Частина 1. Робота зі структурами даних у Cassandra

Завдання:

Ознайомтеся з особливістю моделювання даних у Cassandra:

- https://cassandra.apache.org/doc/latest/cassandra/data_modeling/index.html
- https://www.instaclustr.com/blog/cassandra-data-modeling/

Створіть keyspace з найпростішої стратегією реплікації

https://docs.datastax.com/en/cql/cassandra-5.0/develop/keyspace-create.html

```
┌──(ivan㉿kali)-[~/pvns/5]
└─$ sudo docker network create lab5
[sudo] password for ivan:
75600e629d83452a79022365529d5da852ccac3f01a5d5a3c733e33807450cea

┌──(ivan㉿kali)-[~/pvns/5]
└─$ sudo docker run --name cassandra --network lab5 -d cassandra
Unable to find image 'cassandra:latest' locally
latest: Pulling from library/cassandra
6414378b6477: Pull complete
17da8ec43a12: Pull complete
d12988e90d61: Pull complete
f4d133ca2b7f: Pull complete
143733ae87a4: Pull complete
6717475e96f8: Pull complete
f189a9d82ae7: Pull complete
09bb1bb42e9a: Pull complete
10154685d2bd: Pull complete
2bfd35935537: Pull complete
Digest: sha256:5d4795c41491654e2bda432179e020c7c2cd702bbb22b7d1314747658efd71b4
Status: Downloaded newer image for cassandra:latest
0b8c8b800e98c4231031ce65991948ee6c4defa6cc3019041b16f860ff18f888

┌──(ivan㉿kali)-[~/pvns/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 IF NOT EXISTS cassandra_keyspace WITH REPLICATION = {
   ... 'class': 'SimpleStrategy',
   ... 'replication_factor': 1};
cqlsh>
```

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

```
cqlsh:cassandra_keyspace> CREATE TABLE IF NOT EXISTS cassandra_keyspace.items_ (id uuid, name text, category text, cost decimal, country text, characteristics map<text, text>, PRIMARY KEY (
category, cost, id)) WITH CLUSTERING ORDER BY (cost ASC, id ASC);
cqlsh:cassandra_keyspace>
```

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

```
cqlsh:cassandra_keyspace> INSERT INTO cassandra_keyspace.items_ (id, name, category, cost, country, characteristics) VALUES (uuid(), 'loving_love', 'rose', 100, 'Sweden', {'color': 'red', '
length': '50cm'});
cqlsh:cassandra_keyspace>  INSERT INTO cassandra_keyspace.items_ (id, name, category, cost, country, characteristics) VALUES (uuid(), 'whispering_love', 'rose', 95, 'Sweden', {'color': 'whi
te', 'length': '40cm'});
cqlsh:cassandra_keyspace>  INSERT INTO cassandra_keyspace.items_ (id, name, category, cost, country, characteristics) VALUES (uuid(), 'hugging_face', 'tulp', 35, 'Poland', {'color': 'purple
', 'length': '15cm'});
cqlsh:cassandra_keyspace> INSERT INTO cassandra_keyspace.items_ (id, name, category, cost, country, characteristics) VALUES (uuid(), 'flaming_heart', 'tulp', 35, 'Poland', {'color': 'red',
'length': '15cm'});
cqlsh:cassandra_keyspace> INSERT INTO cassandra_keyspace.items_ (id, name, category, cost, country, characteristics) VALUES (uuid(), 'open_hands', 'tulp', 35, 'Poland', {'color': 'yellow',
'length': '15cm'});
cqlsh:cassandra_keyspace>  INSERT INTO cassandra_keyspace.items_ (id, name, category, cost, country, characteristics) VALUES (uuid(), 'white_flame', 'daisy', 40, 'Ukraine', {'color': 'white
', 'length': '20cm'});
cqlsh:cassandra_keyspace> INSERT INTO cassandra_keyspace.items_ (id, name, category, cost, country, characteristics) VALUES (uuid(), 'yellow_flame', 'daisy', 45, 'Ukraine', {'color': 'yello
w', 'length': '20cm'});
cqlsh:cassandra_keyspace>
```

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)

https://docs.datastax.com/en/cql/cassandra-5.0/develop/collections-create.html#map-collections

https://cassandra.apache.org/doc/4.1/cassandra/cql/types.html#maps

https://cassandra.apache.org/doc/4.1/cassandra/cql/indexes.html#indexes-on-map-keys

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

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

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

```
CREATE TABLE cassandra_keyspace.items_ (
    category text,
    cost decimal,
    id uuid,
    country text,
    name text,
    characteristics map<text, text>,
    PRIMARY KEY (category, cost, id)
) WITH CLUSTERING ORDER BY (cost ASC, id ASC)
    AND additional_write_policy = '99p'
    AND allow_auto_snapshot = true
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', '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:cassandra_keyspace>
```

