6th Lab: Graph Databases (Neo4J)

Júlia Gasull Navarro
Èric Pérez Ibáñez

Design

We made the design keeping in mind the number of nodes we are working with at any given time, without worrying too much about the joins, because in the graph database they are not expensive.

Then, we considered the following points with the nodes of each query:

- For **query 2**, we have kept the nodes of regions that work well to avoid comparisons with query 4.

- For **query 3**, in order to make the select less expensive, we added a MktSegment node.

- For **query 4**, we have kept the nation nodes because they are useful for the restriction on nations it has.

Finally, we also added the part-supplier information in an edge. In this way, we make the queries as selective as possible.

Queries

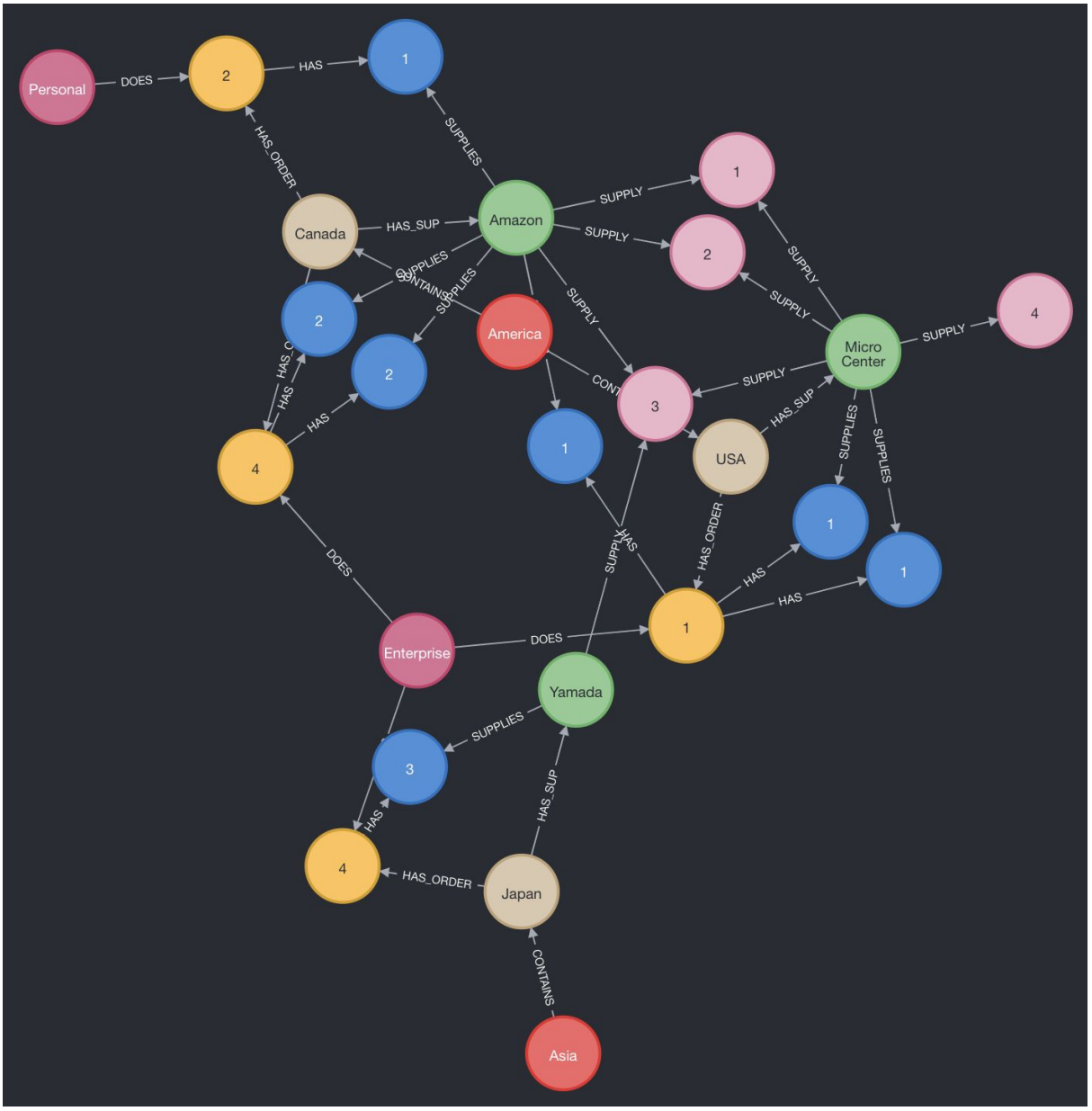
Init

```
// Regions.
CREATE ( America: Region { r_name: "America" } )
CREATE ( Asia: Region    { r_name: "Asia" } )

// Nations.
CREATE ( USA: Nation     { n_name: "USA" } )        <-[:CONTAINS]- ( America )
CREATE ( Canada: Nation  { n_name: "Canada" } )     <-[:CONTAINS]- ( America )
CREATE ( Japan: Nation   { n_name: "Japan" } )      <-[:CONTAINS]- ( Asia )

// Suppliers.
CREATE ( MicroCenter: Supplier
{
       s_acctbal: 40,
       s_name: "Micro Center",
       s_address: "742 Evergreen Terrace",
       s_phone: "123 456 789", s_comment: "American supplier"
} )    <-[:HAS_SUP]- ( USA )
CREATE ( Amazon: Supplier
{
       s_acctbal: 50,
       s_name: "Amazon",
       s_address: "963  Pine Garden Lane",
       s_phone: "987 654 321",
       s_comment: "Canadiense supplier"
} )    <-[:HAS_SUP]- ( Canada )
CREATE ( Yamada: Supplier
{
       s_acctbal: 60,
       s_name: "Yamada",
       s_address: "3286  Coulter Lane",
       s_phone: "842 753 375",
       s_comment: "Japanses supplier"
} )    <-[:HAS_SUP]- ( Japan )

// Parts.
CREATE ( ThinkpadT490: Part      { p_partKey: 1, p_mfgr: "A", p_size: 1,     p_type: "Laptop" } )
CREATE ( ThinkpadL13Yoga: Part   { p_partKey: 2, p_mfgr: "B", p_size: 123,   p_type: "Laptop for home" } )
CREATE ( Zenbook14: Part         { p_partKey: 3, p_mfgr: "C", p_size: 1,     p_type: "Standard Laptop" } )
