**2. Обгрунтування вибраного напряму роботи**

Перш ніж почати проектування та розробку системи необхідно чітко визначити які вимоги така система повинна виконувати. Тільки після повного розуміння та означення вимог програмне рішення може бути правильно спроектованим, реалізованим та відлагодженим. Типові сховища даних можуть забезпечувати не тільки зберігання великих обємів даних в придатному для аналітичних запитів вигляді, але й дозволяти обробку нових даних, що надходять, в реальному часі. Такі сховища даних можуть використовуватися для надання статистичної інформації про певну предметну область з врахуванням нових даних, що постійно надходять. В залежності від того, які вимоги ставляться до сховища даних, його структура та перелік використаних технологій значною мірою відрізнятиметься.

**2.1 Дослідження вимог до сховищ даних**

В загальному випадку розрізняють два основних види сховищ даних:

1. Сховища даних що забезпечують виконання аналітичних запитів до великих обємів даних
2. Сховища даних які забезпечують окрім виконання аналітичних запитів ще й обробку в реальному часі нових вхідних даних.

Перший вид сховищ використовується в більшій мірі в наукових та дослідницьких організаціях. Також значного поширення вони набули в організаціях, які оперують великими обємами даних для побудови прогнозів та прийняття рішень на основі цих прогнозів. Характерною рисою таких систем є те, що вимоги до часу відповіді (латентності) не є критичними. В залежності від обсягів даних та використаної інфраструктури час відповіді таких систем може вимірюватися десятками хвилин або годинами.

Характерною особливістю другого виду систем є забезпечення можливості обробки великих обсягів нових вхідних даних з (або без) використанням даних, що вже збережені в сховищі. До таких програмних рішень ставляться досить жорсткі вимоги по часу відклику. Вони найчастіше використовуються в організаціях, які мають справу з даними, що швидко змінюються в часі і потребують максимально швидкого реагування.

Схему руху даних даних в системі першого виду можна зобразити так, як показано на рис 2.1

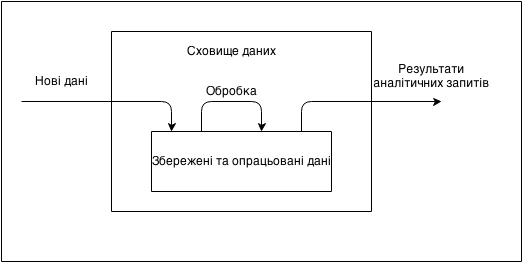


Рис. 2.1 Схема руху даних в сховищах для аналітичних запитів

Як видно зі схеми, в таких системах процес отримання нових даних є незалежним по відношенню до споживання даних зі сховища. Аналітичні запити до сховища даних використовують збережені та опрацьовані в системі дані. Дані можуть надходити в такий вид сховищ з найрізноманітніших джерел: вивантажуватись з систем керування базами даних, отримуватись у вигляді текстових файлів, у спеціальних текстових форматах і т.п. Основною вимогою до таких систем є забезпечити можливість зберігання та подальшого використання завантажених в систему даних. Час на опрацювання вхідних даних та приведення їх до вигляду, придатного для аналітичних запитів може займати суттєвий час. Основною причиною цього можуть бути:

- великі обсяги даних;

- складність трансформацій, що мають бути виконані над вхідними даними;

- необхідність використання всіх, раніше збережених даних для трансформації нових (в такому випадку час опрацювання зростатиме з накопиченням нових записів).

Споживачами даних, збережених у таких сховищах, зазвичай є аналітичні системи, які виконують велику кількість складних запитів до значних частин збереженої інформації. Обсяги інформації, в таких системах можуть сягати кількох десятків або й навіть сотень терабайт. Великою перевагою для проектувальників та розробників таких систем є те, що не ставляться жорсткі вимоги до часу опрацювання та підготовки даних. В першу чергу це дає можливість знехтувати певними оптимізаційними питаннями, які могли б відобразитися у вигляді значних часових затрат на розробку системи. Також це надає можливість зменшити вимоги до апаратної частини системи.

