Проектування високонавантажених систем
Лабораторна робота №5
Робота з базовими функціями БД типу column family на прикладі Cassandra
Мартиненко Денис ФБ-42мп

Частина 1. Робота зі структурами даних у Cassandra

```
PS B:\lab5> docker exec -it cassandra1 nodetool status
Datacenter: dc1
===============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns (effective)  Host ID                               Rack
UN  172.19.0.2  128.86 KiB  256     100.0%            2d984b2b-f9ed-4901-9440-8865098ffa1c  rack1
```

Створіть keyspace з найпростішої стратегією реплікації

```
PS B:\lab5> 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 my_keyspace WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
cqlsh> USE my_keyspace;
cqlsh:my_keyspace>
```

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

```
cqlsh:my_keyspace> CREATE TABLE items (
                ...   category TEXT,
                ...   id UUID,
                ...   name TEXT,
                ...   price DECIMAL,
                ...   manufacturer TEXT,
                ...   properties MAP<TEXT, TEXT>,
                ...   PRIMARY KEY ((category), price, id, name, manufacturer)
                ... );
cqlsh:my_keyspace>
```

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

```
cqlsh:my_keyspace> DESCRIBE TABLE items;

CREATE TABLE my_keyspace.items (
    category text,
    price decimal,
    id uuid,
    name text,
    manufacturer text,
    properties map<text, text>,
    PRIMARY KEY (category, price, id, name, manufacturer)
) WITH CLUSTERING ORDER BY (price ASC, id ASC, name ASC, manufacturer 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:my_keyspace>
```

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

```
cqlsh:my_keyspace> SELECT * FROM items WHERE category = 'electronics' ORDER BY price ASC;

 category    | price | id                                   | name       | manufacturer | properties
-------------+-------+--------------------------------------+------------+--------------+-----------------------------------------------------------------------------------------------
 electronics |   100 | 24c6ead7-a0bc-4934-bc62-28d96fe30c01 | Headphones |        Sony |  {'battery_life': '30 hours', 'color': 'blue', 'type': 'wireless'}
 electronics |   250 | 3a0cf9d8-5842-4fb9-8e55-d6d6890fcc6e | Smartwatch |       Fitbit |          {'color': 'black', 'features': 'heart_rate_monitor, GPS'}
 electronics |   300 | 185e41ae-0e89-4f38-890c-af699b679386 |     Tablet |       Apple | {'battery_life': '10 hours', 'color': 'silver', 'storage': '64GB'}
 electronics |   500 | b1282e24-a776-405e-a223-bf781b7d9954 | Smartphone |      Samsung |          {'camera': '108MP', 'color': 'black', 'storage': '128GB'}
 electronics |  1200 | a4aa02bd-039c-435a-8dfb-19132d71109e |     Laptop |        Dell |       {'RAM': '16GB', 'color': 'silver', 'processor': 'Intel i7'}

(5 rows)
cqlsh:my_keyspace>
```

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

назва,

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

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

cqlsh:my_keyspace> SELECT * FROM my_keyspace.items_by_name WHERE category = 'electronics' AND name = 'Smartphone';

 category    | name       | id                                   | price | manufacturer | properties
-------------+------------+--------------------------------------+-------+--------------+-----------------------------------------------------------
 electronics | Smartphone | b1282e24-a776-405e-a223-bf781b7d9954 |   500 |      Samsung | {'camera': '108MP', 'color': 'black', 'storage': '128GB'}

(1 rows)
cqlsh:my_keyspace>
```

ціна (в проміжку),

```
cqlsh:my_keyspace> CREATE MATERIALIZED VIEW my_keyspace.items_by_price_range AS
               ... SELECT * FROM my_keyspace.items
               ... WHERE category IS NOT NULL AND price IS NOT NULL AND id IS NOT NULL
               ... PRIMARY KEY ((category), price, id)
               ... WITH CLUSTERING ORDER BY (price ASC, id ASC);
```

```
cqlsh:my_keyspace> SELECT * FROM my_keyspace.items_by_price_range
               ... WHERE category = 'electronics' AND price >= 500;

 category    | price | id                                   | manufacturer | name       | properties
-------------+-------+--------------------------------------+--------------+------------+-----------------------------------------------------------------------------------------------------------
 electronics |   500 | 6337da5d-15e4-4926-a869-c7d52ae58c0f |      Samsung | Smartphone |   {'camera': '108MP', 'color': 'black', 'storage': '128GB'}
 electronics |  1200 | 0594a1bf-ffd3-4b9c-a270-87b24c6c8740 |         Dell |     Laptop | {'RAM': '16GB', 'color': 'silver', 'processor': 'Intel i7'}
 electronics |  1500 | 31690391-697b-4af9-a211-d201e7a8ac48 |     Dell AMD |     Laptop |  {'RAM': '16GB', 'color': 'silver', 'processor': 'Ryzen 5'}

