Міністерство освіти і науки України Національний технічний університет України «Київський політехнічний інститут ім. І. Сікорського » Фізико-технічний інститут

Проектування високонавантажених систем

Лабораторна робота №5

Робота з базовими функціями БД типу column family на прикладі Cassandra

Виконав:

Студент групи ФБ-41мп

Заріцький О. В.

Київ 2025 р.

Частина 1. Робота зі структурами даних у Cassandra

Ознайомтеся з особливістю моделювання даних у Cassandra, створіть keyspace з найпростішої стратегією реплікації

compose1.yml

```yaml
version: '3.9'
services:
  cassandra:
    image: cassandra:4.1
    container_name: cassandra
    ports:
      - "9042:9042"
    environment:
      - CASSANDRA_CLUSTER_NAME=TestCluster
      - CASSANDRA_NUM_TOKENS=16
      - CASSANDRA_DC=datacenter1
      - CASSANDRA_RACK=rack1
    volumes:
      - ./cassandra.yaml:/home/bober/zaritskyi_lab5/cassandra.yaml
      - ./data:/var/lib/cassandra
```

CREATE KEYSPACE lab5_keyspace
WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};

```
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra cqlsh
Connected to TestCluster at 127.0.0.1:9042
[cqlsh 6.1.0 | Cassandra 4.1.7 | CQL spec 3.4.6 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE KEYSPACE lab5_keyspace
   ... WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
cqlsh>
```

У таблиці items містить різноманітні товари (тобто у яких різний набір властивостей).

Для набору властивостей товару виберіть базові характеристики однакові для всіх товарів (назва, категорія, ціна, виробник, ...), а для властивостей які відрізняються використовуйте тип map (з індексом для можливості пошуку по її вмісту)

Необхідно, щоб пошук швидко працював для категорії товарів. Ця вимога має бути врахована при створенні ключа для таблиці.

CREATE TABLE lab5_keyspace.items (
category TEXT,
product_id UUID,
name TEXT,
price DECIMAL,
manufacturer TEXT,
specifications MAP<TEXT, TEXT>,
PRIMARY KEY (category, price, product_id)
);

INSERT INTO lab5_keyspace.items (category, product_id, name, price, manufacturer, specifications)
VALUES ('Home Appliances', uuid(), 'Vacuum Cleaner', 150, 'Dyson', {'power': '1200W', 'weight': '3.5kg', 'color': 'Silver'});

INSERT INTO lab5_keyspace.items (category, product_id, name, price, manufacturer, specifications)
VALUES ('Electronics', uuid(), 'Tablet', 250, 'Apple', {'screen_size': '10.2 inch', 'RAM': '4GB', 'processor': 'A13 Bionic'});

INSERT INTO lab5_keyspace.items (category, product_id, name, price, manufacturer, specifications)
VALUES ('Furniture', uuid(), 'Chair', 75, 'Ikea', {'material': 'Wood', 'height': '3.5 ft', 'color': 'Brown'});

```
cqlsh> CREATE TABLE lab5_keyspace.items (
   ...     category TEXT,
   ...     product_id UUID,
   ...     name TEXT,
   ...     price DECIMAL,
   ...     manufacturer TEXT,
   ...     specifications MAP<TEXT, TEXT>,
   ...     PRIMARY KEY (category, price, product_id)
   ... );
cqlsh> INSERT INTO lab5_keyspace.items (category, product_id, name, price, manufacturer, specifications)
   ... VALUES ('Home Appliances', uuid(), 'Vacuum Cleaner', 150, 'Dyson', {'power': '1200W', 'weight': '3.5kg', 'color': 'Silver'});
cqlsh> INSERT INTO lab5_keyspace.items (category, product_id, name, price, manufacturer, specifications)
   ... VALUES ('Electronics', uuid(), 'Tablet', 250, 'Apple', {'screen_size': '10.2 inch', 'RAM': '4GB', 'processor': 'A13 Bionic'});
cqlsh> INSERT INTO lab5_keyspace.items (category, product_id, name, price, manufacturer, specifications)
   ... VALUES ('Furniture', uuid(), 'Chair', 75, 'Ikea', {'material': 'Wood', 'height': '3.5 ft', 'color': 'Brown'});
cqlsh>
```

