GAI- TP4 – Neo4J

Adrián José Zapater Reig

Ejercicio 1 – Crea una base de datos con los datos que se adjuntan en el archivo grafo.txt disponible en el curso virtual.

El fichero grafo.txt contiene una serie de instrucciones en *Cypher*. Se ha añadido un ';' al final de la primera y última línea para poder ejecutarlo desde la cypher-shell.

Tabla 1 – grafo.txt

```
match(n) detach delete(n);

CREATE (Bertoldo:Cliente {nombre:'Bertoldo', edad:33})
CREATE (Herminia:Cliente {nombre:'Herminia', edad:34})
CREATE (Aniceto:Cliente {nombre:'Aniceto', edad:49})
CREATE (Calixta:Cliente {nombre:'Calixta', edad:19})
CREATE (Melibeo:Cliente {nombre:'Melibeo', edad:43})
CREATE (Genara:Cliente {nombre:'Genara', edad:46})
CREATE (Amadea:Cliente {nombre:'Amadea', edad:33})
CREATE (Clotilde:Cliente {nombre:'Clotilde', edad:19})
CREATE (Tiburcio:Cliente {nombre:'Tiburcio', edad:24})
CREATE (Demetrio:Cliente {nombre:'Demetrio', edad:18})
CREATE (Nicodemo:Cliente {nombre:'Nicodemo', edad:51})
CREATE (TV:Producto {id:1,descr:'TV 3D, 200 pulgadas', precio:4000})
CREATE (Maleta:Producto {id:2,descr:'Maleta inteligente, lava la ropa', precio:370})
CREATE (Tostadora:Producto {id:3,descr:'Tostadora 5G, unta la mantequilla sola', precio:230})
CREATE (Clotilde)-[:BAD {id:1}]->(TV),
    (Clotilde)-[:BAD {id:2}]->(Maleta),
    (Clotilde)-[:BAD {id:3}]->(Tostadora),
    (Clotilde)-[:APOYA {id:4}]->(Melibeo),
    (Clotilde)-[:APOYA {id:5}]->(Melibeo),
    (Bertoldo)-[:BAD {id:6}]->(TV),
    (Bertoldo)-[:BAD {id:7}]->(Maleta),
    (Nicodemo)-[:BAD {id:8}]->(TV),
    (Nicodemo)-[:GOOD {id:9}]->(Maleta),
    (Nicodemo)-[:BAD  {id:10}]->(Tostadora),
    (Demetrio)-[:GOOD {id:11}]->(TV),
    (Demetrio)-[:APOYA {id:12}]->(Tiburcio),
    (Demetrio)-[:BAD {id:13}]->(Tostadora),
    (Tiburcio)-[:GOOD {id:14}]->(TV),
    (Tiburcio)-[:GOOD {id:15}]->(Maleta),
    (Tiburcio)-[:BAD {id:16}]->(Tostadora),
    (Amadea)-[:BAD {id:17}] ->(TV),
    (Amadea)-[:APOYA {id:18}] ->(Clotilde),
```

```
(Amadea)-[:BAD {id:19}]->(Tostadora),
(Genara)-[:BAD {id:20}]->(TV),
(Genara)-[:APOYA {id:21}]->(Amadea),
(Genara)-[:BAD {id:22}]->(Maleta),
(Genara)-[:BAD {id:23}]->(Tostadora),
(Melibeo)-[:BAD {id:24}]->(TV),
(Melibeo)-[:APOYA {id:25}]->(Genara),
(Melibeo)-[:BAD {id:26}]->(Tostadora),
(Calixta)-[:GOOD {id:27}]->(TV),
(Calixta)-[:APOYA {id:28}]->(Genara),
(Calixta)-[:BAD {id:29}]->(Tostadora),
(Aniceto)-[:GOOD {id:30}]->(TV),
(Aniceto)-[:APOYA {id:31}]->(Genara),
(Aniceto)-[:BAD {id:32}]->(Tostadora),
(Aniceto)-[:GOOD {id:33}]->(Maleta);
```

En primer lugar, levantamos la BD en un contenedor de docker. Creamos 3 volúmenes: la carpeta *neo4j-data* para que Neo4J persista la información de la BD, *neo4j-logs* para los logs de ejecución y *neo4j-src* para la carpeta /tmp donde depositaremos el fichero de inicialización.

