Task 5 - Робота з базовими функціями БД типу column family на прикладі Cassandra

Частина 1. Робота зі структурами даних у Cassandra

Для створення однієї ноди для виконання подальших завдань використовується compose1.yml та додатковий конфігураційний файл cassandra.yaml

Завдання:

Ознайомтеся з особливістю моделювання даних у Cassandra:

Створіть keyspace з найпростішої стратегією реплікації

```
user@ubuntu:~/Documents/systems/5$ sudo docker exec -it cassandra cqlsh
Connected to Test Cluster 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 lab5_keyspace
{'class': 'SimpleStrategy', 'replication_factor':    ... WITH replication =
{'class': 'SimpleStrategy', 'replication_factor': '1'};
cqlsh>
```

В цьому keyspace необхідно буде створити дві таблиці: *items* та *orders*

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.

```
cqlsh> CREATE TABLE lab5_keyspace.items (
   ...     category TEXT,
   ...     item_id UUID,
   ...     name TEXT,
   ...     price DECIMAL,
```

```
   ...     manufacturer TEXT,
   ...     additional_properties MAP<TEXT, TEXT>,
   ...     PRIMARY KEY (category, price, item_id)
   ... );
cqlsh> INSERT INTO lab5_keyspace.items (category, item_id, name, price,
manufacturer, additional_properties)
   ... VALUES ('Electronics', uuid(), 'Smartphone', 300, 'Samsung',
{'screen_size': '6.5 inch', 'RAM': '8GB', 'processor': 'Exynos 990'});
5_keyspace.items (category, item_id, name, price, manufacturer,
additional_properties)
VALUES ('Eleccqlsh>
cqlsh> INSERT INTO lab5_keyspace.items (category, item_id, name, price,
manufacturer, additional_properties)
tronics', uuid(), 'Laptop', 800, 'Dell', {'screen_   ... VALUES ('Electronics',
uuid(), 'Laptop', 800, 'Dell', {'screen_size': '15.6 inch', 'RAM': '16GB',
'processor': 'Intel i7'});
cqlsh>
```

!!! У запитах заборонено використовувати *ALLOW FILTERING* !!!

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

```
cqlsh> DESCRIBE TABLE lab5_keyspace.items;

CREATE TABLE lab5_keyspace.items (
    category text,
    price decimal,
    item_id uuid,
    manufacturer text,
    name text,
    additional_properties map<text, text>,
    PRIMARY KEY (category, price, item_id)
) WITH CLUSTERING ORDER BY (price ASC, item_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

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

 category    | price | item_id                              | additional_properties                                                                  |
manufacturer | name
-------------+-------+--------------------------------------+----------------------------------------------------------------------------------------+--------------
+------------
 Electronics |   300 | 89d9f550-58d0-4be9-888c-7de58c84081f | {'RAM': '8GB', 'processor': 'Exynos 990',
'screen_size': '6.5 inch'} |      Samsung | Smartphone
 Electronics |   800 | d11b9c18-7407-4c13-9e8e-494d6c93c52e | {'RAM': '16GB', 'processor': 'Intel i7',
'screen_size': '15.6 inch'} |         Dell |     Laptop

(2 rows)
cqlsh>
```

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

```
cqlsh> CREATE MATERIALIZED VIEW lab5_keyspace.items_by_name AS
```

```
   ...   SELECT category, name, price, item_id, manufacturer,
additional_properties
   ...   FROM lab5_keyspace.items
   ...   WHERE category IS NOT NULL AND name IS NOT NULL AND price IS NOT NULL
AND item_id IS NOT NULL
   ...   PRIMARY KEY ((category), name, price, item_id);


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


cqlsh> SELECT * FROM lab5_keyspace.items_by_name
   ... WHERE category = 'Electronics' AND name = 'Smartphone';


 category    | name       | price | item_id                              |
additional_properties                                          |
manufacturer
-------------+------------+-------+--------------------------------------
+----------------------------------------------------------------------
+--------------
 Electronics | Smartphone |   300 | d6a99169-44bb-47e5-8dab-ddb6ceaaa5c4 |
{'RAM': '8GB', 'processor': 'Exynos 990', 'screen_size': '6.5 inch'} |
Samsung


(1 rows)
cqlsh>
```