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

```
cqlsh:cassandra_keyspace> SELECT * FROM items_ WHERE category='daisy' ORDER BY cost;

 category | cost | id                                   | characteristics                          | country | name
----------+------+--------------------------------------+------------------------------------------+---------+--------------
    daisy |   40 | 0ade3004-5860-4c93-acec-037f0b9bffd7 |  {'color': 'white', 'length': '20cm'} | Ukraine |  white_flame
    daisy |   45 | b4cf8e08-583f-4588-a0e1-ee56f2eb3e65 | {'color': 'yellow', 'length': '20cm'} | Ukraine | yellow_flame

(2 rows)
cqlsh:cassandra_keyspace>
```

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

```
cassandra.yaml
/home/ivan/pvns/5

# If dynamic data masking is disabled it won't be allowed to create new column masks, although it will still be possible
# to drop any previously existing masks. Also, any existing mask will be ignored at query time, so all users will see
# the clear values of the masked columns.
# Defaults to false to disable dynamic data masking.
# dynamic_data_masking_enabled: false

########################
# EXPERIMENTAL FEATURES #
########################

# Enables materialized view creation on this node.
# Materialized views are considered experimental and are not recommended for production use.
materialized_views_enabled: true

# Enables SASI index creation on this node.
# SASI indexes are considered experimental and are not recommended for production use.
sasi_indexes_enabled: false

# Enables creation of transiently replicated keyspaces on this node.
```

```
┌──(ivan㉿kali)-[~/pvns/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> use cassandra_keyspace;
cqlsh:cassandra_keyspace> CREATE MATERIALIZED VIEW items_mat_view AS SELECT * FROM items_ WHERE name IS NOT NULL AND category is NOT NULL AND cost IS NOT NULL AND country IS NOT NULL AND id
 IS NOT NULL PRIMARY KEY (category, country, id, cost);

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

cqlsh:cassandra_keyspace>
```

- назва,

```
cqlsh:cassandra_keyspace> SELECT * FROM items_mat_view WHERE category='rose';

 category | country | id                                   | cost | characteristics                                | name
----------+---------+--------------------------------------+------+------------------------------------------------+---------------
     rose |  Sweden | 24d0330a-b20c-427b-9f06-79009c0a5ebe |  100 |   {'color': 'red', 'length': '50cm'} |     loving_love
     rose |  Sweden | e230b205-33f9-4c62-89a3-cd572babde7b |   95 | {'color': 'white', 'length': '40cm'} | whispering_love

(2 rows)
cqlsh:cassandra_keyspace>
```

- ціна (в проміжку),

```
cqlsh:cassandra_keyspace> SELECT * FROM items_ WHERE category = 'rose' AND cost ⩾ 50 AND cost < 100;

 category | cost | id                                   | characteristics                                | country | name
----------+------+--------------------------------------+------------------------------------------------+---------+---------------
     rose |   95 | e230b205-33f9-4c62-89a3-cd572babde7b | {'color': 'white', 'length': '40cm'} |  Sweden | whispering_love

(1 rows)
cqlsh:cassandra_keyspace>
```

- ціна та виробник

```
cqlsh:cassandra_keyspace> CREATE MATERIALIZED VIEW items_mat_view_2 AS SELECT * FROM items_ WHERE name IS NOT NULL AND category IS NOT NULL AND cost IS NOT NULL AND country IS NOT NULL AND
id IS NOT NULL PRIMARY KEY (category, country, cost, id);

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

```
cqlsh:cassandra_keyspace> SELECT * FROM items_mat_view_2 WHERE cost > 97 AND country = 'Sweden' AND category = 'rose';

 category | country | cost | id                                   | characteristics                                | name
----------+---------+------+--------------------------------------+------------------------------------------------+---------------
     rose |  Sweden |  100 | 24d0330a-b20c-427b-9f06-79009c0a5ebe | {'color': 'red', 'length': '50cm'} | loving_love

(1 rows)
cqlsh:cassandra_keyspace>
```

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

```
cqlsh:cassandra_keyspace> CREATE TABLE cassandra_keyspace.orders (id uuid, name text, items_in_list list<uuid>, total_items_cost decimal, date date, PRIMARY KEY(name, date, id)) WITH CLUSTE
RING ORDER BY (date ASC, id ASC);
cqlsh:cassandra_keyspace>
```

```
cqlsh:cassandra_keyspace> SELECT * FROM items_;

 category | cost | id                                   | characteristics                                                  | country | name
----------+------+--------------------------------------+------------------------------------------------------------------+---------+---------------
     tulp |   35 | 6da3df82-94eb-4809-90a1-3fa282de492f | {'color': 'purple', 'length': '15cm'} |  Poland |    hugging_face
     tulp |   35 | 7418725b-34e1-401d-a0c4-9f3f366c6dd8 | {'color': 'yellow', 'length': '15cm'} |  Poland |      open_hands
     tulp |   35 | f86a0b6d-6ce9-4710-93f7-d541b35c161f |    {'color': 'red', 'length': '15cm'} |  Poland |   flaming_heart
     rose |   95 | e230b205-33f9-4c62-89a3-cd572babde7b |  {'color': 'white', 'length': '40cm'} |  Sweden | whispering_love
     rose |  100 | 24d0330a-b20c-427b-9f06-79009c0a5ebe |    {'color': 'red', 'length': '50cm'} |  Sweden |     loving_love
    daisy |   40 | 0ade3004-5860-4c93-acec-037f0b9bffd7 |  {'color': 'white', 'length': '20cm'} | Ukraine |     white_flame
    daisy |   45 | b4cf8e08-583f-4588-a0e1-ee56f2eb3e65 | {'color': 'yellow', 'length': '20cm'} | Ukraine |    yellow_flame