Tabla 2 – Comando de docker para levantar la BD.

```
docker run --name neo4j -d \
--publish=7474:7474 --publish=7687:7687 \
--volume=neo4j-logs:/logs \
--volume=neo4j-data:/data \
--volume=neo4j-tmp:/tmp/src \
neo4j:latest
```

Tabla 3 – Comando de copiado

```
docker cp src/grafo.txt neo4j:/tmp/src/grafo.txt
```

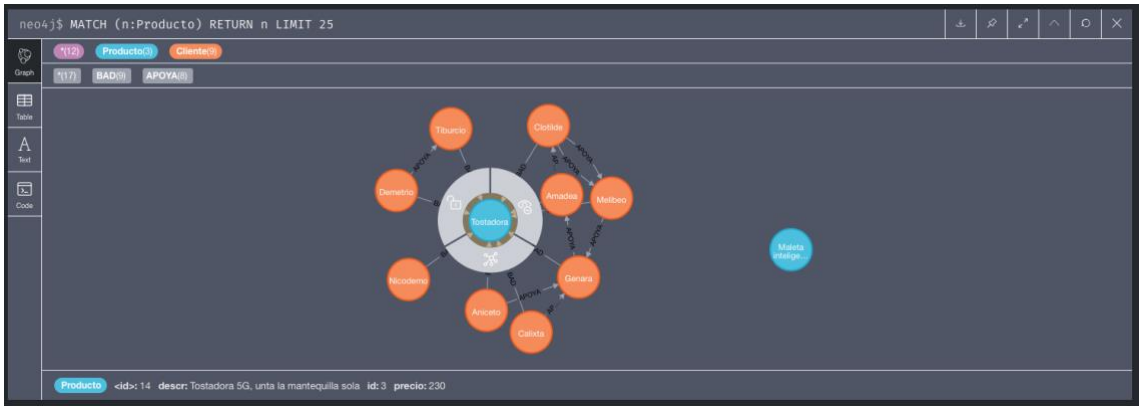
Tabla 4 – Comando de ejecución de fichero grafo.txt

```
docker exec -it neo4j cypher-shell -u neo4j -p uned1 -f /tmp/src/grafo.txt
```

Tras la ejecución del fichero grafo.txt la cypher-shell nos devuelve:

```
0 rows available after 3 ms, consumed after another 0 ms
Deleted 14 nodes, Deleted 33 relationships
0 rows available after 95 ms, consumed after another 0 ms
Added 14 nodes, Created 33 relationships, Set 64 properties, Added 14 labels
```

Y en la herramienta Neo4J browser (http://localhost:7474) se puede comprobar que se han introducido los nodos correctamente:

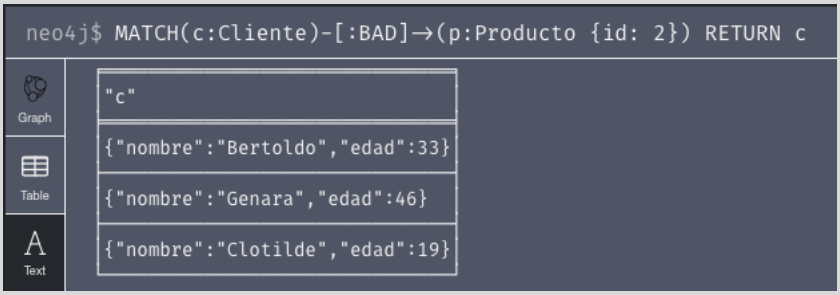


Ejercicio 2 - Sobre dicha base de datos, implementa consultas para conseguir la siguiente información:

2.1. (2 puntos) Queremos conocer nodos tipo Cliente que hayan emitido una opinión tipo BAD del producto con id=2 (hay 3 clientes con estas características).

