Міністерство освіти і науки України
НТУУ «Київський політехнічний інститут»
Фізико-технічний інститут

Протокол лабораторної роботи №5
з дисципліни Проектування високонавантажених систем
на тему: Робота з базовими функціями БД типу column family на прикладі Cassandra

Виконав: студент групи ФІ-21
Грунда Ярослав

Київ, 2025

Завдання

Task 1

1. Створити keyspace з найпростішою стратегією реплікації
2. Створити таблицю items
3. Написати запит, який показує структуру створеної таблиці (команда DESCRIBE)
4. Написати запит, який виводить усі товари в певній категорії відсортовані за ціною
5. Написати запити, які вибирають товари за різними критеріями в межах певної категорії (тут де треба замість індексу використайте MATERIALIZED view):
   (а) назва,
   (б) ціна (в проміжку),
   (в) ціна та виробник
6. Створити таблицю orders
7. Написати запит, який показує структуру створеної таблиці (команда DESCRIBE)
8. Для замовника виведіть всі його замовлення відсортовані за часом коли вони були зроблені
9. Для кожного замовників підрахуйте загальну суму на яку були зроблені усі його замовлення
10. Для кожного замовлення виведіть час коли його ціна були занесена в базу (SELECT WRITETIME)

Task 2

1. Сконфігурувати кластер з 3-х нод
2. Перевірити правильність конфігурації за допомогою nodetool status
3. Використовуючи cqlsh, створити три Keyspace з replication factor 1, 2, 3 з SimpleStrategy
4. В кожному з кейспейсів створити прості таблиці
5. Спробуйте писати і читати в ці таблиці підключаюсь до різних нод через cqlsh
6. Вставте дані в створені таблиці і подивіться на їх розподіл по вузлах кластера окремо для кожного з кейспесов (команда nodetool status) - має бути видно відсоток даних який зберігається на ноді
7. Для якогось запису з кожного з кейспейсу виведіть ноди на яких зберігаються дані - має бути видно ip-адреси вузлів на яких зберігається даний рядок
8. Відключити одну з нод. Для кожного з кейспейсів перевірити з якими рівнями consistency можемо читати та писати
9. Зробити так щоб три ноди працювали, але не бачили одна одну по мережі (заблокуйте чи відключити зв’язок між ними)

10. Для кейспейсу з replication factor 3 задайте рівень consistency рівним 1. Виконайте по черзі запис значення з однаковим primary key, але різними іншими значенням окремо на кожну з нод (тобто створіть конфлікт)

11. Відновіть зв'язок між нодами, і перевірте що вони знову об'єдналися у кластер. Визначте яким чином була вирішений конфлікт даних та яке значення було прийнято кластером та за яким принципом

Task 3

1. Для кластеру налаштованому у попередній частині створити таблицю з каунтером лайків

2. З 10 окремих клієнтів одночасно запустити інкерементацію каунтеру лайків по 10000 на кожного клієнта з різними опціями взаємодії з Cassandra. Виміряйте час виконання та перевірте чи кінцеве значення буде дорівнювати очікуваному - 100К

   (а) Вказавши у параметрах запиту Consistency Level One

   (б) Вказавши у параметрах запиту Consistency Level QUORUM

Зміст

1 Task 1

1.1 docker-compose.yml

```
services:
  cassandra-1:
    image: cassandra:latest
    container_name: cassandra-1
    ports:
      - "9042:9042"
    environment:
      - CASSANDRA_CLUSTER_NAME=MyCluster
      - CASSANDRA_DC=datacenter1
      - CASSANDRA_ENDPOINT_SNITCH=GossipingPropertyFileSnitch
      - CASSANDRA_SEEDS=cassandra-1
      - MAX_HEAP_SIZE=512M
      - HEAP_NEWSIZE=100M
```

```
D:\University\semestr_7\High-Load_Systems\lab5>docker-compose up -d
[+] Running 2/2
 ✓ Network lab5_default   Created
 ✓ Container cassandra-1  Started

D:\University\semestr_7\High-Load_Systems\lab5>docker exec -it cassandra-1 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens  Owns (effective)  Host ID                               Rack
UN  172.20.0.2  309.6 KiB  16      100.0%            338d41c7-900f-4178-9b9a-a5a71c9d3955  rack1
```

1.2 keyspace

```
D:\University\semestr_7\High-Load_Systems\lab5>docker exec -it cassandra-1 cqlsh
Connected to MyCluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.6 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE KEYSPACE store WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
cqlsh> USE store;
cqlsh:store>
```