(7 rows)
cqlsh:cassandra_keyspace> INSERT INTO orders (id, name, items_in_list, total_items_cost, date) VALUES
                      ... (uuid(), 'Ivan', [6da3df82-94eb-4809-90a1-3fa282de492f, f86a0b6d-6ce9-4710-93f7-d541b35c161f], 70, '2024-12-20');
cqlsh:cassandra_keyspace> INSERT INTO orders (id, name, items_in_list, total_items_cost, date) VALUES
                      ... (uuid(), 'Mykola', [6da3df82-94eb-4809-90a1-3fa282de492f, e230b205-33f9-4c62-89a3-cd572babde7b], 130, '2024-12-05');
cqlsh:cassandra_keyspace> INSERT INTO orders (id, name, items_in_list, total_items_cost, date) VALUES
                      ... (uuid(), 'Zhenya', [0ade3004-5860-4c93-acec-037f0b9bffd7], 40, '2024-12-01
                      ... ');
InvalidRequest: Error from server: code=2200 [Invalid query] message="Unable to coerce '2024-12-01
' to a formatted date (long)"
cqlsh:cassandra_keyspace> INSERT INTO orders (id, name, items_in_list, total_items_cost, date) VALUES (uuid(), 'Zhenya', [0ade3004-5860-4c93-acec-037f0b9bffd7], 40, '2024-12-01');
cqlsh:cassandra_keyspace> INSERT INTO orders (id, name, items_in_list, total_items_cost, date) VALUES
                      ... (uuid(), 'Viktoria', [b4cf8e08-583f-4588-a0e1-ee56f2eb3e65, 24d0330a-b20c-427b-9f06-79009c0a5ebe], 145, '2024-11-23');
cqlsh:cassandra_keyspace>
```

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.

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

```
CREATE TABLE cassandra_keyspace.orders (
    name text,
    date date,
    id uuid,
    total_items_cost decimal,
    items_in_list list<uuid>,
    PRIMARY KEY (name, date, id)
) WITH CLUSTERING ORDER BY (date ASC, id ASC)
    AND additional_write_policy = '99p'
    AND allow_auto_snapshot = true
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', '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:cassandra_keyspace>
```

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

Спочатку додам ще один запис, бо в мене унікальні клієнти були:

```
cqlsh:cassandra_keyspace> INSERT INTO orders (id, name, items_in_list, total_items_cost, date) VALUES (uuid(), 'Zhenya', [7418725b-34e1-401d-a0c4-9f3f366c6dd8], 40, '2024-12-02');
cqlsh:cassandra_keyspace>
```

```
cqlsh:cassandra_keyspace> SELECT * FROM orders WHERE name='Zhenya' ORDER BY date;

 name   | date       | id                                   | items_in_list                                | total_items_cost
--------+------------+--------------------------------------+----------------------------------------------+------------------
 Zhenya | 2024-12-01 | 4416561f-c98b-4b5d-970d-eb299d096c30 | [0ade3004-5860-4c93-acec-037f0b9bffd7] |               40
 Zhenya | 2024-12-02 | b47f18ed-2b26-4f6e-9355-3c8359bddc87 | [7418725b-34e1-401d-a0c4-9f3f366c6dd8] |               40

(2 rows)
cqlsh:cassandra_keyspace>
```

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

```
cqlsh:cassandra_keyspace> SELECT name, SUM(total_item_cost) AS total_item_cost_all_time FROM orders GROUP BY name;
InvalidRequest: Error from server: code=2200 [Invalid query] message="Undefined column name total_item_cost in table
cqlsh:cassandra_keyspace> SELECT name, SUM(total_items_cost) AS total_item_cost_all_time FROM orders GROUP BY name;

  name    | total_item_cost_all_time
----------+--------------------------
   Zhenya |                       80
 Viktoria |                      145
     Ivan |                       70
   Mykola |                      130

(4 rows)
```

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

```
cqlsh:cassandra_keyspace> SELECT id, WRITETIME(total_items_cost) AS time FROM orders;

 id                                   | time
--------------------------------------+------------------
 4416561f-c98b-4b5d-970d-eb299d096c30 | 1734706650977694
 b47f18ed-2b26-4f6e-9355-3c8359bddc87 | 1734706795421833
 4b57c297-1d9f-4c59-af88-b57d13961304 | 1734706695445797
 86eb4709-0114-42f4-87b1-a6caf102a094 | 1734706549847661
 6665da3b-b2e9-48e9-8264-d037e75ced2b | 1734706596415412

(5 rows)
cqlsh:cassandra_keyspace>
```

Частина 2. Налаштування реплікації у Cassandra

Завдання

1. Сконфігурувати кластер з 3-х нод:

- https://hub.docker.com/_/cassandra
- https://medium.com/@kayvan.sol2/deploying-apache-cassandra-cluster-3-nodes-with-docker-compose-3634ef8345e8

