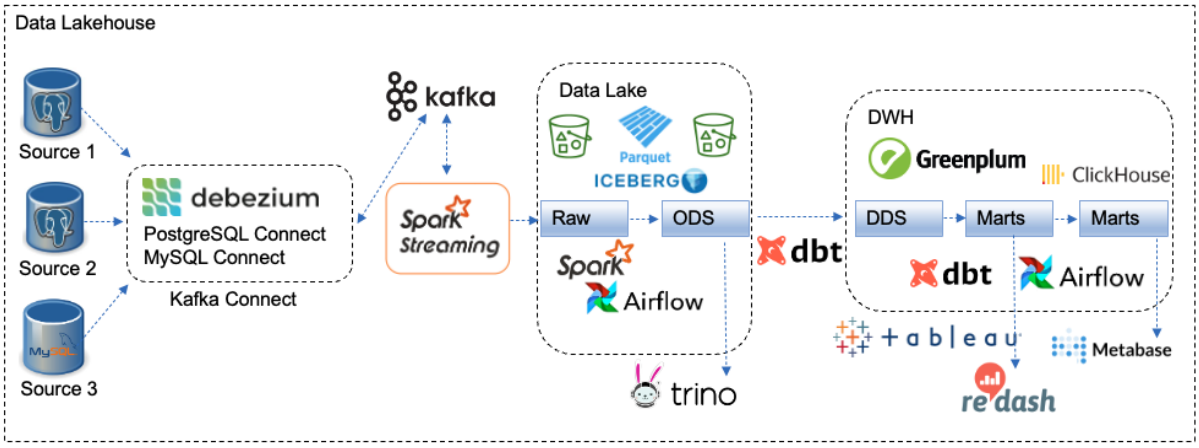

Рисунок 2.7 - Приклад архітектури ELT

На рівні витягу даних (Extract) система працює з різними джерелами, такими як OLTP-бази даних (PostgreSQL, MySQL, Oracle), лог-файли, API-сервіси або потокові дані (Kafka, Apache Flink). Дані витягуються та завантажуються у Data Lake без попередньої трансформації, що дозволяє зберігати їх у первісному вигляді та надавати можливість гнучкої обробки в майбутньому.

На рівні завантаження (Load) інформація потрапляє до сховища, що базується на Object Storage або розподілених файлових системах (MinIO, Hadoop HDFS, ClickHouse). Всі сирі дані зберігаються у Raw Zone, що дозволяє працювати з ними без втрати інформації та історичних змін.

На рівні трансформації (Transform) використовуються такі інструменти, як Apache Spark, dbt (Data Build Tool) або Trino, які виконують необхідні аналітичні операції безпосередньо в Data Lakehouse. Вони

дозволяють нормалізувати, агрегувати та оптимізувати дані, підготувавши їх до аналітичної обробки [12].

Аналітичний рівень у Lakehouse реалізується через BI-системи (Superset, Metabase, Redash), що напряму працюють з обробленими даними, забезпечуючи інтеграцію з ML-моделями й швидкий доступ до аналітики. ELT-архітектура забезпечує обробку великих обсягів структурованих і неструктурованих даних із високою гнучкістю та масштабованістю. Для реального часу використовується CDC, зокрема Debezium, який через Kafka Connect передає зміни з баз даних у Apache Kafka у вигляді подій.

Дані обробляються Spark Streaming або Flink і потрапляють у Raw Layer Data Lake у форматі Parquet з Iceberg. Spark і Airflow забезпечують трансформації та керування процесами в ODS, а Trino надає доступ до даних без дублювання. Для завантаження в DDS та формування аналітичного шару Mart використовуються Airflow та dbt. dbt виконує SQL-трансформації та генерує документацію, а Airflow оркеструє весь процес. Spark залишається ключовим для обробки CDC, проте саме Airflow є стандартом завдяки гнучкості, надійності та зручності управління.

Завдяки цій архітектурі забезпечується масштабоване та ефективне управління даними в Data Lakehouse, поєднуючи автоматизацію обробки, зручність SQL-трансформацій та потужність розподілених обчислень [5].

Такми чином, проаналізовано сучасні архітектурні підходи до побудови DWH, моделі даних (реляційне, розмірне, Data Vault 2.0, Anchor Modeling) та інструменти реалізації ELT-процесів у середовищі Lakehouse. Отримані результати є основою для розробки гнучкої адаптивної моделі гібридного зберігання даних, орієнтованої на роботу з великими обсягами інформації.

3 МОДЕЛЬ ЗБЕРЕЖЕННЯ ІНФОРМАЦІЇ У ГИБРИДНИХ СХОВИЩАХ

3.1 Опис моделі

Запропонована модель базується на ADH-DVL й охоплює комплексну архітектуру, яка поєднує декілька логічних і функціональних рівнів, інтегруючи у собі як сучасні технологічні рішення, так і теоретичні засади моделювання даних для гібридних систем збереження. Основу архітектури складає гібридна модель, що одночасно використовує локальні та хмарні ресурси для забезпечення максимальної гнучкості у розподілі даних, оперативності обробки та високої доступності системи.