- ціна (в проміжку),

```
cqlsh> CREATE MATERIALIZED VIEW lab5_keyspace.items_by_price_range AS
   ...   SELECT category, price, item_id, name, manufacturer,
additional_properties
   ...   FROM lab5_keyspace.items
   ...   WHERE category IS NOT NULL AND price IS NOT NULL AND item_id IS NOT
NULL
   ...   PRIMARY KEY ((category), price, item_id);
 lab5_keyspace.items_by_price_range
WHERE category = 'Electronics' AND price >= 100 AND price <= 500;
Warnings :
Materialized views are experimental and are not recommended for production use.
```

```
cqlsh>
cqlsh> SELECT * FROM lab5_keyspace.items_by_price_range
   ... WHERE category = 'Electronics' AND price >= 100 AND price <= 500;

 category    | price | item_id                                |
additional_properties                                                |
manufacturer | name
-------------+-------+----------------------------------------
+--------------------------------------------------------------------
+--------------+------------
 Electronics |   300 | d6a99169-44bb-47e5-8dab-ddb6ceaaa5c4 | {'RAM': '8GB',
'processor': 'Exynos 990', 'screen_size': '6.5 inch'} |      Samsung |
Smartphone


(1 rows)
cqlsh>
```

- ціна та виробник

```
cqlsh> CREATE MATERIALIZED VIEW lab5_keyspace.items_by_price_and_manufacturer AS
   ...   SELECT category, price, manufacturer, item_id, name,
additional_properties
   ...   FROM lab5_keyspace.items
   ...   WHERE category IS NOT NULL AND price IS NOT NULL AND manufacturer IS
NOT NULL AND item_id IS NOT NULL
   ...   PRIMARY KEY ((category), manufacturer, price, item_id);
FROM lab5_keyspace.items_by_price_and_manufacturer
WHERE category = 'Electronics' AND manufacturer = 'Samsung' AND price = 300;
Warnings :
Materialized views are experimental and are not recommended for production use.

cqlsh>
cqlsh> SELECT * FROM lab5_keyspace.items_by_price_and_manufacturer
   ... WHERE category = 'Electronics' AND manufacturer = 'Samsung' AND price =
300;

 category    | manufacturer | price | item_id                                |
additional_properties                                                | name
-------------+--------------+-------+----------------------------------------
+--------------------------------------------------------------------
+------------
```

```
 Electronics |      Samsung |   300 | d6a99169-44bb-47e5-8dab-ddb6ceaaa5c4 |
{'RAM': '8GB', 'processor': 'Exynos 990', 'screen_size': '6.5 inch'} |
Smartphone

(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.

```
cqlsh> CREATE TABLE lab5_keyspace.orders (
ame TEXT,
    order_id UUID,
    item_ids LIST<UUI   ...     customer_name TEXT,
   ...     order_id UUID,
   ...     item_ids LIST<UUID>,
   ...     total_price DECIMAL,
   ...     order_date TIMESTAMP,
   ...     PRIMARY KEY (customer_name, order_date, order_id)
   ... );
tomer_name, order_id, item_ids, total_price, order_date)
VALUES ('John Doe', uuid(), [uuid(), uuid()], 50.00, '2023-12-20 10:00:00');


INSERT INTO lab5_keyspace.orders (customer_name, order_id, item_ids,
total_price, order_date)
VALUES ('John Doe', uuid(), [uuid()], 20.00, '2023-12-21 14:30:00');


INSERT INTO lab5_keyspace.orders (customer_name, order_id, item_ids,
total_price, order_date)
VALUES ('Jane Smith', uuid(), [uuid(), uuid(), uuid()], 120.00, '2023-12-19
18:45:00');cqlsh>
cqlsh> INSERT INTO lab5_keyspace.orders (customer_name, order_id, item_ids,
total_price, order_date)
   ... VALUES ('John Doe', uuid(), [uuid(), uuid()], 50.00, '2023-12-20
10:00:00');
```

```
cqlsh>

cqlsh> INSERT INTO lab5_keyspace.orders (customer_name, order_id, item_ids,
total_price, order_date)

   ... VALUES ('John Doe', uuid(), [uuid()], 20.00, '2023-12-21 14:30:00');

cqlsh>

cqlsh> INSERT INTO lab5_keyspace.orders (customer_name, order_id, item_ids,
total_price, order_date)