1.3 Part 1: items

```
cqlsh:store> CREATE TABLE items (
         ...     category text,
         ...     price decimal,
         ...     id uuid,
         ...     name text,
         ...     producer text,
         ...     properties map<text, text>,
         ...     PRIMARY KEY ((category), price, id)
         ... ) WITH CLUSTERING ORDER BY (price ASC);
cqlsh:store> CREATE INDEX ON items(KEYS(properties));
```

1. Partition Key (category): Забезпечує миттєвий доступ до групи товарів

2. Clustering Key (price): Підтримує автоматичне сортування та швидкі діапазонні запити.
3. Clustering Key (id): Забезпечує унікальність запису в межах партиції. Це дозволяє зберігати декілька різних товарів з однаковою ціною
4. properties (map<text, text>): Зберігає унікальні властивості товару, що не є загальними для всіх категорій.
5. Secondary Index (`KEYS(properties)`): Створює вторинний індекс саме на *ключах* мапи. Це дозволяє виконувати запити типу `CONTAINS KEY`, щоб знаходити товари, які мають певну характеристику

Додаємо тестові дані:

```
cqlsh:store> BEGIN BATCH
         ...   INSERT INTO items (category, price, id, name, producer, properties) VALUES ('Electronics', 1000, uuid(), 'La
         ...   INSERT INTO items (category, price, id, name, producer, properties) VALUES ('Electronics', 1200, uuid(), 'Sm
         ...   INSERT INTO items (category, price, id, name, producer, properties) VALUES ('Electronics', 800, uuid(), 'Mon
         ...   INSERT INTO items (category, price, id, name, producer, properties) VALUES ('Books', 15, uuid(), 'Cassandra
         ... APPLY BATCH;
cqlsh:store> SELECT * FROM items;

 category    | price | id                                   | name           | producer | properties
-------------+-------+--------------------------------------+----------------+----------+-----------------------------------
       Books |    15 | 9f6226d2-c148-4f6f-8476-b60ca8c6dde2 | Cassandra Guide |  OReilly |                  {'pages':
 Electronics |   800 | 1c6bd689-4332-466b-ad48-fa0b98367e8b |       Monitor Z |  Samsung |               {'resolution':
 Electronics |  1000 | 1c619c8f-416d-40d0-a31f-94052e98fda3 |        Laptop X |     Dell |     {'color': 'silver', 'cpu':
 Electronics |  1200 | a8df27a2-8a7d-4650-b0f6-ab0766997d5a |     Smartphone Y |    Apple | {'color': 'black', 'memory': '12

(4 rows)
```

Опис таблиці:

```
cqlsh:store> DESCRIBE TABLE items;

CREATE TABLE store.items (
    category text,
    price decimal,
    id uuid,
    name text,
    producer text,
    properties map<text, text>,
    PRIMARY KEY (category, price, id)
) WITH CLUSTERING ORDER BY (price ASC, id ASC)
    AND additional_write_policy = '99p'
    AND allow_auto_snapshot = true
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'm
    AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND memtable = 'default'
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND extensions = {}
    AND gc_grace_seconds = 864000
    AND incremental_backups = true
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99p';

CREATE INDEX items_properties_idx ON store.items (keys(properties));
```

Товари в певній категорії відсортовані за ціною:

```
cqlsh:store> SELECT * FROM items WHERE category = 'Electronics';

 category    | price | id                                   | name         | producer | properties
-------------+-------+--------------------------------------+--------------+----------+-----------------------------------
 Electronics |   800 | 1c6bd689-4332-466b-ad48-fa0b98367e8b |    Monitor Z |  Samsung |                {'resolution': '4K
 Electronics |  1000 | 1c619c8f-416d-40d0-a31f-94052e98fda3 |     Laptop X |     Dell |     {'color': 'silver', 'cpu': 'i7
 Electronics |  1200 | a8df27a2-8a7d-4650-b0f6-ab0766997d5a | Smartphone Y |    Apple | {'color': 'black', 'memory': '128GB

(3 rows)
```

Товари в певній категорії в деякому проміжку ціни:

```
cqlsh:store> SELECT * FROM items WHERE category = 'Electronics' AND price > 900 AND price < 1300;

 category    | price | id                                   | name         | producer | properties
-------------+-------+--------------------------------------+--------------+----------+-----------------------------------
 Electronics |  1000 | 1c619c8f-416d-40d0-a31f-94052e98fda3 |     Laptop X |     Dell |     {'color': 'silver', 'cpu': 'i7'
 Electronics |  1200 | a8df27a2-8a7d-4650-b0f6-ab0766997d5a | Smartphone Y |    Apple | {'color': 'black', 'memory': '128GB'

(2 rows)
```