```
┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker run -d --name cassandra_1 --network lab5 -e CASSANDRA_SEEDS=cassandra_1,cassandra_2,cassandra_3 cassandra
[sudo] password for ivan:
1ce6d59bfac704a0c68c3e9956a57afc0a80aa2a280a455bdcc2dea7db483b27
^[[A

┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker run -d --name cassandra_2 --network lab5 -e CASSANDRA_SEEDS=cassandra_1,cassandra_2,cassandra_3 cassandra
73621877bc93eed1c37a572c57c7bda525bf985ef1a8c199a14bf05dde1854b2

┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS          PORTS                                                NAMES
73621877bc93   cassandra   "docker-entrypoint.s…"   4 seconds ago    Up 3 seconds    7000-7001/tcp, 7199/tcp, 9042/tcp, 9160/tcp          cassandra_2
1ce6d59bfac7   cassandra   "docker-entrypoint.s…"   16 seconds ago   Up 14 seconds   7000-7001/tcp, 7199/tcp, 9042/tcp, 9160/tcp          cassandra_1

┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker run -d --name cassandra_3 --network lab5 -e CASSANDRA_SEEDS=cassandra_1,cassandra_2,cassandra_3 cassandra
b2f98bd03c2b4e0022eb4f753b09cb04be6e31b39f4cb4e25e6f1e2a30958cec

┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS          PORTS                                                NAMES
b2f98bd03c2b   cassandra   "docker-entrypoint.s…"   22 seconds ago   Up 20 seconds   7000-7001/tcp, 7199/tcp, 9042/tcp, 9160/tcp          cassandra_3
73621877bc93   cassandra   "docker-entrypoint.s…"   42 seconds ago   Up 41 seconds   7000-7001/tcp, 7199/tcp, 9042/tcp, 9160/tcp          cassandra_2
1ce6d59bfac7   cassandra   "docker-entrypoint.s…"   54 seconds ago   Up 52 seconds   7000-7001/tcp, 7199/tcp, 9042/tcp, 9160/tcp          cassandra_1

┌──(ivan㉿kali)-[~/pvns/1]
└─$
```

2. Перевірити правильність конфігурації за допомогою

nodetool status

https://docs.datastax.com/en/dse/6.9/managing/tools/nodetool/status.html

```
┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_1 nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens  Owns (effective)  Host ID                               Rack
UN  172.21.0.4  80.11 KiB  16      59.5%             eca0bbd5-af55-46be-8752-6d8aa66aa459  rack1
UN  172.21.0.3  80.11 KiB  16      72.2%             eb5b2d79-eb72-4e5d-946a-65ff65b95d2e  rack1
UN  172.21.0.2  80.1 KiB   16      68.3%             3b88ba8a-6cfa-4494-9bfd-60041bb1d5c3  rack1
```

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

https://www.tutorialspoint.com/cassandra/cassandra_create_keyspace.htm

https://docs.datastax.com/en/cql/cassandra-5.0/develop/keyspace-create.html

```
┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_1 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 rep_factor_1 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
cqlsh> CREATE KEYSPACE rep_factor_2 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 2};
cqlsh> CREATE KEYSPACE rep_factor_3 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 3};
cqlsh>
```

4. В кожному з кейспейсів створити прості таблиці

https://www.tutorialspoint.com/cassandra/cassandra_create_table.htm

```
cqlsh> CREATE TABLE IF NOT EXISTS rep_factor_1.simple_table(in_key text, PRIMARY KEY(in_key));
cqlsh> CREATE TABLE IF NOT EXISTS rep_factor_2.simple_table(in_key text, PRIMARY KEY(in_key));
cqlsh> CREATE TABLE IF NOT EXISTS rep_factor_3.simple_table(in_key text, PRIMARY KEY(in_key));
cqlsh>
```

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

Пишу та зчитую з ноди cassandra_1 в/на rep_factor_1:

```
cqlsh> INSERT INTO rep_factor_1.simple_table(in_key) VALUES('In Key');
cqlsh> SELECT * FROM rep_factor_1.simple_table;

 in_key
--------
 In Key

(1 rows)
cqlsh>
```

Пишу та зчитую з ноди cassandra_1 в/на rep_factor_2:

```
cqlsh> INSERT INTO rep_factor_2.simple_table(in_key) VALUES('In Key');
cqlsh> SELECT * FROM rep_factor_2.simple_table;

 in_key
--------
 In Key

(1 rows)
cqlsh>
```

Пишу та зчитую з ноди cassandra_1 в/на rep_factor_3:

```
cqlsh> INSERT INTO rep_factor_3.simple_table(in_key) VALUES('In Key');
cqlsh> SELECT * FROM rep_factor_3.simple_table;

 in_key
--------
 In Key

(1 rows)
cqlsh>
```

Пишу та зчитую з ноди cassandra_2 в/на rep_factor_1:

```
┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_2 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> INSERT INTO rep_factor_1.simple_table(in_key) VALUES('In Key - 2');
cqlsh> SELECT * FROM rep_factor_1.simple_table;

 in_key
------------
 In Key - 2
     In Key

(2 rows)
cqlsh>
```

Пишу та зчитую з ноди cassandra_2 в/на rep_factor_2:

```
cqlsh> INSERT INTO rep_factor_2.simple_table(in_key) VALUES('In Key - 2');
cqlsh> SELECT * FROM rep_factor_1.simple_table;

 in_key
------------
 In Key - 2
     In Key

(2 rows)
cqlsh>
```

Пишу та зчитую з ноди cassandra_2 в/на rep_factor_3:

```
cqlsh> INSERT INTO rep_factor_3.simple_table(in_key) VALUES('In Key - 2');
cqlsh> SELECT * FROM rep_factor_3.simple_table;

 in_key
------------
 In Key - 2
     In Key

(2 rows)
cqlsh>
```

Пишу та зчитую з ноди cassandra_3 в/на rep_factor_1:

```
┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_3 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> INSERT INTO rep_factor_1.simple_table(in_key) VALUES('In Key - 3');
cqlsh> SELECT * FROM rep_factor_1.simple_table;

 in_key
------------
 In Key - 3
 In Key - 2
     In Key

(3 rows)
cqlsh>
```

Пишу та зчитую з ноди cassandra_3 в/на rep_factor_2:

```
cqlsh> INSERT INTO rep_factor_2.simple_table(in_key) VALUES('In Key - 3');
cqlsh> SELECT * FROM rep_factor_2.simple_table;

 in_key
------------
 In Key - 3
 In Key - 2
     In Key
```

Пишу та зчитую з ноди cassandra_3 в/на rep_factor_3:

```
cqlsh> INSERT INTO rep_factor_3.simple_table(in_key) VALUES('In Key - 3');
cqlsh> SELECT * FROM rep_factor_3.simple_table;

 in_key
------------
 In Key - 3
 In Key - 2
     In Key

(3 rows)
```

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

https://docs.datastax.com/en/cql/3.1/cql/cql_reference/insert_r.html

https://docs.datastax.com/en/cql/3.1/cql/cql_reference/select_r.html

https://www.tutorialspoint.com/cassandra/cassandra_create_data.htm

https://www.tutorialspoint.com/cassandra/cassandra_read_data.htm

```
┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_1 nodetool status rep_factor_1
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens  Owns (effective)  Host ID                               Rack
UN  172.21.0.4  80.91 KiB  16      37.6%             eca0bbd5-af55-46be-8752-6d8aa66aa459  rack1
UN  172.21.0.3  80.92 KiB  16      34.5%             eb5b2d79-eb72-4e5d-946a-65ff65b95d2e  rack1
UN  172.21.0.2  80.92 KiB  16      27.9%             3b88ba8a-6cfa-4494-9bfd-60041bb1d5c3  rack1


┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_1 nodetool status rep_factor_2
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens  Owns (effective)  Host ID                               Rack
UN  172.21.0.4  80.91 KiB  16      59.5%             eca0bbd5-af55-46be-8752-6d8aa66aa459  rack1
UN  172.21.0.3  80.92 KiB  16      72.2%             eb5b2d79-eb72-4e5d-946a-65ff65b95d2e  rack1
UN  172.21.0.2  80.92 KiB  16      68.3%             3b88ba8a-6cfa-4494-9bfd-60041bb1d5c3  rack1


┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_1 nodetool status rep_factor_3
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens  Owns (effective)  Host ID                               Rack
UN  172.21.0.4  80.91 KiB  16      100.0%            eca0bbd5-af55-46be-8752-6d8aa66aa459  rack1
UN  172.21.0.3  80.92 KiB  16      100.0%            eb5b2d79-eb72-4e5d-946a-65ff65b95d2e  rack1
UN  172.21.0.2  80.92 KiB  16      100.0%            3b88ba8a-6cfa-4494-9bfd-60041bb1d5c3  rack1


┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_2 nodetool status rep_factor_1
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens  Owns (effective)  Host ID                               Rack
UN  172.21.0.4  80.91 KiB  16      37.6%             eca0bbd5-af55-46be-8752-6d8aa66aa459  rack1
UN  172.21.0.3  80.92 KiB  16      34.5%             eb5b2d79-eb72-4e5d-946a-65ff65b95d2e  rack1
UN  172.21.0.2  80.92 KiB  16      27.9%             3b88ba8a-6cfa-4494-9bfd-60041bb1d5c3  rack1
```

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

https://docs.datastax.com/en/dse/5.1/dse-admin/datastax_enterprise/tools/nodetool/toolsGetEndPoints.html

```
┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_1 nodetool getendpoints rep_factor_1 simple_table in_key
172.21.0.3

┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_1 nodetool getendpoints rep_factor_2 simple_table in_key
172.21.0.3
172.21.0.4

┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_1 nodetool getendpoints rep_factor_3 simple_table in_key
172.21.0.3
172.21.0.4
172.21.0.2

┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_2 nodetool getendpoints rep_factor_1 simple_table in_key
172.21.0.3
```

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