CREATE ( ZenbookFlipS: Part      { p_partKey: 4, p_mfgr: "D", p_size: 1,     p_type: "Laptop" } )

// Line Items.
// Line Items Order 1.
CREATE ( MicroCenterThinkpadT490_Order_1: LineItem
{
        l_returnflag: "1",
        l_linestatus: "1",
        l_quantity: 5,
        l_extendedprice: 5,
        l_discount: 0.6,
        l_tax: 0.21,
        l_shipdate: 20190605
} )
CREATE ( MicroCenterThinkpadL13Yoga_Order_1: LineItem
{
        l_returnflag: "1",
        l_linestatus: "1",
        l_quantity: 7,
        l_extendedprice: 7,
        l_discount: 0.7,
        l_tax: 0.21,
        l_shipdate: 20190602
} )
CREATE ( AmazonZenbook14_Order_1: LineItem
{
        l_returnflag: "1",
        l_linestatus: "2",
        l_quantity: 2,
```

```
        l_extendedprice: 2,
        l_discount: 0.8,
        l_tax: 0.21,
        l_shipdate: 20190603
} )
// Line Items Order 2.
CREATE ( AmazonZenbookFlipS_Order_2: LineItem
{
        l_returnflag: "1",
        l_linestatus: "2",
        l_quantity: 5,
        l_extendedprice: 5,
        l_discount: 0.9,
        l_tax: 0.21,
        l_shipdate: 20190604
} )
// Line Items Order 3.
CREATE ( AmazonThinkpadT490_Order_3: LineItem
{
        l_returnflag: "2",
        l_linestatus: "2",
        l_quantity: 7,
        l_extendedprice: 7,
        l_discount: 0.1,
        l_tax: 0.21,
        l_shipdate: 20190605
} )
CREATE ( AmazonThinkpadL13Yoga_Order_3: LineItem
{
        l_returnflag: "2",
        l_linestatus: "2",
        l_quantity: 2,
        l_extendedprice: 2,
        l_discount: 0.5,
        l_tax: 0.21,
        l_shipdate: 20190606
 } )
// Line Items Order 4.
CREATE ( YamadaZenbook14_Order_4: LineItem
{
        l_returnflag: "3",
        l_linestatus: "3",
        l_quantity: 4,
        l_extendedprice: 4,
        l_discount: 0.4,
        l_tax: 0.21,
        l_shipdate: 20190706
} )

// Orders.
CREATE ( Order_1: Order { o_orderkey: 1, o_orderdate: 20190501, o_shippriority: 1 } ) <-[:HAS_ORDER]- ( USA )
CREATE ( Order_2: Order { o_orderkey: 2, o_orderdate: 20190502, o_shippriority: 2 } ) <-[:HAS_ORDER]- ( Canada )
CREATE ( Order_3: Order { o_orderkey: 3, o_orderdate: 20191201, o_shippriority: 4 } ) <-[:HAS_ORDER]- ( Canada )
