Ширококолонкові СУБД: Apache Cassandra

МЕТА РОБОТИ

Отримати навички роботи з децентралізованою розподіленою СУБД **Apache Cassandra**, зокрема:

1. Засвоїти ключовий принцип **моделювання даних під запити (Query-Driven Modeling)**.
2. Опанувати механізм **де-нормалізації** та багаторазового запису (Multi-Table Write).
3. Продемонструвати гнучкість **розрідженої схеми (Sparse Schema)** для зберігання різнорідних даних.

ПОСТАНОВКА ЗАДАЧІ

Розробити модель даних та клієнтський додаток для зберігання та оперативного доступу до даних великого **каталогу товарів та відгуків** на платформі електронної комерції, використовуючи датасет **Amazon Reviews** (рис. 1). Система повинна забезпечувати **високу пропускну здатність запису** та **надзвичайно низьку затримку** при читанні за ключовими запитами.

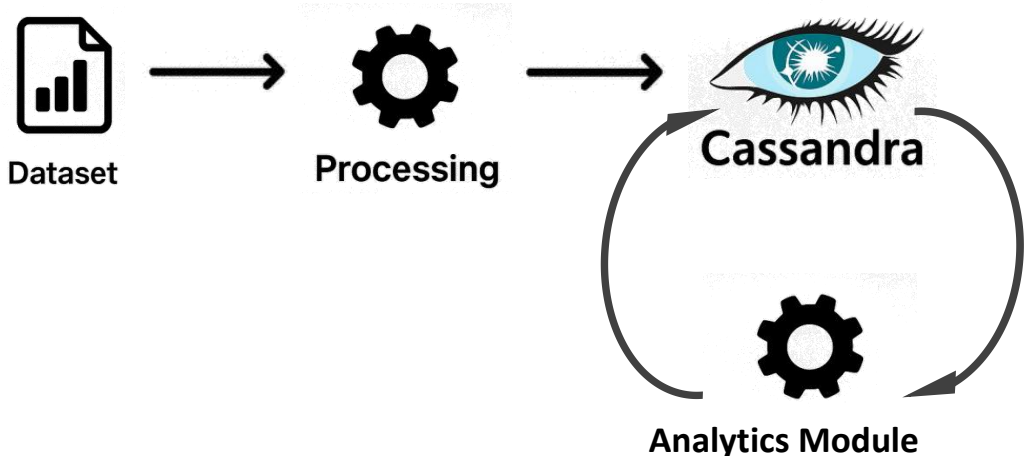

Рис. 1 Архітектура системи

ПОРЯДОК ВИКОНАННЯ

1. Встановлення та конфігурування Cassandra

1.1. Встановити та налаштувати дво-вузловий кластер **Apache Cassandra** (за допомогою Docker). Порядок та параметри налаштування наведені у Додатку 1.

1.2. Створити **Keyspace** (e_commerce) з відповідними налаштуваннями реплікації.

2. Вивчення структури вихідних даних

2.1. Ознайомитися зі структурою датасетів Amazon Reviews, що надаються ресурсом https://amazon-reviews-2023.github.io/ (або будь-яким іншим, подібним до нього).

2.2. **Завантажити** дані (User Reviews та Item Metadata) для обраних 2-3 категорій (наприклад, "Books", "Electronics" и "Apparel").

2.3. Проаналізувати структуру наборів **User Reviews** та визначити, які поля (наприклад, product_id, user_id, review_time, rating, text) будуть використовуватися у якості **ключових та звичайних атрибутів** таблиць reviews_by_product і reviews_by_user із пункту 3.

2.4. Проаналізувати структуру наборів **Item Metadata** та визначити поля для таблиці product_attributes **із пункту 3**. **Зосередитися** на полі details, яке містить атрибути товарів.

3. Моделювання даних (Query-Driven Modeling):

Створити щонайменше **чотири де-нормалізовані таблиці** для підтримки ключових оперативних та аналітичних запитів, використовуючи CQL.

