

# Práctica 4: Neo4j

Martínez Ostoa Néstor Iván  
Bases de Datos No Estructuras - 0600  
Ciencia de Datos, IIMAS, UNAM

Junio 2021



## 1. Introducción

Neo4j es una base de datos gráfica (*graph database*). En ella se pueden implementar algoritmos de gráficas complicados (como el encontrar la ruta mínima entre dos nodos) manteniendo una sencillez a la hora del modelado. Neo4j utiliza *Cypher* como su lenguaje de consultas y el medio principal de comunicación con la base de datos. La principal diferencia que existe con una base de datos relacional es que Neo4j almacena la información como un grafo.

### 1.1. Modelado en Neo4j

Para realizar el modelado de la base de datos, tenemos que tener en consideración los siguientes elementos:

- **Nodos:** son la entidad más elemental en Neo4j y son el serían el equivalente a una tabla en el modelado relacional. Estos nodos pueden tener propiedades que los diferencien de otros
- **Relaciones:** son el segundo elemento fundamental para el modelado en Neo4j. Las relaciones son un aspecto fundamental porque indican la manera de interconexión entre los nodos. Las relaciones, al igual que los nodos, pueden tener propiedades que ayuden a identificar patrones en los nodos

### 1.2. Cypher

*Cypher* es el lenguaje de programación con el que programadores interactúan con Neo4j para realizar todas las operaciones principales. Tiene una notación muy peculiar, a pesar de realizar actividades muy similares a SQL. Derivado de esto, a continuación se muestra un ejemplo de algunas operaciones básicas en Neo4j.

- Creación de un nodo

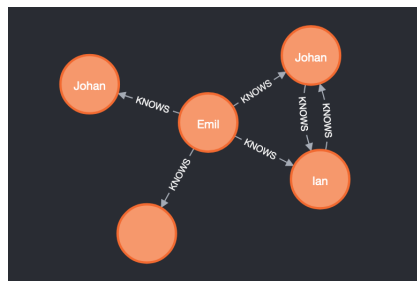
```
1 CREATE (ee:Person { name: "Emil", from: "Sweden", klout: 99 })
```

- Creación de otros nodos junto con relaciones básicas

```
1 MATCH (ee:Person) WHERE ee.name = "Emil"
2 CREATE (js:Person { name: "Johan", from: "Sweden", learn: "surfing" }),
3 (ir:Person { name: "Ian", from: "England", title: "author" }),
4 (rvb:Person { name: "Rik", from: "Belgium", pet: "Orval" }),
5 (ally:Person { name: "Allison", from: "California", hobby: "surfing" }),
6 (ee)-[:KNOWS {since: 2001}]->(js),(ee)-[:KNOWS {rating: 5}]->(ir),
7 (js)-[:KNOWS]->(ir),(js)-[:KNOWS]->(rvb),
8 (ir)-[:KNOWS]->(js),(ir)-[:KNOWS]->(ally),
9 (rvb)-[:KNOWS]->(ally)
```

- Búsqueda en el grafo

```
1 MATCH (ee:Person)-[:KNOWS]-(friends)
2 WHERE ee.name = "Emil" RETURN ee, friends
```

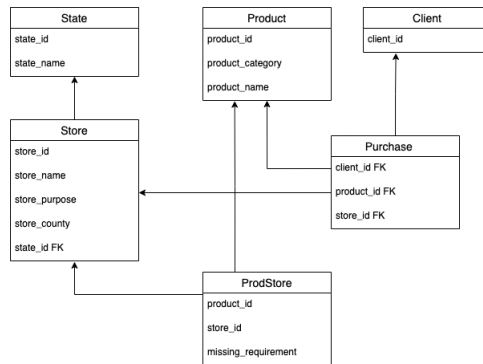


## 2. Desarrollo

Con base en los datos de Normatividad Mercantil de la PROFECO para el 2015, realicé una base de datos tipo grafo con los apartados descritos a continuación.