На початковому етапі всі дані, які надходять із різноманітних джерел, включаючи транзакційні системи, сенсорні мережі, журнали подій та сторонні API, зберігаються у сирому вигляді у озері даних Raw Data Lake. Це озеро даних реалізується із залученням як локальних, так і хмарних сховищ, що дозволяє поєднувати переваги обох середовищ. Локальна складова гарантує безпеку, контроль доступу та мінімізацію затримок при роботі з чутливими або критичними даними, тоді як хмарна складова забезпечує масштабованість, еластичність та високу доступність при роботі з великими обсягами інформації. Дані на цьому рівні зберігаються у початковому форматі без попередньої трансформації, що забезпечує швидкість надходження та збереження нової інформації та відкладає етап складної обробки на наступні шари архітектури.

Після накопичення сирих даних наступним етапом формується логічний рівень інтеграції, реалізований за методологією Data Vault 2.0. Цей рівень забезпечує нормалізацію даних, історизацію змін, формування ключових бізнес-об’єктів та їх взаємозв’язків. Структура Data Vault складається із хабів, які представляють собою сутності бізнес-об’єктів, що ідентифікуються унікальними бізнес-ключами. Взаємозв’язки між цими

об’єктами моделюються за допомогою зв’язків, які фіксують усі типи асоціацій між сутностями. Атрибутивна інформація та історичні зміни фіксуються у сателітах, що дозволяє зберігати повну історію розвитку кожної бізнес-сутності. На основі цього формуються стабільні, гнучкі та стійкі до змін моделі корпоративних сховищ даних. Додатково формується бізнес-рівень Business Vault, у якому здійснюється трансформація даних згідно з внутрішніми бізнес-правилами, проводиться валідація інформації, обчислюються агрегати, створюються розширені атрибути та формуються золоті набори даних для подальшої аналітики. Саме на цьому рівні дані набувають цілісності та семантичної зрілості, придатної для управлінського аналізу.

Після завершення інтеграції даних формується аналітичний рівень Information Marts, який забезпечує користувачам доступ до оптимізованих наборів даних для виконання складних запитів та побудови звітності. На цьому рівні здійснюється денормалізація структур, застосовуються агрегати, формується предметно-орієнтована аналітика, що забезпечує ефективну роботу аналітичних інструментів, бізнес-інтелекту та OLAP-систем. Інформаційні вітрини адаптовані до потреб кінцевих користувачів, забезпечуючи високу продуктивність аналітичних операцій та мінімальні затримки у формуванні звітності [14].

Особливу роль у моделі відіграє шар потокової обробки Streaming Layer, який дозволяє здійснювати оперативне захоплення змін у джерелах даних у режимі близькому до реального часу. Захоплення змін реалізується за допомогою технології Change Data Capture, яка моніторить транзакційні журнали джерел і фіксує усі операції модифікації даних. Потокові події далі транспортуються через Kafka Connect та Apache Kafka, що забезпечує надійний транспортний механізм для масштабованої передачі даних у потоковому режимі. На фінальному етапі Apache Flink або Spark Streaming здійснюють обробку потоків даних, виконуючи трансформації, агрегації, очищення та маршрутизацію інформації до відповідних рівнів сховища.

Завдяки такій архітектурі (рисунок 3.1) система здатна в режимі реального часу оновлювати дані в аналітичних вітринах, підтримувати їх актуальність та дозволяє будувати високодинамічні бізнес-процеси.

