

Міністерство освіти і науки України  
Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря 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-42мп  
Осіпчук Антон

Київ – 2024




# 1. Робота зі структурами даних у Cassandra





**cassandra1**  
cassandra:latest  
9042:9042

**cassandra2**  
cassandra:latest

**cassandra3**  
cassandra:latest

View Configurations





6:53,485 OutboundConnection.java:1153 - cassandra1/172.20.0.2:7000(/172.20.0.2:57644)->/172.20.0.4:7000-URGENT\_MESSAGES-3da7f19d successfully connected, version = 12, framing = LZ4, encryption = unencrypted  
2024-12-31 15:56:54 cassandra1 | INFO [GossipStage:1] 2024-12-31 13:56:54,825 Gossipper.java:1507 - Node /172.20.0.3:7000 is now part of the cluster  
2024-12-31 15:56:54 cassandra1 | INFO [GossipStage:1] 2024-12-31 13:56:54,827 Gossipper.java:1452 - InetAddress /172.20.0.3:7000 is now UP  
2024-12-31 15:56:55 cassandra1 | INFO [GossipStage:1] 2024-12-31 13:56:55,508 Gossipper.java:1507 - Node /172.20.0.4:7000 is now part of the cluster  
2024-12-31 15:56:55 cassandra1 | INFO [GossipStage:1] 2024-12-31 13:56:55,510 Gossipper.java:1452 - InetAddress /172.20.0.4:7000 is now UP  
2024-12-31 15:56:55 cassandra1 | WARN [GossipTasks:1] 2024-12-31 13:56:55,656 FailureDetector.java:346 - Not marking nodes down due to local pause of 5766854413 ns > 5000000000ns  
2024-12-31 15:56:55 cassandra3 | WARN [GossipTasks:1] 2024-12-31 13:56:55,822 FailureDetector.java:346 - Not marking nodes down due to local pause of 5610677991 ns > 5000000000ns  
2024-12-31 15:56:56 cassandra2 | INFO [GossipStage:1] 2024-12-31 13:56:56,826 Gossipper.java:1507 - Node /172.20.0.2:7000 is now part of the cluster  
2024-12-31 15:56:56 cassandra2 | INFO [GossipStage:1] 2024-12-31 13:56:56,828 Gossipper.java:1507 - Node /172.20.0.3:7000 is now part of the cluster  
2024-12-31 15:56:56 cassandra2 | INFO [GossipStage:1] 2024-12-31 13:56:56,829 Gossipper.java:1452 - InetAddress /172.20.0.2:7000 is now UP  
2024-12-31 15:56:56 cassandra2 | INFO [GossipStage:1] 2024-12-31 13:56:56,829 Gossipper.java:1452 - InetAddress /172.20.0.3:7000 is now UP  
2024-12-31 15:56:57 cassandra2 | WARN [GossipTasks:1] 2024-12-31 13:56:57,507 FailureDetector.java:346 - Not marking nodes down due to local pause of 7728270344 ns > 5000000000ns

```
C:\Users\User>docker exec -it cassandra1 nodetool status
Datacenter: dc1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address      Load          Tokens   Owns (effective)  Host ID                               Rack
UJ 172.20.0.3    119.19 KiB    256      ?                  dbc4139e-df1c-4122-8dd6-6e4e38cf7b77 rack1
UN 172.20.0.2    124.23 KiB    256      100.0%             301125d7-5083-44dc-bce5-8285744b9fc6 rack1
```

## Створіть keyspace з найпростішої стратегією реплікації

```
C:\Users\User>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 test WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
cqlsh> USE test;
cqlsh:test> DESCRIBE KEYSPACES;

system      system_distributed  system_traces      system_virtual_schema
system_auth  system_schema       system_views        test
```