Сховища такого виду є поширеними уже на протязі тривалого часу і зазвичай будуються з використанням БД та великих серверних станцій. Використання таких технологій цілком достатньо і раціонально у випадку, якщо обєми вхідної і оброблюваної інформації не перевищують кількасот гігабайт. З ростом даних такі системи стають надто повільними, а програмних оптимізацій стає недостатньо щоб задовольнити вимоги. Єдиним шляхом є покращення апаратного забезпечення, а це надто дорогий крок.

Системи другого виду забезпечують не тільки можливість виконання аналітичних запитів, але й обробку та видачу вхідної інформації у реальному часі. Такі системи здатні реагувати на події, що виникають, та реагувати на них з надзвичайно низькою латентністю. Сховища такого виду зазвичай мають кілька інтеграційних точок: інтерфейс для виконання аналітичних запитів, та інструмент для передачі результатів обробки вхідної інформації, що безпереревно надходить в систему.

Схема руху даних, які потребують обробки в реальному часі зображена на рис. 2.2.



Рис. 2.2. Схема руху даних в сховищах з підтримкою відповіді в реальному часі

Цей вид сховищ даних є розширеним варіантом першого виду. Такі системи забезпечують можливість не тільки виконання аналітичних запитів, але й надають інтерфейс для виконання обробки вхідних даних в реальному часі.

Інформація в сховище надходить кількома різними шляхами. Це можуть бути великі завантаження файлів з файлових систем або таблиць з баз даних. Ці дані зберігаються та опрацьовуються для подальшого використання інструментами побудови аналітичних звітів. Окрім цього існує також канал, по якому в систему надходить інформація, яка потребує дуже швидкої обробки і передачі вже опрацьованих даних користувачам системи. Такі дані можуть оброблятися за наборами складних правил а також потребувати доступ до уже наявних даних. Вимоги до латентності такої системи є критичними. На відміну від першого типу сховищ даних, в яких дозволений час для опрацювання і перетворення даних може становити десятки хвилин або й години, в системах другого типу дані повинні опрацьовуватися за лічені секунди. В певних випадках час відклику має становити кількасот мілісекунд. Складністю побудови таких систем є в першу чергу те, що при значному збільшені вимог до часу відклику, вимоги до обсягів даних, які мають опрацьовуватися не зменшується. Це означає, що має бути досягнута надзвичайно велика паралелізація обчислень. Потік вхідних даних, які потребують обробки в реальному часі є безперервним, тому не допускаються жодні затримки або збої в системі. Швидкість такого потоку може досягати кількох Мб/с.

Сховища такого виду можуть використовуватися в організаціях, що надають оперативну інформцію в сферах, де потрібно опрацьовувати значні обєми даних в реальному часі. Наприклад метеостанції, медичні заклади, дослідницькі центри, які працюють з вимірюваннями швидкоплинних процесів.

Перелік вимог до систем значною мірою вплине на структуру програмного рішення. В магістерській кваліфікаційній роботі буде досліджено та розроблено шляхи побудови як першого так і другого виду сховищ даних. Для першого типу також буде запропонована реалізація, на основі платформи розподілених обчислень Hadoop.

2.2. Огляд шляхів побудови сховищ даних на основі платформи Hadoop

Вимоги до системи є найбільш визначальним фактором, що впливає на весь процес проектування та розробки програмного рішення. Так і у випадку з сховищами даних, від способу обробки даних залежитиме набір технологій і спосіб їх взаємодії в межах системи. Платформа Hadoop надає можливість зберігання та паралельної обробки великих масивів даних на кластерах зі звичайного недорого обладнання. Такі кластери є набагато дешевшими аніж великі серверні станції, що використовуються для встановлення на них баз даних. Системи на основі Hadoop володіють майже лінійною масштабованістю, а це в свою чергу дозволяє легко і недорого розширювати апаратне забезпечення.