Tabla 5 – Solución 2.1

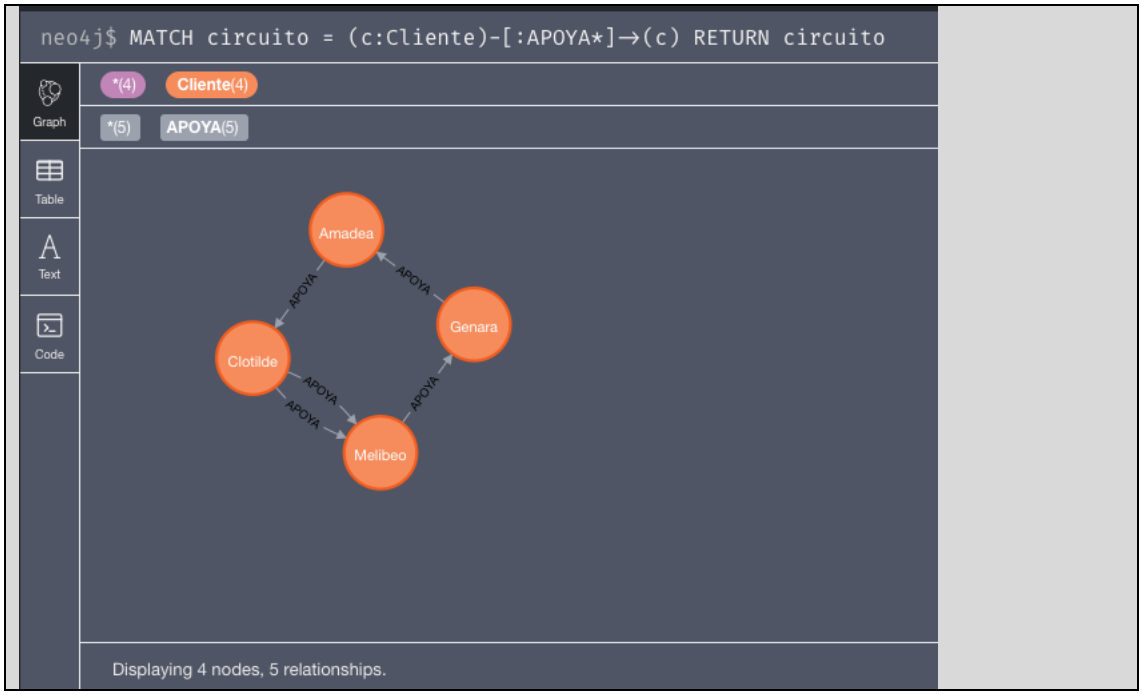
```
MATCH(c:Cliente)-[:BAD]->(p:Producto {id: 2})
RETURN c;
```



2.2. (2 puntos) Queremos buscar clientes que formen un “circuito” de relaciones de tipo “APOYA”, es decir una secuencia de relaciones, todas en la misma dirección, que acaba y empieza en el mismo Cliente.

Tabla 6 – Solución 2.2

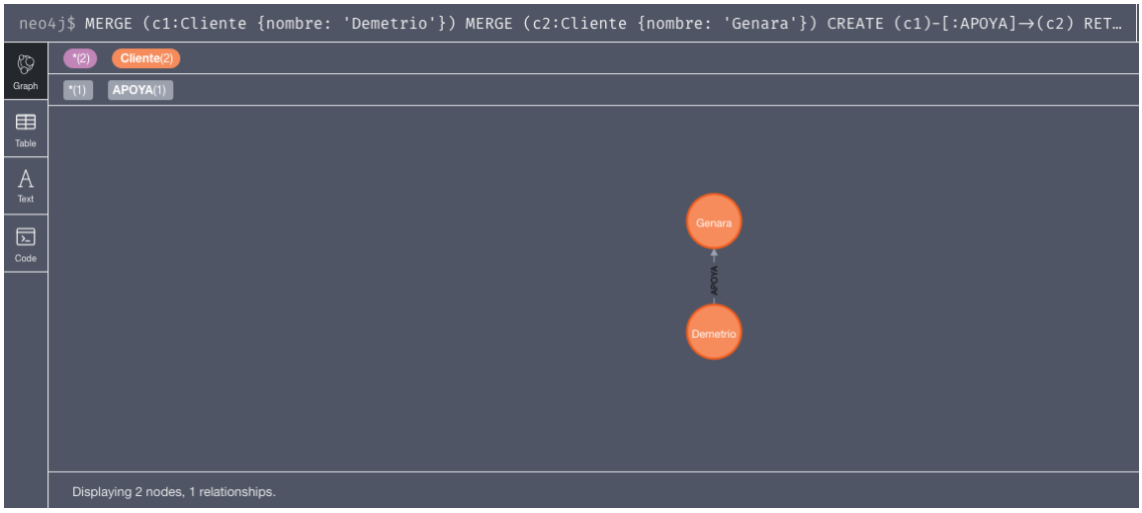
```
MATCH circuito = (c:Cliente)-[:APOYA*]->(c)
RETURN circuito;
```



2.3. (2 puntos) Demetrio ha decidido mostrar su apoyo a Genara. Añadir la relación APOYA entre los nodos correspondientes. No se debe suponer que el nodo existe (ni que no existe), solo se debe asegurar que tras la instrucción ambos existen (sin duplicarlos) y con la relación APOYA indicada.