(3 rows)
cqlsh:my_keyspace> SELECT * FROM my_keyspace.items_by_price_range WHERE category = 'electronics' AND price >= 1000;

 category    | price | id                                   | manufacturer | name   | properties
-------------+-------+--------------------------------------+--------------+--------+-----------------------------------------------------------------------------
 electronics |  1200 | 0594a1bf-ffd3-4b9c-a270-87b24c6c8740 |         Dell | Laptop | {'RAM': '16GB', 'color': 'silver', 'processor': 'Intel i7'}
 electronics |  1500 | 31690391-697b-4af9-a211-d201e7a8ac48 |     Dell AMD | Laptop |  {'RAM': '16GB', 'color': 'silver', 'processor': 'Ryzen 5'}

(2 rows)
cqlsh:my_keyspace>
```

ціна та виробник

```
cqlsh:my_keyspace> CREATE MATERIALIZED VIEW my_keyspace.items_by_manufacturer AS
               ... SELECT * FROM my_keyspace.items
               ... WHERE category IS NOT NULL AND manufacturer IS NOT NULL AND price IS NOT NULL AND id IS NOT NULL
               ... PRIMARY KEY ((category), manufacturer, price, id);

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

cqlsh:my_keyspace> SELECT * FROM my_keyspace.items_by_manufacturer
               ... WHERE category = 'electronics' AND manufacturer = 'Samsung';

 category    | manufacturer | price | id                                   | name       | properties
-------------+--------------+-------+--------------------------------------+------------+-----------------------------------------------------------------------------------------
 electronics |      Samsung |   500 | 6337da5d-15e4-4926-a869-c7d52ae58c0f | Smartphone | {'camera': '108MP', 'color': 'black', 'storage': '128GB'}

(1 rows)
cqlsh:my_keyspace>
```

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

Напишіть запит, який показує структуру створеної таблиці

```
cqlsh:my_keyspace> CREATE TABLE orders (
               ...     customer_name TEXT,
               ...     order_id UUID,
               ...     item_ids LIST<UUID>,
               ...     total_price DECIMAL,
               ...     order_date TIMESTAMP,
               ...     PRIMARY KEY (customer_name, order_date)
               ... ) WITH CLUSTERING ORDER BY (order_date DESC);
cqlsh:my_keyspace> DESCRIBE TABLE orders;

CREATE TABLE my_keyspace.orders (
    customer_name text,
    order_date timestamp,
    order_id uuid,
    total_price decimal,
    item_ids list<uuid>,
    PRIMARY KEY (customer_name, order_date)
) WITH CLUSTERING ORDER BY (order_date DESC)
    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:my_keyspace>
```

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

```
cqlsh:my_keyspace> SELECT * FROM orders
               ... WHERE customer_name = 'John Doe';

 customer_name | order_date                      | item_ids                                                                     | order_id                             | total_price
---------------+---------------------------------+------------------------------------------------------------------------------+--------------------------------------+-------------
      John Doe | 2025-01-03 10:00:00.000000+0000 | [bbb0e909-b236-471e-827c-f13f07835d3a, 18867753-e7fa-496c-af86-61f273c30a0a] | 3392253e-4d78-4669-a31a-e1687e297382 |      300.50
      John Doe | 2025-01-02 15:30:00.000000+0000 | [336e83aa-3060-4652-9e9f-e675d21aeec4, 05ef2afe-dae2-4bd4-a316-bbd113504934] | ef2a7d94-bc54-47ea-a12b-7fa45284b73d |      150.75

(2 rows)
cqlsh:my_keyspace>
```

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

```
(2 rows)
cqlsh:my_keyspace> SELECT customer_name, SUM(total_price) FROM my_keyspace.orders GROUP BY customer_name;

 customer_name | system.sum(total_price)
---------------+-------------------------
   Alice Brown |                  450.00
      John Doe |                  451.25
   Emily Green |                  120.00
     Bob White |                  200.25

(4 rows)

Warnings :
Aggregation query used without partition key
```

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

