



*Міністерство освіти і науки України Національний технічний університет України
«Київський політехнічний інститут ім. І. Сікорського» Фізико-технічний інститут*

ПРОЕКТУВАННЯ ВИСОКОНАВАНТАЖЕНИХ СИСТЕМ

ЛАБОРАТОРНА РОБОТА №3

РОБОТА З БАЗОВИМИ ФУНКЦІЯМИ ГРАФ-ОРІЄНТОВАНОЇ БД НА ПРИКЛАДІ NEO4J

Виконав:

Студент групи ФБ-41мп

Заріцький О. В.

Київ 2024 р.

Встановлення Neo4j

```
compose.yml
1  services:
2    neo4j:
3      image: neo4j
4      ports:
5        - "7474:7474"
6        - "7687:7687"
7      volumes:
8        - ./neo4j/data:/data
9        - ./neo4j/logs:/logs
10     environment:
11       - NEO4J_AUTH=neo4j/zaritsky_fb41mp
12       - NEO4J_ACCEPT_LICENSE_AGREEMENT=yes
13       - NEO4JLABS_PLUGINS=["graph-data-science", "apoc"]
14  volumes:
15    neo4j:
16      driver: local
17
```

\$:server connect

Connect to Neo4j

Database access might require an authenticated connection

Connect URL

neo4j//

Database - leave empty for default

Authentication type

Username / Password

Username

Password

Connect

Змодельовати наступну предметну область:

Є: Items, Customers, Orders

```
CREATE (i1:Item {id: 1, name: 'Electric Guitar', price: 15000})
CREATE (i2:Item {id: 2, name: 'Drum Set', price: 30000})
CREATE (i3:Item {id: 3, name: 'Synthesizer', price: 20000})
CREATE (i4:Item {id: 4, name: 'Ukulele', price: 5000})
CREATE (i5:Item {id: 5, name: 'Violin', price: 12000})

CREATE (c1:Customer {id: 1, nickname: 'RockStar', email: 'rockstar@example.com'})
CREATE (c2:Customer {id: 2, nickname: 'BeatMaster', email: 'beatmaster@example.com'})
CREATE (c3:Customer {id: 3, nickname: 'PianoKing', email: 'pianoking@example.com'})

CREATE (o1:Order {id: 1, date: '2024-10-15'})
CREATE (o2:Order {id: 2, date: '2024-10-20'})
CREATE (o3:Order {id: 3, date: '2024-11-05'})
CREATE (o4:Order {id: 4, date: '2024-11-18'})
CREATE (o5:Order {id: 5, date: '2024-12-01'})
```

neo4j\$ MATCH (n) RETURN n	
Graph	[n]
Table	[(:Item {price: 15000,name: "Electric Guitar",id: 1})]
Text	[(:Item {price: 30000,name: "Drum Set",id: 2})]
Code	[(:Item {price: 20000,name: "Synthesizer",id: 3})]
	[(:Item {price: 5000,name: "Ukulele",id: 4})]
	[(:Item {price: 12000,name: "Violin",id: 5})]
	[(:Customer {nickname: "RockStar",id: 1,email: "rockstar@example.com"})]
	[(:Customer {nickname: "BeatMaster",id: 2,email: "beatmaster@example.com"})]
	[(:Customer {nickname: "PianoKing",id: 3,email: "pianoking@example.com"})]
	[(:Order {date: "2024-10-15",id: 1})]
	[(:Order {date: "2024-10-20",id: 2})]
	[(:Order {date: "2024-11-05",id: 3})]
	[(:Order {date: "2024-11-18",id: 4})]
	[(:Order {date: "2024-12-01",id: 5})]

Customer може додати Item(s) до Order (тобто купити Товар)