Tabla 7

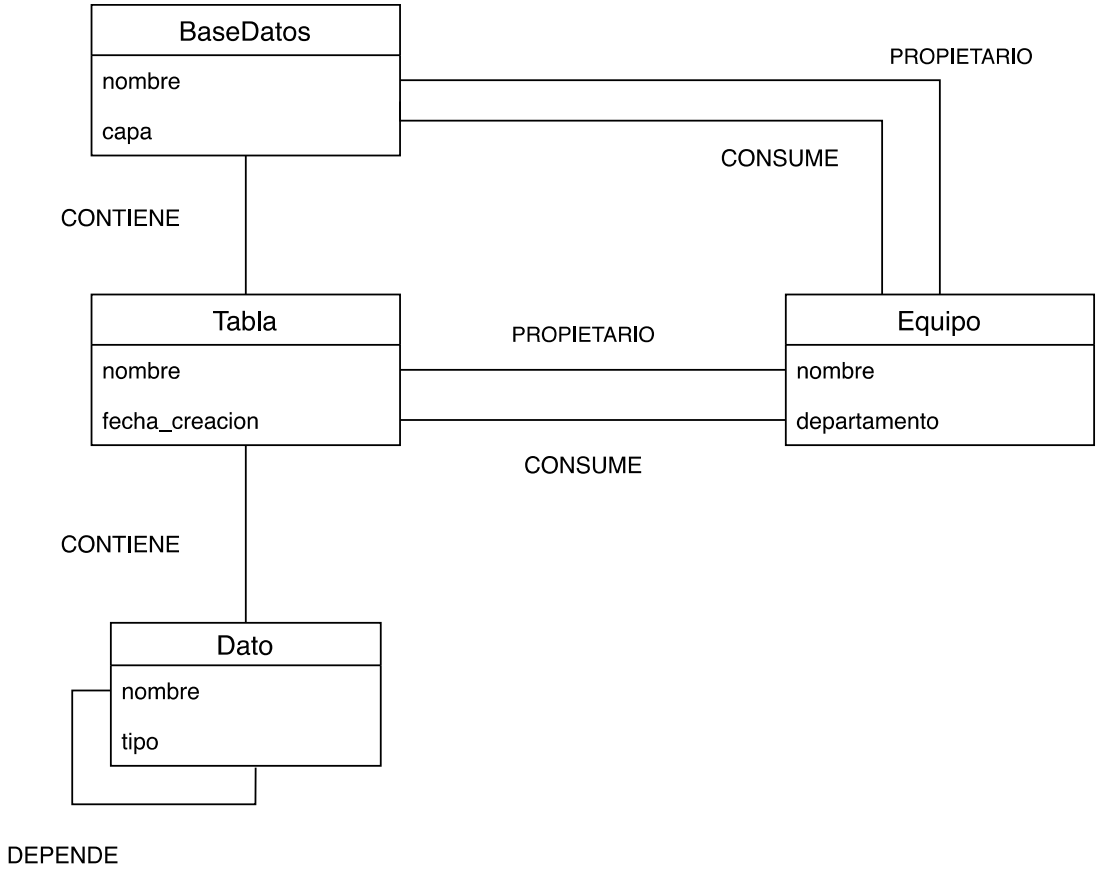
```
MERGE (c1:Cliente {nombre: 'Demetrio'})
MERGE (c2:Cliente {nombre: 'Genara'})
CREATE (c1)-[:APOYA]->(c2)
RETURN c1,c2;
```



3. (3 puntos) Crea tu propia base de datos sobre Neo4j, e implementa al menos 2 consultas sobre dichos datos. Se valorará la originalidad y la complejidad de la base de datos y de las consultas creadas.

