Міністерство освіти і науки України
НТУУ «Київський політехнічний інститут»
Фізико-технічний інститут

**Проектування високонавантажених систем**
Лабораторна робота No5
Робота з базовими функціями БД типу column family на прикладі Cassandra

**Виконав:**
Студент 4-го курсу
групи ФІ-21
Климентьєв Максим
**Перевірив:**
_____

# Зміст

# 1 Код реалізації

task_1.cql

```
DROP KEYSPACE IF EXISTS shop;

/* IF NOT EXISTS */
CREATE KEYSPACE shop
WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};

USE shop;

/* IF NOT EXISTS */
CREATE TABLE items (
    id uuid,
    name text,
    producer text,
    category text,
    price decimal,
    features map<text, text>,
    PRIMARY KEY ((category), price, id)
);

/* IF NOT EXISTS */
CREATE INDEX items_features_index ON items(ENTRIES(features));
CREATE INDEX items_name_index ON items(name);

/* IF NOT EXISTS */
CREATE MATERIALIZED VIEW items_by_producer AS
SELECT category, producer, price, id, name, features
FROM items
WHERE category IS NOT NULL AND producer IS NOT NULL AND price IS NOT NULL AND id IS NOT NULL
PRIMARY KEY ((category), producer, price, id);

INSERT INTO items (category, price, id, name, producer, features) VALUES ('Конструктори', 450, uuid(), 'Classic
    Creative Bricks', 'LEGO', {'pieces': '200', 'age': '4+', 'material': 'plastic'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Конструктори', 899, uuid(), 'Technic Race
    Car', 'LEGO', {'pieces': '450', 'age': '9+', 'difficulty': 'medium'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Конструктори', 3200, uuid(), 'Star Wars
    Millennium Falcon', 'LEGO', {'pieces': '1351', 'age': '12+', 'theme': 'Star Wars'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Конструктори', 1500, uuid(), 'City Police
    Station', 'LEGO', {'pieces': '800', 'age': '6+', 'theme': 'City'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Конструктори', 650, uuid(), 'Minecraft The
    Cave', 'LEGO', {'pieces': '300', 'age': '8+', 'theme': 'Minecraft'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Конструктори', 300, uuid(), 'Tank T-34', '
    Sluban', {'pieces': '150', 'age': '6+', 'origin': 'China'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Конструктори', 2500, uuid(), 'Harry Potter
    Hogwarts', 'LEGO', {'pieces': '950', 'age': '10+', 'theme': 'Fantasy'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Конструктори', 120, uuid(), 'Mini Plane', '
    Sluban', {'pieces': '50', 'age': '5+'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Конструктори', 4200, uuid(), 'Technic
    Liebherr Excavator', 'LEGO', {'pieces': '2000', 'age': '16+', 'motor': 'yes'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Конструктори', 950, uuid(), 'Friends Beach
    House', 'LEGO', {'pieces': '550', 'age': '7+', 'series': 'Friends'});

INSERT INTO items (category, price, id, name, producer, features) VALUES ('Ляльки', 600, uuid(), 'Barbie Fashionistas',
     'Mattel', {'series': 'Fashion', 'material': 'plastic', 'height': '29cm'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Ляльки', 1200, uuid(), 'Barbie Dreamhouse
    Adventures', 'Mattel', {'set': 'doll+accessories', 'age': '3+'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Ляльки', 450, uuid(), 'Spider-Man Action
    Figure', 'Hasbro', {'hero': 'Marvel', 'articulation': 'yes', 'height': '15cm'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Ляльки', 2100, uuid(), 'Baby Born
    Interactive', 'Zapf Creation', {'functions': 'cry,eat,sleep', 'battery': 'no'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Ляльки', 350, uuid(), 'LOL Surprise Mini', '
    MGA', {'type': 'blind_box', 'series': 'Glitter'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Ляльки', 900, uuid(), 'Elsa Frozen II', '
    Hasbro', {'movie': 'Frozen', 'singing': 'yes'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Ляльки', 550, uuid(), 'Ken Doll', 'Mattel',
    {'series': 'Beach', 'gender': 'male'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Ляльки', 1800, uuid(), 'Rainbow High Fashion
     Doll', 'MGA', {'hair_color': 'rainbow', 'accessories': 'included'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Ляльки', 300, uuid(), 'Transformer Bumblebee
    ', 'Hasbro', {'transform': 'car', 'difficulty': 'easy'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Ляльки', 850, uuid(), 'Monster High
    Draculaura', 'Mattel', {'theme': 'Gothic', 'year': '2023'});

INSERT INTO items (category, price, id, name, producer, features) VALUES ('Настільні ігри', 1200, uuid(), 'Monopoly
    Classic', 'Hasbro', {'players': '2-6', 'time': '60-120min', 'lang': 'ua'});
```

```
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Настільні ігри', 250, uuid(), 'UNO Cards', '
    Mattel', {'players': '2-10', 'type': 'card_game', 'age': '7+'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Настільні ігри', 1500, uuid(), 'Catan', '
    Kosmos', {'players': '3-4', 'strategy': 'economic', 'time': '90min'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Настільні ігри', 800, uuid(), 'Dixit', '
    Libellud', {'type': 'associative', 'art': 'abstract', 'players': '3-6'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Настільні ігри', 400, uuid(), 'Jenga', '
    Hasbro', {'material': 'wood', 'type': 'dexterity'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Настільні ігри', 950, uuid(), 'Alias Party',
     'Tactic', {'lang': 'ua', 'type': 'words', 'players': '4+'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Настільні ігри', 1800, uuid(), 'Scythe', '
    Stonemaier Games', {'complexity': 'hard', 'theme': 'dieselpunk', 'time': '120min+'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Настільні ігри', 350, uuid(), 'Dobble', '
    Asmodee', {'type': 'reaction', 'compact': 'yes'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Настільні ігри', 600, uuid(), 'Carcassonne',
     'Hans im Gluck', {'type': 'tile_placement', 'theme': 'medieval'});
INSERT INTO items (category, price, id, name, producer, features) VALUES ('Настільні ігри', 2200, uuid(), 'Gloomhaven
    Jaws of the Lion', 'Cephalofair', {'type': 'rpg', 'campaign': 'yes', 'weight': 'heavy'});

/* IF NOT EXISTS */
CREATE TABLE orders (
    customer_name text,
    order_time timestamp,
    order_id uuid,
    item_ids list<uuid>,
    total_price decimal,
    PRIMARY KEY ((customer_name), order_time, order_id)
) WITH CLUSTERING ORDER BY (order_time DESC);

CREATE INDEX orders_item_ids_index ON orders(item_ids);
CREATE INDEX orders_total_price_index ON orders(total_price);

INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Ivan', '2023-10-01 10:00:00',
    uuid(), [uuid(), uuid()], 2000);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Ivan', '2023-10-05 14:30:00',
    uuid(), [uuid()], 500);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Ivan', '2023-11-10 09:15:00',
    uuid(), [uuid(), uuid(), uuid()], 4500);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Ivan', '2023-12-01 18:20:00',
    uuid(), [uuid()], 120);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Ivan', '2023-12-25 08:00:00',
    uuid(), [uuid(), uuid(), uuid(), uuid(), uuid()], 8000);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Ivan', '2024-01-05 11:45:00',
    uuid(), [uuid()], 300);

INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Maria', '2023-09-15 12:00:00',
     uuid(), [uuid(), uuid()], 1500);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Maria', '2023-10-20 16:10:00',
     uuid(), [uuid()], 600);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Maria', '2023-11-25 10:30:00',
     uuid(), [uuid(), uuid()], 1200);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Maria', '2023-12-19 07:00:00',
     uuid(), [uuid(), uuid(), uuid(), uuid()], 3500);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Maria', '2024-01-10 13:20:00',
     uuid(), [uuid()], 400);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Maria', '2024-02-14 15:00:00',
     uuid(), [uuid(), uuid()], 950);

INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Petro', '2023-11-01 19:00:00',
     uuid(), [uuid()], 2200);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Petro', '2023-11-02 19:05:00',
     uuid(), [uuid()], 1800);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Petro', '2023-11-15 20:30:00',
     uuid(), [uuid(), uuid()], 600);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Petro', '2023-12-05 14:00:00',
     uuid(), [uuid()], 950);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Petro', '2024-01-02 12:00:00',
     uuid(), [uuid(), uuid(), uuid()], 2500);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Petro', '2024-01-15 16:45:00',
     uuid(), [uuid()], 350);

INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Oksana', '2023-08-20 10:00:00'
    , uuid(), [uuid()], 4200);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Oksana', '2023-09-10 11:30:00'
    , uuid(), [uuid()], 3200);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Oksana', '2023-12-12 15:15:00'
    , uuid(), [uuid()], 899);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Oksana', '2024-01-01 12:00:00'
    , uuid(), [uuid(), uuid()], 1500);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Oksana', '2024-01-20 18:00:00'
    , uuid(), [uuid()], 2100);
```

```
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Oksana', '2024-02-01 09:30:00'
    , uuid(), [uuid(), uuid()], 5500);

INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Andriy', '2023-10-10 08:30:00'
    , uuid(), [uuid()], 250);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Andriy', '2023-10-12 08:45:00'
    , uuid(), [uuid()], 150);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Andriy', '2023-10-15 09:00:00'
    , uuid(), [uuid()], 450);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Andriy', '2023-11-20 17:20:00'
    , uuid(), [uuid(), uuid()], 800);
INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Andriy', '2023-12-30 20:00:00'
    , uuid(), [uuid()], 120);

INSERT INTO orders (customer_name, order_time, order_id, item_ids, total_price) VALUES ('Olena', '2024-01-25 14:00:00',
     uuid(), [uuid(), uuid(), uuid()], 5000);
```