CREATE ( Order_4: Order { o_orderkey: 4, o_orderdate: 20190501, o_shippriority: 4 } ) <-[:HAS_ORDER]- ( Japan )

// Marketing Segments.
CREATE ( Enterprise: MktSegment   { m_name: "Enterprise" } )
CREATE ( Personal: MktSegment     { m_name: "Personal" } )

// Suppliers -SUPPLY-> Parts relations.
CREATE ( MicroCenter )   -[:SUPPLY { ps_supplycost: 1 }]-> ( ThinkpadT490 )
CREATE ( MicroCenter )   -[:SUPPLY { ps_supplycost: 1 }]-> ( ThinkpadL13Yoga )
CREATE ( MicroCenter )   -[:SUPPLY { ps_supplycost: 1 }]-> ( Zenbook14 )
CREATE ( MicroCenter )   -[:SUPPLY { ps_supplycost: 1 }]-> ( ZenbookFlipS )
CREATE ( Amazon )        -[:SUPPLY { ps_supplycost: 1 }]-> ( Zenbook14 )
CREATE ( Amazon )        -[:SUPPLY { ps_supplycost: 1 }]-> ( ThinkpadT490 )
CREATE ( Amazon )        -[:SUPPLY { ps_supplycost: 1 }]-> ( ThinkpadL13Yoga )
CREATE ( Yamada )        -[:SUPPLY { ps_supplycost: 1 }]-> ( Zenbook14 )

// Suppliers -SUPPLIES-> LineItem relations.
CREATE ( MicroCenter )   -[:SUPPLIES]-> ( MicroCenterThinkpadT490_Order_1 )
CREATE ( MicroCenter )   -[:SUPPLIES]-> ( MicroCenterThinkpadL13Yoga_Order_1 )
CREATE ( Amazon )        -[:SUPPLIES]-> ( AmazonZenbook14_Order_1 )
CREATE ( Amazon )        -[:SUPPLIES]-> ( AmazonZenbookFlipS_Order_2 )
CREATE ( Amazon )        -[:SUPPLIES]-> ( AmazonThinkpadT490_Order_3 )
CREATE ( Amazon )        -[:SUPPLIES]-> ( AmazonThinkpadL13Yoga_Order_3 )
```

```
CREATE ( Yamada )        -[:SUPPLIES]-> ( YamadaZenbook14_Order_4 )

// Orders -HAS-> LineItem relations.
CREATE ( Order_1 ) -[:HAS]-> ( MicroCenterThinkpadT490_Order_1 )
CREATE ( Order_1 ) -[:HAS]-> ( MicroCenterThinkpadL13Yoga_Order_1 )
CREATE ( Order_1 ) -[:HAS]-> ( AmazonZenbook14_Order_1 )
CREATE ( Order_2 ) -[:HAS]-> ( AmazonZenbookFlipS_Order_2 )
CREATE ( Order_3 ) -[:HAS]-> ( AmazonThinkpadT490_Order_3 )
CREATE ( Order_3 ) -[:HAS]-> ( AmazonThinkpadL13Yoga_Order_3 )
CREATE ( Order_4 ) -[:HAS]-> ( YamadaZenbook14_Order_4 )

// Marketing Segments -DOES-> Orders relations.
CREATE ( Enterprise ) -[:DOES]-> ( Order_1 )
CREATE ( Personal )   -[:DOES]-> ( Order_2 )
CREATE ( Enterprise ) -[:DOES]-> ( Order_3 )
CREATE ( Enterprise ) -[:DOES]-> ( Order_4 )
```