Propongo una BD que ayude al organismo de Gobierno del Dato de una empresa a registrar las dependencias entre proyectos que trabajan en un mismo Data-Lake y que comparten datos. Esto puede ser muy útil cuando el número de proyectos crece mucho (y con ello el número de tablas con información en el lago) y se quiere averiguar cuantos proyectos se ven afectados por una modificación en alguna tabla del lago.

El modelo de la BD podría ser algo así:



Donde cada BD y Tabla tiene un equipo propietario y varios equipos consumidores. Una BD está compuesta por Tablas, y una Tabla está compuesta por Datos. Existe una relación entre Dato y sí mismo que refleja la dependencia que puede tener este Dato con otro Dato de otra tabla, por ejemplo:
La Tabla *provision_ventas* contiene el Dato *provision_economica* que se genera en un proceso del equipo *analíticas_ventas*, y la Tabla *inversiones*, del equipo *inversiones_financieras*, tiene el Dato *provision_presupuesto_inversion* que se calcula en un proceso posterior con la siguiente fórmula:
*provision_ventas.provision_economica * 0.6.* Existiría una relación DEPENDE desde *provision_presupuesto_inversion* hasta *provision_economica*.

Si el equipo *analíticas_ventas* decide modificar el Dato *provision_economica*, *provision_presupuesto_inversion* se verá afectado y por lo tanto, deberán contactar con el equipo *inversiones_financieras* (PROPIETARIO de la tabla a la que pertenece el Dato) y los equipos que consuman la Tabla *provision_ventas* antes de realizar el cambio.
Asimismo, al verse afectado el Dato *provision_presupuesto_inversion*, se deberá contactar con los equipos propietarios de las Tablas que contengan Datos que tengan relación DEPENDE en dirección al Dato *provision_presupuesto_inversion* y a los equipos que consuman dichas tablas.

Las sentencias para crear la BD y poblarla con datos ejemplo sería así:

2 Bases de Datos
4 Equipos
4 Tablas con 4 Datos cada tabla.

Tabla 8 – Sentencias de creación de grafo.

```
CREATE (e1: Equipo {nombre:'UXportalWeb', departamento:'UX'})
CREATE (e2: Equipo {nombre:'UXportalInterno', departamento:'UX'})
CREATE (e3: Equipo {nombre:'ventasMayorista', departamento:'Ventas'})
CREATE (e4: Equipo {nombre:'recomendaciones', departamento:'DEV'})

CREATE (db1: BaseDatos {nombre:'bu_portal_interno', capa:'Business Unit'})
CREATE (db2: BaseDatos {nombre:'cd_portal_web', capa:'Common Data'})

CREATE (t1: Tabla {nombre:'security_threat', fecha_creacion:'2019-10-05'})
CREATE (d1_1: Dato {nombre:'id_cliente', tipo:'String'})
CREATE (d1_2: Dato {nombre:'fecha_login', tipo:'Date'})
CREATE (d1_3: Dato {nombre:'ip', tipo:'String'})
CREATE (d1_4: Dato {nombre:'navegador', tipo:'String'})

CREATE (t2: Tabla {nombre:'logins_portal_web', fecha_creacion:'2019-10-07'})
CREATE (d2_1: Dato {nombre:'id_cliente', tipo:'String'})
CREATE (d2_2: Dato {nombre:'fecha_login', tipo:'Date'})
CREATE (d2_3: Dato {nombre:'ip', tipo:'String'})
CREATE (d2_4: Dato {nombre:'navegador', tipo:'String'})

CREATE (t3: Tabla {nombre:'kpi_recomendacion', fecha_creacion:'2019-10-07'})
CREATE (d3_1: Dato {nombre:'id_cliente', tipo:'String'})
CREATE (d3_2: Dato {nombre:'id_producto', tipo:'Int'})
CREATE (d3_3: Dato {nombre:'kpi_base', tipo:'Decimal(16,15)'})
CREATE (d3_4: Dato {nombre:'kpi_prediccion', tipo:'Decimal(16,15)'})

CREATE (t4: Tabla {nombre:'pedidos_portal_web', fecha_creacion:'2019-10-07'})
CREATE (d4_1: Dato {nombre:'id_cliente', tipo:'String'})
CREATE (d4_2: Dato {nombre:'id_producto', tipo:'Int'})
CREATE (d4_3: Dato {nombre:'precio', tipo:'Decimal(3,2)'})
CREATE (d4_4: Dato {nombre:'is_recomendado', tipo:'Boolean'})
CREATE (db1)-[:CONTIENE]->(t1),
 (db2)-[:CONTIENE]->(t2),
```

```
(db2)-[:CONTIENE]->(t3),
(db2)-[:CONTIENE]->(t4),
(t1)-[:CONTIENE]->(d1_1),
(t1)-[:CONTIENE]->(d1_2),
(t1)-[:CONTIENE]->(d1_3),
(t1)-[:CONTIENE]->(d1_4),
(t2)-[:CONTIENE]->(d2_1),
(t2)-[:CONTIENE]->(d2_2),
(t2)-[:CONTIENE]->(d2_3),
(t2)-[:CONTIENE]->(d2_4),
(t3)-[:CONTIENE]->(d3_1),
(t3)-[:CONTIENE]->(d3_2),
(t3)-[:CONTIENE]->(d3_3),
(t3)-[:CONTIENE]->(d3_4),
(t4)-[:CONTIENE]->(d4_1),
(t4)-[:CONTIENE]->(d4_2),
(t4)-[:CONTIENE]->(d4_3),
(t4)-[:CONTIENE]->(d4_4),
(e1)-[:PROPIETARIO]->(t2),
(e2)-[:PROPIETARIO]->(db1),
(e2)-[:PROPIETARIO]->(t1),
(e4)-[:PROPIETARIO]->(db2),
(e4)-[:PROPIETARIO]->(t3),
(e4)-[:PROPIETARIO]->(t4),
(e1)-[:CONSUME]->(t4),
(e2)-[:CONSUME]->(t2),
(e2)-[:CONSUME]->(t4),
(e3)-[:CONSUME]->(t3),
(e3)-[:CONSUME]->(t4),
(e4)-[:CONSUME]->(t2),
(d3_4)<-[:DEPENDE]-(d4_4)-[:DEPENDE]->(d3_3),
(d2_1)<-[:DEPENDE]-(d3_1)-[:DEPENDE]->(d1_1),
(d1_3)-[:DEPENDE]->(d2_3);
```