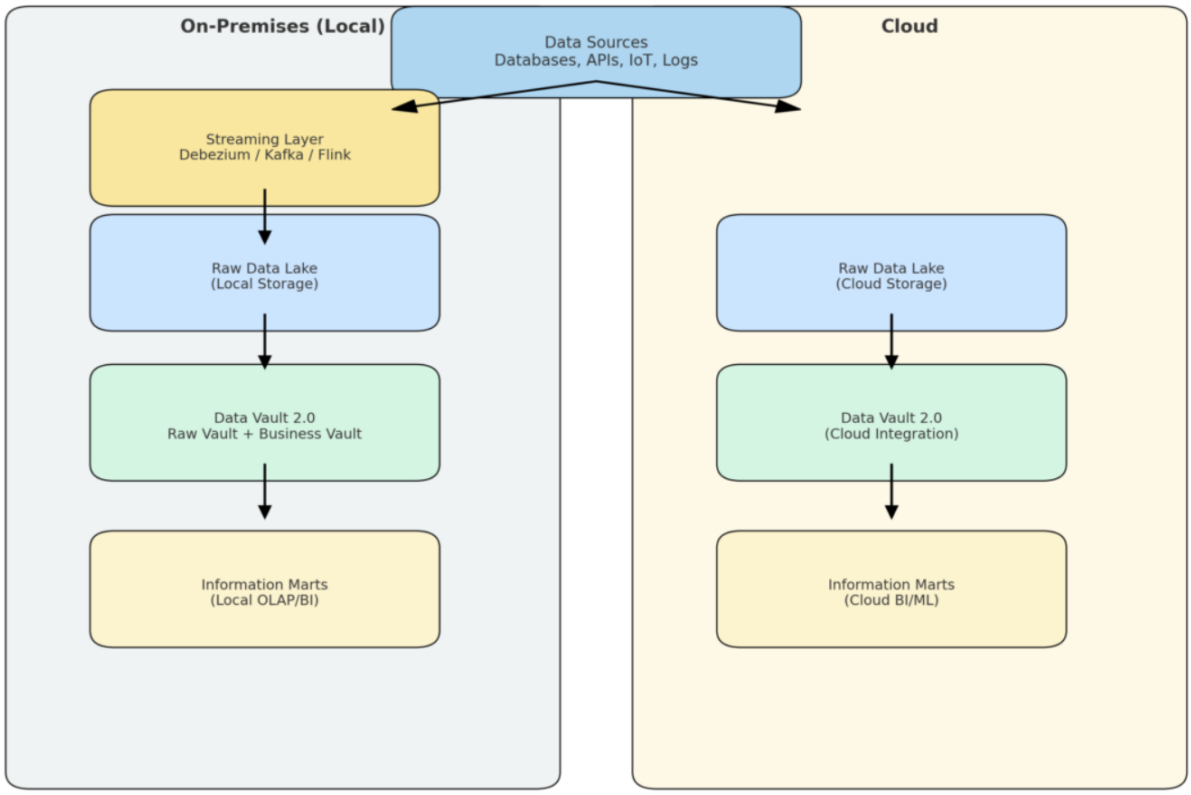

Рисунок 3.1 – Архітектура запропонованої моделі

Таким чином, модель на базі ADH-DVL формує єдину адаптивну екосистему обробки та збереження даних, що забезпечує гнучкий баланс між безпекою, продуктивністю, масштабованістю та оперативністю прийняття рішень у складних розподілених інформаційних системах.

3.2 Розширена математична модель розподілу даних

Побудова адаптивної гібридної системи збереження великих обсягів даних потребує точного математичного формалізму, що дозволяє описати основні процеси балансування між локальним та хмарним середовищем, а також врахувати динаміку навантаження, політики безпеки, частоту доступу

до даних та часові затримки в обробці. Запропонована математична модель на базі ADH-DVL описує ключові закономірності розподілу даних у гібридній інфраструктурі.

У центрі моделі лежить концепція поділу повного обсягу даних між локальною та хмарною частинами системи. Формально це відображається через базове рівняння балансу обсягів збереження:

$$V = V_L - V_C. \tag{3.1}$$

У цьому виразі величина V позначає загальний обсяг усіх даних, які необхідно зберігати у системі в даний момент часу. Змінна V_L описує обсяг інформації, який розміщується у локальній частині сховища, тоді як V_C відображає частину даних, що передається у хмарну інфраструктуру. Це рівняння дозволяє встановити базову умову збереження повноти інформації у системі та забезпечує основу для подальших адаптивних розрахунків розподілу даних.

Розподіл даних між середовищами залежить від ряду факторів. Одним із ключових є формула для визначення обсягу локального зберігання, яка враховує політики безпеки, чутливість даних та інтенсивність їх використання:

$$V_L = \alpha V(f_A + S). \tag{3.2}$$

Тут параметр $\alpha \in [0,1]$ й є керованим коефіцієнтом політики локалізації даних. Він дозволяє адміністратору системи або автоматизованому оптимізатору встановлювати загальний рівень пріоритетності локального зберігання залежно від обмежень безпеки або вимог регуляторів. Чим більшим є значення α, тим більша частина інформації залишається у локальній інфраструктурі. Змінна f_A описує нормалізовану частоту доступу до даних, тобто відображає інтенсивність використання конкретних

інформаційних наборів кінцевими користувачами або додатками аналітики. Величина S задає рівень чутливості даних з точки зору конфіденційності, безпеки та критичності для бізнесу. Обидва параметри f_A та S нормалізовані у діапазоні від 0 до 1, що дозволяє створити єдину шкалу вагомості для комбінованої оцінки важливості локального зберігання для кожного набору даних. Таким чином, обсяг даних, які повинні зберігатися локально, є результатом поєднання об'єктивних бізнесових і технічних характеристик з пріоритетами управління сховищем.

З урахуванням того, що навантаження на систему змінюється у часі, модель має включати механізм адаптації до динамічних змін частоти доступу. Для цього вводиться адаптивна функція зміни обсягу локального зберігання:

$$R(t) = \gamma_L(f_A(t) - \bar{f_A}). \tag{3.3}$$

У формулі (3.3) змінна R(t) описує динамічну складову зміни обсягу локального зберігання у часі t, яка формується як різниця між поточною частотою доступу $f_A(t)$ та середнім значенням цієї частоти $\bar{f_A}$) за певний період спостереження. Параметр γ є коефіцієнтом адаптації, що регулює швидкість та амплітуду реакції системи на зміни активності користувачів. Таким чином, якщо спостерігається аномальне зростання доступу до певного масиву даних, система автоматично підвищує його частку у локальному середовищі, зменшуючи залежність від хмарної інфраструктури та знижуючи затримки у доступі.

Крім розподілу обсягів збереження, для оцінки ефективності роботи системи необхідно враховувати часові витрати на обробку даних у всіх ключових компонентах. Це відображено через рівняння загальної затримки системи:

$$L_{total} = L_{ingest} + L_{stream} + L_{ETL} + L_{query}. \tag{3.4}$$

У рівнянні (3.4) складова L_{ingest} описує затримку при захопленні змін у джерелах даних, що реалізується за допомогою механізмів CDC (наприклад, через Debezium). Компонент L_{stream} фіксує затримки стрімінгової обробки даних, яка виконується потоковими рушіями типу Apache Flink або Spark Streaming. Наступна складова L_{ETL} характеризує затрати часу на трансформацію та обробку даних у Data Vault, де здійснюється нормалізація, очищення, історизація та збагачення інформації згідно з бізнес-правилами. Завершальна частина L_{query} включає затримки виконання аналітичних запитів на рівні BI-платформ, що працюють з Information Marts. Повна сума цих затримок дозволяє кількісно оцінити загальний час відповіді системи на запити користувачів та дає можливість оптимізувати конфігурацію системи відповідно до цільових показників продуктивності [15].