Включаємо можливість створювати materialized views та перезавантажуємо ноду

```
D:\University\semestr_7\High-Load_Systems\lab5>docker cp cassandra-1:/etc/cassandra/cassandra.yaml .
Successfully copied 115kB to D:\University\semestr_7\High-Load_Systems\lab5\.

D:\University\semestr_7\High-Load_Systems\lab5>docker cp cassandra.yaml cassandra-1:/etc/cassandra/cassandra.yaml
Successfully copied 115kB to cassandra-1:/etc/cassandra/cassandra.yaml

D:\University\semestr_7\High-Load_Systems\lab5>docker restart cassandra-1
cassandra-1
```

```
1949
1950    # Enables materialized view creation on this node.
1951    # Materialized views are considered experimental and are not recommended for production use.
1952    materialized_views_enabled: true
1953
```

Створюємо materialized view та виконуємо запит: шукаємо конкретний товар за назвою в категорії

```
cqlsh:store> CREATE MATERIALIZED VIEW items_by_name AS
         ... SELECT category, price, id, name, producer
         ... FROM items
         ... WHERE category IS NOT NULL
         ...   AND price IS NOT NULL
         ...   AND id IS NOT NULL
         ...   AND name IS NOT NULL
         ... PRIMARY KEY ((category), name, price, id);

Warnings :
Materialized views are experimental and are not recommended for production use.

cqlsh:store> SELECT * FROM items_by_name
         ... WHERE category = 'Electronics'
         ...   AND name = 'Laptop X';

 category    | name     | price | id                                   | producer
-------------+----------+-------+--------------------------------------+----------
 Electronics | Laptop X |  1000 | 1c619c8f-416d-40d0-a31f-94052e98fda3 |     Dell
```

Створюємо materialized view та виконуємо запит: шукаємо товар за ціною та виробником в категорії

```
cqlsh:store> CREATE MATERIALIZED VIEW items_by_producer AS
         ... SELECT category, price, id, name, producer
         ... FROM items
         ... WHERE category IS NOT NULL
         ...   AND price IS NOT NULL
         ...   AND id IS NOT NULL
         ...   AND producer IS NOT NULL
         ... PRIMARY KEY ((category), producer, price, id);

Warnings :
Materialized views are experimental and are not recommended for production use.

cqlsh:store> SELECT * FROM items_by_producer
         ... WHERE category = 'Electronics'
         ...   AND producer = 'Dell'
         ...   AND price > 500;

 category    | producer | price | id                                   | name
-------------+----------+-------+--------------------------------------+----------
 Electronics |     Dell |  1000 | 1c619c8f-416d-40d0-a31f-94052e98fda3 | Laptop X

(1 rows)
```

1.4 Part 2: orders

Створюємо табличку та додаємо дані

```
cqlsh:store> CREATE TABLE orders (
         ...     customer_name text,
         ...     order_time timestamp,
         ...     order_id uuid,
         ...     items_ids list<uuid>,
         ...     amount decimal,
         ...     PRIMARY KEY ((customer_name), order_time, order_id)
         ... ) WITH CLUSTERING ORDER BY (order_time DESC);
cqlsh:store> BEGIN BATCH
         ...     INSERT INTO orders (customer_name, order_time, order_id, items_ids, amount) VALUES ('Ivan', '2023-10-25 10
         ...     INSERT INTO orders (customer_name, order_time, order_id, items_ids, amount) VALUES ('Ivan', '2023-10-25 12
         ...     INSERT INTO orders (customer_name, order_time, order_id, items_ids, amount) VALUES ('Ivan', '2023-10-26 09
         ...     INSERT INTO orders (customer_name, order_time, order_id, items_ids, amount) VALUES ('Oleg', '2023-10-25 14
         ... APPLY BATCH;
cqlsh:store> SELECT * FROM orders;

 customer_name | order_time                      | order_id                             | amount  | items_ids
---------------+---------------------------------+--------------------------------------+---------+-----------------------
----------------------------
          Oleg | 2023-10-25 14:00:00.000000+0000 | 52b86e8f-0f7e-40a7-bdad-02299779b18e | 1000.00 |
9e4-4c1c-8c3f-243bebad5cab]
          Ivan | 2023-10-26 09:15:00.000000+0000 | 1f315038-65bb-452e-8d14-27f077172e32 |  550.00 | [96e66e66-64f8-4dc1-9dc
717-481f-8f8c-3ab60e6706a2]
          Ivan | 2023-10-25 12:30:00.000000+0000 | 226d1a44-c792-407f-a8ae-058557d60a24 |  200.00 |
2de-4001-bea2-c941471bf2df]
          Ivan | 2023-10-25 10:00:00.000000+0000 | 09d9d5ee-263d-441f-bbe1-33aba7a9bb77 |  150.50 |
8f8-4112-8f6b-f8e4d7b2d150]
```