Для правильної побудови сховища даних потрібно враховувати те, як дані мають рухатись в системі. Під параметрами руху маються на увазі:

* Кількість джерел даних
* Інтерфейси, які ці джерела надають;
* Обсяги даних;
* Швидкість надходження;
* Правила перетворень
* Вимоги до часу зберігання
* Інтерфейси систем-споживачів
* Вимоги до часу запиту

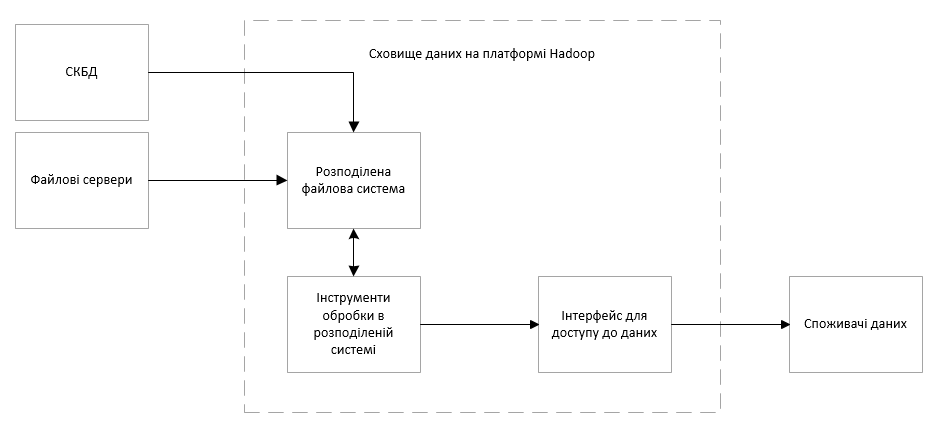
Всі ці параметри визначають спосіб взаємодії між усіма елементами системи та алгоритми їх роботи. Також від значень цих параметрів залежить спектр конкретних інструментів та технологій, які будуть використовуватися для побудови сховища даних.

Для початку розглянемо структуру програмного рішення сховища даних на основі платформи Hadoop.В звязку з тим, що сховища даних є предметно-орієнтованими, кількість та природа джерел даних буде різною в кожному окремому випадку. Але не зважаючи на це, існують певні аспекти, спільні для всіх систем такого типу. Перелік цих аспектів може повторно використовуватися в сховищах з різних прикладних сфер. Характеристики, якими системи відрізняються можуть бути винесені в окремий незалежний модуль. До таких характеристик можна віднести логічну модель даних, що зберігаються та правила обробки цих даних. Інтерфейси, інфраструктура та набір технологій можуть бути однаковими у найрізноманітніших предметних областях. Розділювання одної системи на такі незалежні модулі дозволить зробити її конфігурованою і забезпечити можливість повнорного використання її компонентів.

Спершу варто розглянути структуру сховища даних для виконання аналітичних запитів на ньому. Основними вимогами до такої системи є:

* Інтерфейс для отримання даних з БД;
* Інтерфейс для отримання текстових файлів з файлових серверів;
* Можливість надійного зберігання надвеликих обємів даних;
* Опрацювання та перетворення всіх даних за прийнятний час;
* Інтерфейс для виконання аналітичних запитів на збережених опрацьованих даних;
* Прийнятний час виконання запитів, що звертаються до всього обєму збережених даних.

Цей перелік вимог визначає структуру такої системи. Архітектура такого програмного рішення зображена на рис 2.3.

 Рис. 2.3. Структурна схема сховища даних для побудови аналітичних звітів на основі збережених даних

Далі буде детально розглянуто роль кожного елемента в системі та способи їх взаємодії. В наступному розділі цієї МКР для кожного елемента структурної схеми буде підібрано відповідну технологію і запропоновано більш детальну архітектурну діаграму.

Така система має два основних інтерфейси вхідних даних:

1. Інтерфейс для роботи з СКБД;
2. Інтерфейс для роботи з файловими серверами.

Наявність саме таких інтерфейсів зумовлена тим, що в даний момент майже у всіх предметних областях бази даних є найбільш використовуваним способом збереження даних. Тому при побудові нових програмних рішень варто враховувати, що великі обсяги даних мають бути перенесені з уже існуючих БД і можливо перетворені в інші логічні структури. Також важливим фактором є те, інформація, яка отримується від клієнтів та буде збережена в сховищі даних, в першу чергу заносить в бази даних. Це спричинено тим, що БД є досить оптимальним рішенням для швидкого збереження невеликої та середньої кількості даних. А сфери генерування інформації в яких безпосередньо задіяна людина характеризуються невеликою кількістю згенерованих даних. Потрібно також враховувати що існує велика кількість різних СКБД, тому інтерфейс сховища має забезпечувати сумісність з найпопулярнішим та найпоширенішими СКБД.