В цьому keyspace необхідно буде створити таблиці

```
cqlsh:test> CREATE TABLE items (category TEXT, price DECIMAL, manufacturer TEXT, item_id UUID, name TEXT, attributes MAP<TEXT, TEXT>, PRIMARY KEY ((category), price, manufacturer, item_id));
cqlsh:test> DESCRIBE TABLE items;
```

```
CREATE TABLE test.items (
  category text,
  price decimal,
  manufacturer text,
  item_id uuid,
  name text,
  attributes map<text, text>,
  PRIMARY KEY (category, price, manufacturer, item_id)
) WITH CLUSTERING ORDER BY (price ASC, manufacturer 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:test> INSERT INTO items (category, item_id, name, price, manufacturer, attributes) VALUES ('phone', uuid(), 'iPhone 11', 400, 'Apple', {'storage': '128GB'});
cqlsh:test> INSERT INTO items (category, item_id, name, price, manufacturer, attributes) VALUES ('phone', uuid(), 'iPhone 12', 500, 'Apple', {'storage': '256GB'});
cqlsh:test> INSERT INTO items (category, item_id, name, price, manufacturer, attributes) VALUES ('phone', uuid(), 'iPhone 15', 900, 'Apple', {'storage': '256GB', 'connector': 'type-c'});
cqlsh:test> INSERT INTO items (category, item_id, name, price, manufacturer, attributes) VALUES ('phone', uuid(), 'S24', 850, 'Samsung', {'storage': '256GB', 'connector': 'type-c'});
cqlsh:test> INSERT INTO items (category, item_id, name, price, manufacturer, attributes) VALUES ('phone', uuid(), 'S24 Ultra', 1000, 'Samsung', {'storage': '512GB', 'connector': 'type-c'});
cqlsh:test> INSERT INTO items (category, item_id, name, price, manufacturer, attributes) VALUES ('laptop', uuid(), 'Macbook Air M1', 600, 'Apple', {'storage': '256GB', 'ram': '8GB'});
cqlsh:test> INSERT INTO items (category, item_id, name, price, manufacturer, attributes) VALUES ('laptop', uuid(), 'Macbook Air M2', 800, 'Apple', {'storage': '256GB', 'ram': '8GB'});
cqlsh:test> INSERT INTO items (category, item_id, name, price, manufacturer, attributes) VALUES ('laptop', uuid(), 'Macbook Air M3', 1000, 'Apple', {'storage': '512GB', 'ram': '8GB'});
cqlsh:test> INSERT INTO items (category, item_id, name, price, manufacturer, attributes) VALUES ('headphones', uuid(), 'Airpods 1', 150, 'Apple', {'year': '2019'});
cqlsh:test> INSERT INTO items (category, item_id, name, price, manufacturer, attributes) VALUES ('headphones', uuid(), 'WH-1000XM5', 400, 'Sony', {'year': '2023'});
cqlsh:test> INSERT INTO items (category, item_id, name, price, manufacturer, attributes) VALUES ('headphones', uuid(), 'WH-1000XM4', 200, 'Sony', {'year': '2022'});
cqlsh:test> SELECT * FROM items;
```

category	price	manufacturer	item_id	attributes	name
laptop	600	Apple	21af46de-567e-4882-b919-7946641d3b7c	{ "ram": "8GB", "storage": "256GB" }	Macbook Air M1
laptop	800	Apple	ea77961e-d8d7-4ab6-85d7-f43686a57225	{ "ram": "8GB", "storage": "256GB" }	Macbook Air M2
laptop	1000	Apple	54f8e3d3-a837-4038-980f-b2a9f7f3cf75	{ "ram": "8GB", "storage": "512GB" }	Macbook Air M3
phone	400	Apple	3b4f9134-6c9b-48b3-8631-980a8e23f800	{ "storage": "128GB" }	iPhone 11
phone	500	Apple	c8c10cfe-300d-4709-b152-eeed98a4d07f	{ "storage": "256GB" }	iPhone 12
phone	850	Samsung	fe27cb21-0507-47b1-a5a0-cf83fc6e9560	{ "connector": "type-c", "storage": "256GB" }	S24
phone	900	Apple	6809c843-0510-4186-ba89-9c0b77542d8d	{ "connector": "type-c", "storage": "256GB" }	iPhone 15
phone	1000	Samsung	7baab5e-5cab-4f4a-a7da-4cb5fc7824a2	{ "connector": "type-c", "storage": "512GB" }	S24 Ultra
headphones	150	Apple	f69a280e-34a7-41ec-8225-48735635270a	{ "year": "2019" }	Airpods 1
headphones	200	Sony	d7eb119e-050d-47ba-b6ca-6f75b69834ae	{ "year": "2022" }	WH-1000XM4
headphones	400	Sony	d3259e0c-b5b5-418a-9a56-d67bb0f40385	{ "year": "2023" }	WH-1000XM5