### 2.1. Modelo de grafo

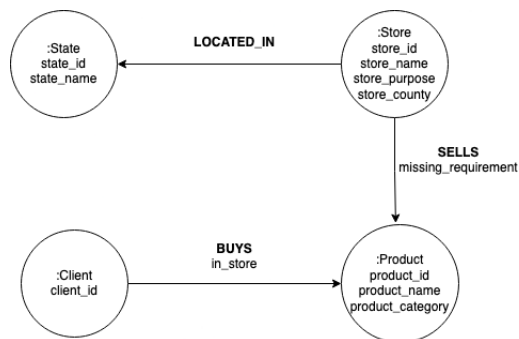
Para modelar el grafo solución a este problema, empecé un punto atrás. En un modelo relacional que me ayudara a estructurar las diversas entidades. El modelo se muestra a continuación:



El modelo consta de las siguientes entidades:

- **State**: representa una entidad federativa mexicana
- **Store**: representa cada una de las tiendas encontradas en los datos originales. Esta entidad contiene un atributo llamado **store\_county** que representa el municipio en el que se encuentra la tienda. La razón de hacer esto se debe a que existían múltiples tiendas con el mismo nombre (tiendas OXXO por ejemplo) y en el preprocesamiento de los datos, ocupé el atributo del municipio para diferenciarlas
- **Product**: representa un producto. La **product\_category** corresponde al campo **tipo producto** de la fuente de datos original y el campo **product\_name** corresponde al campo **descripcion producto**
- **Client**: representa un cliente
- **Purchase**: representa una tabla encargada de almacenar el producto que compró un cliente junto con la tienda en donde lo compró
- **ProdStore**: representa una entidad encargada de almacenar los productos que ofrece cada tienda. El campo **missing\_requirement** se encarga de almacenar si el producto, para esa tienda en específico, cuenta con algún incumplimiento reglamentario

Una vez diseñado el modelo anterior, podemos pasar a diseñar el modelo del grafo que cargaremos a Neo4j. De nuevo, el modelo de grafo se muestra primero y luego su explicación:



En este modelo podemos ver que tenemos solo cuatro nodos. A continuación se describen con mayor detalle:

- **:State**: representa una entidad federativa y se relaciona con el nodo **:Store** a través de la relación **LOCATED\_IN**
- **:Store**: representa una tienda y se relaciona con el nodo **:Product** mediante la relación **SELLS** la cual tiene como atributo **missing\_requirement** que es una cadena que indicará el resultado de la visita de la PROFECO. Es decir, en este campo se guardará si el producto tiene o no incumplimiento con alguna norma oficial
- **:Client**: representan a los consumidores de productos y se relaciona con el nodo **:Product** por medio de la relación **BUYS** la cual tiene como propiedad **in\_store** el cual almacenará el id de la tienda en donde se realizó la compra de ese producto
- **:Product**: representan los productos ofrecidos por las tiendas

## 2.2. Preprocesamiento de datos

El preprocesamiento de los datos lo realicé en el archivo `code.ipynb`, en él, se encuentra con mayor detalle las etapas en Python para limpiar y preprocesar los datos. A grandes rasgos, lo que hice fue lo siguiente:

1. Generar un catálogo de estados con el campo **ENTIDAD**
2. Generar un catálogo de tiendas tomando los campos **RAZON SOCIAL** y **MUNICIPIO** como identificador único para cada una de las tiendas. Esto lo hice para evitar duplicados en las tiendas
3. Generar un catálogo de productos tomando los campos **TIPO PRODUCTO** y **DESCRIPCION PRODUCTO** como llave única para evitar duplicados
4. Crear una entidad que contuviera la relación descrita por la entidad **ProdStore** del modelo relacional en donde se almacenan todos los productos vendidos por cada tienda al igual que un posible incumplimiento de alguna norma
5. Generar un catálogo de 50,000 clientes
6. Simular compras para cada uno de los 50,000 clientes. Esto lo realicé con números aleatorios para establecer:
  - Cantidad de compras realizadas por cliente
  - Tiendas en las que el cliente realizó una compra
  - Productos aleatorios vendidos por las tiendas previamente seleccionadas como parte de su historia de compras