DESCRIBE TABLE

```
(4 rows)
cqlsh:store> DESCRIBE TABLE orders;

CREATE TABLE store.orders (
    customer_name text,
    order_time timestamp,
    order_id uuid,
    amount decimal,
    items_ids list<uuid>,
    PRIMARY KEY (customer_name, order_time, order_id)
) WITH CLUSTERING ORDER BY (order_time DESC, order_id ASC)
    AND additional_write_policy = '99p'
    AND allow_auto_snapshot = true
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'm
    AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND memtable = 'default'
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND extensions = {}
    AND gc_grace_seconds = 864000
    AND incremental_backups = true
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99p';
```

Всі замовлення замовника (відсортовані за часом)

```
 customer_name | order_time                      | order_id                             | amount | items_ids
---------------+---------------------------------+--------------------------------------+--------+-------------
--------------------------
          Ivan | 2023-10-26 09:15:00.000000+0000 | 1f315038-65bb-452e-8d14-27f077172e32 | 550.00 | [96e66e66-64
17-481f-8f8c-3ab60e6706a2]
          Ivan | 2023-10-25 12:30:00.000000+0000 | 226d1a44-c792-407f-a8ae-058557d60a24 | 200.00 |
de-4001-bea2-c941471bf2df]
          Ivan | 2023-10-25 10:00:00.000000+0000 | 09d9d5ee-263d-441f-bbe1-33aba7a9bb77 | 150.50 |
f8-4112-8f6b-f8e4d7b2d150]

(3 rows)
```

Загальну суму на яку були зроблені усі його замовлення для кожного замовника

```
cqlsh:store> SELECT customer_name, sum(amount) as total_spent
         ... FROM orders
         ... WHERE customer_name = 'Ivan';

 customer_name | total_spent
---------------+-------------
          Ivan |      900.50

(1 rows)
cqlsh:store> SELECT customer_name, sum(amount) as total_spent
         ... FROM orders
         ... WHERE customer_name = 'Oleg';

 customer_name | total_spent
---------------+-------------
          Oleg |     1000.00

(1 rows)
```

Для кожного замовлення час коли його ціна були занесена в базу

```
cqlsh:store> SELECT order_time, amount, WRITETIME(amount)  FROM orders;

 order_time                      | amount  | writetime(amount)
---------------------------------+---------+------------------
 2023-10-25 14:00:00.000000+0000 | 1000.00 |  1764119213917424
 2023-10-26 09:15:00.000000+0000 |  550.00 |  1764119213917424
 2023-10-25 12:30:00.000000+0000 |  200.00 |  1764119213917424
 2023-10-25 10:00:00.000000+0000 |  150.50 |  1764119213917424

(4 rows)
```

2 Task 2

2.1 docker-compose.yml

```
services:
  cassandra-1:
    image: cassandra:latest
    container_name: cassandra-1
    ports:
      - "9042:9042"
    environment:
      - CASSANDRA_CLUSTER_NAME=MyCluster
      - CASSANDRA_DC=datacenter1
      - CASSANDRA_ENDPOINT_SNITCH=GossipingPropertyFileSnitch
      - CASSANDRA_SEEDS=cassandra-1
      - MAX_HEAP_SIZE=512M
      - HEAP_NEWSIZE=100M
    networks:
      - cassandra-net
    healthcheck:
      test: ["CMD", "cqlsh", "-e", "describe keyspaces"]
      interval: 20s
      timeout: 20s
      retries: 10

  cassandra-2:
    image: cassandra:latest
    container_name: cassandra-2
    environment:
      - CASSANDRA_CLUSTER_NAME=MyCluster
      - CASSANDRA_DC=datacenter1
      - CASSANDRA_ENDPOINT_SNITCH=GossipingPropertyFileSnitch
      - CASSANDRA_SEEDS=cassandra-1
      - MAX_HEAP_SIZE=512M
      - HEAP_NEWSIZE=100M
    depends_on:
      cassandra-1:
        condition: service_healthy
    networks:
      - cassandra-net
    healthcheck:
```

```
      test: ["CMD", "cqlsh", "-e", "describe keyspaces"]
      interval: 20s
      timeout: 20s
      retries: 10

  cassandra-3:
    image: cassandra:latest
    container_name: cassandra-3
    environment:
      - CASSANDRA_CLUSTER_NAME=MyCluster
      - CASSANDRA_DC=datacenter1
      - CASSANDRA_ENDPOINT_SNITCH=GossipingPropertyFileSnitch
      - CASSANDRA_SEEDS=cassandra-1
      - MAX_HEAP_SIZE=512M
      - HEAP_NEWSIZE=100M
    depends_on:
      cassandra-2:
        condition: service_healthy
    networks:
      - cassandra-net

networks:
  cassandra-net:
    driver: bridge
```

```
D:\University\semestr_7\High-Load_Systems\lab5>docker exec -it cassandra-1 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens  Owns (effective)  Host ID                               Rack
UN  172.19.0.4  114.62 KiB  16      76.0%             d125b999-a19d-423c-8355-c9d0b29d68b7  rack1
UN  172.19.0.2  119.84 KiB  16      64.7%             c1844ebd-9a86-4038-b9d0-fd23b0fb105c  rack1
UN  172.19.0.3  80.05 KiB   16      59.3%             2a9a931d-e471-4510-906f-eed84048315a  rack1
```

