

Завдання:

Змодельювати 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

```
neo4j$ CREATE (:Customer {id: 3, name: 'Sam'})
```

Added 1 label, created 1 node, set 2 properties, completed after 9 ms.

```
neo4j$ CREATE (:Customer {id: 2, name: 'Alex'})
```

Added 1 label, created 1 node, set 2 properties, completed after 5 ms.

```
neo4j$ CREATE (:Customer {id: 1, name: 'John'})
```

Added 1 label, created 1 node, set 2 properties, completed after 2 ms.

```
neo4j$ CREATE (:Order {id: 3})
```

Added 1 label, created 1 node, set 1 property, completed after 8 ms.

```
neo4j$ CREATE (:Order {id: 2})
```

Added 1 label, created 1 node, set 1 property, completed after 7 ms.

```
neo4j$ CREATE (:Order {id: 1})
```

Added 1 label, created 1 node, set 1 property, completed after 17 ms.

```
neo4j$ CREATE (:Item {id: 1, name: 'Keyboard', price: 140}) CREATE (:Item {id: 2, name: 'Mouse', price: 100}) CREATE (:Item {id: 3, name: 'Monitor', price: 300})
```

Added 3 labels, created 3 nodes, set 9 properties, completed after 42 ms.

```
neo4j$ MATCH (o:Order {id: 3}), (i:Item {id: 2}) CREATE (o)-[:CONTAINS]→(i);
```

Created 1 relationship, completed after 15 ms.

```
neo4j$ MATCH (o:Order {id: 2}), (i:Item {id: 3}) CREATE (o)-[:CONTAINS]→(i);
```

Created 1 relationship, completed after 11 ms.

```
neo4j$ MATCH (o:Order {id: 1}), (i:Item {id: 2}) CREATE (o)-[:CONTAINS]→(i);
```

Created 1 relationship, completed after 18 ms.

```
neo4j$ MATCH (o:Order {id: 1}), (i:Item {id: 1}) CREATE (o)-[:CONTAINS]→(i);
```

Created 1 relationship, completed after 25 ms.

```
neo4j$ MATCH (c:Customer {id: 3}), (o:Order {id: 3}) CREATE (c)-[:PLACED]→(o);
```

Created 1 relationship, completed after 18 ms.

```
neo4j$ MATCH (c:Customer {id: 2}), (o:Order {id: 2}) CREATE (c)-[:PLACED]→(o);
```

Created 1 relationship, completed after 18 ms.

```
neo4j$ MATCH (c:Customer {id: 1}), (o:Order {id: 1}) CREATE (c)-[:PLACED]→(o);
```

Created 1 relationship, completed after 47 ms.

```
neo4j$ MATCH (c:Customer {id: 3}), (i:Item {id: 2}) CREATE (c)-[:VIEWED]→(i);
```

Created 1 relationship, completed after 19 ms.

```
neo4j$ MATCH (c:Customer {id: 2}), (i:Item {id: 1}) CREATE (c)-[:VIEWED]→(i);
```

Created 1 relationship, completed after 28 ms.

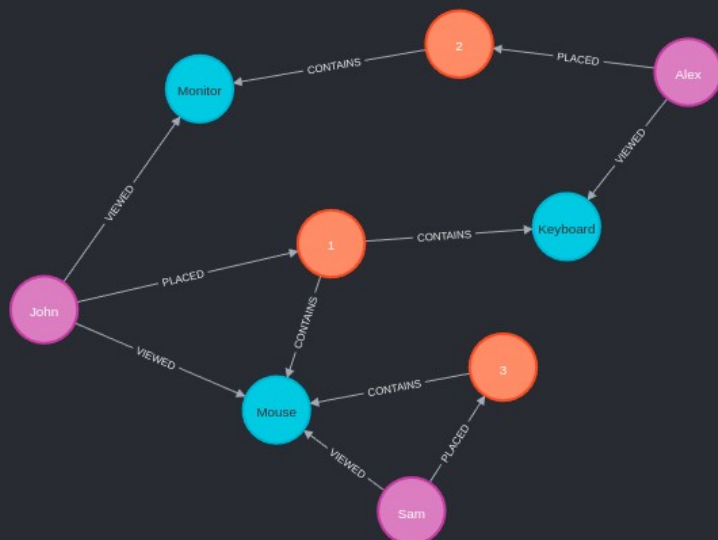
```
neo4j$ MATCH (c:Customer {id: 1}), (i:Item {id: 2}) CREATE (c)-[:VIEWED]→(i);
```

Created 1 relationship, completed after 22 ms.

```
neo4j$ MATCH (c:Customer {id: 1}), (i:Item {id: 3}) CREATE (c)-[:VIEWED]→(i);
```

Created 1 relationship, completed after 20 ms.