   ... VALUES ('Jane Smith', uuid(), [uuid(), uuid(), uuid()], 120.00, '2023-12-
19 18:45:00');

cqlsh>
```

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

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

 customer_name | order_date                      | order_id
| item_ids                                                                    |
total_price
---------------+---------------------------------
+-------------------------------------
+-------------------------------------------------------------------------------
+-------------
      John Doe | 2023-12-21 14:30:00.000000+0000 | c2422709-ab82-4dc0-a0ff-
007d5b45a271 |                                        [f0a66266-ec55-4672-89f6-
b54244f996e4] |       20.00
      John Doe | 2023-12-20 10:00:00.000000+0000 | f9803d85-99a5-4bf4-94c5-
325b79711886 | [590b6a49-f64b-4b30-9f98-8c31f908d76b, 6c0c0242-d753-4704-af2a-
6d40ed24aa83] |       50.00

(2 rows)
cqlsh>
```

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

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

 customer_name | total_spent
---------------+-------------
    Jane Smith |      120.00
      John Doe |       70.00

(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)

```
cqlsh> SELECT WRITETIME(total_price) AS write_time, total_price
   ... FROM lab5_keyspace.orders
   ... WHERE customer_name = 'John Doe' AND order_date = '2023-12-20 12:00:00';

 write_time | total_price
------------+-------------

(0 rows)
cqlsh>
```

Частина 2. Налаштування реплікації у Cassandra

Завдання

1. Сконфігурувати кластер з 3-х нод:

Для конфігурації кластера використовуються compose2.yml та відповідні конфігурації сервісу cassandra.

2. Перевірити правильність конфігурації за допомогою *nodetool status*

```
user@ubuntu:~/Documents/systems/5$ sudo 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  172.30.0.2  119.52 KiB  1       0.0%            5baad37f-1a5a-45ec-9aec-
9cb25e08fdf9  rack1
UN  172.30.0.4  84.78 KiB   1       100.0%          09b5fafd-0045-490d-8b3c-
0bc598a91c1e  rack1
UN  172.30.0.3  84.78 KiB   1       100.0%          657e670a-3620-4e1f-b4cf-
6e0b19e655c3  rack1
```

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

```
user@ubuntu:~/Documents/systems/5$ sudo docker exec -it cassandra1 cqlsh
Connected to MyCluster 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};
{'class': 'SimpleStrategy', 'replication_factor': 2};
CREATE KEYSPACE keyspace_rf3 WITH replication = {'class': 'SimpleStrategy',
'replication_factor': 3};
cqlsh> CREATE KEYSPACE keyspace_rf2 WITH replication = {'class':
'SimpleStrategy', 'replication_factor': 2};
cqlsh> CREATE KEYSPACE keyspace_rf3 WITH replication = {'class':
'SimpleStrategy', 'replication_factor': 3};
cqlsh>
```

4. В кожному з кейспейсів створити прості таблиці

```
cqlsh> USE keyspace_rf1;
 PRIMARY KEY, value TEXT);

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

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

```
user@ubuntu:~/Documents/systems/5$ sudo docker exec -it cassandra2 cqlsh
cassandra2
Connected to MyCluster at cassandra2:9042
[cqlsh 6.2.0 | Cassandra 5.0.2 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> INSERT INTO keyspace_rf1.test_table (id, value) VALUES (uuid(), 'RF1');
cqlsh> quit
user@ubuntu:~/Documents/systems/5$ sudo docker exec -it cassandra3 cqlsh
cassandra3
Connected to MyCluster at cassandra3:9042
[cqlsh 6.2.0 | Cassandra 5.0.2 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> INSERT INTO keyspace_rf3.test_table (id, value) VALUES (uuid(), 'RF3');
cqlsh> quit
user@ubuntu:~/Documents/systems/5$ sudo docker exec -it cassandra1 cqlsh
cassandra1
Connected to MyCluster at cassandra1:9042
[cqlsh 6.2.0 | Cassandra 5.0.2 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> INSERT INTO keyspace_rf2.test_table (id, value) VALUES (uuid(), 'RF2');
cqlsh> quit
```

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

```
user@ubuntu:~/Documents/systems/5$ sudo docker exec -it cassandra1 nodetool
status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns  Host ID
Rack
UN  172.30.0.2  120.75 KiB  1       ?     5baad37f-1a5a-45ec-9aec-9cb25e08fdf9
rack1
UN  172.30.0.4  85.67 KiB   1       ?     09b5fafd-0045-490d-8b3c-0bc598a91c1e
rack1
UN  172.30.0.3  85.66 KiB   1       ?     657e670a-3620-4e1f-b4cf-6e0b19e655c3
rack1