У Customer може бути багато Orders

Item може входити в багато Orders, і у Item є вартість

Customer може переглядати (view), але при цьому не купувати Items

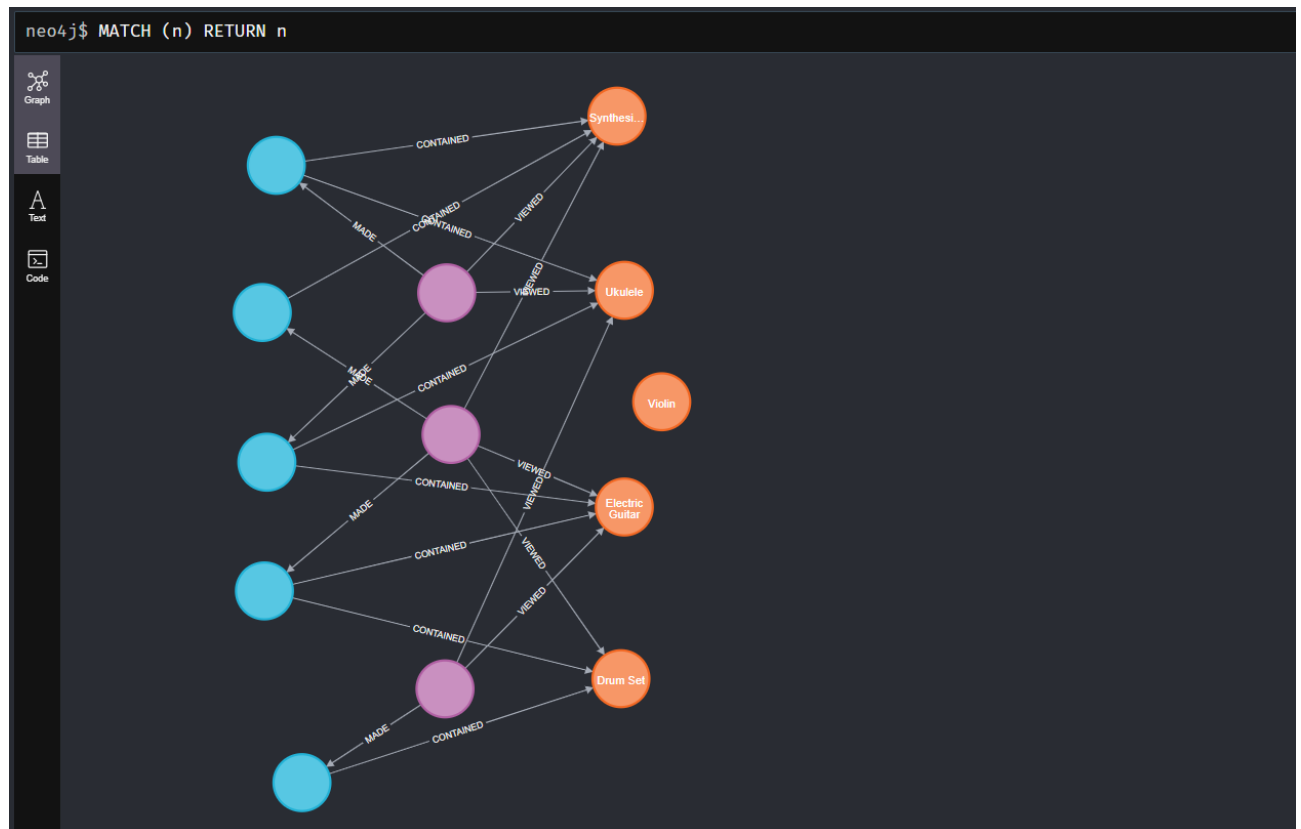
```
// Підбір покупців і замовлень
MATCH (c1:Customer {id: 1}), (c2:Customer {id: 2}), (c3:Customer {id: 3}),
      (o1:Order {id: 1}), (o2:Order {id: 2}), (o3:Order {id: 3}),
      (o4:Order {id: 4}), (o5:Order {id: 5})

CREATE
(c1)-[:MADE]->(o1),
(c1)-[:MADE]->(o2),
(c2)-[:MADE]->(o3),
(c3)-[:MADE]->(o4),
(c2)-[:MADE]->(o5);

// Зв'язки між замовленнями і товарами
MATCH (o1:Order {id: 1}), (o2:Order {id: 2}), (o3:Order {id: 3}),
      (o4:Order {id: 4}), (o5:Order {id: 5}),
      (i1:Item {id: 1}), (i2:Item {id: 2}), (i3:Item {id: 3}), (i4:Item {id: 4})

CREATE
(o1)-[:CONTAINED]->(i1),
(o1)-[:CONTAINED]->(i2),
(o2)-[:CONTAINED]->(i3),
(o3)-[:CONTAINED]->(i4),
(o3)-[:CONTAINED]->(i1),
(o4)-[:CONTAINED]->(i2),
(o5)-[:CONTAINED]->(i3),
(o5)-[:CONTAINED]->(i4);
```

```
// Перегляд товарів
MATCH (c1:Customer {id: 1}), (c2:Customer {id: 2}), (c3:Customer {id: 3}),
      (i1:Item {id: 1}), (i2:Item {id: 2}), (i3:Item {id: 3}), (i4:Item {id: 4})
CREATE
(c1)-[:VIEWED {viewDate: datetime()}]->(i1),
(c1)-[:VIEWED {viewDate: datetime()}]->(i2),
(c1)-[:VIEWED {viewDate: datetime()}]->(i3),
(c2)-[:VIEWED {viewDate: datetime()}]->(i3),
(c2)-[:VIEWED {viewDate: datetime()}]->(i4),
(c3)-[:VIEWED {viewDate: datetime()}]->(i1),
(c3)-[:VIEWED {viewDate: datetime()}]->(i4);
```