2.2 Keyspace

```
D:\University\semestr_7\High-Load_Systems\lab5>docker exec -it cassandra-1 cqlsh
Connected to MyCluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.6 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE KEYSPACE ks_rf1 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
cqlsh> CREATE KEYSPACE ks_rf2 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 2};
cqlsh> CREATE KEYSPACE ks_rf3 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 3};
cqlsh>
```

2.3 Створення таблиць

```
cqlsh> CREATE TABLE IF NOT EXISTS ks_rf1.users (
   ... user_id int PRIMARY KEY,
   ... username text,
   ... email text
   ... );
cqlsh> CREATE TABLE IF NOT EXISTS ks_rf1.orders (
   ...     order_id uuid PRIMARY KEY,
   ...     user_id int,
   ...     amount decimal
   ... );
cqlsh> CREATE TABLE IF NOT EXISTS ks_rf2.users (
   ...
                  <identifier>       <new_column_name>  <quot
   ...     user_id int PRIMARY KEY,
   ...     username text,
   ...     email text
   ... );
cqlsh> CREATE TABLE IF NOT EXISTS ks_rf2.orders (
   ...     order_id uuid PRIMARY KEY,
   ...     user_id int,
   ...     amount decimal
   ... );
cqlsh> CREATE TABLE IF NOT EXISTS ks_rf3.users (
   ...     user_id int PRIMARY KEY,
   ...     username text,
   ...     email text
   ... );
cqlsh> CREATE TABLE IF NOT EXISTS ks_rf3.orders (
   ...     order_id uuid PRIMARY KEY,
   ...     user_id int,
   ...     amount decimal
   ... );
```

2.4 Запис та читання з різних нод

Запис на першу ноду

```
D:\University\semestr_7\High-Load_Systems\lab5>docker exec -it cassandra-1 cqlsh
Connected to MyCluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.6 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> INSERT INTO ks_rf1.users (user_id, username, email) VALUES (1, 'User1', 'u1@test.com');
cqlsh> INSERT INTO ks_rf1.users (user_id, username, email) VALUES (2, 'User2', 'u2@test.com');
cqlsh> INSERT INTO ks_rf1.users (user_id, username, email) VALUES (3, 'User3', 'u3@test.com');
cqlsh> INSERT INTO ks_rf1.orders (order_id, user_id, amount) VALUES (uuid(), 1, 100.50);
cqlsh> INSERT INTO ks_rf1.orders (order_id, user_id, amount) VALUES (uuid(), 2, 200.00);
cqlsh>
```

Запис на другу ноду та читання

```
D:\University\semestr_7\High-Load_Systems\lab5>docker exec -it cassandra-2 cqlsh
Connected to MyCluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.6 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> select * from ks_rf2.users;

 user_id | email | username
---------+-------+----------

(0 rows)
cqlsh> select * from ks_rf1.users;

 user_id | email       | username
---------+-------------+----------
       1 | u1@test.com |    User1
       2 | u2@test.com |    User2
       3 | u3@test.com |    User3

(3 rows)
cqlsh> INSERT INTO ks_rf2.users (user_id, username, email) VALUES (10, 'User10', 'u10@test.com');
cqlsh> INSERT INTO ks_rf2.users (user_id, username, email) VALUES (11, 'User11', 'u11@test.com');
cqlsh> INSERT INTO ks_rf2.users (user_id, username, email) VALUES (12, 'User12', 'u12@test.com');
cqlsh> INSERT INTO ks_rf2.orders (order_id, user_id, amount) VALUES (uuid(), 10, 500.00);
```