After initializing the system, the resulting graph is as follows:



Query 1

This first query is the simplest compared to the others, where we basically have to specify the aggregation functions within the additional WITH clause so that it does the grouping we want.

Code

```
MATCH
        ( li:LineItem )
WHERE
        li.l_shipdate <= 20190606
WITH
        li.l_returnflag AS l_returnflag,
        li.l_linestatus AS l_linestatus,
        SUM( li.l_quantity ) AS sum_qty,
        SUM( li.l_extendedprice ) AS sum_base_price,
        SUM( li.l_extendedprice * ( 1 - li.l_discount ) ) AS sum_disc_price,
        SUM( li.l_extendedprice * ( 1 - li.l_discount ) * ( 1 + li.l_tax ) ) AS sum_charge,
        AVG( li.l_quantity ) AS avg_qty,
        AVG( li.l_extendedprice ) AS avg_price,
        AVG( li.l_discount ) AS avg_disc,
        COUNT( * ) AS count_order
RETURN l_returnflag, l_linestatus, sum_qty, sum_base_price, sum_disc_price, sum_charge, avg_qty, avg_price,
avg_disc, count_order
ORDER BY l_returnflag, l_linestatus;
```

Result

	l_returnflag	l_linestatus	sum_qty	sum_base_price	sum_disc_price	sum_charge	avg_qty	avg_price	avg_disc	count_order
1	"1"	"1"	12	12	4.1000000000000005	4.961	6.0	6.0	0.6499999999999999	2
2	"1"	"2"	7	7	0.8999999999999998	1.0889999999999997	3.5	3.5	0.8500000000000001	2
3	"2"	"2"	9	9	7.3	8.832999999999998	4.5	4.5	0.30000000000000004	2

Query 2

What we could highlight from this second query, would be the fact that in Neo4j, a regular expression can be put in the WHERE clause thanks to the operand p.p_type = ~ ". * Laptop $"

Also, we can emphasize that we try to make the query more optimal using the first MATCH to filter the REGION that interests us.

Then, the concept of multiple MATCH-WHERE clauses nested is used in the same query.

Code

```
MATCH
	(p:Part { p_size : 1 }) <-[ps:SUPPLY]- (:Supplier) <-[:HAS_SUP]- (n) <-[:CONTAINS]- (r:Region{r_name: "America"})
WHERE
	p.p_type =~ ".*Laptop$"
WITH
	p,
	MIN( ps.ps_supplycost ) AS minps
MATCH
	(p)<-[ps:SUPPLY{ps_supplycost:minps}]-(s:Supplier)<-[:HAS_SUP]-(n)<-[:CONTAINS]-(r:Region{r_name:"America"})
RETURN s.s_acctbal, s.s_name, n.n_name, p.p_partKey, p.p_mfgr, s.s_address, s.s_phone, s.s_comment
ORDER BY s.s_acctbal DESC, n.n_name, s.s_name, p.p_partkey;
```