## task_2.cql

```
DROP KEYSPACE IF EXISTS shop_rf1;
DROP KEYSPACE IF EXISTS shop_rf2;
DROP KEYSPACE IF EXISTS shop_rf3;

CREATE KEYSPACE shop_rf1
WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
CREATE TABLE shop_rf1.items (id int PRIMARY KEY, name text);

CREATE KEYSPACE shop_rf2
WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 2};
CREATE TABLE shop_rf2.items (id int PRIMARY KEY, name text);

CREATE KEYSPACE shop_rf3
WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 3};
CREATE TABLE shop_rf3.items (id int PRIMARY KEY, name text);

INSERT INTO shop_rf1.items (id, name) VALUES (1, 'Item One');
INSERT INTO shop_rf2.items (id, name) VALUES (1, 'Item One');
INSERT INTO shop_rf3.items (id, name) VALUES (1, 'Item One');
SELECT * FROM shop_rf1.items;
SELECT * FROM shop_rf2.items;
SELECT * FROM shop_rf3.items;
```

## task_3.cql

```
DROP KEYSPACE IF EXISTS shop_rf3;

CREATE KEYSPACE shop_rf3
WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 3};

USE shop_rf3;

CREATE TABLE likes_counter (
    id int PRIMARY KEY,
    likes counter
);

SELECT * FROM likes_counter;
```