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

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

category	price	manufacturer	item_id	attributes	name
phone	400	Apple	3b4f9134-6c9b-48b3-8631-980a8e23f800	{'storage': '128GB'}	iPhone 11
phone	500	Apple	c8c10cff-300d-47b9-b152-eeed9ae4dd7f	{'storage': '256GB'}	iPhone 12
phone	850	Samsung	fe27cb22-0507-47b1-a5a8-cf83fc669560	{'connector': 'type-c', 'storage': '256GB'}	S24
phone	900	Apple	68e9c843-0510-4186-ba89-9c0b77542d8d	{'connector': 'type-c', 'storage': '256GB'}	iPhone 15
phone	1000	Samsung	7baabb5e-5cab-4f4a-a7da-4cb5fc7824a2	{'connector': 'type-c', 'storage': '512GB'}	S24 Ultra

(5 rows)

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

```
cqlsh:test> CREATE MATERIALIZED VIEW items_by_name2 AS SELECT category, name, item_id, price, manufacturer FROM items WHERE category IS NOT NULL AND name IS NOT NULL AND item_id IS NOT NULL AND price IS NOT NULL AND manufacturer IS NOT NULL PRIMARY KEY (category, name, price, manufacturer, item_id);
```

```
cqlsh:test> SELECT * FROM items_by_name WHERE category = 'phone' AND name = 'S24';
```

category	name	price	manufacturer	item_id
phone	S24	850	Samsung	fe27cb22-0507-47b1-a5a8-cf83fc669560

(1 rows)

Створіть таблицю orders

```
cqlsh:test> CREATE TABLE orders (customer TEXT, order_id UUID, item_ids LIST(UUID), price DECIMAL, order_date TIMESTAMP, PRIMARY KEY (customer, order_date, order_id));
cqlsh:test> DESCRIBE TABLE orders;
```

```
CREATE TABLE test.orders (
  customer text,
  order_date timestamp,
  order_id uuid,
  price decimal,
  item_ids list(uuid),
  PRIMARY KEY (customer, 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 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:test> INSERT INTO orders (customer, order_id, item_ids, price, order_date) VALUES ('Anton Osipchuk', uuid(), [uuid(), uuid()], 1000, '2024-09-01 12:23:34');
cqlsh:test> INSERT INTO orders (customer, order_id, item_ids, price, order_date) VALUES ('Vitali Klitschko', uuid(), [uuid(), uuid()], 5000, '2024-09-10 12:23:34');
cqlsh:test> INSERT INTO orders (customer, order_id, item_ids, price, order_date) VALUES ('Valerii Zaluzhnyi', uuid(), [uuid(), uuid()], 500, '2024-09-12 12:23:34');
```

```
cqlsh:test> select * from orders;
```