Таким чином, запропонована модель забезпечує цілісне формалізоване представлення процесів управління гібридними сховищами даних у динамічних інформаційних системах великого масштабу. Вона дозволяє не лише проектувати архітектуру таких систем, а й проводити оптимізаційне налаштування у процесі експлуатації відповідно до поточних бізнес-вимог, навантажень та політик безпеки.

4 ДОСЛІДЖЕННЯ МОДЕЛІ ЗБЕРЕЖЕННЯ ІНФОРМАЦІЇ У ГІБРИДНИХ СХОВИЩАХ

4.1 Оцінка ефективності моделі

Для оцінки ефективності запропонованої адаптивної математичної моделі було проведено моделювання на основі трьох експериментальних сценаріїв, які відображають різні варіанти чутливості даних, інтенсивності доступу до них та політики локалізації [16].

У першому сценарії розглянуто ситуацію, коли вся інформація має високий рівень чутливості. Загальний обсяг даних становив 10 ТБ, при цьому коефіцієнт чутливості дорівнював 0.8, що відображає високу критичність збереження даних. Частота доступу до інформації була максимальною і становила 1.0, що відповідає постійному активному використанню. Політика розподілу, яка задається коефіцієнтом α, дорівнювала 0.8, що свідчить про пріоритет локального розміщення. Підставляючи вхідні параметри у формулу (3.2) визначення локального обсягу V_L =14.4ТБ.

Оскільки отриманий результат перевищує фактичний обсяг даних, модель коригує результат до максимально допустимого значення. Таким чином, у даному випадку весь обсяг інформації у розмірі 10 ТБ розміщується виключно у локальному середовищі.

Другий сценарій моделює середній рівень чутливості інформації. Загальний обсяг даних дорівнював 100 ТБ. Коефіцієнт чутливості становив 0.5, частота доступу — 0.5, що відповідає помірній активності використання. Значення α дорівнювало 0.6, що відображає помірковану політику розподілу між локальним та хмарним середовищами. Підставляючи відповідні параметри у формулу, отримано V_L =60ТБ.

Таким чином, у локальній частині залишається 60 ТБ, а решта 40 ТБ зберігаються у хмарній інфраструктурі. Це демонструє ефективне поєднання

локальних ресурсів для активних та чутливих даних із використанням хмарних потужностей для зберігання менш активної частини.

У третьому сценарії було змодельовано ситуацію з мінімальною чутливістю даних та низькою інтенсивністю доступу. Загальний обсяг даних складав 100 ТБ, параметри чутливості та частоти доступу становили по 0.2. Значення коефіцієнта α було обрано на рівні 0.4, що імітує економічно орієнтовану стратегію максимальної передачі даних у хмару. Розрахунок дає наступне значення V_L =16ТБ. Таким чином, у локальному середовищі залишаються лише 16 ТБ інформації, тоді як основна частина — 84 ТБ — розміщується у хмарному сховищі.

Отримані результати демонструють адаптивність моделі до змінних параметрів даних, забезпечуючи динамічний баланс між вимогами безпеки, ефективності доступу та економічної доцільності використання обчислювальних ресурсів [17]. Таким чином, модель на базі ADH-DVL дозволяє гнучко масштабувати гібридну інфраструктуру під різні типи бізнес-даних.

4.2 Дослідження поведінки гібридної моделі на базі ADH-DVL

Для поглибленої верифікації адаптивної гібридної моделі ADH-DVL було реалізовано серію числових експериментів, які відображають ключові закономірності функціонування системи в умовах змінних параметрів даних та політик управління. Побудовані графічні залежності дозволяють наочно продемонструвати гнучкість і балансувальні механізми моделі при різних сценаріях роботи гібридних сховищ.

Перша серія графічного моделювання дозволила оцінити, як змінюється частка локального зберігання при варіації чутливості даних. Ця залежність (рисунок 4.1) ілюструє логіку того, як модель реагує на підвищення конфіденційності чи бізнес-критичності інформації. Зі зростанням чутливості система закономірно нарощує локалізацію даних,

оскільки збереження таких масивів у локальних сегментах дозволяє мінімізувати ризики витоку, несанкціонованого доступу та зовнішніх загроз. Особливо помітно посилюється цей ефект у поєднанні з високою частотою доступу.

