

Дмитренко ФБ-42мп

Змодельовати наступну предметну область:

- Є: Items, Customers, Orders
- 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

```
CREATE (:Customer {id: 1, name: "Alice"});
```

```
CREATE (:Customer {id: 2, name: "Bob"});
```

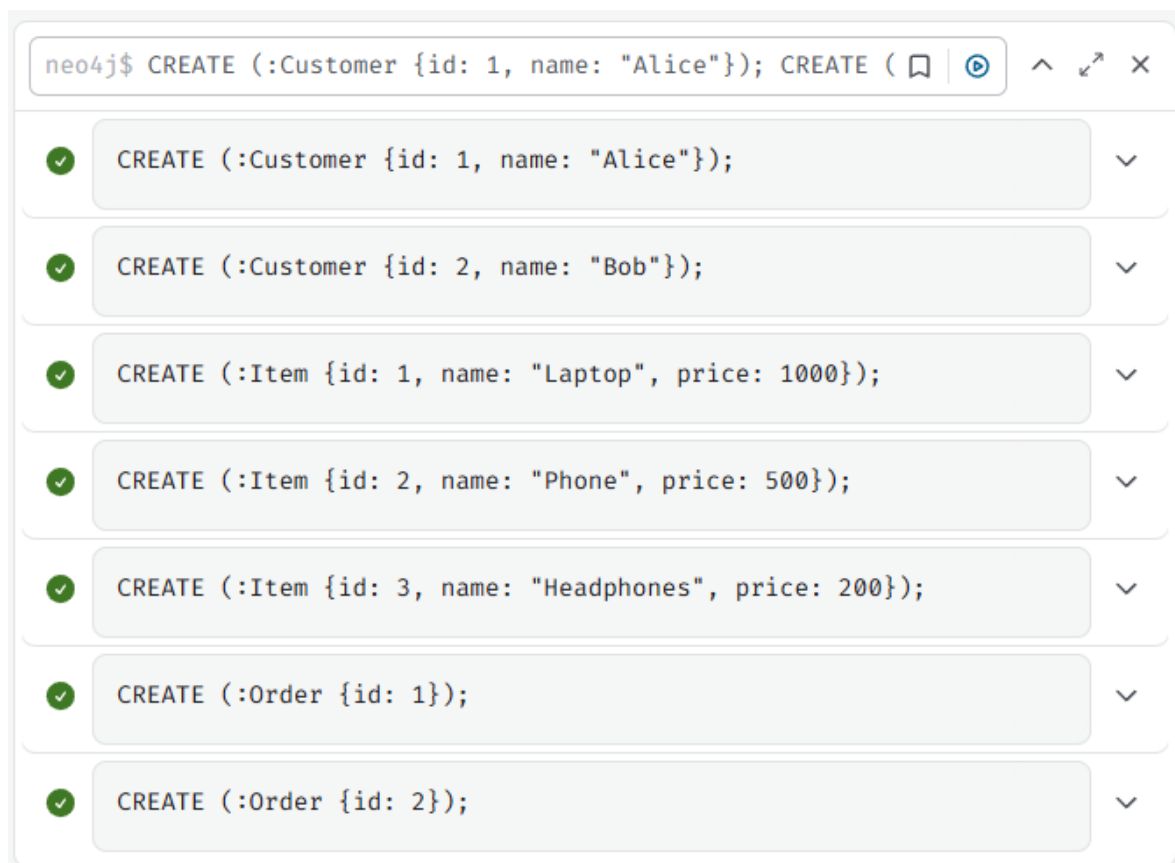
```
CREATE (:Item {id: 1, name: "Laptop", price: 1000});
```

```
CREATE (:Item {id: 2, name: "Phone", price: 500});
```

```
CREATE (:Item {id: 3, name: "Headphones", price: 200});
```

```
CREATE (:Order {id: 1});
```

```
CREATE (:Order {id: 2});
```



```
MATCH (c:Customer {id: 1}), (o:Order {id: 1})
```

```
CREATE (c)-[:CREATED]->(o);
```

```
MATCH (o:Order {id: 1}), (i:Item {id: 1})
```

```
CREATE (o)-[:CONTAINS]->(i);
```

```
MATCH (o:Order {id: 1}), (i:Item {id: 2})
```

```
CREATE (o)-[:CONTAINS]->(i);
```

```
MATCH (c:Customer {id: 1}), (i:Item {id: 3})
```

```
CREATE (c)-[:VIEWED]->(i);
```

```
MATCH (c:Customer {id: 2}), (o:Order {id: 2})
```