customer	order_date	order_id	item_ids	price
Anton Osipchuk	2024-09-01 12:23:34.000000+0000	b0c60fdc-de71-461e-ab63-5442f0ec6ac9	[057cce2c-db79-4f54-b04a-2b7434c5b017, 6a6b9ac2-b7a0-4de6-903d-f4bbfcc77e1a]	1000
Anton Osipchuk	2024-09-01 13:23:34.000000+0000	8a01825e-c135-41e8-ba6a-c7bdc20880e1	[39557769-0f67-48d6-8fea-a93fe6659072, 002ea370-f493-4d54-9086-e500c2604f83]	100
Valerii Zaluzhnyi	2024-09-12 12:23:34.000000+0000	bcdd09b4-b570-4389-a216-a35390cb173c	[3b60575e-b6d8-4720-bba7-e33c3711a9dd, dd048048-d2d7-4b25-b456-4352585da6dc]	500
Vitali Klitschko	2024-09-10 12:23:34.000000+0000	c649a100-b450-4da2-a23b-c480f6e6a04	[b96aecf7-bd2c-43ce-8d69-5fcf18bd853f, 640f6db8-7a62-42bc-99ef-6d1d99316d8c]	5000

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

```
cqlsh:test> SELECT * FROM orders WHERE customer = 'Anton Osipchuk' ORDER BY order_date ASC;
```

customer	order_date	order_id	item_ids	price
Anton Osipchuk	2024-09-01 12:23:34.000000+0000	b0c60fdc-de71-461e-ab63-5442f0ec6ac9	[057cce2c-db79-4f54-b04a-2b7434c5b017, 6a6b9ac2-b7a0-4de6-903d-f4bbfcc77e1a]	1000
Anton Osipchuk	2024-09-01 13:23:34.000000+0000	8a01825e-c135-41e8-ba6a-c7bdc20886e1	[39557769-0f67-48d6-8fea-a93fe6659872, 002ea370-f493-4d54-9086-e500c2604f83]	100

(2 rows)

```
cqlsh:test> SELECT * FROM orders WHERE customer = 'Anton Osipchuk' ORDER BY order_date DESC;
```

customer	order_date	order_id	item_ids	price
Anton Osipchuk	2024-09-01 13:23:34.000000+0000	8a01825e-c135-41e8-ba6a-c7bdc20886e1	[39557769-0f67-48d6-8fea-a93fe6659872, 002ea370-f493-4d54-9086-e500c2604f83]	100
Anton Osipchuk	2024-09-01 12:23:34.000000+0000	b0c60fdc-de71-461e-ab63-5442f0ec6ac9	[057cce2c-db79-4f54-b04a-2b7434c5b017, 6a6b9ac2-b7a0-4de6-903d-f4bbfcc77e1a]	1000

(2 rows)

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

```
cqlsh:test> SELECT customer, SUM(price) AS total FROM orders GROUP BY customer;
```

customer	total
Anton Osipchuk	1100
Valerii Zaluzhnyi	500
Vitali Klitschko	5000

(3 rows)

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


```
cqlsh:test> SELECT order_id, WRITETIME(price) AS write_time FROM orders;
```

order_id	write_time
b0c60fdc-de71-461e-ab63-5442f0ec6ac9	1735659998998698
8a01825e-c135-41e8-ba6a-c7bdc20886e1	1735660305457270
bcdd09b4-b570-4389-a216-a35390cb173c	1735660112914237
c649a100-b450-4da2-a23b-c4840f6e6a04	1735660062169076




(4 rows)