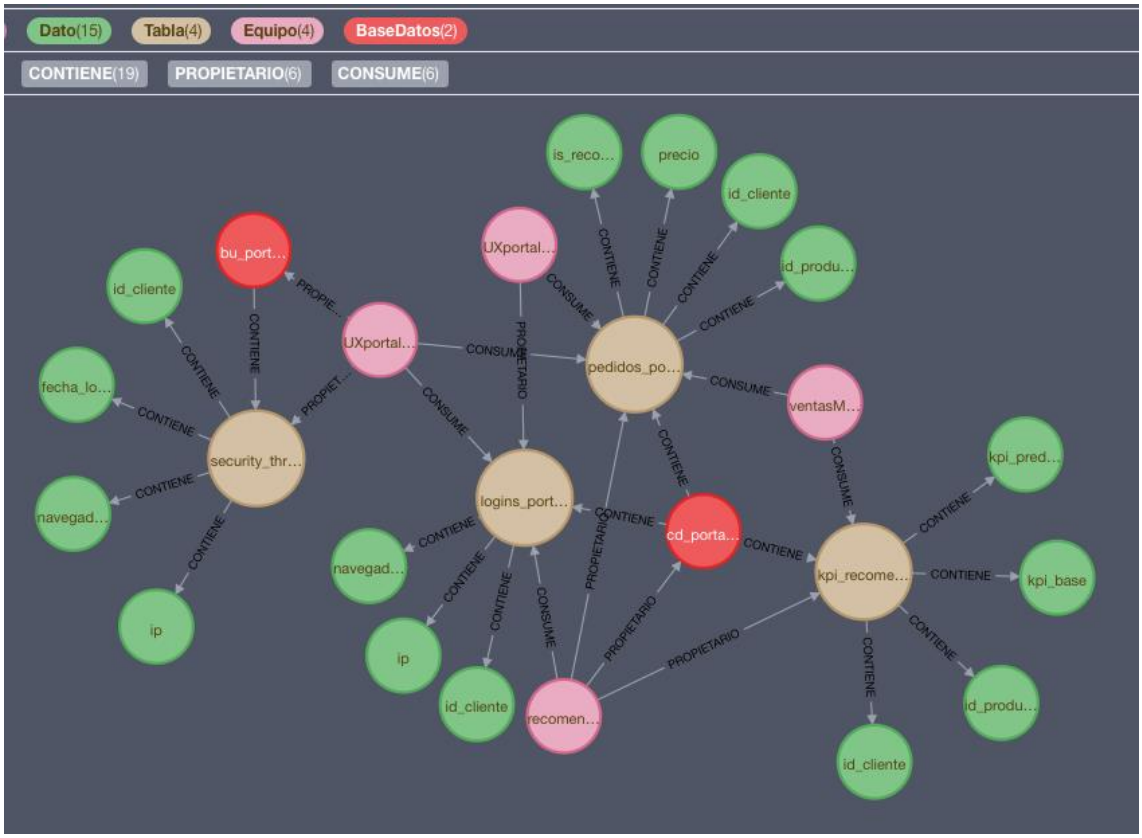


Figura 1 - Grafo que muestra todos los nodos y relaciones menos la relación DEPENDE