## task_3.py

```
"""Частина
 3. Аналіз продуктивності та перевірка цілісностіАналогічно

 попереднім завданням, необхідно, для кластеру налаштованому у попередній частині, створити таблицю з каунтером лайків.
     Далі
 з 10 окремих клієнтів одночасно запустити інкерементацію каунтеру лайків по 10_000 на кожного клієнта з різними
     опціями взаємодії з Cassandra.Таблиця

 має бути створена у Keyspace з replication factor 3.Для

 створення каунтеру використовуйте спеціальний тип колонки - counter цей( тип буде підтримувати операції increment/
     decrement in-place):

1. Вказавши у параметрах запиту Consistency Level One це( буде означати, що запис відбувається синхронно тільки на
     одну ноду),
    запустіть 10 клієнтів з інкрементом по 10_000 на кожному з них.
```

```
    Виміряйте час виконання та перевірте чи кінцеве значення буде дорівнювати очікуваному - К100
2. Вказавши у параметрах запиту Consistency Level QUORUM це( буде означати, що запис відбувається синхронно на
     більшість нод),
    запустіть 10 клієнтів з інкрементом по 10_000 на кожному з них.
    Виміряйте час виконання та перевірте чи кінцеве значення буде дорівнювати очікуваному - К100
"""
import threading
import time
from cassandra.cluster import Cluster
from cassandra import ConsistencyLevel
from cassandra.query import SimpleStatement

def run_benchmark(session, consistency_level, test_name, clients=10, increments_per_client=10000, id=1):
    target_likes = clients * increments_per_client
    print(f"[TEST] {test_name}")
    print("[DELETE] Deleting previous data...")
    session.execute(f"TRUNCATE shop_rf3.likes_counter")
    time.sleep(4)

    query = f"UPDATE shop_rf3.likes_counter SET likes = likes + 1 WHERE id = {id}"
    statement = SimpleStatement(query, consistency_level=consistency_level)

    def client_worker():
        for _ in range(increments_per_client):
            try:
                session.execute(statement)
            except Exception as e:
                print(f"Error: {e}")


    threads = []
    start_time = time.time()

    print(f"[START] Clients: {clients} | Increments per client: {increments_per_client}")
    for _ in range(clients):
        t = threading.Thread(target=client_worker)
        threads.append(t)
        t.start()

    for t in threads:
        t.join()

    end_time = time.time()
    duration = end_time - start_time

    row = session.execute(f"SELECT likes FROM shop_rf3.likes_counter WHERE id = {id}").one()
    final_count = row.likes if row else 0

    print(f"[RESULTS] Duration: {duration:.2f} сек")
    print(f"[RESULTS] Throughput: {target_likes / duration:.0f} op/sec")
    print(f"[RESULTS] Expected: {target_likes:_}")
    print(f"[RESULTS] Received:  {final_count:_}")

if __name__ == "__main__":
    cluster = Cluster(['127.0.0.1'], port=9042)
    session = cluster.connect()

    try:
        run_benchmark(session, ConsistencyLevel.ONE, "CONSISTENCY ONE")
        run_benchmark(session, ConsistencyLevel.QUORUM, "CONSISTENCY QUORUM")
    finally:
        cluster.shutdown()
```

# 2 Результати

## 2.1 Частина 1. Робота зі структурами даних у Cassandra

Створіть keyspace з найпростішої стратегією реплікації

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

В цьому keyspace необхідно буде створити дві таблиці: items та orders Таблиця items містить різноманітні товари (тобто у яких різний набір властивостей).

Для набору властивостей товару виберіть базові характеристики однакові для всіх товарів (name, category, price, producer, ...), а для властивостей які відрізняються використовуйте тип map (створивши індекс для можливості пошуку по її вмісту)

Увага: Необхідно, щоб пошук швидко працював для категорії (category) товарів. Ця вимога має бути врахована при створенні ключа для таблиці (тобто, category має бути partition key).

```
CREATE TABLE items (
    id uuid,
    name text,
    producer text,
    category text,
    price decimal,
    features map<text, text>,
    PRIMARY KEY ((category), price, id)
);
CREATE INDEX items_features_index ON items(ENTRIES(features));
```

Наповніть таблиці тестовими данними

| category | price | id | features | name | producer |
|---|---|---|---|---|---|
| Ляльки | 300 | cefe6187-8bf4-4b38-a4b1-87deeaae43e1 | {'difficulty': 'easy', 'transform': 'car'} | Transformer Bumblebee | Hasbro |
| Ляльки | 350 | 556dc9dd-8d51-47c2-ac95-948c26a80ace | {'series': 'Glitter', 'type': 'blind_box'} | LOL Surprise Mini | MGA |
| Ляльки | 450 | 8c95d39a-382a-4f84-b42f-83f3b7b55fd8 | {'articulation': 'yes', 'height': '15cm', 'hero': 'Marvel'} | Spider-Man Action Figure | Hasbro |
| Ляльки | 550 | 206243ad-435c-498d-9997-be1015bfd102 | {'gender': 'male', 'series': 'Beach'} | Ken Doll | Mattel |
| Ляльки | 600 | 91b4fac8-d5ae-4910-bc84-bbd5bab08b2c | {'height': '29cm', 'material': 'plastic', 'series': 'Fashion'} | Barbie Fashionistas | Mattel |
| Ляльки | 850 | cf107892-dcd5-4962-95f4-d5045af9e120 | {'theme': 'Gothic', 'year': '2023'} | Monster High Draculaura | Mattel |
| Ляльки | 900 | bf12cc9b-dccd-4eb0-8d31-d39a5ab46240 | {'movie': 'Frozen', 'singing': 'yes'} | Elsa Frozen II | Hasbro |
| Ляльки | 1200 | 76db4601-d4c0-4d7b-b715-740e4a1ca3e1 | {'age': '3+', 'set': 'doll+accessories'} | Barbie Dreamhouse Adventures | Mattel |
| Ляльки | 1800 | 381e78b0-a79d-467c-a13d-db8bc45b0a1c | {'accessories': 'included', 'hair_color': 'rainbow'} | Rainbow High Fashion Doll | MGA |
| Ляльки | 2100 | b0c0dfca-b793-4ab5-8aac-cc6062676314 | {'battery': 'no', 'functions': 'cry,eat,sleep'} | Baby Born Interactive | Zapf Creation |
| Настільні ігри | 250 | 44ed9683-cdaa-4d92-9c54-c585e2371500 | {'age': '7+', 'players': '2-10', 'type': 'card_game'} | UNO Cards | Mattel |
| Настільні ігри | 350 | 69b40f6c-b2a7-48ab-870d-796a0d1d242a | {'compact': 'yes', 'type': 'reaction'} | Dobble | Asmodee |
| Настільні ігри | 400 | 4d062a91-2e11-4846-b8ca-af7aaa5b8fb0 | {'material': 'wood', 'type': 'dexterity'} | Jenga | Hasbro |
| Настільні ігри | 600 | 816bb3a4-9a2e-437b-83a5-3676be879856 | {'theme': 'medieval', 'type': 'tile_placement'} | Carcassonne | Hans im Gluck |
| Настільні ігри | 800 | 71256f78-70f7-48cb-a59e-fc15471bbbf6 | {'art': 'abstract', 'players': '3-6', 'type': 'associative'} | Dixit | Libellud |
| Настільні ігри | 950 | 8e6c23fa-4dd5-4e61-81e7-e4aa880cd391 | {'lang': 'ua', 'players': '4+', 'type': 'words'} | Alias Party | Tactic |
| Настільні ігри | 1200 | 32227102-a40c-4184-846d-a7be4d549c4e | {'lang': 'ua', 'players': '2-6', 'time': '60-120min'} | Monopoly Classic | Hasbro |
| Настільні ігри | 1500 | d1f9da51-e764-44a2-ba92-320a5cd3312a | {'players': '3-4', 'strategy': 'economic', 'time': '90min'} | Catan | Kosmos |
| Настільні ігри | 1800 | 4d424425-3fee-4b37-a617-1d3d1db8b9d0 | {'complexity': 'hard', 'theme': 'dieselpunk', 'time': '120min+'} | Scythe | Stonemaier Games |
| Настільні ігри | 2200 | 79bc9515-0f9a-4017-b6bc-bd56726ed167 | {'campaign': 'yes', 'type': 'rpg', 'weight': 'heavy'} | Gloomhaven Jaws of the Lion | Cephalofair |
| Конструктори | 120 | 80a55ce5-ef3e-480d-bebe-6a56376e05a1 | {'age': '5+', 'pieces': '50'} | Mini Plane | Sluban |
| Конструктори | 300 | 80a13e97-d080-4477-9da5-6526c186774c | {'age': '6+', 'origin': 'China', 'pieces': '150'} | Tank T-34 | Sluban |
| Конструктори | 450 | 1a75682a-64e8-48f7-9873-8e7589780560 | {'age': '4+', 'material': 'plastic', 'pieces': '200'} | Classic Creative Bricks | LEGO |
| Конструктори | 650 | afd1e3ed-01c6-4cb9-bfee-781674f31055 | {'age': '8+', 'pieces': '300', 'theme': 'Minecraft'} | Minecraft The Cave | LEGO |
| Конструктори | 899 | 8730a950-dd78-432b-bdc8-21578cfe4d03 | {'age': '9+', 'difficulty': 'medium', 'pieces': '450'} | Technic Race Car | LEGO |
| Конструктори | 950 | dcc56451-3e23-4818-9b99-efbe9051e889 | {'age': '7+', 'pieces': '550', 'series': 'Friends'} | Friends Beach House | LEGO |
| Конструктори | 1500 | db21a50d-03c9-45e2-adc8-35c91a4f1b89 | {'age': '6+', 'pieces': '800', 'theme': 'City'} | City Police Station | LEGO |
| Конструктори | 2500 | 2635630b-82e9-49a8-9e41-cfc0a066403a | {'age': '10+', 'pieces': '950', 'theme': 'Fantasy'} | Harry Potter Hogwarts | LEGO |
| Конструктори | 3200 | 9ceaa9a7-3a67-499a-9dd5-ef6841d942aa | {'age': '12+', 'pieces': '1351', 'theme': 'Star Wars'} | Star Wars Millennium Falcon | LEGO |
| Конструктори | 4200 | ec49333c-c370-4bd1-81b9-f6df8ea62bb1 | {'age': '16+', 'motor': 'yes', 'pieces': '2000'} | Technic Liebherr Excavator | LEGO |