1. Напишіть запит, який показує структуру створеної таблиці (команда DESCRIBE)

DESCRIBE TABLE lab5_keyspace.items;

```
cqlsh> DESCRIBE TABLE lab5_keyspace.items;

CREATE TABLE lab5_keyspace.items (
    category text,
    price decimal,
    product_id uuid,
    manufacturer text,
    name text,
    specifications map<text, text>,
    PRIMARY KEY (category, price, product_id)
) WITH CLUSTERING ORDER BY (price ASC, product_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', 'min_threshold': '4'}
    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';
cqlsh>
```

2. Напишіть запит, який виводить усі товари в певній категорії відсортовані за ціною

*SELECT * FROM lab5_keyspace.items*
WHERE category = 'Electronics'
ORDER BY price ASC;

```
cqlsh> SELECT * FROM lab5_keyspace.items
   ... WHERE category = 'Electronics'
   ... ORDER BY price ASC;

 category    | price | product_id                           | manufacturer | name   | specifications
-------------+-------+--------------------------------------+--------------+--------+-----------------------------------------------------------------------------------
 Electronics |   250 | e30e693b-4f45-4c2d-8509-bc722185c722 |        Apple | Tablet | {'RAM': '4GB', 'processor': 'A13 Bionic', 'screen_size': '10.2 inch'}

(1 rows)
cqlsh>
```

3. Напишіть запити, які вибирають товари за різними критеріями в межах певної категорії (тут де треба замість індексу використайте Matirialized view):
    - назва,

CREATE MATERIALIZED VIEW lab5_keyspace.items_by_name AS
SELECT category, name, price, product_id, manufacturer, specifications
FROM lab5_keyspace.items
WHERE category IS NOT NULL AND name IS NOT NULL AND price IS NOT NULL AND product_id IS NOT NULL
PRIMARY KEY ((category), name, price, product_id);

*SELECT * FROM lab5_keyspace.items_by_name*
WHERE category = 'Furniture' AND name = 'Chair';

```
cqlsh> SELECT * FROM lab5_keyspace.items_by_name
   ... WHERE category = 'Furniture' AND name = 'Chair';

 category  | name  | price | product_id                           | manufacturer | specifications
-----------+-------+-------+--------------------------------------+--------------+-------------------------------------------------------------
 Furniture | Chair |    75 | f258f150-f4b7-48ca-887a-0c1924335e0a |         Ikea | {'color': 'Brown', 'height': '3.5 ft', 'material': 'Wood'}

(1 rows)
cqlsh>
```

    - ціна (в проміжку)

CREATE MATERIALIZED VIEW lab5_keyspace.items_by_price_range AS
SELECT category, price, product_id, name, manufacturer, specifications
FROM lab5_keyspace.items
WHERE category IS NOT NULL AND price IS NOT NULL AND product_id IS NOT NULL
PRIMARY KEY ((category), price, product_id);

*SELECT * FROM lab5_keyspace.items_by_price_range*
WHERE category = 'Electronics' AND price >= 200 AND price <= 600;

```
cqlsh> SELECT * FROM lab5_keyspace.items_by_price_range
   ... WHERE category = 'Electronics' AND price >= 200 AND price <= 600;

 category    | price | product_id                           | manufacturer | name   | specifications
-------------+-------+--------------------------------------+--------------+--------+-----------------------------------------------------------------------------------
 Electronics |   250 | e30e693b-4f45-4c2d-8509-bc722185c722 |        Apple | Tablet | {'RAM': '4GB', 'processor': 'A13 Bionic', 'screen_size': '10.2 inch'}

(1 rows)
cqlsh>
```

- ціна та виробник

CREATE MATERIALIZED VIEW lab5_keyspace.items_by_price_and_manufacturer AS
SELECT category, price, manufacturer, product_id, name, specifications
FROM lab5_keyspace.items
WHERE category IS NOT NULL AND price IS NOT NULL AND manufacturer IS NOT NULL AND product_id IS NOT NULL
PRIMARY KEY ((category), manufacturer, price, product_id);