Важливим питанням є те, що СКБД хоч і забезпечують виконання великої кількості одночасних запитів, та можуть не справлятися з їх обробкою, якщо такі запити буде виконувати цілий кластер машин (а у випадку розподіленої системи це буде саме так). Тому інтерфейс повинен забезпечувати одночасні запити до СКБД-джерел тільки обмеженої кількості машин кластера. Особливо гострим таке питання може стати, якщо час вивантаження даних з БД буде співпадати з часом, коли БД забезпечує доступ до даних своїм першочерговим клієнтам (користувачам системи де основні дані продукуються). Це може призвести до відмови системи на достатньо тривалий час. А таке є недопустимим, адже може призвести до значних збитків.

Іншим не менш важливим інтерфейсом є інтерфейс з файловими серверами (наприклад FTP). В багатьох предметних областях основні відомості зберігаються в текстових файлах, де записи розділяються комами або іншими розділювачами. Файли з такими даними можуть досягати колосальних розмірів, а тому не можуть бути нормально опрацьовані на звичайних компютерах. Сучасні засоби обробки текстових файлів не дозволяються запускати опрацювання таких великих документів на кількох машинах і це ставить в безвихідне становище системи, яким така обробка необхідна. Файлові сервери можуть працювати за різними протоколами, тому інтерфейс сховища даних має враховувати це і надавати можливість працювати з найпоширенішими протоколами.

Після пересилання інформації з БД або файлових серверів вона має бути надійно збережена і готова для подальшої обробки. Для цього система повинна володіти надійною розподіленою файловою системою. Окрім надійності зберігання даних також має бути передбачений захист цих даних. Дуже часто дані, що мають бути опрацьовані, носять конфіденційний характер і вимагають надійного захисту від несанкціонованого доступу. Це можуть бути медична, фінансова та інша інформація.

Розподілені системи можуть містити десятки а то і сотні машин в свому складі. А це значно збільшує ймовірність відмови однієї або кількох машин кластера. Це має бути враховано при побудові розподіленої файлової системи. Адже при виході з ладу одного або кількох вузлів кластера можлива безповоротна втрата частини збережених даних. В більшості предметних областей це недопустимо. Існує кілька шляхів забезпечення надійності в розподілених файлових системах. Одним з найпоширеніших є реплікація даних. При такому підході вся інформація, що надходить в ФС реплікується і розсилається на кілька вузлів. Тому у випадку виходу з ладу однієї або кількох машин втрачені дані можуть бути відновлені з копій, що знаходилися на інших машинах. Недоліком таких систем є інтенсивне використання зовнішньої памяті. В залежності від степеня реплікацій, обсяги памяті, що займається може зрости в кілька разів по відношенню до розміру початкових даних. Також недоліком подібних систем є те, що потрібно вирішувати проблему несинхронізованості копій. Тобто у випадку редагування файлу, потрібно забезпечити щоб усі копії цього файлу були однаковими і містили останні зміни. Важливим елементом таких розподілених ФС є необхідність в наявності координатора та сховища метаданих. Їх призначенням є координування звертань до файлів у системі. Вони мають забезпечувати направлення звертання саме до того вузла, який містить необхідний файл. Координатор та сховище метаданих є найвразливішим місцем всієї файлової системи. Адже при їхньому виході з ладу втрачається можливість доступу до будь-якого з раніше збережених файлів.

Для забезпечення захисту збереженої інформації може використовуватися шифрування усіх файлів. Такий шлях захисту хоч і вберігає файли від несанкіонованого доступу, але значною мірою впливає на продуктивність системи. Шифрування та дешифрування при кожному звертанні до даних може в кілька разів зменшити швидкодію системи. Можливе також шифрування на апаратному рівні, але такий спосіб може мати дуже великий вплив на вартість обладнання. В такому випадку нівелюється можливість побудови розподіленого середовища з використанням звичайних машин.