## 2. Налаштування реплікації у Cassandra


### Сконфігурувати кластер з 3-х нод


**lab5**  
C:\Users\User\Desktop\pvs\_git\lab5

View Configurations



**cassandra1**  
cassandra:latest  
9042:9042

**cassandra2**  
cassandra:latest

**cassandra3**  
cassandra:latest

```
6:53,485 OutboundConnection.java:1153 - cassandra1/172.20.0.2:7000(/172.20.0.2:57644)->/172.20.0.4:7000-URGENT_MESSAGES-3da7f19d successfully connected, version = 12, framing = LZ4, encryption = unencrypted
2024-12-31 15:56:54 cassandra1 | INFO [GossipStage:1] 2024-12-31 13:56:54,825 Gossipper.java:1507 - Node /172.20.0.3:7000 is now part of the cluster
2024-12-31 15:56:54 cassandra1 | INFO [GossipStage:1] 2024-12-31 13:56:54,827 Gossipper.java:1452 - InetAddress /172.20.0.3:7000 is now UP
2024-12-31 15:56:55 cassandra1 | INFO [GossipStage:1] 2024-12-31 13:56:55,508 Gossipper.java:1507 - Node /172.20.0.4:7000 is now part of the cluster
2024-12-31 15:56:55 cassandra1 | INFO [GossipStage:1] 2024-12-31 13:56:55,510 Gossipper.java:1452 - InetAddress /172.20.0.4:7000 is now UP
2024-12-31 15:56:55 cassandra1 | WARN [GossipTasks:1] 2024-12-31 13:56:55,656 FailureDetector.java:346 - Not marking nodes down due to local pause of 5766854413ns > 5000000000ns
2024-12-31 15:56:55 cassandra3 | WARN [GossipTasks:1] 2024-12-31 13:56:55,822 FailureDetector.java:346 - Not marking nodes down due to local pause of 5610677991ns > 5000000000ns
2024-12-31 15:56:56 cassandra2 | INFO [GossipStage:1] 2024-12-31 13:56:56,826 Gossipper.java:1507 - Node /172.20.0.2:7000 is now part of the cluster
2024-12-31 15:56:56 cassandra2 | INFO [GossipStage:1] 2024-12-31 13:56:56,828 Gossipper.java:1507 - Node /172.20.0.3:7000 is now part of the cluster
2024-12-31 15:56:56 cassandra2 | INFO [GossipStage:1] 2024-12-31 13:56:56,829 Gossipper.java:1452 - InetAddress /172.20.0.2:7000 is now UP
2024-12-31 15:56:56 cassandra2 | INFO [GossipStage:1] 2024-12-31 13:56:56,829 Gossipper.java:1452 - InetAddress /172.20.0.3:7000 is now UP
2024-12-31 15:56:57 cassandra2 | WARN [GossipTasks:1] 2024-12-31 13:56:57,507 FailureDetector.java:346 - Not marking nodes down due to local pause of 7728270344ns > 5000000000ns
```

### Перевірити правильність конфігурації за допомогою nodetool status

```
Datcenter: dc1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address      Load          Tokens   Owns (effective)  Host ID                               Rack
UN 172.23.0.3    108.82 KiB    256      60.0%             22a01ceb-06c0-4527-87f9-ebd17e713941 rack1
UN 172.23.0.4    169.93 KiB    256      74.0%             410ab6b1-ab18-4f08-a210-18bc6fd29842 rack1
UN 172.23.0.2    113.81 KiB    256      66.0%             1cff18cc-c574-4061-82d6-2d73dcc94242 rack1
```

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

```
[cqlsh> CREATE KEYSPACE keyspace1 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
cqlsh>
cqlsh> CREATE KEYSPACE keyspace2 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 2};
cqlsh>
cqlsh> CREATE KEYSPACE keyspace3 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 3};
```

В кожному з кейспейсів створити прості таблиці

```
[cqlsh:keyspace1> CREATE TABLE test_table (id UUID PRIMARY KEY, value text);
cqlsh:keyspace1> USE keyspace2;
cqlsh:keyspace2> CREATE TABLE test_table (id UUID PRIMARY KEY, value text);
cqlsh:keyspace2> USE keyspace3;
cqlsh:keyspace3> CREATE TABLE test_table (id UUID PRIMARY KEY, value text);
```

Спробуйте писати і читати в ці таблиці підключаючись на різні ноди.

```
[cqlsh:keyspace1> INSERT INTO test_table (id, value) VALUES (uuid(), 'Data in keyspace1');
cqlsh:keyspace1> USE keyspace2;
cqlsh:keyspace2> INSERT INTO test_table (id, value) VALUES (uuid(), 'Data in keyspace2');
cqlsh:keyspace2> USE keyspace3;
cqlsh:keyspace3> INSERT INTO test_table (id, value) VALUES (uuid(), 'Data in keyspace3');
```