*SELECT * FROM lab5_keyspace.items_by_price_and_manufacturer*
WHERE category = 'Electronics' AND manufacturer = 'Apple' AND price = 250;

```
cqlsh> SELECT * FROM lab5_keyspace.items_by_price_and_manufacturer
   ... WHERE category = 'Electronics' AND manufacturer = 'Apple' AND price = 250;

 category    | manufacturer | price | product_id                           | name   | specifications
-------------+--------------+-------+--------------------------------------+--------+-----------------------------------------------------------------------------------------
 Electronics |        Apple |   250 | e30e693b-4f45-4c2d-8509-bc722185c722 | Tablet | {'RAM': '4GB', 'processor': 'A13 Bionic', 'screen_size': '10.2 inch'}

(1 rows)
cqlsh>
```

Створіть таблицю orders в якій міститься ім'я замовника і інформація про замовлення: перелік id-товарів у замовленні, вартість замовлення, дата замовлення, ...

Для кожного замовника повинна бути можливість швидко шукати його замовлення і виконувати по них запити. Ця вимога має бути врахована при створенні ключа для таблиці.

CREATE TABLE lab5_keyspace.orders (
customer_name TEXT,
order_date TIMESTAMP,
order_id UUID,
total_price DECIMAL,
item_ids LIST<UUID>,
PRIMARY KEY (customer_name, order_date, order_id)
);

INSERT INTO lab5_keyspace.orders (customer_name, order_id, item_ids, total_price, order_date)
VALUES ('Alice', uuid(), [uuid(), uuid()], 100.00, '2025-01-14 10:00:00');

INSERT INTO lab5_keyspace.orders (customer_name, order_id, item_ids, total_price, order_date)
VALUES ('Bob', uuid(), [uuid()], 50.00, '2025-01-13 14:30:00');

INSERT INTO lab5_keyspace.orders (customer_name, order_id, item_ids, total_price, order_date)
VALUES ('Alice', uuid(), [uuid()], 150.00, '2025-01-12 18:45:00');

1. Напишіть запит, який показує структуру створеної таблиці (команда DESCRIBE)

```
cqlsh> DESCRIBE TABLE lab5_keyspace.orders;

CREATE TABLE lab5_keyspace.orders (
    customer_name text,
    order_date timestamp,
    order_id uuid,
    total_price decimal,
    item_ids list<uuid>,
    PRIMARY KEY (customer_name, order_date, order_id)
) WITH CLUSTERING ORDER BY (order_date ASC, order_id ASC)
    AND additional_write_policy = '99p'
    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', 'min_threshold': '4'}
    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 max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99p';
cqlsh>
```

2. Для замовника виведіть всі його замовлення відсортовані за часом коли вони були зроблені

*SELECT * FROM lab5_keyspace.orders*
WHERE customer_name = 'Alice'
ORDER BY order_date DESC;

```
cqlsh> SELECT * FROM lab5_keyspace.orders
   ... WHERE customer_name = 'Alice'
   ... ORDER BY order_date DESC;

 customer_name | order_date                      | order_id                             | item_ids                                                                     | total_price
---------------+---------------------------------+--------------------------------------+------------------------------------------------------------------------------+-------------
         Alice | 2025-01-14 10:00:00.000000+0000 | 448713fd-a077-4c78-900e-fac209fba2b4 | [138c94f5-3b29-4455-a1bf-4702c292e5af, 15f6b4ea-3ec7-4c41-916d-181748abe04f] |      100.00
         Alice | 2025-01-12 18:45:00.000000+0000 | 538b4186-1cb4-404e-88d6-ab1061514a2c |                                       [869c2f9a-85da-4330-a6e9-310925d493f2] |      150.00

(2 rows)
cqlsh>
```