Result

	s.s_acctbal	s.s_name	n.n_name	p.p_partKey	p.p_mfgr	s.s_address	s.s_phone	s.s_comment
1	50	"Amazon"	"Canada"	3	"C"	"963 Pine Garden Lane"	"987 654 321"	"Canadiense supplier"
2	50	"Amazon"	"Canada"	1	"A"	"963 Pine Garden Lane"	"987 654 321"	"Canadiense supplier"
3	40	"Micro Center"	"USA"	3	"C"	"742 Evergreen Terrace"	"123 456 789"	"American supplier"
4	40	"Micro Center"	"USA"	1	"A"	"742 Evergreen Terrace"	"123 456 789"	"American supplier"
5	40	"Micro Center"	"USA"	4	"D"	"742 Evergreen Terrace"	"123 456 789"	"American supplier"

- p_partKey 2: Stayed out by p_type = "Laptop for home"
- YamadaZenbook14_Order_4 is left out because the relationship Suppliers -SUPPLY-> Parts is with the Supplier "Yamada", which is from the Region! = "America"

Query 3

In query 3, we start with the MATCH clause to access the MktSegment nodes. Therefore, the number of comparisons we will have will be low because we expect very few MktSegment. Then we get the Orders and filter them by o_orderdate in the WHERE clause. Then, we get the LineItems and filter them by l_shipdate in the following MATCH and WHERE. Finally, we do the grouping in the WITH clause and return the ordered values.

Code

```
MATCH
        ( m:MktSegment { m_name:"Enterprise" } ) - [:DOES] -> ( o:Order )
WHERE
        o.o_orderdate < 20190509
MATCH
        (o) - [:HAS] -> ( li:LineItem )
WHERE
        li.l_shipdate > 20190600
WITH
        o,
        SUM( li.l_extendedprice * ( 1 - li.l_discount ) ) AS revenue
RETURN
        o.o_orderkey,
        revenue,
        o.o_orderdate,
        o.o_shippriority
ORDER BY
        revenue DESC,
        o.o_orderdate;
```

Result

	o.o_orderkey	revenue	o.o_orderdate	o.o_shippriority
1	1	4.5	20190501	1
2	4	2.4	20190501	4

As we can see in the image above, some orders are within the selection and others are not:

- o_orderkey 1: Inside
- o_orderkey 2: Stayed out by c_mktsegment = "Personal"
- o_orderkey 3: Stayed out by o_orderdate = 20191201
- o_orderkey 4: Inside

Query 4

In this query, we start with a MATCH clause to enter by Region and get the Nation from it. Then, for each Nation, we get the Order.

Then, in the WHERE clause we filter the Orders by o_orderdate. In the following MATCH, we obtain the LineItems of the Order that we have filtered and from these we obtain the Supplier.

Finally, we do the grouping in the WITH clause and return the ordered results.

Code

```
MATCH
        ( r:Region { r_name:"America" } ) -[:CONTAINS]-> ( n:Nation ) -[:HAS_ORDER]-> ( o:Order )
WHERE
        o.o_orderdate >= 20190500
        AND o.o_orderdate <  20190566
MATCH
        (o) -[:HAS]-> ( l:LineItem ) <-[:SUPPLIES]- ( s:Supplier ) <-[:HAS_SUP]- ( n )
WITH
        n,
        SUM( l.l_extendedprice * ( 1 - l.l_discount ) ) AS revenue
RETURN
        n.n_name,
        revenue
ORDER BY
        revenue DESC;
```

Result

	n.n_name	revenue
1	"USA"	4.1000000000000005
2	"Canada"	0.4999999999999999

As we can see in the image above, some orders are within the selection and others are not:

- o_orderkey 1: Inside
- o_orderkey 2: Inside
- o_orderkey 3: Stayed out by + o_orderdate + = 1999999999991
- o_orderkey 4: Stayed out by + r_name + (+ Order <- [: HAS_ORDER] - Region +)! = "Europe"