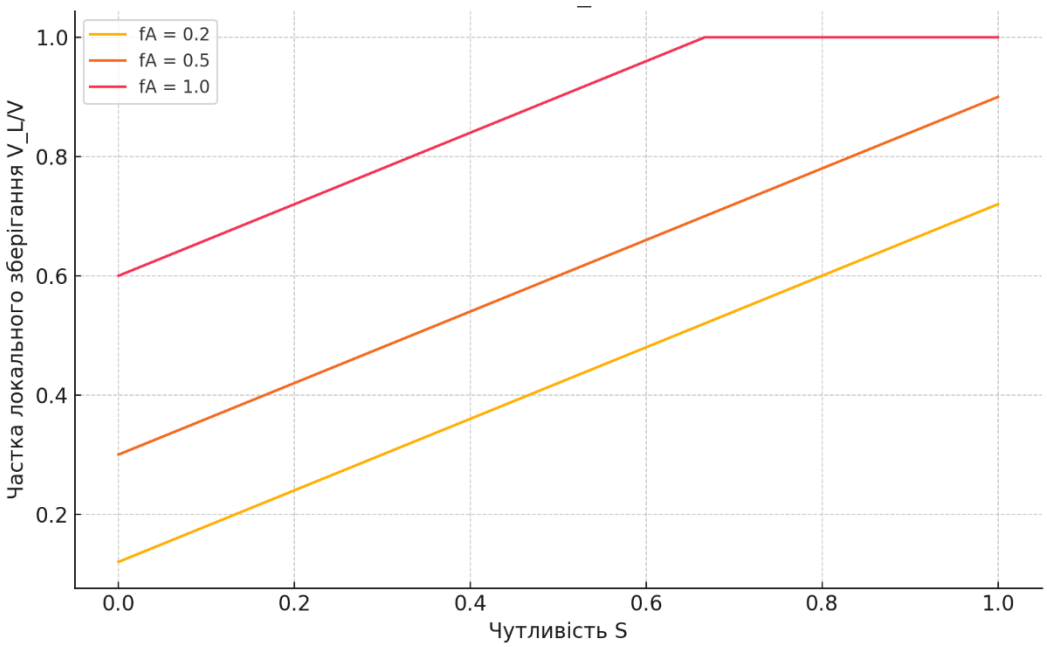

Рисунок 4.1 - Залежність частки локального зберігання від чутливості

Наприклад, для даних, що активно використовуються аналітичними системами чи транзакційними платформами, навіть незначна чутливість даних вимагає значної локальної присутності. Модель демонструє адекватну поведінку — вона не нав’язує повну локалізацію за низьких параметрів доступу, що економить ресурси, але швидко реагує при зростанні навантажень. У практичних системах таке балансування дозволяє водночас оптимізувати витрати й виконувати регуляторні вимоги.

Друге графічне дослідження фокусувалося на аналізі залежності загальної затримки роботи системи від обсягу інформації, що виноситься у хмарні ресурси. Отримана картина (рисунок 4.2) дозволяє чітко відслідкувати прямий компроміс між гнучкістю масштабування і ціною втрачених мілісекунд доступу. Поведінка системи цілком передбачувана — при нарощуванні частки хмарних компонент поступово зростають часові втрати,

зумовлені особливостями мережевих комунікацій, маршрутизації пакетів, розподіленої обробки запитів у хмарних обчислювальних кластерах. У промислових системах ця закономірність стає критичною при обслуговуванні задач реального часу: наприклад, у фінансових платформах, телемедичних системах, оперативних виробничих контролерах, де затримки понад визначений поріг можуть спричинити не лише фінансові втрати, але й порушення логіки керування технологічними процесами.

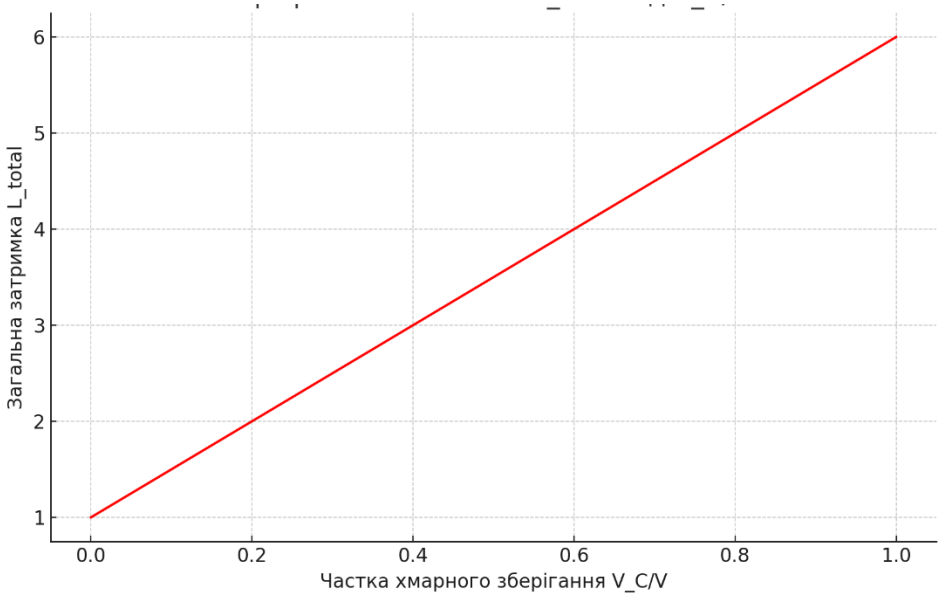

Рисунок 4.2 - Залежність затримки від частки хмарного зберігання

У рамках третього експериментального блоку було досліджено вплив адміністративного параметра політики локалізації (рисунок 4.3). Цей параметр дає змогу без зміни властивостей самих даних швидко та гнучко регулювати архітектурну поведінку сховища відповідно до стратегічних змін у бізнес-середовищі. Наприклад, під впливом нових нормативних актів державного контролю, при появі нових ризиків витоків інформації або в разі зміни економічних умов тарифікації хмарних послуг, компанія отримує можливість оперативно змінювати частку локального зберігання. Особливо це актуально для банківських, державних, оборонних та медичних систем, де внутрішні регламенти збереження можуть змінюватися буквально впродовж

тижнів, змушуючи IT-інфраструктури динамічно адаптуватися до зовнішніх викликів. Отриманий лінійний характер залежності підтверджує простоту й ефективність реалізації цього механізму у реальних умовах експлуатації великих датацентрів [17].