3. Для кожного замовників визначте суму на яку були зроблені усі його замовлення

SELECT customer_name, SUM(total_price) AS total_spent
FROM lab5_keyspace.orders
GROUP BY customer_name;

```
cqlsh> SELECT customer_name, SUM(total_price) AS total_spent
   ... FROM lab5_keyspace.orders
   ... GROUP BY customer_name;

 customer_name | total_spent
---------------+-------------
           Bob |       50.00
         Alice |      250.00

(2 rows)

Warnings :
Aggregation query used without partition key

cqlsh>
```

4. Для кожного замовлення виведіть час коли його ціна були занесена в базу (SELECT WRITETIME)

   SELECT WRITETIME(total_price) AS write_time, total_price
   FROM lab5_keyspace.orders
   WHERE customer_name = 'Alice' AND order_date = '2025-01-14 10:00:00';

```
cqlsh> SELECT WRITETIME(total_price) AS write_time, total_price
   ... FROM lab5_keyspace.orders
   ... WHERE customer_name = 'Alice' AND order_date = '2025-01-14 10:00:00';

 write_time       | total_price
------------------+-------------
 1736888813472259 |      100.00

(1 rows)
cqlsh>
```

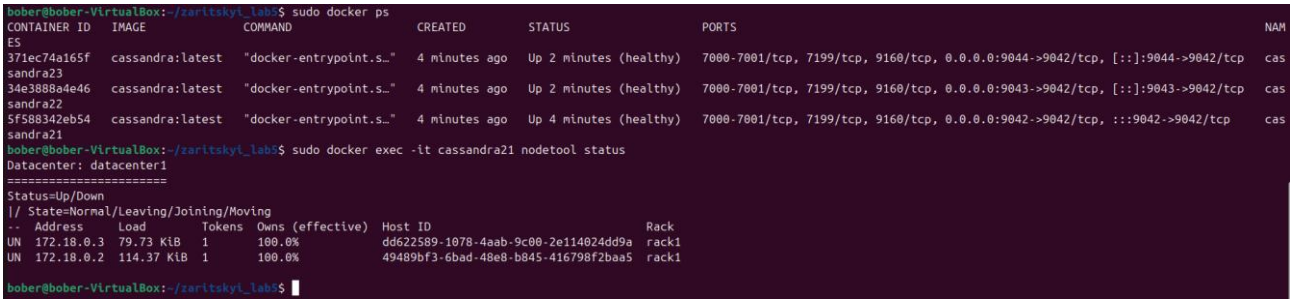
Частина 2. Налаштування реплікації у Cassandra

1. Сконфігурувати кластер з 3-х нод:
   compose2.yml

```yaml
services:
  cassandra1:
    image: cassandra:latest
    container_name: cassandra21
    environment:
      - CASSANDRA_CLUSTER_NAME=Test2Cluster
      - CASSANDRA_SEEDS=cassandra1
      - CASSANDRA_START_RPC=true
      - CASSANDRA_NUM_TOKENS=1
      - JVM_OPTS=-Dcassandra.skip_schema_check=true
    networks:
      - cassandra_net
    ports:
      - "9042:9042"
    healthcheck:
      test: ["CMD", "nodetool", "status"]
      interval: 30s
      timeout: 10s
      retries: 3
    volumes:
      - ./cassandra/cassandra1.yaml:/etc/cassandra/cassandra21.yaml

  cassandra2:
    image: cassandra:latest
    container_name: cassandra22
    environment:
      - CASSANDRA_CLUSTER_NAME=Test2Cluster
      - CASSANDRA_SEEDS=cassandra1
      - CASSANDRA_START_RPC=true
      - CASSANDRA_NUM_TOKENS=1
      - JVM_OPTS=-Dcassandra.skip_schema_check=true
    networks:
      - cassandra_net
    depends_on:
      cassandra1:
        condition: service_healthy
    volumes:
      - ./cassandra/cassandra2.yaml:/etc/cassandra/cassandra22.yaml

  cassandra3:
    image: cassandra:latest
    container_name: cassandra23
    environment:
      - CASSANDRA_CLUSTER_NAME=Test2Cluster
```

```
      - CASSANDRA_SEEDS=cassandra1
      - CASSANDRA_START_RPC=true
      - CASSANDRA_NUM_TOKENS=1
      - JVM_OPTS=-Dcassandra.skip_schema_check=true
    networks:
      - cassandra_net
    depends_on:
      cassandra1:
        condition: service_healthy
    volumes:
      - ./cassandra/cassandra3.yaml:/etc/cassandra/cassandra23.yaml

networks:
  cassandra_net:
    driver: bridge
```