Написати наступні види запитів:

Знайти Items які входять в конкретний Order

```
1 MATCH (o:Order {id: 5})-[:CONTAINED]->(i:Item)
2 RETURN i
```

i
(:Item {price: 5000,name: "Ukulele",id: 4})
(:Item {price: 20000,name: "Synthesizer",id: 3})

Підрахувати вартість конкретного Order

```
1 MATCH (o:Order {id: 5})-[:CONTAINED]→(i:Item)
2 RETURN SUM[i.price]
```

Table

SUM(i.price)
25000

Text

Знайти всі Orders конкретного Customer

```
1 MATCH (c:Customer {nickname: 'PianoKing'})-[:MADE]→(o:Order)
2 RETURN o
```

Graph

Table

o
(:Order {date: "2024-11-18",id: 4})

Text

Знайти всі Items куплені конкретним Customer (через Order)

```
1 MATCH (c:Customer {nickname: 'BeatMaster'})-[:MADE]→(o:Order)-[:CONTAINED]→(i:Item)
2 RETURN i
```

Graph

Table

i
(:Item {price: 5000,name: "Ukulele",id: 4})
(:Item {price: 20000,name: "Synthesizer",id: 3})
(:Item {price: 15000,name: "Electric Guitar",id: 1})
(:Item {price: 5000,name: "Ukulele",id: 4})

Text

Code

Знайти кількість Items куплені конкретним Customer (через Order)

```
1 MATCH (c:Customer {nickname: 'RockStar'})-[:MADE]→(o:Order)-[:CONTAINED]→(i:Item)
2 RETURN COUNT[i]
```

Table

COUNT(i)
3

Text

Знайти для Customer на яку суму він придбав товарів (через Order)

```
1 MATCH (c:Customer {nickname: 'RockStar'})-[:MADE]-(o:Order)-[:CONTAINED]-(i:Item)
2 RETURN SUM(i.price)
```

Table

SUM(i.price)
65000

Text

Знайти скільки разів кожен товар був придбаний, відсортувати за цим значенням

```
neo4j$ MATCH (o:Order)-[:CONTAINED]-(i:Item) RETURN i, COUNT(i) AS n ORDER BY n DESC
```