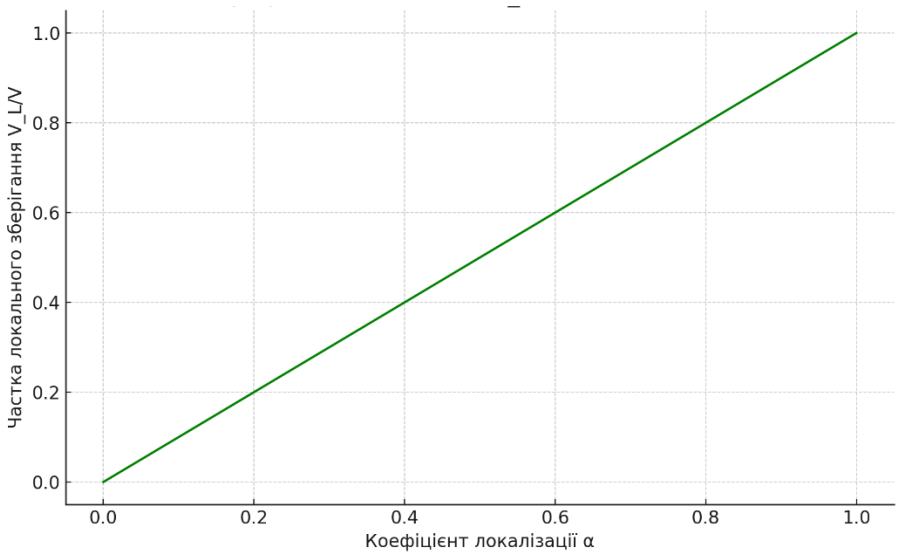

Рисунок 4.3 - Залежність частки хмарного зберігання від коефіцієнту локалізації

Нарешті, четверта модель демонструє складну взаємодію одразу двох критичних факторів: чутливості даних та частки хмарної компоненти. Побудована тривимірна картина (рисунок 4.4) формує повноцінний аналітичний простір компромісів між продуктивністю, безпекою й інфраструктурними витратами. Контурна карта чітко показує, що найбільші затримки виникають у тих конфігураціях, де з одного боку зберігається велика частка інформації у хмарних ресурсах, а з іншого — дані мають низьку бізнес-критичність, через що система допускає таке винесення. Водночас система автоматично забезпечує кращі показники затримок у випадках, коли чутливість даних зростає, адже у таких ситуаціях модель сама переводить значні частки інформації у локальні контури обробки [18]. Практична цінність цього дослідження полягає у тому, що воно дозволяє підприємствам готувати повноцінні сценарні стратегії побудови своїх

гібридних сховищ під конкретні класи інформаційних потоків, оптимізуючи сукупну вартість володіння інфраструктурою без втрат у критичних SLA.

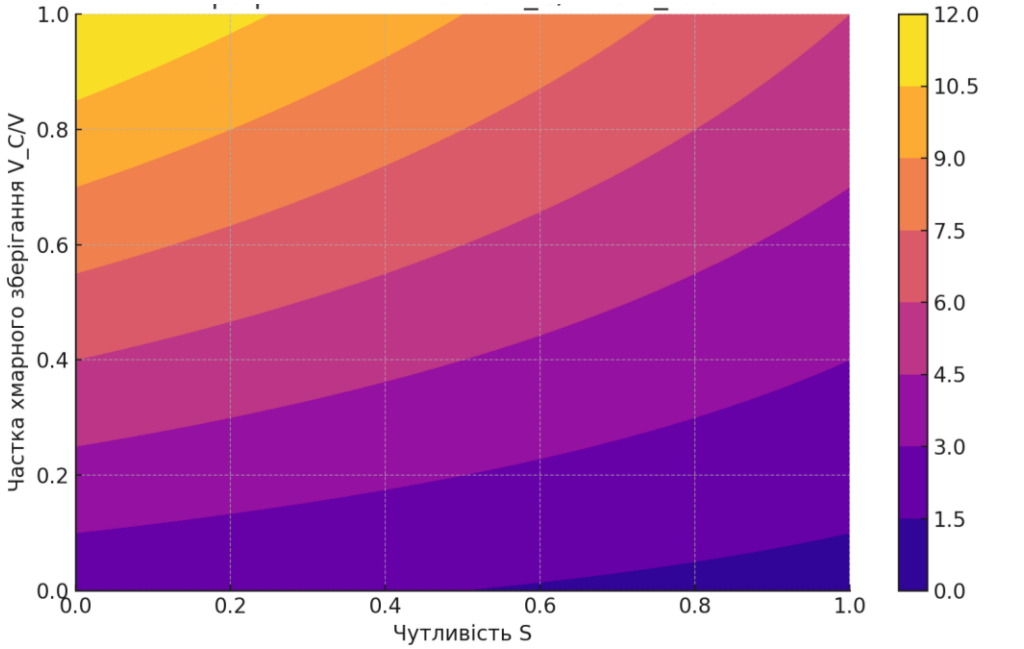

Рисунок 4.4 - Вплив чутливості та хмарного зберігання на затримку

Таким чином, проведене числове та графічне моделювання дозволило здійснити комплексну оцінку функціональних характеристик адаптивної гібридної моделі збереження інформації ADH-DVL. Отримані результати повністю підтверджують здатність моделі забезпечувати ефективне динамічне балансування між локальним та хмарним середовищем зберігання залежно від зміни параметрів чутливості даних, інтенсивності доступу та управлінських політик.