2. Перевірити правильність конфігурації за допомогою *nodetool status*



3. Використовуючи cqlsh, створити три Keyspace з replication factor 1, 2, 3 з SimpleStrategy

CREATE KEYSPACE keyspace_rf1
WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};

CREATE KEYSPACE keyspace_rf2
WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 2};

CREATE KEYSPACE keyspace_rf3
WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 3};

```
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra21 cqlsh
Connected to Test2Cluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.2 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE KEYSPACE keyspace_rf1
   ... WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};

cqlsh>
cqlsh> CREATE KEYSPACE keyspace_rf2
   ... WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 2};

cqlsh>
cqlsh> CREATE KEYSPACE keyspace_rf3
   ... WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 3};


cqlsh>
cqlsh>
cqlsh> DESCRIBE KEYSPACES;

keyspace_rf1  system              system_schema  system_virtual_schema
keyspace_rf2  system_auth         system_traces
keyspace_rf3  system_distributed  system_views

cqlsh>
```

4. В кожному з кейспейсів створити прості таблиці

```
USE keyspace_rf1;

CREATE TABLE test_table (
    id UUID PRIMARY KEY,
    value TEXT
);

USE keyspace_rf2;

CREATE TABLE test_table (
    id UUID PRIMARY KEY,
    value TEXT
);

USE keyspace_rf3;

CREATE TABLE test_table (
    id UUID PRIMARY KEY,
    value TEXT
);
```

```
cqlsh:keyspace_rf1> USE keyspace_rf2;
cqlsh:keyspace_rf2> CREATE TABLE test_table (
                ...     id UUID PRIMARY KEY,
                ...     value TEXT
                ... );

cqlsh:keyspace_rf2>
cqlsh:keyspace_rf2> USE keyspace_rf3;
cqlsh:keyspace_rf3> CREATE TABLE test_table (
                ...     id UUID PRIMARY KEY,
                ...     value TEXT
                ... );

cqlsh:keyspace_rf3>
cqlsh:keyspace_rf3> USE keyspace_rf1;
cqlsh:keyspace_rf1> CREATE TABLE test_table (
                ...     id UUID PRIMARY KEY,
                ...     value TEXT
                ... );
```

5. Спробуйте писати і читати в ці таблиці підключаюсь на різні ноди.

```
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra21 cqlsh
Connected to Test2Cluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.2 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> USE keyspace_rf1;
cqlsh:keyspace_rf1> INSERT INTO test_table (id, value) VALUES (uuid(), 'RF1');
cqlsh:keyspace_rf1> exit
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra22 cqlsh
Connected to Test2Cluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.2 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> USE keyspace_rf2;
cqlsh:keyspace_rf2> INSERT INTO test_table (id, value) VALUES (uuid(), 'RF1');
cqlsh:keyspace_rf2> INSERT INTO test_table (id, value) VALUES (uuid(), 'RF2');
cqlsh:keyspace_rf2> exit
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra23 cqlsh
Connected to Test2Cluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.2 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> USE keyspace_rf3;
cqlsh:keyspace_rf3> INSERT INTO test_table (id, value) VALUES (uuid(), 'RF3');
cqlsh:keyspace_rf3>
```