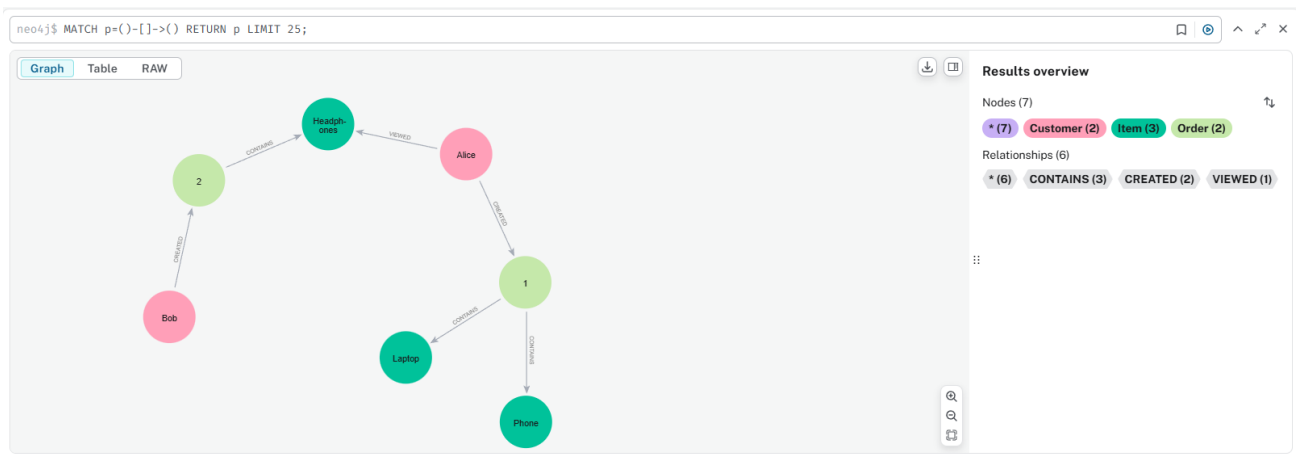
```
CREATE (c)-[:CREATED]->(o);
```

```
MATCH (o:Order {id: 2}), (i:Item {id: 3})
```

```
CREATE (o)-[:CONTAINS]->(i);
```

The image shows a Neo4j Cypher console interface. At the top, there is a command input field containing the text: `neo4j$ MATCH (c:Customer {id: 1}), (o:Order {id: 1}) CRE`. Below the input field, there is a list of six executed queries, each preceded by a green checkmark icon. The queries are: `MATCH (c:Customer {id: 1}), (o:Order {id: 1})`, `MATCH (o:Order {id: 1}), (i:Item {id: 1})`, `MATCH (o:Order {id: 1}), (i:Item {id: 2})`, `MATCH (c:Customer {id: 1}), (i:Item {id: 3})`, `MATCH (c:Customer {id: 2}), (o:Order {id: 2})`, and `MATCH (o:Order {id: 2}), (i:Item {id: 3})`. Each query entry has a dropdown arrow on the right side.

Status	Query	Action
✓	<code>MATCH (c:Customer {id: 1}), (o:Order {id: 1})</code>	▼
✓	<code>MATCH (o:Order {id: 1}), (i:Item {id: 1})</code>	▼
✓	<code>MATCH (o:Order {id: 1}), (i:Item {id: 2})</code>	▼
✓	<code>MATCH (c:Customer {id: 1}), (i:Item {id: 3})</code>	▼
✓	<code>MATCH (c:Customer {id: 2}), (o:Order {id: 2})</code>	▼
✓	<code>MATCH (o:Order {id: 2}), (i:Item {id: 3})</code>	▼



Написати наступні види запитів:

- Знайти Items які входять в конкретний Order

```
MATCH (o:Order {id: 1})-[:CONTAINS]->(i:Item)
```

```
RETURN i;
```



- Підрахувати вартість конкретного Order

```
MATCH (o:Order {id: 1})-[:CONTAINS]->(i:Item)
```

```
RETURN o.id AS OrderId, SUM(i.price) AS TotalPrice;
```

neo4j\$ MATCH (o:Order {id: 1})-[:CONTAINS]->(i:Item) RET

Table RAW

OrderId	TotalPrice
1	1500

Started streaming 1 record after 51 ms and completed after 58 ms.

- Знайти всі Orders конкретного Customer

```
MATCH (c:Customer {id: 1})-[:CREATED]->(o:Order)
```

```
RETURN o;
```

neo4j\$ MATCH (c:Customer {id: 1})-[:CREATED]->(o:Order)

Graph Table RAW

o

(:Order {id: 1})

Started streaming 1 record after 43 ms and completed after 45 ms.

- Знайти всі Items куплені конкретним Customer (через Order)

```
MATCH (c:Customer {id: 1})-[:CREATED]->(o:Order)-[:CONTAINS]->(i:Item)
```

```
RETURN i;
```

neo4j\$ MATCH (c:Customer {id: 1})-[:CREATED]->(:Order)-[

Graph Table RAW

i

¹ (:Item {price: 1000, name: "Laptop", id: 1})

² (:Item {price: 500, name: "Phone", id: 2})

Started streaming 2 records after 53 ms and completed after 55 ms.