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

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

```
DESCRIBE TABLE items;
```

```
CREATE TABLE shop.items (
    category text,
    price decimal,
    id uuid,
    name text,
    producer text,
    features map<text, text>,
    PRIMARY KEY (category, price, id)
) WITH CLUSTERING ORDER BY (price 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';

CREATE INDEX items_features_index ON shop.items (entries(features));

CREATE INDEX items_name_index ON shop.items (name);

CREATE MATERIALIZED VIEW shop.items_by_producer AS
    SELECT category, producer, price, id, features, name
    FROM shop.items
    WHERE category IS NOT NULL AND producer IS NOT NULL AND price IS NOT NULL AND id IS NOT NULL
    PRIMARY KEY (category, producer, price, id)
 WITH CLUSTERING ORDER BY (producer ASC, price 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 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';
```

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

```
select * from items where category = 'Настільні ігри';
```

```
cqlsh:shop> select * from items where category = 'Настільні ігри';
```

| category | price | id | features | name | producer |
|---|---|---|---|---|---|
| Настільні ігри | 250 | 7cff1657-9eda-44cf-933c-8f7109331348 | {'age': '7+', 'players': '2-10', 'type': 'card_game'} | UNO Cards | Mattel |
| Настільні ігри | 350 | 734a398c-28f4-4c05-8efe-3c7226152c15 | {'compact': 'yes', 'type': 'reaction'} | Dobble | Asmodee |
| Настільні ігри | 400 | 0c673cef-4bfe-4ddc-a8a7-982a3cc96364 | {'material': 'wood', 'type': 'dexterity'} | Jenga | Hasbro |
| Настільні ігри | 600 | 219709fd-9709-4928-b5b2-0cae81e2e696 | {'theme': 'medieval', 'type': 'tile_placement'} | Carcassonne | Hans im Gluck |
| Настільні ігри | 800 | 9e711a89-30c9-463f-8cbb-edbb1c293144 | {'art': 'abstract', 'players': '3-6', 'type': 'associative'} | Dixit | Libellud |
| Настільні ігри | 950 | efa6dfa0-bf57-4cb7-bcfc-fd5de0d06e97 | {'lang': 'ua', 'players': '4+', 'type': 'words'} | Alias Party | Tactic |
| Настільні ігри | 1200 | 71420877-6b34-4a43-8b0c-c50006feecfd | {'lang': 'ua', 'players': '2-6', 'time': '60-120min'} | Monopoly Classic | Hasbro |
| Настільні ігри | 1500 | 3f199ea5-453c-402f-906b-86424b985e84 | {'players': '3-4', 'strategy': 'economic', 'time': '90min'} | Catan | Kosmos |
| Настільні ігри | 1800 | b9d1dc2e-cf8e-40ef-88d4-da3cd82afb3d | {'complexity': 'hard', 'theme': 'dieselpunk', 'time': '120min+'} | Scythe | Stonemaier Games |
| Настільні ігри | 2200 | a7d2d61f-6456-418b-bf9f-ede909f1c76e | {'campaign': 'yes', 'type': 'rpg', 'weight': 'heavy'} | Gloomhaven Jaws of the Lion | Cephalofair |