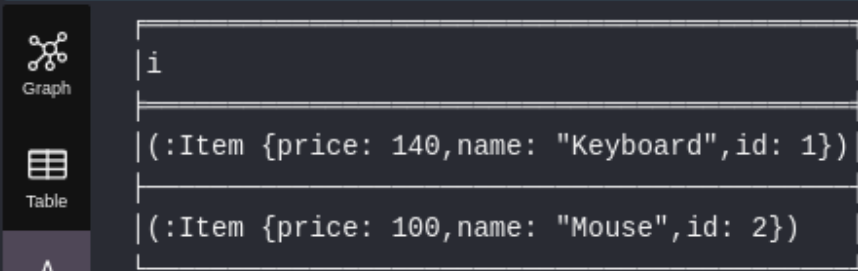
```
neo4j$ MATCH (n)-[r]→(m) RETURN n, r, m;
```



1. Написати наступні види запитів:

- Знайти Items які входять в конкретний Order

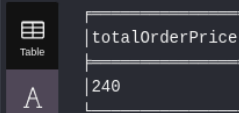
```
neo4j$ MATCH (o:Order {id: 1})-[:CONTAINS]→(i:Item) RETURN i;
```



i
{:Item {price: 140,name: "Keyboard",id: 1}}
{:Item {price: 100,name: "Mouse",id: 2}}

- Підрахувати вартість конкретного Order

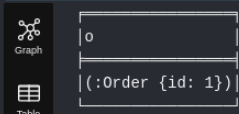
```
neo4j$ MATCH (o:Order {id: 1})-[:CONTAINS]→(i:Item) RETURN SUM(i.price) AS totalOrderPrice;
```



totalOrderPrice
240

- Знайти всі Orders конкретного Customer

```
neo4j$ MATCH (c:Customer {id: 1})-[:PLACED]→(o:Order) RETURN o;
```



o
{:Order {id: 1}}

- Знайти всі Items куплені конкретним Customer (через Order)

```
neo4j$ MATCH (o:Order {id: 1})-[:CONTAINS]→(i:Item) RETURN i;
```

Graph
Table
Text
Code

i
(:Item {price: 140,name: "Keyboard",id: 1})
(:Item {price: 100,name: "Mouse",id: 2})

- Знайти кількість Items куплені конкретним Customer (через Order)

```
neo4j$ MATCH (c:Customer {id: 1})-[:PLACED]→(o:Order)-[:CONTAINS]→(i:Item) RETURN COUNT(i) AS totalItemsBought;
```

Table
Text
Code

totalItemsBought
2

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

```
neo4j$ MATCH (c:Customer {id: 1})-[:PLACED]→(o:Order)-[:CONTAINS]→(i:Item) RETURN SUM(i.price) AS totalSpentByCustomer;
```

Table
Text
Code

totalSpentByCustomer
240

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

```
neo4j$ MATCH (o:Order)-[:CONTAINS]→(i:Item) RETURN i.name AS itemName, COUNT(o) AS timesBought ORDER BY timesBought DESC;
```

Table
Text
Code

itemName	timesBought
"Mouse"	2
"Keyboard"	1
"Monitor"	1

- Знайти всі Items переглянуті (view) конкретним Customer

```
neo4j$ MATCH (c:Customer {id: 1})-[:VIEWED]->(i:Item) RETURN i;
```

i
(:Item {price: 100,name: "Mouse",id: 2})
(:Item {price: 300,name: "Monitor",id: 3})

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

```
neo4j$ MATCH (i:Item {id: 1})<-[:CONTAINS]-(o:Order)-[:CONTAINS]->(other:Item) WHERE other.id <> i.id RETURN DISTINCT other;
```

other
(:Item {price: 100,name: "Mouse",id: 2})

- Знайти Customers які купили даний конкретний Item

```
neo4j$ MATCH (c:Customer)-[:PLACED]->(o:Order)-[:CONTAINS]->(i:Item {id: 1}) RETURN DISTINCT c;
```

c
(:Customer {name: "John",id: 1})

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

```
neo4j$ MATCH (c:Customer {id: 1})-[:VIEWED]->(i:Item) WHERE NOT EXISTS { MATCH (c)-[:PLACED]->(o:Order)-[:CONTAINS]->(i) } RETURN i;
```

i
(:Item {price: 300,name: "Monitor",id: 3})

2. 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

```
04j$ MATCH (i:Item {id: 1}) SET i.likes = 0 RETURN i;
```

```
{
  "i": {
    "price": 140,
    "name": "Keyboard",
    "id": 1,
    "likes": 0
  }
}
```

```
/usr/bin/python3.12 /home/perry/kp11m/pvs/lab3/main.py
```

```
Фінальне значення likes для Item 1: 100000
```

```
Час виконання: 167.90 секунд
```