```
┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS                     PORTS                                       NAMES
b2f98bd03c2b   cassandra   "docker-entrypoint.s…"   28 minutes ago   Up 28 minutes (Paused)     7000-7001/tcp, 7199/tcp, 9042/tcp, 9160/tcp   cassandra_3
73621877bc93   cassandra   "docker-entrypoint.s…"   28 minutes ago   Up 28 minutes              7000-7001/tcp, 7199/tcp, 9042/tcp, 9160/tcp   cassandra_2
1ce6d59bfac7   cassandra   "docker-entrypoint.s…"   28 minutes ago   Up 28 minutes              7000-7001/tcp, 7199/tcp, 9042/tcp, 9160/tcp   cassandra_1
```

- 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**

https://docs.datastax.com/en/cql/3.1/cql/cql_reference/consistency_r.html

CONSISTENCY ONE для keyspace з усіма replication_factor 1, 2 та 3:

```
┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_1 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> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh> INSERT INTO rep_factor_1.simple_table(in_key) VALUES('In Key - con 1 in rep 1');
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 datacenter1>: Unavailable('Error from server: code=1000 [Unava
stency\': \'ONE\', \'required_replicas\': 1, \'alive_replicas\': 0}')})
cqlsh> INSERT INTO rep_factor_2.simple_table(in_key) VALUES('In Key - con 1 in rep 2');
cqlsh> INSERT INTO rep_factor_3.simple_table(in_key) VALUES('In Key - con 1 in rep 3');
cqlsh> SELECT * FROM rep_factor_1.simple_table;
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 datacenter1>: Unavailable('Error from server: code=1000 [Unava
stency\': \'ONE\', \'required_replicas\': 1, \'alive_replicas\': 0}')})
cqlsh> SELECT * FROM rep_factor_2.simple_table;

 in_key
------------------------
             In Key - 3
             In Key - 2
 In Key - con 1 in rep 2
                In Key

(4 rows)
cqlsh> SELECT * FROM rep_factor_3.simple_table;

 in_key
------------------------
             In Key - 3
             In Key - 2
                In Key
 In Key - con 1 in rep 3

(4 rows)
cqlsh>
```

CONSISTENCY TWO для keyspace з replication factor 2 та 3:

```
cqlsh> CONSISTENCY TWO;
Consistency level set to TWO.
cqlsh> INSERT INTO rep_factor_2.simple_table(in_key) VALUES('In Key - con 2 in rep 2');
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 datacenter1>: Unavailable('Error from server: code=1000 [Unava
ilable exception] message="Cannot achieve consistency level TWO" info={\'consistency\': \'TWO\', \'required_replicas\': 2, \'alive_replicas\': 1}')})
cqlsh> INSERT INTO rep_factor_3.simple_table(in_key) VALUES('In Key - con 2 in rep 3');
cqlsh> SELECT * FROM rep_factor_2.simple_table;
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 datacenter1>: Unavailable('Error from server: code=1000 [Unava
ilable exception] message="Cannot achieve consistency level TWO" info={\'consistency\': \'TWO\', \'required_replicas\': 2, \'alive_replicas\': 1}')})
cqlsh> SELECT * FROM rep_factor_3.simple_table;

 in_key
------------------------
             In Key - 3
             In Key - 2
 In Key - con 2 in rep 3
                In Key
 In Key - con 1 in rep 3

(5 rows)
cqlsh>
```

CONSISTENCY THREE для keyspace з replication factor 3:

```
cqlsh> CONSISTENCY THREE;
Consistency level set to THREE.
cqlsh> INSERT INTO rep_factor_3.simple_table(in_key) VALUES('In Key - con 3 in rep 3');
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 datacenter1>: Unavailable('Error from server: code=1000 [Unava
ilable exception] message="Cannot achieve consistency level THREE" info={\'consistency\': \'THREE\', \'required_replicas\': 3, \'alive_replicas\': 2}')})
cqlsh> SELECT * FROM rep_factor_3.simple_table;
NoHostAvailable: ('Unable to complete the operation against any hosts', {<Host: 127.0.0.1:9042 datacenter1>: Unavailable('Error from server: code=1000 [Unava
ilable exception] message="Cannot achieve consistency level THREE" info={\'consistency\': \'THREE\', \'required_replicas\': 3, \'alive_replicas\': 2}')})
cqlsh>
```

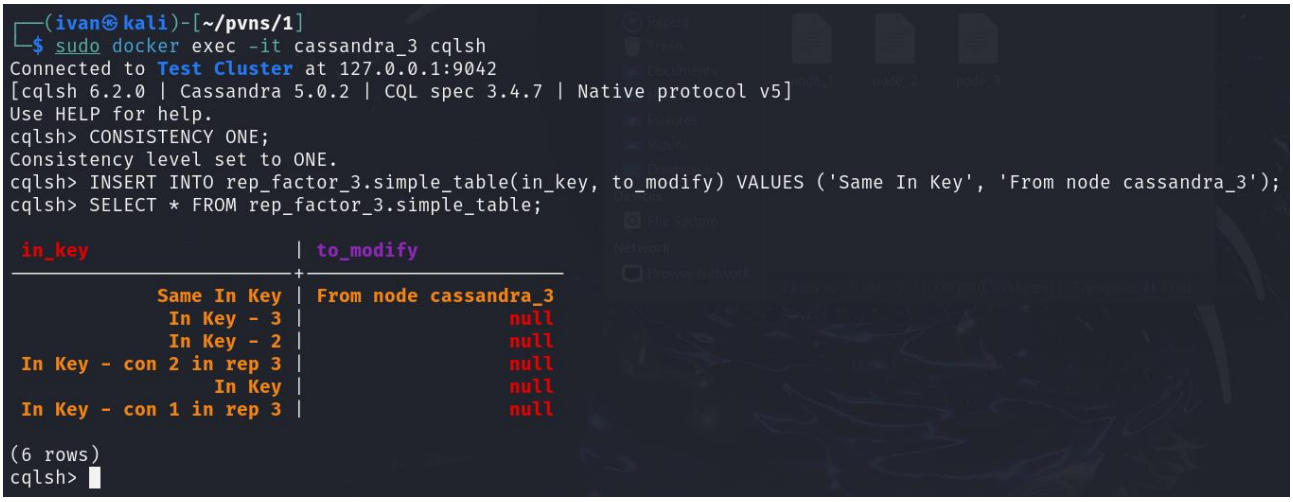
9. Зробить так щоб три ноди працювали, але не бачили одна одну по мережі (заблокуйте чи відключити зв'язок між ними)