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

   - назва,
   - ціна (в проміжку),
   - ціна та виробник

```
CREATE INDEX items_name_index ON items(name);

CREATE MATERIALIZED VIEW items_by_producer AS
SELECT category, producer, price, id, name, features
FROM items
WHERE category IS NOT NULL AND producer IS NOT NULL AND price IS NOT NULL AND id IS NOT NULL
PRIMARY KEY ((category), producer, price, id);

SELECT * FROM items WHERE name = 'UNO Cards';
SELECT * FROM items WHERE category = 'Настільні ігри' AND price >= 500 AND price <= 1000;
SELECT * FROM items_by_producer WHERE category = 'Настільні ігри' AND producer = 'Hasbro' AND price >= 500 AND
    price <= 3000;
```

```
cqlsh:shop> SELECT * FROM items WHERE name = 'UNO Cards';
```

| category | price | id | features | name | producer |
|---|---|---|---|---|---|
| Настільні ігри | 250 | 7cff1657-9eda-44cf-933c-8f7109331348 | {'age': '7+', 'players': '2-10', 'type': 'card_game'} | UNO Cards | Mattel |

```
cqlsh:shop> SELECT * FROM items WHERE category = 'Настільні ігри' AND price >= 500 AND price <= 1000;
```

| category | price | id | features | name | producer |
|---|---|---|---|---|---|
| Настільні ігри | 600 | 219709fd-9709-4928-b5b2-0cae81e2e696 | {'theme': 'medieval', 'type': 'tile_placement'} | Carcassonne | Hans im Gluck |
| Настільні ігри | 800 | 9e711a89-30c9-463f-8cbb-edbb1c293144 | {'art': 'abstract', 'players': '3-6', 'type': 'associative'} | Dixit | Libellud |
| Настільні ігри | 950 | efa6dfa0-bf57-4cb7-bcfc-fd5de0d06e97 | {'lang': 'ua', 'players': '4+', 'type': 'words'} | Alias Party | Tactic |

```
cqlsh:shop> SELECT * FROM items_by_producer WHERE category = 'Настільні ігри' AND producer = 'Hasbro' AND price >= 500 AND price <= 3000;

 category       | producer | price | id                                   | features                                                          | name
----------------+----------+-------+--------------------------------------+-------------------------------------------------------------------+-----------------
 Настільні ігри |   Hasbro |  1200 | 71420877-6b34-4a43-8b0c-c50006feecfd | {'lang': 'ua', 'players': '2-6', 'time': '60-120min'} | Monopoly Classic
```

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 (аналогічно як для items).

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

CREATE INDEX orders_item_ids_index ON orders(item_ids);
CREATE INDEX orders_total_price_index ON orders(total_price);
```

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

```
DESCRIBE TABLE orders;
```

```
CREATE TABLE shop.orders (
    customer_name text,
    order_time timestamp,
    order_id uuid,
    total_price decimal,
    item_ids list<uuid>,
    PRIMARY KEY (customer_name, order_time, order_id)
) WITH CLUSTERING ORDER BY (order_time DESC, 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';

CREATE INDEX orders_item_ids_index ON shop.orders (values(item_ids));

CREATE INDEX orders_total_price_index ON shop.orders (total_price);
```

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

```
EXPAND ON;
select * from orders where customer_name = 'Andriy';
EXPAND OFF;
select * from orders where customer_name = 'Andriy';
```

```
cqlsh:shop> select * from orders where customer_name = 'Andriy';

@ Row 1
---------------+-------------------------------------------------------------------------
 customer_name | Andriy
 order_time    | 2023-12-30 20:00:00.000000+0000
 order_id      | 60a7328c-12ce-4d59-8712-f0a6c1ce3387
 item_ids      | [60c9409a-303b-403d-a22c-d420045f7a94]
 total_price   | 120

@ Row 2
---------------+-------------------------------------------------------------------------
 customer_name | Andriy
 order_time    | 2023-11-20 17:20:00.000000+0000
 order_id      | d5d8ee2f-1bb1-44d2-88d3-cdee77a466bd
 item_ids      | [b01516ad-3754-4108-add8-c82aad7a3d27, 33223a0c-0f78-4bb1-8932-50f38a58c5e5]
 total_price   | 800

@ Row 3
---------------+-------------------------------------------------------------------------
 customer_name | Andriy
 order_time    | 2023-10-15 09:00:00.000000+0000
 order_id      | f1127d26-ef7a-4cef-b8ac-3603ebbbfc86
 item_ids      | [2d7c93cf-bf68-46a5-8ba7-c918fe3528af]
 total_price   | 450

@ Row 4
---------------+-------------------------------------------------------------------------
 customer_name | Andriy
 order_time    | 2023-10-12 08:45:00.000000+0000
 order_id      | bdd1e186-603a-4bfa-9240-96cf40451477
 item_ids      | [a32de239-9096-4bb2-8df1-89a595e8c3fe]
 total_price   | 150

@ Row 5
---------------+-------------------------------------------------------------------------
 customer_name | Andriy
 order_time    | 2023-10-10 08:30:00.000000+0000
 order_id      | 49f0b86d-7bb3-4dab-8bae-4996568d50f8
 item_ids      | [72223d1b-7a81-49e8-9d96-b172026be6a5]
 total_price   | 250
```

```
cqlsh:shop> EXPAND Off;
EXPAND set to OFF
cqlsh:shop> select * from orders where customer_name = 'Andriy';

 customer_name | order_time                      | order_id                             | item_ids                                                                                | total_price
---------------+---------------------------------+--------------------------------------+-----------------------------------------------------------------------------------------+-------------
       Andriy | 2023-12-30 20:00:00.000000+0000 | 60a7328c-12ce-4d59-8712-f0a6c1ce3387 |                                                [60c9409a-303b-403d-a22c-d420045f7a94] |         120
       Andriy | 2023-11-20 17:20:00.000000+0000 | d5d8ee2f-1bb1-44d2-88d3-cdee77a466bd | [b01516ad-3754-4108-add8-c82aad7a3d27, 33223a0c-0f78-4bb1-8932-50f38a58c5e5] |         800
       Andriy | 2023-10-15 09:00:00.000000+0000 | f1127d26-ef7a-4cef-b8ac-3603ebbbfc86 |                                                [2d7c93cf-bf68-46a5-8ba7-c918fe3528af] |         450
       Andriy | 2023-10-12 08:45:00.000000+0000 | bdd1e186-603a-4bfa-9240-96cf40451477 |                                                [a32de239-9096-4bb2-8df1-89a595e8c3fe] |         150
       Andriy | 2023-10-10 08:30:00.000000+0000 | 49f0b86d-7bb3-4dab-8bae-4996568d50f8 |                                                [72223d1b-7a81-49e8-9d96-b172026be6a5] |         250
```

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