Таблиця	**Призначення**	**Ключові поля**
reviews_by_product	Відображення **всіх відгуків** на сторінці конкретного товару.	**PRIMARY KEY** (**product_id**, review_time)
reviews_by_user	Побудова **історії активності** для профілю конкретного користувача.	**PRIMARY KEY** (**user_id**, review_time)
product_attributes	Зберігання атрибутів продукту з демонстрацією розрідженої схеми.	**PRIMARY KEY** (**product_id**)
user_top_categories	Зберігання попередньо розрахованих результатів аналітики (**топ-категорія/категорії, загальна кількість відгуків**) для забезпечення **миттєвого доступу (low-latency)** сервісів персоналізації.	**PRIMARY KEY** (**user_id**)

Вказівки:

1) При визначенні структури таблиці **product_attributes пропонується обрати** 5-10 найпоширеніших атрибутів продуктів (з поля полі details набору **Item Metadata**).
2) Зазвичай, потенційно невідомі атрибути зберігаються у колонці типу **Map** (Map<text, text>), де ключ — це назва атрибута, а значення — його вміст. Але в поточній роботі це не є обов'язковим.
3) В обох таблицях відгуків використовуйте **review_time** як **кластеризуючий ключ (Clustering Key)** для сортування за часом.

4) У таблиці product_attributes передбачити колонки, які будуть **розрідженими** (наприклад, author для книг та screen_size для електроніки).

4. Розробка сервісу завантаження даних (Ingestion Service)

Розробити застосунок (наприклад, на **Python** з використанням бібліотеки cassandra-driver, або на будь-якій іншій мові програмування), в межах якого реалізувати функціонал завантаження даних (або підмножини даних) з датасету Amazon Reviews до Apache Cassandra, зокрема:

4.1. Реалізувати **мультизапис (Multi-Table Write)**: при обробці кожного відгуку виконати запис **одночасно** у таблиці **reviews_by_product** та **reviews_by_user**.

4.2. Завантажити метадані товару (поле details) у таблицю **product_attributes**, демонструючи, як розріджена схема дозволяє зберігати різні типи атрибутів в одній таблиці.

Примітка: У реальних Big Data архітектурах цей етап (завантаження даних) часто представлений окремою платформою потокової передачі даних (Streaming Platform), яка включає в себе Apache Kafka або AWS Kinesis Streams, які дозволяють приймати дані в реальному часі.

5. Розробка сервісу аналізу даних (Analytics Engine): написати **окремий функціональний блок, який:**

5.1. **Читає** дані з **reviews_by_user**

5.2. Для невеликої вибірки користувачів **програмно розрахувує** їхню "топ-категорію/категорії" (наприклад, шляхом простого підрахунку).

5.3. **Заносить** розраховані агреговані дані (топ-категорії, total_reviews, last_analysis_time в таблицю **user_top_categories**.

Примітка! У реальних системах задачі аналізу даних як правило виконуються окремими аналітичними платформами на базі Apache Spark.

6. Валідація та тестування.

6.1. Виконати запит, що симулює **оперативну вибірку відгуків** на сторінці товару (з використанням product_id та обмеженням LIMIT).

6.2. Виконати запит, що симулює **побудову профілю користувача** (з використанням user_id).

6.3. Продемонструвати, що запити з використанням лише Partition Key виконуються з **надзвичайно низькою затримкою**.

7. Оформити звіт, в який включити розроблений код, схеми таблиць CQL, скріншоти результатів виконання оперативних запитів та відповіді на контрольні запитання.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Поясніть, у чому полягає філософія **"Моделювання даних під запити"** у Cassandra. Чим вона відрізняється від моделювання в РСУБД?
2. Яку роль відіграють **ключ розділу (Partition Key)** та **кластеризуючий ключ (Clustering Key)** у забезпеченні масштабованості та низької затримки при читанні?
3. На прикладі таблиці product_attributes поясніть, що таке **розріджена схема (Sparse Schema)** і як вона забезпечується на рівні одиниці зберігання даних (т.зв. трійки: **колонка, значення, мітка часу**).
4. Поясніть, що таке **незмінність (Immutability)** файлів SSTable і як вона дозволяє Cassandra досягати високої пропускної здатності запису.
5. Поясніть, як у типовій **Big Data архітектурі** вирішуються задачі аналітики, адже аналітичні запити на величезних масивах даних в Cassandra можуть бути доволі повільними?
6. Поясніть, як працює механізм шардування/партиціонування та реплікації в Cassandra.
7. До якої моделі з CAP-теореми воліє Cassandra: CA, CP чи AP? Чи є у Cassandra механізм, який дозволяє керувати рівнем цілісності даних (Consistency)?