```
┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED         STATUS         PORTS                                       NAMES
b2f98bd03c2b   cassandra   "docker-entrypoint.s…"   38 minutes ago   Up 38 minutes   7000-7001/tcp, 7199/tcp, 9042/tcp, 9160/tcp   cassandra_3
73621877bc93   cassandra   "docker-entrypoint.s…"   38 minutes ago   Up 38 minutes   7000-7001/tcp, 7199/tcp, 9042/tcp, 9160/tcp   cassandra_2
1ce6d59bfac7   cassandra   "docker-entrypoint.s…"   38 minutes ago   Up 38 minutes   7000-7001/tcp, 7199/tcp, 9042/tcp, 9160/tcp   cassandra_1

┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_1 nodetool disablegossip

┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_2 nodetool disablegossip

┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_3 nodetool disablegossip
```

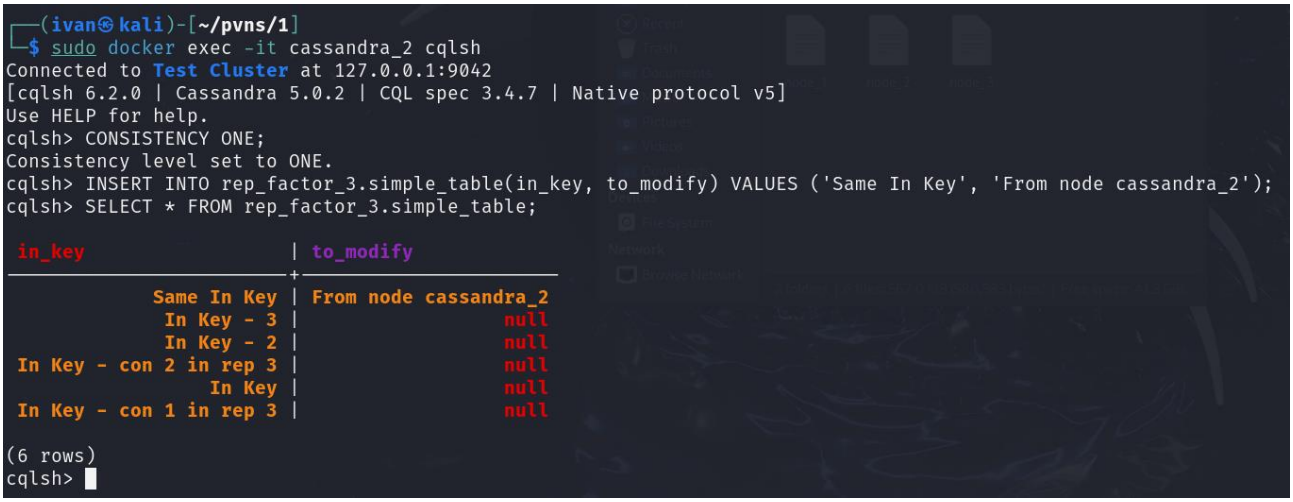
10. Для кейспейсу з *replication factor* 3 задайте рівень consistency рівним 1. Виконайте по черзі запис значення з однаковим primary key, але різними іншими значенням окремо на кожну з нод (тобто створіть конфлікт)

Модифікував прості таблиці на кожному replication factor:

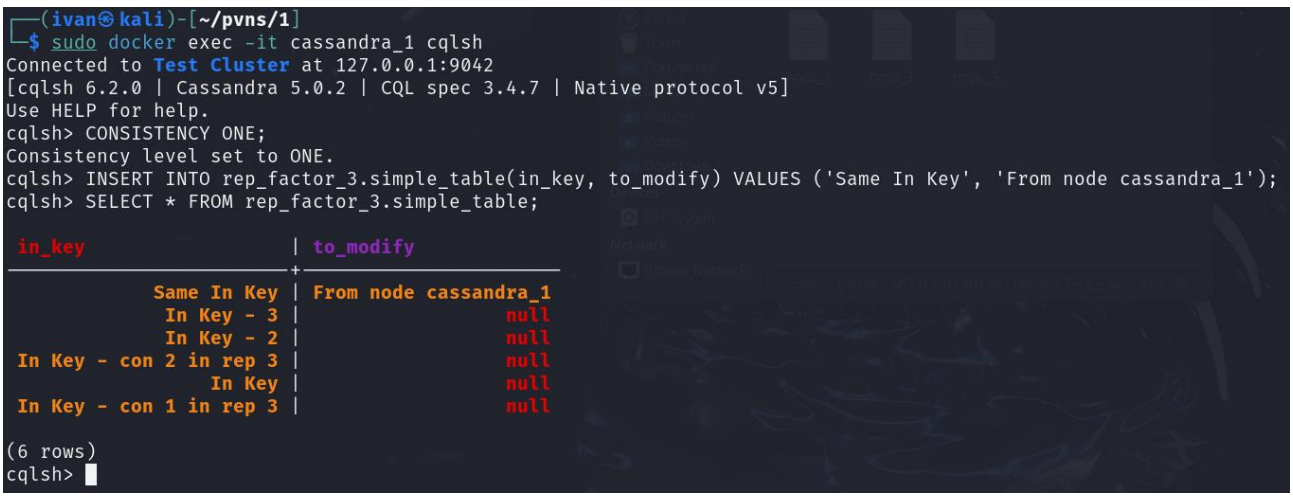
```
┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_1 cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.2 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> SELECT * FROM rep_factor_1.simple_table;