Запис на третю ноду та читання

```
D:\University\semestr_7\High-Load_Systems\lab5>docker exec -it cassandra-3 cqlsh
Connected to MyCluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.6 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> INSERT INTO ks_rf3.users (user_id, username, email) VALUES (20, 'User20', 'u20@test.com');
cqlsh> INSERT INTO ks_rf3.users (user_id, username, email) VALUES (21, 'User21', 'u21@test.com');
cqlsh> INSERT INTO ks_rf3.users (user_id, username, email) VALUES (22, 'User22', 'u22@test.com');
cqlsh> INSERT INTO ks_rf3.orders (order_id, user_id, amount) VALUES (uuid(), 20, 999.99);
cqlsh> select * from ks_rf1.users;

 user_id | email       | username
---------+-------------+----------
       1 | u1@test.com |    User1
       2 | u2@test.com |    User2
       3 | u3@test.com |    User3

(3 rows)
cqlsh> select * from ks_rf2.users;

 user_id | email        | username
---------+--------------+----------
      10 | u10@test.com |   User10
      11 | u11@test.com |   User11
      12 | u12@test.com |   User12

(3 rows)
```

2.5 Розподіл даних по вузлах

```
D:\University\semestr_7\High-Load_Systems\lab5>docker exec -it cassandra-1 nodetool status ks_rf1
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns (effective)  Host ID                               Rack
UN  172.19.0.4  86.47 KiB   16      35.7%             d125b999-a19d-423c-8355-c9d0b29d68b7  rack1
UN  172.19.0.2  205.69 KiB  16      32.7%             c1844ebd-9a86-4038-b9d0-fd23b0fb105c  rack1
UN  172.19.0.3  177.75 KiB  16      31.6%             2a9a931d-e471-4510-906f-eed84048315a  rack1


D:\University\semestr_7\High-Load_Systems\lab5>docker exec -it cassandra-1 nodetool status ks_rf2
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns (effective)  Host ID                               Rack
UN  172.19.0.4  177.28 KiB  16      76.0%             d125b999-a19d-423c-8355-c9d0b29d68b7  rack1
UN  172.19.0.2  205.69 KiB  16      64.7%             c1844ebd-9a86-4038-b9d0-fd23b0fb105c  rack1
UN  172.19.0.3  177.75 KiB  16      59.3%             2a9a931d-e471-4510-906f-eed84048315a  rack1


D:\University\semestr_7\High-Load_Systems\lab5>docker exec -it cassandra-1 nodetool status ks_rf3
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns (effective)  Host ID                               Rack
UN  172.19.0.4  177.28 KiB  16      100.0%            d125b999-a19d-423c-8355-c9d0b29d68b7  rack1
UN  172.19.0.2  205.69 KiB  16      100.0%            c1844ebd-9a86-4038-b9d0-fd23b0fb105c  rack1
UN  172.19.0.3  177.75 KiB  16      100.0%            2a9a931d-e471-4510-906f-eed84048315a  rack1
```

- Для `ks_rf1` приблизно 33% на кожну з нод, це означає дані розмазані по кластеру без дублювання.
- Для `ks_rf2` кожна нода показує ≈66%, бо вона зберігає свої дані + репліку однієї іншої ноди.
- Для `ks_rf3` кожна нода показує 100%, тобто кожна нода має повну копію всіх даних цього кейспейсу.

Це підтверджується наступними запитами:

```
D:\University\semestr_7\High-Load_Systems\lab5>docker exec -it cassandra-1 nodetool getendpoints ks_rf1 users 1
172.19.0.3

D:\University\semestr_7\High-Load_Systems\lab5>docker exec -it cassandra-1 nodetool getendpoints ks_rf2 users 10
172.19.0.3
172.19.0.4

D:\University\semestr_7\High-Load_Systems\lab5>docker exec -it cassandra-1 nodetool getendpoints ks_rf3 users 20
172.19.0.4
172.19.0.2
172.19.0.3
```

2.6 Перевірка рівней consistency

Стопаємо ноду - docker stop cassandra-3.