6. Вставте дані в створені таблиці і подивіться на їх розподіл по вузлах кластера для кожного з кейспесов (команда nodetool status)

```
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra21 cqlsh
Connected to Test2Cluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.2 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> USE keyspace_rf1;
cqlsh:keyspace_rf1> select * from test_table;

 id                                   | value
--------------------------------------+-------
 5cc0c245-2c2b-487f-bf3c-23562fce0736 |   RF1
 fc1e4894-f30f-46af-a0f8-cda58b59bd5d |   RF1

(2 rows)
cqlsh:keyspace_rf1> exit
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra21 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns  Host ID                               Rack
UN  172.18.0.3  80.63 KiB   1       ?     dd622589-1078-4aab-9c00-2e114024dd9a  rack1
UN  172.18.0.2  115.63 KiB  1       ?     49489bf3-6bad-48e8-b845-416798f2baa5  rack1

Note: Non-system keyspaces don't have the same replication settings, effective ownership information is meaningless
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra21 nodetool getendpoints keyspace_rf1 test_table 5cc0c245-2c2b-487f-bf3c-23562fce0736
172.18.0.3
bober@bober-VirtualBox:~/zaritskyi_lab5$
```

```
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra22 cqlsh
Connected to Test2Cluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.2 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> USE keyspace_rf2;
cqlsh:keyspace_rf2> select * from test_table;

 id                                   | value
--------------------------------------+-------
 07179457-198d-4709-a7ca-ee27e7d36442 |   RF2
 803cdee0-f8dc-4705-91f0-d50bf9784bd3 |   RF1

(2 rows)
cqlsh:keyspace_rf2> exit
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra22 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns  Host ID                               Rack
UN  172.18.0.4  80.66 KiB   1       ?     ffc010cf-c440-43e9-b1d0-f5df1dcd9802  rack1
UN  172.18.0.2  115.63 KiB  1       ?     49489bf3-6bad-48e8-b845-416798f2baa5  rack1

Note: Non-system keyspaces don't have the same replication settings, effective ownership information is meaningless
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra22 nodetool getendpoints keyspace_rf2 test_table 07179457-198d-4709-a7ca-ee27e7d36442
172.18.0.4
172.18.0.2
bober@bober-VirtualBox:~/zaritskyi_lab5$
```

```
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra23 cqlsh
Connected to Test2Cluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.2 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> USE keyspace_rf3;
cqlsh:keyspace_rf3> select * from test_table;

 id                                   | value
--------------------------------------+-------
 3ae9b657-22de-4e0e-89d5-cb30ea6d039a |   RF3

(1 rows)
cqlsh:keyspace_rf3> exit
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra23 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns  Host ID                               Rack
UN  172.18.0.3  80.63 KiB   1       ?     dd622589-1078-4aab-9c00-2e114024dd9a  rack1
UN  172.18.0.2  115.63 KiB  1       ?     49489bf3-6bad-48e8-b845-416798f2baa5  rack1

Note: Non-system keyspaces don't have the same replication settings, effective ownership information is meaningless
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra23 nodetool getendpoints keyspace_rf3 test_table 3ae9b657-22de-4e0e-89d5-cb30ea6d039a
172.18.0.3
172.18.0.2
```

7. Для якогось запису з кожного з кейспейсу виведіть ноди на яких зберігаються дані

```
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra21 nodetool getendpoints keyspace_rf1 test_table 5cc0c245-2c2b-487f-bf3c-23562fce0736
172.18.0.3
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra21 nodetool getendpoints keyspace_rf2 test_table 07179457-198d-4709-a7ca-ee27e7d36442
172.18.0.3
172.18.0.2
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra21 nodetool getendpoints keyspace_rf3 test_table 3ae9b657-22de-4e0e-89d5-cb30ea6d039a
172.18.0.3
172.18.0.2
bober@bober-VirtualBox:~/zaritskyi_lab5$
```