Розроблена програмна реалізація (Додаток В) демонструє функціонування керуючого модуля системи гібридного зберігання даних, в основі якого лежить адаптивний підхід до розподілу ресурсів з урахуванням характеристик інформаційних потоків [19]. Система організована за об’єктно-орієнтованим принципом з використанням класу керування збереженням даних.

Клас HybridStorageSystem інкапсулює основні параметри сховища: загальну ємність системи, ємність локального сегменту, а також поточний

обсяг використаних ресурсів у локальному та хмарному середовищах. Така структура забезпечує чітке розділення відповідальностей між зберіганням системного стану і виконанням обчислювальних операцій над даними.

Метод adaptive_store() виконує динамічне прийняття рішення щодо розміщення чергового блоку даних. На основі вхідних характеристик нових даних, які містять параметри доступу, чутливості та політики розподілу, розраховується рекомендований обсяг локального зберігання. Далі система здійснює перевірку доступності локальних ресурсів. Якщо локальна ємність достатня — дані частково або повністю розміщуються локально; у разі дефіциту вільного простору — залишкові обсяги автоматично прямують у хмарне середовище.

Метод status() реалізує допоміжний функціонал моніторингу поточного стану розподілу даних у гібридному сховищі, відображаючи накопичений обсяг інформації у кожному середовищі зберігання.

Такий підхід дозволяє моделювати реальну поведінку гібридних систем збереження інформації, де управління ресурсами має відбуватись динамічно в залежності від характеристик вхідних потоків даних та поточного стану інфраструктури. Програмна структура легко масштабована для інтеграції з реальними інформаційними потоками, API зовнішніх сервісів, систем моніторингу стану ресурсів, а також для подальшої автоматизації процесів балансування навантаження у гібридних інфраструктурах зберігання великих обсягів інформації.

Ключовим підтвердженим ефектом моделі є її здатність автоматично масштабувати обсяг локального зберігання залежно від зростання чутливості інформації та активності її використання. Такий адаптивний механізм дозволяє гнучко знижувати затримки доступу до критичних даних, мінімізуючи ризики безпекових інцидентів, і водночас не перевантажувати локальну інфраструктуру обробкою неактуальних або архівних масивів, які економічно доцільніше розміщувати у хмарних середовищах.

Результати моделювання затримок показали очікувану закономірність

лінійного зростання часу обробки при збільшенні частки хмарного зберігання. Це ще раз підтверджує необхідність ретельного врахування параметрів SLA, особливо для систем із вимогами до роботи в режимі реального часу. Водночас модель дозволяє сформувати гнучкі сценарії управління затримками шляхом динамічного зміщення інформаційних обсягів між середовищами при зміні навантаження чи бізнес-критичності потоків даних.

Дослідження впливу політики локалізації показало, що параметр α є дієвим інструментом стратегічного керування інфраструктурою, дозволяючи централізовано адаптувати модель до зовнішніх регуляторних, юридичних або економічних факторів. Це створює базу для подальшого застосування моделі у системах з підвищеною нормативною динамікою, включно з банківською сферою, охороною здоров’я, урядовим управлінням, промисловістю та обороною.

Результати комплексного багатофакторного аналізу підтвердили здатність моделі на базі ADH-DVL оптимально адаптуватися до змінних умов експлуатації складних гібридних систем зберігання інформації. Модель створює умови для побудови балансованих архітектур, що забезпечують високу продуктивність, мінімізацію ризиків інформаційної безпеки та ефективне використання обчислювальних ресурсів в умовах великомасштабних інформаційних навантажень.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було проведено комплексне дослідження сучасних підходів, архітектур та моделей збереження великих обсягів інформації в умовах розвитку гібридних інформаційних сховищ, що поєднують локальні та хмарні компоненти.

На етапі аналітичного огляду проаналізовано класичні та сучасні архітектури побудови систем збереження даних. Було досліджено еволюцію підходів від традиційних централізованих DWH до сучасних гібридних систем, які включають концепції Data Lakes, Data Lakehouse, Data Fabric, а також децентралізованих моделей типу Data Mesh. Особливу увагу приділено особливостям інтеграції структурованих, напівструктурованих та неструктурованих даних у єдиному середовищі, що відображає актуальні виклики зростаючих обсягів інформації в умовах стрімкого цифрового розвитку.

У другому розділі було системно розглянуто принципи побудови шарової архітектури сучасних DWH, зокрема в контексті гібридних моделей на базі Data Lakehouse. Докладно вивчено методики моделювання даних, включаючи реляційне моделювання, розмірне моделювання, механізми роботи з повільно-змінюваними вимірюваннями, а також сучасні концепції Data Vault та Anchor Modeling, які дозволяють забезпечити гнучкість, історичність та масштабованість моделей даних у складних динамічних середовищах. Особливу увагу приділено також аналізу інструментальних засобів реалізації процесів ETL/ELT, що є критично важливими для забезпечення стабільної роботи сучасних гібридних платформ.

У третьому розділі на основі проведеного аналізу було запропоновано нову адаптивну модель збереження інформації у гібридних сховищах — ADH-DVL, яка поєднує ідеї Lakehouse-архітектури, концепцію Data Vault 2.0, елементи стрімінгової обробки та розподіленого зберігання даних у хмарних

та локальних середовищах. Запропонована математична модель дозволяє враховувати параметри чутливості даних, частоти доступу до них та політик локалізації, забезпечуючи гнучке балансування інформаційних обсягів між середовищами з урахуванням поточних вимог до безпеки, продуктивності та вартості експлуатації.