```
cqlsh:my_keyspace> SELECT order_id, WRITETIME(total_price) AS write_time FROM orders;

 order_id                             | write_time
--------------------------------------+------------------
 78031710-5f3c-4c2c-a99f-95dbea05cfcb | 1735914387399453
 3392253e-4d78-4669-a31a-e1687e297382 | 1735914387368721
 ef2a7d94-bc54-47ea-a12b-7fa45284b73d | 1735914387383783
 dc54885d-3979-4382-83dc-bda0f75196bc | 1735914387907028
 3a765ff3-5330-4f86-9ba3-7375dbd75a65 | 1735914387414112

(5 rows)
```

Частина 2. Налаштування реплікації у Cassandra

Сконфігурувати кластер з 3-х нод, перевірити правильність конфігурації

```
PS B:\lab5> docker exec -it cassandra1 nodetool status
Datacenter: dc1
===============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns (effective)  Host ID                               Rack
UN  172.19.0.4  171.62 KiB  256     35.2%             60a64ab2-aacb-4658-97e6-b03d173e8cc1  rack1
UN  172.19.0.2  528.11 KiB  256     34.0%             2d984b2b-f9ed-4901-9440-8865098ffa1c  rack1
UN  172.19.0.3  120.33 KiB  256     30.9%             59382681-9a1b-4e72-8b79-a44a45f6b55c  rack1
```

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

```
PS B:\lab5> 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
class': 'SimpleStrategy', 'replication_factor': 1}   ... WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
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>
```

В кожному з кейспейсів створити прості таблиці

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

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

```
cqlsh:keyspace_rf3> SELECT * FROM test_table
                ... ;

 id                                   | value
--------------------------------------+----------------
 92305822-aaac-4956-aefd-c2f82c2bf999 | Record for RF3

(1 rows)
cqlsh:keyspace_rf3> USE keyspace_rf2;
cqlsh:keyspace_rf2> SELECT * FROM test_table ;

 id                                   | value
--------------------------------------+----------------
 f82d9b4b-879d-4619-ac91-f8e5f6c5c8c0 | Record for RF2

(1 rows)
cqlsh:keyspace_rf2> USE keyspace_rf1;
cqlsh:keyspace_rf1> SELECT * FROM test_table ;

 id                                   | value
--------------------------------------+----------------
 842c7cc8-97f4-4840-a25f-8246d9f32c9e | Record for RF1

(1 rows)
cqlsh:keyspace_rf1>
```

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

```
PS B:\lab5> docker exec -it cassandra1 nodetool getendpoints keyspace_rf1 test_table 842c7cc8-97f4-4840-a25f-8246d9f32c9e
172.19.0.2
PS B:\lab5> docker exec -it cassandra1 nodetool getendpoints keyspace_rf2 test_table f82d9b4b-879d-4619-ac91-f8e5f6c5c8c0
172.19.0.3
172.19.0.4
PS B:\lab5> docker exec -it cassandra1 nodetool getendpoints keyspace_rf3 test_table 92305822-aaac-4956-aefd-c2f82c2bf999
172.19.0.4
172.19.0.3
172.19.0.2
PS B:\lab5>
```

Відключити одну з нод. Для кожного з кейспейсів перевірити з якими рівнями *consistency* можемо читати та писати

```
PS B:\lab5> docker stop cassandra3
cassandra3
PS B:\lab5> docker exec -it cassandra1 nodetool status
Datacenter: dc1
===============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns  Host ID                               Rack
DN  172.19.0.4  132.91 KiB  256     ?     60a64ab2-aacb-4658-97e6-b03d173e8cc1  rack1
UN  172.19.0.2  528.86 KiB  256     ?     2d984b2b-f9ed-4901-9440-8865098ffa1c  rack1
UN  172.19.0.3  126.57 KiB  256     ?     59382681-9a1b-4e72-8b79-a44a45f6b55c  rack1

Note: Non-system keyspaces don't have the same replication settings, effective ownership information is meaningless
PS B:\lab5>
```

```
cqlsh:keyspace_rf1> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh:keyspace_rf1> SELECT * FROM test_table
                ...
cqlsh:keyspace_rf1> SELECT * FROM test_table;
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 dc1>: Unavailable('Error from server: code=1000 [Unavailable exception] message="Cannot achieve
consistency level ONE" info={\'consistency\': \'ONE\', \'required_replicas\': 1, \'alive_replicas\': 0}')})
cqlsh:keyspace_rf1>
```