8. Відключити одну з нод. Для кожного з кейспейсів перевірити з якими рівнями consistency можемо читати та писати
    - для Keyspace з replication factor 1 - CONSISTENCY ONE
    - для Keyspace з replication factor 2 - CONSISTENCY ONE/TWO
    - для Keyspace з replication factor 3 - CONSISTENCY ONE/TWO/THREE

```
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker stop cassandra23
cassandra23
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra21 cqlsh
Connected to Test2Cluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.2 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh> SELECT * FROM keyspace_rf1.test_table;
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 datacenter1>: Unavailable('Error from server: code=1000 [Unavailable exception] message="
Cannot achieve consistency level ONE" info={\'consistency\': \'ONE\', \'required_replicas\': 1, \'alive_replicas\': 0}')})
cqlsh> CONSISTENCY TWO;
Consistency level set to TWO.
cqlsh> SELECT * FROM keyspace_rf2.test_table;
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 datacenter1>: Unavailable('Error from server: code=1000 [Unavailable exception] message="
Cannot achieve consistency level TWO" info={\'consistency\': \'TWO\', \'required_replicas\': 2, \'alive_replicas\': 1}')})
cqlsh> CONSISTENCY THREE;
Consistency level set to THREE.
cqlsh> SELECT * FROM keyspace_rf3.test_table;
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 datacenter1>: Unavailable('Error from server: code=1000 [Unavailable exception] message="
Cannot achieve consistency level THREE" info={\'consistency\': \'THREE\', \'required_replicas\': 3, \'alive_replicas\': 1}')})
cqlsh>
```

9. Зробить так щоб три ноди працювали, але не бачили одна одну по мережі (заблокуйте чи відключити зв'язок між ними)

```
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker network disconnect zaritskyi_lab5_cassandra_net cassandra22
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker network disconnect zaritskyi_lab5_cassandra_net cassandra23
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra21 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns  Host ID                               Rack
DN  172.18.0.3  121.2 KiB   1       ?     dd622589-1078-4aab-9c00-2e114024dd9a  rack1
UN  172.18.0.2  115.63 KiB  1       ?     49489bf3-6bad-48e8-b845-416798f2baa5  rack1

Note: Non-system keyspaces don't have the same replication settings, effective ownership information is meaningless
bober@bober-VirtualBox:~/zaritskyi_lab5$
```

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)

```
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra22 cqlsh
Connected to Test2Cluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.2 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh> USE keyspace_rf3;
cqlsh:keyspace_rf3> INSERT INTO test_table (id, value) VALUES (uuid(), 'Test_Value1');
cqlsh:keyspace_rf3> exit
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra23 cqlsh
Connected to Test2Cluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.2 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh> USE keyspace_rf3;
cqlsh:keyspace_rf3> INSERT INTO test_table (id, value) VALUES (uuid(), 'Test_Value2');
cqlsh:keyspace_rf3> exit
bober@bober-VirtualBox:~/zaritskyi_lab5$
```

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

```
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker network connect zaritskyi_lab5_cassandra_net cassandra22
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker network connect zaritskyi_lab5_cassandra_net cassandra23
bober@bober-VirtualBox:~/zaritskyi_lab5$
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra21 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns  Host ID                               Rack
UN  172.18.0.3  121.2 KiB   1       ?     dd622589-1078-4aab-9c00-2e114024dd9a  rack1
UN  172.18.0.2  164.53 KiB  1       ?     49489bf3-6bad-48e8-b845-416798f2baa5  rack1

Note: Non-system keyspaces don't have the same replication settings, effective ownership information is meaningless
bober@bober-VirtualBox:~/zaritskyi_lab5$ S
```

```
bober@bober-VirtualBox:~/zaritskyi_lab5$ sudo docker exec -it cassandra21 cqlsh
Connected to Test2Cluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.2 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> USE keyspace_rf3;
cqlsh:keyspace_rf3> select id from test_table;

 id
--------------------------------------
 3ae9b657-22de-4e0e-89d5-cb30ea6d039a
 0a5fb708-8eec-4601-9bce-c63d359ce8bd
 c9514ab0-34e0-4301-b2d1-ce1f58d9d76a
 c02976d9-3c97-401c-bb79-eb8bfd4e7fbd

(4 rows)
cqlsh:keyspace_rf3> SELECT * FROM keyspace_rf3.test_table WHERE id = 0a5fb708-8eec-4601-9bce-c63d359ce8bd;

 id                                   | value
--------------------------------------+-------------
 0a5fb708-8eec-4601-9bce-c63d359ce8bd | Test_Value2

(1 rows)
cqlsh:keyspace_rf3>
```

Отже, cassandra вирішила конфлікт даних за принципом Last Write Wins (LWW), обираючи останнє записане значення(запис із найпізнішою меткою часу) як остаточне.