 in_key      | to_modify
-------------+-----------
 In Key - 3 |      null
 In Key - 2 |      null
     In Key |      null

(3 rows)
cqlsh> SELECT * FROM rep_factor_2.simple_table;

 in_key                     | to_modify
----------------------------+-----------
              In Key - 3 |      null
              In Key - 2 |      null
 In Key - con 1 in rep 2 |      null
                  In Key |      null

(4 rows)
cqlsh> SELECT * FROM rep_factor_3.simple_table;

 in_key                     | to_modify
----------------------------+-----------
              In Key - 3 |      null
              In Key - 2 |      null
 In Key - con 2 in rep 3 |      null
                  In Key |      null
 In Key - con 1 in rep 3 |      null

(5 rows)
cqlsh>
```



```
┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_3 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> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh> INSERT INTO rep_factor_3.simple_table(in_key, to_modify) VALUES ('Same In Key', 'From node cassandra_3');
cqlsh> SELECT * FROM rep_factor_3.simple_table;

 in_key                  | to_modify
-------------------------+-------------------------
             Same In Key | From node cassandra_3
              In Key - 3 |                    null
              In Key - 2 |                    null
 In Key - con 2 in rep 3 |                    null
                In Key |                    null
 In Key - con 1 in rep 3 |                    null

(6 rows)
cqlsh>
```



```
┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_2 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> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh> INSERT INTO rep_factor_3.simple_table(in_key, to_modify) VALUES ('Same In Key', 'From node cassandra_2');
cqlsh> SELECT * FROM rep_factor_3.simple_table;

 in_key                  | to_modify
-------------------------+-------------------------
             Same In Key | From node cassandra_2
              In Key - 3 |                    null
              In Key - 2 |                    null
 In Key - con 2 in rep 3 |                    null
                In Key |                    null
 In Key - con 1 in rep 3 |                    null

(6 rows)
cqlsh>
```



```
┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_1 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> CONSISTENCY ONE;
Consistency level set to ONE.
cqlsh> INSERT INTO rep_factor_3.simple_table(in_key, to_modify) VALUES ('Same In Key', 'From node cassandra_1');
cqlsh> SELECT * FROM rep_factor_3.simple_table;

 in_key                  | to_modify
-------------------------+-------------------------
             Same In Key | From node cassandra_1
              In Key - 3 |                    null
              In Key - 2 |                    null
 In Key - con 2 in rep 3 |                    null
                In Key |                    null
 In Key - con 1 in rep 3 |                    null

(6 rows)
cqlsh>
```

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

```
┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_1 nodetool enablegossip

┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_2 nodetool enablegossip

┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_3 nodetool enablegossip
```

```
┌──(ivan㉿kali)-[~/pvns/1]
└─$ sudo docker exec -it cassandra_1 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> SELECT * FROM rep_factor_3.simple_table;

 in_key                    | to_modify
---------------------------+-------------------------
             Same In Key | From node cassandra_1
              In Key - 3 |                    null
              In Key - 2 |                    null
 In Key - con 2 in rep 3 |                    null
                  In Key |                    null
 In Key - con 1 in rep 3 |                    null

(6 rows)
cqlsh>
```

Кластер обрав значення «From node cassandra_1» для за принципом «last-write-wins», оскільки я спочатку записував значення на ноду cassandra_3, потім на ноду cassandra_2 і лише потім на ноду cassandra_1.