```
cqlsh:keyspace_rf2> USE keyspace_rf2;
cqlsh:keyspace_rf2> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh:keyspace_rf2> SELECT * FROM test_table;

 id                                   | value
--------------------------------------+----------------
 f82d9b4b-879d-4619-ac91-f8e5f6c5c8c0 | Record for RF2

(1 rows)
cqlsh:keyspace_rf2> CONSISTENCY TWO;
Consistency level set to TWO.
cqlsh:keyspace_rf2> SELECT * FROM test_table;
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 dc1>: Unavailable('Error from server: code=1000 [Unavailable exception] message="Cannot achieve
consistency level TWO" info={\'consistency\': \'TWO\', \'required_replicas\': 2, \'alive_replicas\': 1}')})
cqlsh:keyspace_rf2>
```

```
cqlsh:keyspace_rf2> USE keyspace_rf3;
cqlsh:keyspace_rf3> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh:keyspace_rf3> SELECT * FROM test_table;

 id                                   | value
--------------------------------------+----------------
 92305822-aaac-4956-aefd-c2f82c2bf999 | Record for RF3

(1 rows)
cqlsh:keyspace_rf3> CONSISTENCY TWO;
Consistency level set to TWO.
cqlsh:keyspace_rf3> SELECT * FROM test_table;

 id                                   | value
--------------------------------------+----------------
 92305822-aaac-4956-aefd-c2f82c2bf999 | Record for RF3

(1 rows)
cqlsh:keyspace_rf3> CONSISTENCY THREE;
Consistency level set to THREE.
cqlsh:keyspace_rf3> SELECT * FROM test_table;
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 dc1>: Unavailable('Error from server: code=1000 [Unavailable exception] message="Cannot achieve
consistency level THREE" info={\'consistency\': \'THREE\', \'required_replicas\': 3, \'alive_replicas\': 2}')})
cqlsh:keyspace_rf3>
```

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

```
PS B:\lab5> docker network disconnect lab5_cassandra_network cassandra2
PS B:\lab5> docker network disconnect lab5_cassandra_network cassandra3
```

```
PS B:\lab5> docker network inspect lab5_cassandra_network
[
    {
        "Name": "lab5_cassandra_network",
        "Id": "ca25e49b9cf34609d3044f3be091606c59e445fd83b0e13b98564622a97e7910",
        "Created": "2025-01-03T12:30:26.019452179Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.19.0.0/16",
                    "Gateway": "172.19.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "56658d07d636c595a454bb7a85c29f86dbd3b7b5799c77a0b075bb271e29b2c8": {
                "Name": "cassandra1",
                "EndpointID": "f4cc084f09616a5d79487a72ca80ed3394f378d88aadf6cef9c8ce4fecdae7b2",
                "MacAddress": "02:42:ac:13:00:02",
                "IPv4Address": "172.19.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {
            "com.docker.compose.network": "cassandra_network",
            "com.docker.compose.project": "lab5",
            "com.docker.compose.version": "2.23.0"
        }
    }
]
PS B:\lab5>
```

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)

```
PS B:\lab5> 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>
cqlsh>
cqlsh> USE keyspace_rf3;
cqlsh:keyspace_rf3> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh:keyspace_rf3> INSERT INTO test_table (id, value) VALUES (11111111-1111-1111-1111-111111111111, 'Value from node 1');
cqlsh:keyspace_rf3>
```

```
PS B:\lab5> 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> USE keyspace_rf3;
cqlsh:keyspace_rf3> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh:keyspace_rf3> INSERT INTO test_table (id, value) VALUES (11111111-1111-1111-1111-111111111111, 'Value from node 2');
cqlsh:keyspace_rf3>
PS B:\lab5> 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> USE keyspace_rf3;
cqlsh:keyspace_rf3> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh:keyspace_rf3> INSERT INTO test_table (id, value) VALUES (11111111-1111-1111-1111-111111111111, 'Value from node 3');
cqlsh:keyspace_rf3>
```

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

```
PS B:\lab5> docker network connect lab5_cassandra_network cassandra3
PS B:\lab5> docker network connect lab5_cassandra_network cassandra2
PS B:\lab5>
```

```
cqlsh:keyspace_rf3> SELECT * FROM test_table WHERE id=11111111-1111-1111-1111-111111111111;

 id                                   | value
--------------------------------------+-------------------
 11111111-1111-1111-1111-111111111111 | Value from node 1

(1 rows)
cqlsh:keyspace_rf3>
```