2.6.1 Для Keyspace з replication factor 3

Consistency level: ONE

```
cqlsh> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh> INSERT INTO ks_rf3.users (user_id, username) VALUES (999, 'TestOne');
cqlsh> SELECT * FROM ks_rf3.users;

 user_id | email         | username
---------+---------------+----------
      22 | u22@test.com |   User22
      20 | u20@test.com |   User20
      21 | u21@test.com |   User21
     999 |          null |  TestOne

(4 rows)
```

Consistency level: QUORUM (TWO)

```
cqlsh> CONSISTENCY QUORUM;
Consistency level set to QUORUM.
cqlsh> INSERT INTO ks_rf3.users (user_id, username) VALUES (888, 'TestQuorum');
cqlsh> SELECT * FROM ks_rf3.users;

 user_id | email         | username
---------+---------------+------------
      22 | u22@test.com |     User22
     888 |          null | TestQuorum
      20 | u20@test.com |     User20
      21 | u21@test.com |     User21
     999 |          null |    TestOne

(5 rows)
```

Consistency level: ALL (THREE)

```
cqlsh> CONSISTENCY ALL;
Consistency level set to ALL.
cqlsh> INSERT INTO ks_rf3.users (user_id, username) VALUES (777, 'TestAll');
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 datacenter1>: Unavailable('E
not achieve consistency level ALL" info={\'consistency\': \'ALL\', \'required_replicas\': 3, \'alive_replicas\': 2}')})
cqlsh> SELECT * FROM ks_rf3.users;
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 datacenter1>: Unavailable('E
not achieve consistency level ALL" info={\'consistency\': \'ALL\', \'required_replicas\': 3, \'alive_replicas\': 2}')})
```

2.6.2 Для Keyspace з replication factor 2

Consistency level: ONE

```
cqlsh> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh> SELECT * FROM ks_rf2.users LIMIT 1;

 user_id | email        | username
---------+--------------+----------
      10 | u10@test.com |   User10

(1 rows)
cqlsh> INSERT INTO ks_rf2.users (user_id, username) VALUES (2001, 'WriteOne');
cqlsh> SELECT * FROM ks_rf2.users;

 user_id | email        | username
---------+--------------+----------
      10 | u10@test.com |   User10
      11 | u11@test.com |   User11
    2001 |         null | WriteOne
      12 | u12@test.com |   User12

(4 rows)
```

Consistency level: ALL (TWO)

```
(4 rows)
cqlsh> CONSISTENCY ALL;
Consistency level set to ALL.
cqlsh> INSERT INTO ks_rf2.users (user_id, username) VALUES (2002, 'WriteAll_1');
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 datacenter1>: Unavailable('
not achieve consistency level ALL" info={\'consistency\': \'ALL\', \'required_replicas\': 2, \'alive_replicas\': 1}')})
cqlsh>  SELECT * FROM ks_rf2.users;
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 datacenter1>: Unavailable('
not achieve consistency level ALL" info={\'consistency\': \'ALL\', \'required_replicas\': 2, \'alive_replicas\': 1}')})
```

2.6.3 Для Keyspace з replication factor 1

Consistency level: ONE

```
cqlsh> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh> INSERT INTO ks_rf1.users (user_id, username) VALUES (100, 'RiskyData');
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 da
not achieve consistency level ONE" info={\'consistency\': \'ONE\', \'required_replicas\': 1, \'al
cqlsh> INSERT INTO ks_rf1.users (user_id, username) VALUES (100, 'RiskyData');
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 da
not achieve consistency level ONE" info={\'consistency\': \'ONE\', \'required_replicas\': 1, \'al
cqlsh> INSERT INTO ks_rf1.users (user_id, username) VALUES (30, 'RiskyData');
```

Як видно для варіанта запису з UserId 100 вишла помилка оскільки хеш ключа UserId 100 потрапив у діапазон токенів, за який відповідає саме відключена нода, а для UserId 30 воно записалось тому що хеш цього ключа вказує на одну з живих нод кластера, яка успішно прийняла та обробила запит.

2.7 Блокування зв’язків

```
D:\University\semestr_7\High-Load_Systems\lab5>docker network disconnect lab5_cassandra-net cassandra-1

D:\University\semestr_7\High-Load_Systems\lab5>docker network disconnect lab5_cassandra-net cassandra-2

D:\University\semestr_7\High-Load_Systems\lab5>docker network disconnect lab5_cassandra-net cassandra-3

D:\University\semestr_7\High-Load_Systems\lab5>docker exec -it cassandra-1 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns  Host ID                               Rack
DN  172.19.0.4  153.62 KiB  16      ?     d125b999-a19d-423c-8355-c9d0b29d68b7  rack1
UN  172.19.0.2  203.51 KiB  16      ?     c1844ebd-9a86-4038-b9d0-fd23b0fb105c  rack1
DN  172.19.0.3  174.82 KiB  16      ?     2a9a931d-e471-4510-906f-eed84048315a  rack1

Note: Non-system keyspaces don't have the same replication settings, effective ownership information is meaningless

D:\University\semestr_7\High-Load_Systems\lab5>docker exec -it cassandra-2 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns  Host ID                               Rack
DN  172.19.0.4  153.62 KiB  16      ?     d125b999-a19d-423c-8355-c9d0b29d68b7  rack1
DN  172.19.0.2  203.51 KiB  16      ?     c1844ebd-9a86-4038-b9d0-fd23b0fb105c  rack1
UN  172.19.0.3  174.82 KiB  16      ?     2a9a931d-e471-4510-906f-eed84048315a  rack1

Note: Non-system keyspaces don't have the same replication settings, effective ownership information is meaningless
```