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

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

```
[cqlsh:keyspace3> SELECT * FROM test_table;

id | value
-----+-----
37550b59-d1af-4362-b0e7-d387c6ad4e23 | Data in keyspace3
(1 rows)
```

```
172.23.0.3
172.23.0.2
172.23.0.4
```

```
[cqlsh:keyspace1> SELECT id, value FROM test_table;

id | value
-----+-----
da08a0c6-3cfd-4faa-8ea6-3c028f5b7bdd | Data in keyspace1
```

```
172.23.0.2
172.23.0.4
172.23.0.3
```

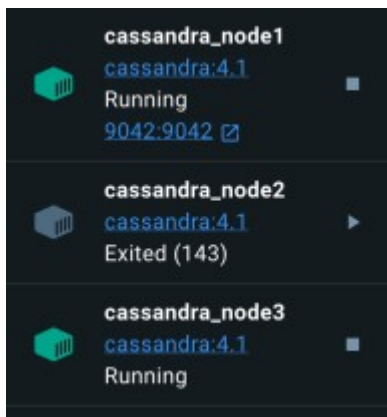
```
[cqlsh:keyspace2> SELECT id, value FROM test_table;

id | value
-----+-----
93b9888f-f339-4da1-8e5c-34f9a755b782 | Data in keyspace2
```

```
172.23.0.2
172.23.0.4
172.23.0.3
```



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



```
Datacenter: dc1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address      Load          Tokens     Owns    Host ID                               Rack
DN 172.23.0.3    230.21 KiB    256        ?       22a01ceb-06c0-4527-87f9-ebd17e713941 rack1
UN 172.23.0.4    276.93 KiB    256        ?       410ab6b1-ab18-4f08-a210-18bc6fd29842 rack1
UN 172.23.0.2    177.44 KiB    256        ?       1cff18cc-c574-4061-82d6-2d73dcc94242 rack1
```

```
[cqlsh:keyspace1> INSERT INTO test_table (id, value) VALUES (uuid(), 'Test for RF=1');
[cqlsh:keyspace1> 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:keyspace2> INSERT INTO test_table (id, value) VALUES (uuid(), 'Test for RF=2 consistency ONE');
[cqlsh:keyspace2> SELECT * FROM test_table;

id | value
-----+-----
80729315-8d88-40e4-ba98-6bf4f58893ef | Test for RF=2 consistency ONE
93b9888f-f339-4da1-8e5c-34f9a755b782 | Data in keyspace2

(2 rows)
```

```
[cqlsh:keyspace3> CONSISTENCY ONE;
Consistency level set to ONE.
[cqlsh:keyspace3> INSERT INTO test_table (id, value) VALUES (uuid(), 'Test for RF=3 consistency ONE');
[cqlsh:keyspace3> CONSISTENCY TWO;
Consistency level set to TWO.
[cqlsh:keyspace3> INSERT INTO test_table (id, value) VALUES (uuid(), 'Test for RF=3 consistency TWO');
[cqlsh:keyspace3> CONSISTENCY THREE;
Consistency level set to THREE.
[cqlsh:keyspace3> INSERT INTO test_table (id, value) VALUES (uuid(), 'Test for RF=3 consistency THREE')
;
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:keyspace3> 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}\')})
```



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

Node1

```
cqlsh:keyspace3> INSERT INTO test_table (id, value) VALUES (11111111-1111-1111-1111-111111111111, 'A');
```

Node2

```
cqlsh:keyspace3> INSERT INTO test_table (id, value) VALUES (11111111-1111-1111-1111-111111111111, 'B');
```

Node3

```
cqlsh:keyspace3> INSERT INTO test_table (id, value) VALUES (11111111-1111-1111-1111-111111111111, 'C');
```

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

```
cqlsh:keyspace3> SELECT * FROM test_table WHERE id = 11111111-1111-1111-1111-111111111111;
```

id	value
11111111-1111-1111-1111-111111111111	C

Після відновлення зв'язку ноди автоматично синхронізувались, конфлікт був вирішений за допомогою принципу «Last Write Wins»