Graph

Table

i	n
(:Item {price: 15000,name: "Electric Guitar",id: 1})	2
(:Item {price: 30000,name: "Drum Set",id: 2})	2
(:Item {price: 20000,name: "Synthesizer",id: 3})	2
(:Item {price: 5000,name: "Ukulele",id: 4})	2

Text

Code

Знайти всі Items переглянуті (view) конкретним Customer

```
1 MATCH (c:Customer {nickname: 'BeatMaster'})-[:VIEWED]-(i:Item)
2 RETURN i
```

Graph

Table

i
(:Item {price: 5000,name: "Ukulele",id: 4})
(:Item {price: 20000,name: "Synthesizer",id: 3})

Text

Знайти інші Items що купувались разом з конкретним Item (тобто всі Items що входять до Order-s разом з даними Item)

```
1 MATCH (i:Item {id: 1})<-[:CONTAINED]-(o:Order)-[:CONTAINED]-(in_one_cart:Item)
2 RETURN in_one_cart
```

Graph

Table

in_one_cart
(:Item {price: 5000,name: "Ukulele",id: 4})
(:Item {price: 30000,name: "Drum Set",id: 2})

Text

Знайти Customers які купили даний конкретний Item

```
1 MATCH (i:Item {id: 2})<-[[:CONTAINED]]-(o:Order)<-[[:MADE]]-(who:Customer)
2 RETURN who
```

Graph

Table

Text

who
(:Customer {nickname: "PianoKing",id: 3,email: "pianoking@example.com"})
(:Customer {nickname: "RockStar",id: 1,email: "rockstar@example.com"})

Знайти для певного Customer(a) товари, які він переглядав, але не купив

```
1 MATCH (c:Customer {nickname: 'PianoKing'})-[:VIEWED]->(i:Item)
2 WHERE NOT (c)-[:MADE]->(:Order)-[:CONTAINED]->(i)
3 RETURN i
```

Graph

Table

Text

i
(:Item {price: 5000,name: "Ukulele",id: 4})
(:Item {price: 15000,name: "Electric Guitar",id: 1})

Як і в попередніх завданнях, для якогось одного обраного Item додайте поле з кількістю його лайків.

Виконання:

```
uri = "bolt://localhost:7687"
user = "neo4j"
password = "zaritsky_fb4lmp"

driver = GraphDatabase.driver(uri, auth=(user, password))

def initialize_likes_field(driver, item_id):
    with driver.session() as session:
        session.run("""
            MATCH (i:Item {id: $item_id})
            SET i.likes = 0
            """, item_id=item_id)

def increment_likes(driver, item_id, increments):
    with driver.session() as session:
        for _ in range(increments):
            session.run("""
                MATCH (i:Item {id: $item_id})
                SET i.likes = i.likes + 1
                """, item_id=item_id)

def get_likes(driver, item_id):
    with driver.session() as session:
        result = session.run("""
            MATCH (i:Item {id: $item_id})
            RETURN i.likes AS likes
            """, item_id=item_id)
    return result.single()["likes"]
```

```
def safe_increment_likes(driver, item_id, increments):
    retries = 5
    for _ in range(increments):
        success = False
        attempts = 0
        while not success and attempts < retries:
            try:
                increment_likes(driver, item_id, 1)
                success = True
            except TransientError:
                attempts += 1
                time.sleep(0.1)
```

Результат:

```
PS D:\KPI\mag\hisp\lab3> & "C:/Users/Ede1/AppData/Local/Programs/Python/Python311/python.exe"
Initializing 'likes' for the item with ID: 1
Starting threads for incrementing...
Final 'likes' count for item with ID 1: 100000
Execution time: 158.51 seconds
PS D:\KPI\mag\hisp\lab3> □
```