```
select customer_name, sum(total_price) as all_price from orders GROUP BY customer_name;
```

```
cqlsh:shop> select customer_name, sum(total_price) as all_price from orders GROUP BY customer_name;

 customer_name | all_price
---------------+-----------
        Andriy |      1770
         Olena |      5000
         Petro |      8400
         Maria |      8150
          Ivan |     15420
        Oksana |     17399
```

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

```
SELECT customer_name, order_time, order_id, total_price, WRITETIME(total_price) as write_time FROM orders;
```

```
cqlsh:shop> SELECT customer_name, order_time, order_id, total_price, WRITETIME(total_price) as write_time FROM orders;

 customer_name | order_time                      | order_id                             | total_price | write_time
---------------+---------------------------------+--------------------------------------+-------------+------------------
        Andriy | 2023-12-30 20:00:00.000000+0000 | 60a7328c-12ce-4d59-8712-f0a6c1ce3387 |         120 | 1765834416002435
        Andriy | 2023-11-20 17:20:00.000000+0000 | d5d8ee2f-1bb1-44d2-88d3-cdee77a466bd |         800 | 1765834416001116
        Andriy | 2023-10-15 09:00:00.000000+0000 | f1127d26-ef7a-4cef-b8ac-3603ebbbfc86 |         450 | 1765834416000147
        Andriy | 2023-10-12 08:45:00.000000+0000 | bdd1e186-603a-4bfa-9240-96cf40451477 |         150 | 1765834415999090
        Andriy | 2023-10-10 08:30:00.000000+0000 | 49f0b86d-7bb3-4dab-8bae-4996568d50f8 |         250 | 1765834415998113
         Olena | 2024-01-25 14:00:00.000000+0000 | 3c6a7337-4582-4406-a767-4dea2ec07333 |        5000 | 1765834416004026
         Petro | 2024-01-15 16:45:00.000000+0000 | d3624ea8-f191-4484-8ca2-52efbf3b0eac |         350 | 1765834415990878
         Petro | 2024-01-02 12:00:00.000000+0000 | f6d68dcf-6980-4061-9ed4-12b5860f08c7 |        2500 | 1765834415989883
         Petro | 2023-12-05 14:00:00.000000+0000 | 65a603ef-4477-4544-b4e5-f535a988b152 |         950 | 1765834415988776
         Petro | 2023-11-15 20:30:00.000000+0000 | 3e0e2766-c957-4982-aa35-9f461bbfbe5a |         600 | 1765834415987745
         Petro | 2023-11-02 19:05:00.000000+0000 | 9bc52335-f51f-4e08-a407-5e736fe7aa97 |        1800 | 1765834415986731
         Petro | 2023-11-01 19:00:00.000000+0000 | 12d51bd0-3b76-4405-91bb-890b62091459 |        2200 | 1765834415985724
         Maria | 2024-02-14 15:00:00.000000+0000 | a7044ab1-f4d6-44a5-8255-f2a0c3d3d599 |         950 | 1765834415984162
         Maria | 2024-01-10 13:20:00.000000+0000 | e9819bad-aa57-4b64-82a5-0ed440bfa181 |         400 | 1765834415983109
         Maria | 2023-12-19 07:00:00.000000+0000 | ab511e8b-c284-4489-9771-12409e110593 |        3500 | 1765834415982089
         Maria | 2023-11-25 10:30:00.000000+0000 | 65b94de2-1d2b-4a9e-afb6-bea2e9cce1ea |        1200 | 1765834415981126
         Maria | 2023-10-20 16:10:00.000000+0000 | 088b2171-303b-4c63-9eee-207c3fb05488 |         600 | 1765834415980148
         Maria | 2023-09-15 12:00:00.000000+0000 | e2196db1-117c-4f80-b3dd-c2bf055db610 |        1500 | 1765834415979102
          Ivan | 2024-01-05 11:45:00.000000+0000 | 66027604-380f-485b-b1f9-23218df2589b |         300 | 1765834415977805
          Ivan | 2023-12-25 08:00:00.000000+0000 | e8f2adfc-ef15-4f4e-b4b5-548dce62f755 |        8000 | 1765834415976567
          Ivan | 2023-12-01 18:20:00.000000+0000 | a2c803ea-8bb2-4036-ba05-7c97875ee661 |         120 | 1765834415975531
          Ivan | 2023-11-10 09:15:00.000000+0000 | b89a398c-5452-4f30-af69-015d79547fb1 |        4500 | 1765834415974509
          Ivan | 2023-10-05 14:30:00.000000+0000 | 6ea41414-77d8-4413-9671-9d44c61a045f |         500 | 1765834415973276
          Ivan | 2023-10-01 10:00:00.000000+0000 | bad2bd5a-6de4-4054-bf6a-8c56120f5a6f |        2000 | 1765834415966869
        Oksana | 2024-02-01 09:30:00.000000+0000 | 2708972b-ecad-4e6b-98a6-8e8034e84b9c |        5500 | 1765834415997072
        Oksana | 2024-01-20 18:00:00.000000+0000 | cb8031a5-f167-4263-a8b2-8ae08939323d |        2100 | 1765834415996015
        Oksana | 2024-01-01 12:00:00.000000+0000 | 7dd14e9a-e37f-4f24-8b56-741eba8d24c3 |        1500 | 1765834415994968
        Oksana | 2023-12-12 15:15:00.000000+0000 | d021b60b-4280-4719-86a7-3f3f0648d58f |         899 | 1765834415993939
        Oksana | 2023-09-10 11:30:00.000000+0000 | afcec8a4-5c26-405c-b2d1-6686bc4b26de |        3200 | 1765834415992845
        Oksana | 2023-08-20 10:00:00.000000+0000 | 6b188434-1e8c-4387-8a13-89b66700de37 |        4200 | 1765834415991840
```

## 2.2 Частина 2. Налаштування реплікації у Cassandra

1. Сконфігурувати кластер з 3-х нод:

docker-compose.yml