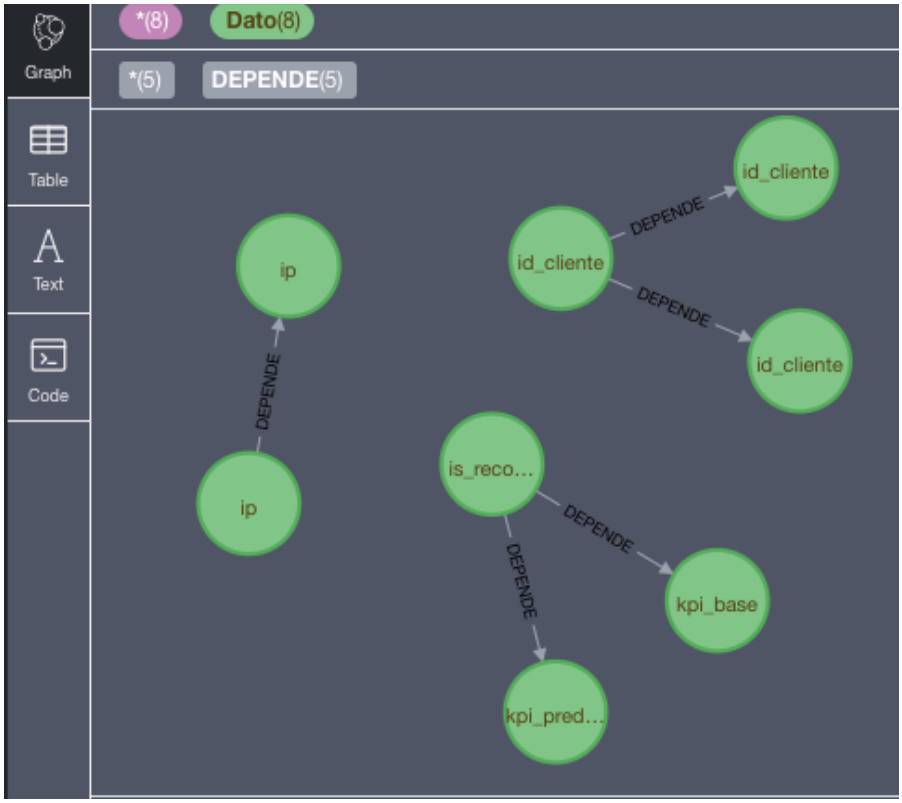


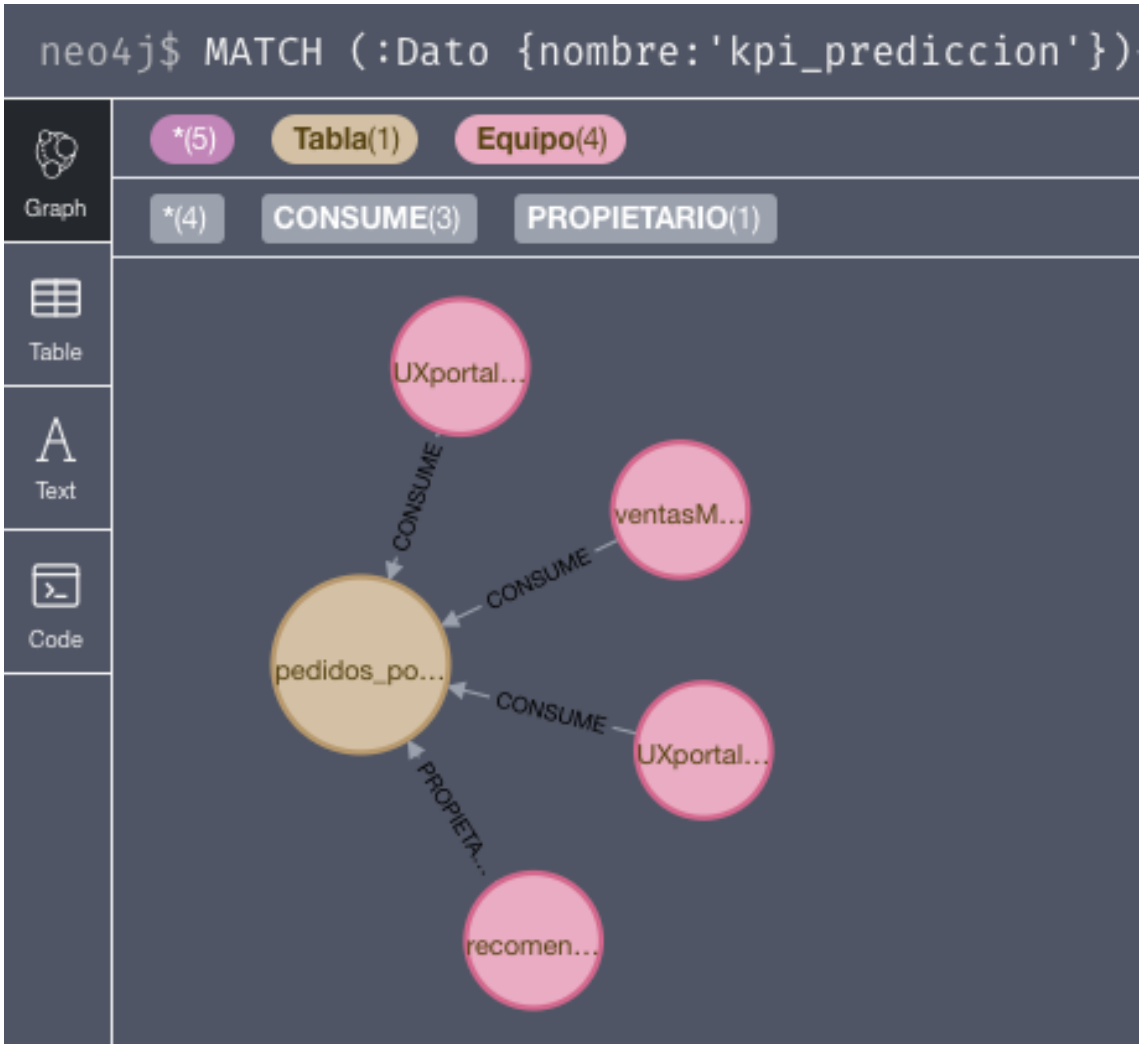
Figura 2 - Grafo que muestra los nodos Dato con relaciones de tipo DEPENDE

Query 1:

Averiguar los equipos y tablas afectados por una modificación de la columna *kpi_predicción*:

Tabla 9 – Query 1

```
MATCH (:Dato {nombre:'kpi_prediccion'})<-[:DEPENDE*]-(:Dato)<-
[:CONTIENE]-(t:Tabla)<--(e:Equipo)
RETURN t,e
```



Query 1.1:

Si el dato aparece en mas de una columna, como por ejemplo id_cliente, debemos filtrar por nombre de tabla:

Tabla 10 – Query 1.1

```
MATCH (t_orig:Tabla {nombre:'security_threat'})-[:CONTIENE]->(d:Dato
{nombre:'id_cliente'})<-[:DEPENDE*]-(:Dato)<-[:CONTIENE]-(t:Tabla)<--
(e:Equipo)
RETURN t,e
```

Query 2:

Averigua todas las tablas y datos que pertenecen al equipo *recomendaciones*:

Tabla 11- Query 2

```
MATCH (t:Tabla)<-[:PROPIETARIO]-(:Equipo {nombre:'recomendaciones'})
RETURN t
```