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

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

```
user@ubuntu:~/Documents/systems/5$ sudo docker exec -it cassandra1 nodetool
getendpoints keyspace_rf1 test_table fc25a95b-41cc-48ad-aef7-d97d6aec7b5c
172.30.0.4
user@ubuntu:~/Documents/systems/5$ sudo docker exec -it cassandra1 nodetool
getendpoints keyspace_rf2 test_table  335236f2-b9cd-4781-9407-7eaa72f79bfa
172.30.0.4
172.30.0.3
user@ubuntu:~/Documents/systems/5$ sudo docker exec -it cassandra1 nodetool
getendpoints keyspace_rf3 test_table 6ceeef91-a346-4864-8e39-b88c5851c89f
172.30.0.4
172.30.0.3
172.30.0.2
```

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**

```
user@ubuntu:~/Documents/systems/5$ sudo docker stop cassandra2
cassandra2
user@ubuntu:~/Documents/systems/5$ sudo docker exec -it cassandra1 cqlsh
Connected to MyCluster 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;
able;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\':
2}')})
cqlsh>
```

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

```
user@ubuntu:~/Documents/systems/5$ sudo docker network disconnect
5_cassandra_net cassandra2
user@ubuntu:~/Documents/systems/5$ sudo docker network disconnect
5_cassandra_net cassandra3
```

```
user@ubuntu:~/Documents/systems/5$ sudo docker exec -it cassandra1 nodetool
status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns  Host ID
Rack
UN  172.30.0.2  120.75 KiB  1       ?     5baad37f-1a5a-45ec-9aec-9cb25e08fdf9
rack1
DN  172.30.0.4  85.67 KiB   1       ?     09b5fafd-0045-490d-8b3c-0bc598a91c1e
rack1
DN  172.30.0.3  152.42 KiB  1       ?     657e670a-3620-4e1f-b4cf-6e0b19e655c3
rack1
```

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)

```
user@ubuntu:~/Documents/systems/5$ sudo docker exec -it cassandra2 cqlsh
Connected to MyCluster 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> INSERT INTO keyspace_rf3.test_table (id, value) VALUES (uuid(),
'Value1');
cqlsh> quit
user@ubuntu:~/Documents/systems/5$ sudo docker exec -it cassandra3 cqlsh
Connected to MyCluster 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> INSERT INTO keyspace_rf3.test_table (id, value) VALUES (uuid(),
'Value2');
cqlsh> quit
```

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

```
user@ubuntu:~/Documents/systems/5$ sudo docker network connect 5_cassandra_net
cassandra2
user@ubuntu:~/Documents/systems/5$ sudo docker network connect 5_cassandra_net
cassandra3
user@ubuntu:~/Documents/systems/5$ sudo docker exec -it cassandra1 nodetool
status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns  Host ID
Rack
UN  172.30.0.2  120.75 KiB  1       ?     5baad37f-1a5a-45ec-9aec-9cb25e08fdf9
rack1
UN  172.30.0.4  85.67 KiB   1       ?     09b5fafd-0045-490d-8b3c-0bc598a91c1e
rack1
UN  172.30.0.3  131.51 KiB  1       ?     657e670a-3620-4e1f-b4cf-6e0b19e655c3
rack1

user@ubuntu:~/Documents/systems/5$ sudo docker exec -it cassandra1 cqlsh
Connected to MyCluster 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> SELECT id FROM keyspace_rf3.test_table;

 id
--------------------------------------
 343dd3a5-2b67-4b67-ada5-8eb6a1a41acb
 6ceeef91-a346-4864-8e39-b88c5851c89f
```

```
 5142164b-3634-43a8-a673-8ad859bb3f4b

(3 rows)
cqlsh> SELECT * FROM keyspace_rf3.test_table WHERE id = 5142164b-3634-43a8-
a673-8ad859bb3f4b;

 id                                   | value
--------------------------------------+--------
 5142164b-3634-43a8-a673-8ad859bb3f4b | Value1

(1 rows)
cqlsh>
```

Отже, залишилось значення, яке було записано раніше ніж запис з Value2.