2.8 Створення конфлікта

Для розірваного кластера створюємо три версії реальності для ключа 9999

```
D:\University\semestr_7\High-Load_Systems\lab5>docker exec -it cassandra-1 cqlsh
Connected to MyCluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.6 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh> INSERT INTO ks_rf3.users (user_id, username, email) VALUES (9999, 'User_from_Node_1', 'mail1@test.com');
cqlsh>
D:\University\semestr_7\High-Load_Systems\lab5>docker exec -it cassandra-2 cqlsh
Connected to MyCluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.6 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> CONSISTENCY ONE; INSERT INTO ks_rf3.users (user_id, username, email) VALUES (9999, 'User_from_Node_2', 'mail2@test.co
Consistency level set to ONE.
cqlsh>
D:\University\semestr_7\High-Load_Systems\lab5>docker exec -it cassandra-3 cqlsh
Connected to MyCluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.6 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> CONSISTENCY ONE; INSERT INTO ks_rf3.users (user_id, username, email) VALUES (9999, 'User_from_Node_3', 'mail3@test.co
Consistency level set to ONE.
```

Відновлюємо кластер

```
cqlsh>
D:\University\semestr_7\High-Load_Systems\lab5>docker network connect lab5_cassandra-net cassandra-1

D:\University\semestr_7\High-Load_Systems\lab5>docker network connect lab5_cassandra-net cassandra-2

D:\University\semestr_7\High-Load_Systems\lab5>docker network connect lab5_cassandra-net cassandra-3

D:\University\semestr_7\High-Load_Systems\lab5>docker exec -it cassandra-1 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens  Owns  Host ID                               Rack
UN  172.19.0.4  153.62 KiB  16      ?     d125b999-a19d-423c-8355-c9d0b29d68b7  rack1
UN  172.19.0.2  203.51 KiB  16      ?     c1844ebd-9a86-4038-b9d0-fd23b0fb105c  rack1
UN  172.19.0.3  174.82 KiB  16      ?     2a9a931d-e471-4510-906f-eed84048315a  rack1
```

Бачимо, що Cassandra використовує стратегію Last Write Wins, тобто вибирається остання змінена версія - в нашому випадку User_from_Node_3 бо ми його якраз і додавали останнім.

```
D:\University\semestr_7\High-Load_Systems\lab5>docker exec -it cassandra-1 cqlsh
Connected to MyCluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.6 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> CONSISTENCY ALL;
Consistency level set to ALL.
cqlsh> SELECT user_id, username, WRITETIME(username) FROM ks_rf3.users WHERE user_id = 9999;

 user_id | username         | writetime(username)
---------+------------------+---------------------
    9999 | User_from_Node_3 |    1764351852750148
```

3 Task 3

```
(venv7s) D:\University\semestr_7\High-Load_Systems\lab5>docker run --rm --network lab5_cassandra-net cassandra-tester
--- 1. Налаштування інфраструктури ---
Keyspace 'ks_rf3' перевірено/створено.
Таблиця 'likes_counter' перевірена/створена.

ЗАПУСК ТЕСТУ: CONSISTENCY LEVEL ONE
Лічильник скинуто.
Час виконання: 68.90 сек
Очікувано: 100000
Отримано:  100000
РЕЗУЛЬТАТ: цілісність збережено

ЗАПУСК ТЕСТУ: CONSISTENCY LEVEL QUORUM
Лічильник скинуто.
Час виконання: 72.65 сек
Очікувано: 100000
Отримано:  100000
РЕЗУЛЬТАТ: цілісність збережено
```

Як бачимо, обидва Consistency Levels забезпечили 100% збереження даних. Відсутність втрат при використанні Consistency Level ONE пояснюється тим, що під час тестування кластер працював стабільно, і збоїв обладнання чи мережі не відбулося. Ризик втрати даних при цьому рівні існує лише у випадку раптового відключення ноди до завершення фонової реплікації.

Посилання на репозиторій: GitHub