```
services:
  # cassandra-task1:
  #   image: cassandra:5.0.6
  #   container_name: cassandra-task1
  #   hostname: cassandra-task1
  #   ports:
  #     - "9042:9042"
  #   environment:
  #     - CASSANDRA_CLUSTER_NAME=ShopCluster
  #   volumes:
  #     - ./scripts:/scripts
  #   user: root
  #   command: [ "bash", "/scripts/mater-view.sh"]
  #   healthcheck:
  #     test: ["CMD", "cqlsh", "-e", "describe keyspaces"]
  #     interval: 10s
  #     timeout: 10s
  #     retries: 10
  #   networks:
  #     - cassandra-net-task1

  # cassandra-task1-setup:
  #   image: cassandra:5.0.6
  #   container_name: cassandra-setup
  #   depends_on:
  #     cassandra-task1:
  #       condition: service_healthy
  #   volumes:
  #     - ./scripts:/scripts
  #   networks:
  #     - cassandra-net-task1
  #   # restart: unless-stopped
  #   entrypoint: ["cqlsh", "cassandra-task1", "-f",  "/scripts/
   task_1.cql"]

  cassandra-node1:
    image: cassandra:5.0.6
    container_name: cassandra-node1
    environment:
      - CASSANDRA_CLUSTER_NAME=shop_cluster
      - CASSANDRA_RPC_ADDRESS=0.0.0.0
      - CASSANDRA_BROADCAST_RPC_ADDRESS=cassandra-node1
      - CASSANDRA_SEEDS=cassandra-node1,cassandra-node2
      - CASSANDRA_ENDPOINT_SNITCH=GossipingPropertyFileSnitch
```

```
      - MAX_HEAP_SIZE=512M
      - HEAP_NEWSIZE=100M
    ports:
      - "9042:9042"
    volumes:
      - ./scripts:/scripts
    networks:
      - cassandra-net

  cassandra-node2:
    image: cassandra:5.0.6
    container_name: cassandra-node2
    environment:
      - CASSANDRA_CLUSTER_NAME=shop_cluster
      - CASSANDRA_RPC_ADDRESS=0.0.0.0
      - CASSANDRA_BROADCAST_RPC_ADDRESS=cassandra-node2
      - CASSANDRA_SEEDS=cassandra-node1,cassandra-node2
      - CASSANDRA_ENDPOINT_SNITCH=GossipingPropertyFileSnitch
      - MAX_HEAP_SIZE=512M
      - HEAP_NEWSIZE=100M
    ports:
      - "9043:9042"
    volumes:
      - ./scripts:/scripts
    networks:
      - cassandra-net

  cassandra-node3:
    image: cassandra:5.0.6
    container_name: cassandra-node3
    environment:
      - CASSANDRA_CLUSTER_NAME=shop_cluster
      - CASSANDRA_RPC_ADDRESS=0.0.0.0
      - CASSANDRA_BROADCAST_RPC_ADDRESS=cassandra-node3
      - CASSANDRA_SEEDS=cassandra-node1,cassandra-node2
      - CASSANDRA_ENDPOINT_SNITCH=GossipingPropertyFileSnitch
      - MAX_HEAP_SIZE=512M
      - HEAP_NEWSIZE=100M
    ports:
      - "9044:9042"
    volumes:
      - ./scripts:/scripts
    networks:
      - cassandra-net

networks:
  cassandra-net:
  cassandra-net-task1:
```

2. Перевірити правильність конфігурації за допомогою nodetool status

```
docker exec -it cassandra-node1 nodetool status
```

```
Datacenter: dc1
===============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens  Owns (effective)  Host ID                               Rack
UN  172.18.0.3  85.22 KiB  16      72.2%             6eec0752-ea2d-423c-ba6d-83211e6a73c5  rack1
UN  172.18.0.2  194.38 KiB 16      59.3%             4f7f8a4c-893e-4815-bee1-da6090b8a56c  rack1
UN  172.18.0.4  80.09 KiB  16      68.5%             34511972-165a-4c27-8448-9f92768af587  rack1

```

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

```
CREATE KEYSPACE shop_rf1
WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};

CREATE KEYSPACE shop_rf2
WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 2};

CREATE KEYSPACE shop_rf3
WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 3};
```

4. В кожному з кейспейсів створити прості таблиці

```
CREATE TABLE shop_rf1.items (id int PRIMARY KEY, name text);

CREATE TABLE shop_rf2.items (id int PRIMARY KEY, name text);

CREATE TABLE shop_rf3.items (id int PRIMARY KEY, name text);
```

1. Спробуйте писати і читати в ці таблиці підключаюсь до різних нод через cqlsh.

```
INSERT INTO shop_rf1.items (id, name) VALUES (1, 'Item One');
INSERT INTO shop_rf2.items (id, name) VALUES (1, 'Item One');
INSERT INTO shop_rf3.items (id, name) VALUES (1, 'Item One');
SELECT * FROM shop_rf1.items;
SELECT * FROM shop_rf2.items;
SELECT * FROM shop_rf3.items;

docker exec -it cassandra-node1 cqlsh cassandra-node1 -f /scripts/task_2.cql
```

2. Вставте дані в створені таблиці і подивіться на їх розподіл по вузлах кластера окремо для кожного з кейспесов (команда nodetool status) - має бути видно відсоток даних який зберігається на ноді

```
docker exec -it cassandra-node1 nodetool status shop_rf1
docker exec -it cassandra-node1 nodetool status shop_rf2
docker exec -it cassandra-node1 nodetool status shop_rf3
```

```
Datacenter: dc1
===============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens  Owns (effective)  Host ID                               Rack
UN  172.18.0.3  86.01 KiB  16      30.3%             6eec0752-ea2d-423c-ba6d-83211e6a73c5  rack1
UN  172.18.0.2  80.09 KiB  16      40.3%             4f7f8a4c-893e-4815-bee1-da6090b8a56c  rack1
UN  172.18.0.4  80.09 KiB  16      29.4%             34511972-165a-4c27-8448-9f92768af587  rack1

Datacenter: dc1
===============
```

```
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns (effective)  Host ID                               Rack
UN  172.18.0.3  86.01 KiB  16      72.2%             6eec0752-ea2d-423c-ba6d-83211e6a73c5  rack1
UN  172.18.0.2  80.09 KiB  16      59.3%             4f7f8a4c-893e-4815-bee1-da6090b8a56c  rack1
UN  172.18.0.4  80.09 KiB  16      68.5%             34511972-165a-4c27-8448-9f92768af587  rack1