ДОДАТОК 1

Встановлення та налаштування дво-вузлового кластеру **Apache Cassandra**

1. Створення файлу docker-compose.yml

Створіть файл з назвою docker-compose.yml та додайте до нього наступну конфігурацію. Цей файл визначає два сервіси, де cassandra1 є **Seed Node** (початковим вузлом), а другий вузол приєднуються до нього.

Лістинг 1. docker-compose.yml

```yaml
version: "3.7"

# Визначаємо дві служби (контейнери): node1 та node2
services:
  cassandra1:
    image: cassandra:4.1
    container_name: cassandra1
    hostname: cassandra1
    ports:
      - "9042:9042"
    volumes:
      - cassandra1_data:/var/lib/cassandra
    environment:
      - CASSANDRA_CLUSTER_NAME=UniversityCluster
      - CASSANDRA_SEEDS=cassandra1 # Цей вузол є Seed Node
```

```
    networks:
      - cassandra_network

  cassandra2:
    image: cassandra:4.1
    container_name: cassandra2
    hostname: cassandra2
    volumes:
      - cassandra2_data:/var/lib/cassandra
    environment:
      - CASSANDRA_CLUSTER_NAME=UniversityCluster
      - CASSANDRA_SEEDS=cassandra1 # Приєднується до першого вузла

    networks:
      - cassandra_network
    depends_on:
      - cassandra1 # cassandra2 запуститься після cassandra1

# Визначаємо мережу
networks:
  cassandra_network:
    driver: bridge

# Оголошуємо томи
volumes:
  cassandra1_data:
  cassandra2_data:
```

Примітки!

1. **Volumes** (томи) гарантують, що дані, які Cassandra записує усередині контейнера (/var/lib/cassandra), фізично зберігалися на вашій хост-машині. Це робить дані **персистентними** (стійкими) після перезапуску контейнерів.
2. Зверніть увагу: кількість томів співпадає з кількістю вузлів Cassandra. На кожному томі зберігаються не тільки оригінальні дані, але й репліковані.

2. Запуск кластера

Виконайте команду в тому ж каталозі, де знаходиться файл docker-compose.yml:

```
docker-compose up -d
```

3. Перевірка статусу кластера

Дочекайтеся 1–2 хвилини, поки всі вузли запустяться та приєднаються до кластера. Потім виконайте команду **nodetool status** через контейнер першого вузла:

```
docker exec -it cassandra1 nodetool status
```

Очікуваний результат: Ви повинні побачити таблицю з трьома вузлами (Node 1, Node 2), і їхній статус має бути **UN** (Up, Normal — Активний, Нормальний).

```
PS C:\Temp\Cassandra> docker exec -it cassandra1 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens  Owns (effective)  Host ID                               Rack
UN  172.18.0.3  128.17 KiB  16      100.0%            cd21ebb1-8ca1-4d5c-b296-613ec1b7bcc0  rack1
UN  172.18.0.2  109.42 KiB  16      100.0%            62b35dc9-a690-4176-9430-f53e22258c18  rack1
```

4. Підключення до CQL (cqlsh)

Підключіться до інтерфейсу запитів CQL, використовуючи порт 9042:

```
docker exec -it cassandra1 cqlsh
```

Примітки!

1. cqlsh є стандартним клієнтом з графічним інтерфейсом для СУБД Apache Cassandra і поставляється разом і з самою СУБД.
2. Для роботи з Cassandra може бути використаний і якийсь стандартний Клієнт з графічним інтерфейсом, наприклад, DBeaver. У випадку з Apache Cassandra треба буде окремо встановити драйвер. Наприклад, безкоштовний драйвер cassandra-jdbc-wrapper (от ING), доступний на https://github.com/ing-bank/cassandra-jdbc-wrapper/releases

5. Створення Keyspace з реплікацією

Створіть **Keyspace** з фактором реплікації RF=2 (що означає, що кожна копія даних буде записана на два різні вузли):

```
CREATE KEYSPACE e_commerce WITH replication = {
    'class': 'SimpleStrategy',
    'replication_factor': '2'
};
```