У четвертому розділі було проведено детальне дослідження поведінки запропонованої моделі ADH-DVL. Виконано числове моделювання для різних сценаріїв роботи системи з високою, середньою та низькою чутливістю інформації, що дозволило кількісно оцінити зміну обсягів локального та хмарного зберігання залежно від параметрів задачі. Окрім числових експериментів, проведено серію графічних досліджень, що дозволили візуально продемонструвати динаміку адаптації моделі до зміни частоти доступу, чутливості, політик управління локалізацією та їх спільного впливу на загальну продуктивність системи. Встановлено, що модель демонструє здатність ефективно знижувати затримки доступу до даних при збереженні балансу між безпекою, продуктивністю та економічною ефективністю використання гібридних ресурсів.

Таким чином, в рамках кваліфікаційної роботи було реалізовано комплексний підхід до аналізу, проектування, математичного моделювання та дослідження поведінки сучасних гібридних систем збереження великих обсягів інформації. Розроблена модель на базі ADH-DVL може бути використана як методологічна основа для побудови реальних промислових рішень у сферах фінансів, охорони здоров’я, промислових IT-систем, державних інформаційних платформ та інших складних інформаційних середовищ, де гібридна архітектура є критичною умовою ефективного управління даними.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. James Serra. Deciphering Data Architectures: Choosing Between a Modern Data Warehouse, Data Fabric, Data Lakehouse, and Data Mesh 1st Edition.: Oreilly and Associates. 2024. – 226 p.

2. Daniel Linstedt, Michael Olschimke. Building a Scalable Data Warehouse with Data Vault 2.0.: Morgan Kaufmann, 2015. – 684 p.

3. Lawrence Corr, Jim Stagnitto. Agile Data Warehouse Design: Collaborative Dimensional Modeling, from Whiteboard to Star Schema. DecisionOne Consulting, 2011 - 330 p.

4. Хмарні технології. Хмарна платформа для управління. [Електронний
ресурс] – Режим доступу до ресурсу: http://j.parus.ua/ua/358/.

5. National Institute of Standards and Technology [Электронный ресурс] - URL: https://www.nist.gov / (Дата звернення 10.05.2025).

6. Grolinger K., Higashino W.A., Tiwari A., Capretz M.A.M. Data management in cloud environments: NoSQL and NewSQL data stores // Journal of Cloud Computing. – 2013. – Vol. 2. – Art. 22. DOI: https://doi.org/10.1186/2192-113X-2-22.

7. Inmon W.H. Building the Data Warehouse. – 4th ed. – New York: John Wiley & Sons, 2005. – ISBN 978-0-7645-9945-6.

8. Kimball R., Ross M. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. – 3rd ed. – New York: John Wiley & Sons, 2013. – ISBN 978-1-118-53480-3.

9. Linstedt D. The Data Vault Guru: Building the Data Vault 2.0 Architecture. – Boulder: Technics Publications, 2015. – ISBN 978-1-63431-026-2.

10. Golfarelli M., Rizzi S. Data Warehouse Design: Modern Principles and Methodologies. – New York: McGraw-Hill, 2009. – ISBN 978-0-07-161039-8.

11. Madera C., Laurent A. The next information architecture evolution: the

data lake wave // Proceedings of the 8th International Conference on Management of Digital EcoSystems (MEDES). – Biarritz, France, 2016. – P. 174–180. DOI: https://doi.org/10.1145/3012071.3012091.

12. Sawadogo P.N., Darmont J. On data lake architectures and metadata management // Journal of Intelligent Information Systems. – 2021. – Vol. 56. – P. 97–122. DOI: https://doi.org/10.1007/s10844-019-00580-3.

13. Dehghani Z. Data Mesh: Delivering Data-Driven Value at Scale. – O'Reilly Media, 2022. – ISBN 978-1-098-10707-6.

14. Armbrust M. et al. Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics // Proceedings of CIDR 2021. – January 2021.

15. Karajan M., Sorenson P., Saleh K. Hybrid Cloud Storage: Research Status, Challenges, and Future Directions // Journal of Cloud Computing: Advances, Systems and Applications. – 2020. – Vol. 9. – Art. 49. DOI: https://doi.org/10.1186/s13677-020-00188-y.

16. Al-Doghman F., Muthanna A. Hybrid Cloud Storage Models: A Survey and Framework // Journal of Computer Networks and Communications. – 2021. – Art. 5583984. DOI: https://doi.org/10.1155/2021/5583984.

17. Marjani M. et al. Big IoT Data Analytics: Architecture, Opportunities, and Open Research Challenges // IEEE Access. – 2017. – Vol. 5. – P. 5247–5261. DOI: https://doi.org/10.1109/ACCESS.2017.2689040.

18. Ghosh R., Niyogi R. Big data storage and retrieval: storage architecture and efficient query processing // Journal of Cloud Computing. – 2017. – Vol. 6. – Art. 16. DOI: https://doi.org/10.1186/s13677-017-0090-5.

19. Rittinghouse J.W., Ransome J.F. Cloud Computing: Implementation, Management, and Security. – 2nd ed. – Boca Raton: CRC Press, 2017. – ISBN 978-1-4987-3950-2.