- Знайти кількість Items куплені конкретним Customer (через Order)

```
MATCH (c:Customer {id: 1})-[:CREATED]->(:Order)-[:CONTAINS]->(i:Item)
```

```
RETURN COUNT(i) AS TotalItems;
```

neo4j\$ MATCH (c:Customer {id: 1})-[:CREATED]->(:Order)-[

Table RAW

TotalItems

¹ 2

Started streaming 1 record after 54 ms and completed after 56 ms.

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

```
MATCH (c:Customer {id: 1})-[:CREATED]->(:Order)-[:CONTAINS]->(i:Item)
```

```
RETURN SUM(i.price) AS TotalSpent;
```

neo4j\$ MATCH (c:Customer {id: 1})-[:CREATED]->(:Order)-[

Table RAW

TotalSpent

1 1500

Started streaming 1 record after 60 ms and completed after 61 ms.

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

MATCH (:Order)-[:CONTAINS]->(i:Item) RETURN i.name AS ItemName, COUNT(*) AS PurchaseCount ORDER BY PurchaseCount DESC;

neo4j\$ MATCH (:Order)-[:CONTAINS]->(i:Item) RETURN i.name

Table RAW

	ItemName	PurchaseCount
1	"Laptop"	1
2	"Phone"	1
3	"Headphones"	1

Started streaming 3 records after 79 ms and completed after 82 ms.

- Знайти всі Items переглянуті (view) конкретним Customer

MATCH (c:Customer {id: 1})-[:VIEWED]->(i:Item)

RETURN i;

neo4j\$ MATCH (c:Customer {id: 1})-[:VIEWED]->(i:Item) RE

Graph Table RAW

i

```
{(:Item {price: 200, name: "Headphones", id: 3})}
```

Started streaming 1 record after 40 ms and completed after 41 ms.

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

```
MATCH (:Item {id: 1})<-[:CONTAINS]-(o:Order)-[:CONTAINS]->(i:Item)
```

```
WHERE i.id <> 1
```

```
RETURN i;
```

neo4j\$ MATCH (:Item {id: 1})<-[:CONTAINS]-(o:Order)-[:CO

Graph Table RAW

i

```
{(:Item {price: 500, name: "Phone", id: 2})}
```

Started streaming 1 record after 62 ms and completed after 63 ms.

- Знайти Customers які купили даний конкретний Item

```
MATCH (c:Customer)-[:CREATED]->(o:Order)-[:CONTAINS]->(i:Item {id: 1})
```

```
RETURN c;
```

The screenshot shows the Neo4j Cypher query editor. The query is: `neo4j$ MATCH (c:Customer)-[:CREATED]->(o:Order)-[:CONTAINS]->(i:Item) WHERE c.name = 'Alice' RETURN i`. The interface has tabs for Graph, Table, and RAW. The Table tab is selected, showing a single record: `(:Customer {name: "Alice", id: 1})`. A status message at the bottom says: "Started streaming 1 record after 80 ms and completed after 82 ms."

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

```
MATCH (c:Customer {id: 1})-[:VIEWED]->(i:Item)
```

```
WHERE NOT EXISTS {
```

```
  MATCH (c)-[:CREATED]->(o:Order)-[:CONTAINS]->(i)
```

```
}
```

```
RETURN i;
```

The screenshot shows the Neo4j Cypher query editor. The query is: `neo4j$ MATCH (c:Customer {id: 1})-[:VIEWED]->(i:Item) WHERE NOT EXISTS { MATCH (c)-[:CREATED]->(o:Order)-[:CONTAINS]->(i) } RETURN i`. The interface has tabs for Graph, Table, and RAW. The Table tab is selected, showing a single record: `(:Item {price: 200, name: "Headphones", id: 3})`. A status message at the bottom says: "Started streaming 1 record after 146 ms and completed after 148 ms."

1. Як і в попередніх завданнях, для якогось одного обраного Item додайте поле з кількістю його лайків.
 - З 10 окремих клієнтів одночасно запустити інкрементацію каунтеру лайків по 10_000 на кожного клієнта
 - зробіть так щоб не було втрат та перевірте щоб фінальне значення було 100_000
 - заміряйте час роботи


```
neo4j$ MATCH (i:Item {id: 1}) SET i.likes = 0;
```

✓ Set 1 property

```
neo4j$ MATCH (i:Item {id: 1}) RETURN i.likes;
```

Table

RAW

i.likes

1 0

```
(kali@kali)-[~/Desktop/pvs]
$ /bin/python3 /home/kali/Desktop/pvs/lab3.py
Total likes: 0
Total likes: 100000
time: 2.6375820636749268 seconds
```

```
neo4j$ MATCH (i:Item {id: 1}) RETURN i.likes;
```

Table RAW

i.likes

1 100000

Started streaming 1 record after 3 ms and completed after 4 ms.