Datacenter: dc1
===============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns (effective)  Host ID                               Rack
UN  172.18.0.3  86.01 KiB  16      100.0%            6eec0752-ea2d-423c-ba6d-83211e6a73c5  rack1
UN  172.18.0.2  80.09 KiB  16      100.0%            4f7f8a4c-893e-4815-bee1-da6090b8a56c  rack1
UN  172.18.0.4  80.09 KiB  16      100.0%            34511972-165a-4c27-8448-9f92768af587  rack1
```

3. Для якогось запису з кожного з кейспейсу виведіть ноди на яких зберігаються дані - має бути видно ip-адреси вузлів на яких зберігається даний рядок

```
docker exec -it cassandra-node1 nodetool getendpoints shop_rf1 items 1
docker exec -it cassandra-node1 nodetool getendpoints shop_rf2 items 1
docker exec -it cassandra-node1 nodetool getendpoints shop_rf3 items 1
```

```
172.18.0.2

172.18.0.2
172.18.0.4

172.18.0.2
172.18.0.4
172.18.0.3
```

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

```
docker stop cassandra-node3
docker exec -it cassandra-node1 cqlsh
```

- для Keyspace з replication factor 1 - CONSISTENCY ONE

```
USE shop_rf1;
CONSISTENCY ONE;
INSERT INTO items (id, name) VALUES (100, 'Risky Data');
```

```
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}')})
```

- для Keyspace з replication factor 2 - CONSISTENCY ONE/TWO

```
USE shop_rf2;
CONSISTENCY ONE;
INSERT INTO items (id, name) VALUES (200, 'Risky Data');
select * from items;

CONSISTENCY TWO;
INSERT INTO items (id, name) VALUES (201, 'Risky Data 2');
```

```
 id  | name
-----+------------
   1 |   Item One
 200 | Risky Data

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}')})
```

- для Keyspace з replication factor 3 - CONSISTENCY ONE/TWO/THREE

```
USE shop_rf3;
CONSISTENCY ONE;
INSERT INTO items (id, name) VALUES (300, 'Risky Data');
select * from items;

CONSISTENCY TWO;
INSERT INTO items (id, name) VALUES (301, 'Risky Data 2');
select * from items;

CONSISTENCY THREE;
INSERT INTO items (id, name) VALUES (302, 'Risky Data 2');
```

```
 id  | name
-----+------------
   1 |   Item One
 300 | Risky Data

 id  | name
-----+--------------
   1 |     Item One
 300 |   Risky Data
 301 | Risky Data 2

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}')})
```

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

```
docker start cassandra-node3
docker network disconnect lab5_cassandra-net cassandra-node1
docker network disconnect lab5_cassandra-net cassandra-node2
docker network disconnect lab5_cassandra-net cassandra-node3
```

2. 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)

```
docker exec -it cassandra-node1 cqlsh -e "USE shop_rf3; CONSISTENCY ONE; INSERT INTO items (id, name) VALUES
    (777, 'ISLAND_1');"
docker exec -it cassandra-node2 cqlsh -e "USE shop_rf3; CONSISTENCY ONE; INSERT INTO items (id, name) VALUES
    (777, 'ISLAND_2');"
docker exec -it cassandra-node3 cqlsh -e "USE shop_rf3; CONSISTENCY ONE; INSERT INTO items (id, name) VALUES
    (777, 'ISLAND_3');"
```

3. 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

```
docker network connect lab5_cassandra-net cassandra-node1
docker network connect lab5_cassandra-net cassandra-node2
docker network connect lab5_cassandra-net cassandra-node3

docker exec -it cassandra-node1 nodetool status

docker restart cassandra-node1 cassandra-node2 cassandra-node3

docker exec -it cassandra-node1 nodetool status

docker exec -it cassandra-node1 cqlsh -e "SELECT id, name, WRITETIME(name) FROM shop_rf3.items WHERE id = 777;"
```

```
 id  | name     | writetime(name)
-----+----------+------------------
 777 | ISLAND_3 | 1765903107702957
```

## 2.3 Частина 3. Аналіз продуктивності та перевірка цілісності

Аналогічно попереднім завданням, необхідно, для кластеру налаштованому у попередній частині, створити таблицю з каунтером лайків. Далі з 10 окремих клієнтів одночасно запустити інкерементацію каунтеру лайків по 10_000 на кожного клієнта з різними опціями взаємодії з Cassandra.

Таблиця має бути створена у Keyspace з replication factor 3.

Для створення каунтеру використовуйте спеціальний тип колонки - counter (цей тип буде підтримувати операції increment/decrement in-place):

```
CREATE KEYSPACE shop_rf3
WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 3};
USE shop_rf3;

CREATE TABLE likes_counter (
    id int PRIMARY KEY,
    likes counter
);

docker exec -it cassandra-node1 cqlsh cassandra-node1 -f /scripts/task_3.cql
```

1. Вказавши у параметрах запиту Consistency Level One (це буде означати, що запис відбувається синхронно тільки на одну ноду), запустіть 10 клієнтів з інкрементом по 10_000 на кожному з них. Виміряйте час виконання та перевірте чи кінцеве значення буде дорівнювати очікуваному - 100К

```
[TEST] CONSISTENCY ONE
[DELETE] Deleting previous data...
[START] Clients: 10 | Increments per client: 10000
[RESULTS] Duration: 30.21 сек
[RESULTS] Throughput: 3311 op/sec
[RESULTS] Expected: 100_000
[RESULTS] Received:  100_000
```

2. Вказавши у параметрах запиту Consistency Level QUORUM (це буде означати, що запис відбувається синхронно на більшість нод), запустіть 10 клієнтів з інкрементом по 10_000 на кожному з них. Виміряйте час виконання та перевірте чи кінцеве значення буде дорівнювати очікуваному - 100К

```
[TEST] CONSISTENCY QUORUM
[DELETE] Deleting previous data...
[START] Clients: 10 | Increments per client: 10000
[RESULTS] Duration: 34.70 сек
[RESULTS] Throughput: 2882 op/sec
[RESULTS] Expected: 100_000
[RESULTS] Received:  100_000
```