Todos estos pasos los almacené en archivos separados por coma (csv). A continuación se muestran los links para visualizar cada uno de estos:

- [Datos originales](#)
- [Catálogos de estados](#)

- Catálogos de tiendas
- Catálogos de productos
- Productos vendidos por tienda
- Catálogo de clientes
- Historial de compras por cliente

## 2.3. Inserción de datos

Para la inserción de datos en Neo4j <sup>1</sup> seguí los siguientes pasos:

### 1. Carga de estados (entidades federativas)

```
1 LOAD CSV WITH HEADERS FROM "https://raw.githubusercontent.com/nestorivanmo/iimas-data-science/master/spring-21/bne/practicas/neo4j/data/states.csv" AS
  row
2 CREATE (n:State)
3 SET n = row
```

neo4j\$ MATCH (n:State) RETURN n LIMIT 25

"n"
({"state_name":"aguascalientes","state_id":"1"})
({"state_name":"baja california","state_id":"2"})
({"state_name":"baja california sur","state_id":"3"})
({"state_name":"campeche","state_id":"4"})
({"state_name":"distrito federal","state_id":"5"})
({"state_name":"chiapas","state_id":"6"})
({"state_name":"chihuahua","state_id":"7"})
({"state_name":"sonora","state_id":"8"})
({"state_name":"coahuila de zaragoza","state_id":"9"})
({"state_name":"colima","state_id":"10"})
({"state_name":"mexico","state_id":"11"})

### 2. Carga de tiendas

```
1 LOAD CSV WITH HEADERS FROM "https://raw.githubusercontent.com/nestorivanmo/iimas-data-science/master/spring-21/bne/practicas/neo4j/data/stores.csv" AS
  row
2 CREATE (n:Store)
3 SET n = row
```

---

<sup>1</sup>s-01-load-data.cql

```
neo4j$ MATCH (n:Store) RETURN n LIMIT 25
```

"n"
{ "store_id": "1", "store_county": "aguascalientes", "store_name": "julio cepeda jugueterias s.a. de c.v.", "state_id": "1", "store_purpose": "jugueteria" }
{ "store_id": "2", "store_county": "aguascalientes", "store_name": "c&a de mexico s. de r.l.", "state_id": "1", "store_purpose": "tienda departamental" }
{ "store_id": "3", "store_county": "aguascalientes", "store_name": "asfeixas s.a. de c.v.", "state_id": "1", "store_purpose": "motel" }
{ "store_id": "4", "store_county": "aguascalientes", "store_name": "motel villa miro- miro hoteleria s.a. de c.v.", "state_id": "1", "store_purpose": "motel" }
{ "store_id": "5", "store_county": "aguascalientes", "store_name": "motel luna- compaia hotelera pescadores s.a. de c.v.", "state_id": "1", "store_purpose": "motel" }

### 3. Carga de productos

```
1 LOAD CSV WITH HEADERS FROM "https://raw.githubusercontent.com/nestorivanmo/iimas-data-science/master/spring-21/bne/practicas/neo4j/data/products.csv"
  AS row
2 CREATE (n:Product)
3 SET n = row
```

```
neo4j$ MATCH (n:Product) RETURN n LIMIT 25
```

"n"
{ "product_name": "en empaque de carton estampado y plastico rigido mu0e ca para ni0a de la marca barbie de origen china. contenido una pieza m odelo barbie basica life fashion con numero de lote t7439 indica infor macion importante para consulta impo", "product_category": "mu0ecos", "pr oduct_id": "1" }
{ "product_name": "camisa para caballero en color azul marino de la marc a c & a de origen bangladesh importado por c & a mexico de composicion en 64% algodn 32% poliamida y 4% elastano con numero de lote 252959. contiene instrucciones de cuidado", "product_category": "ropa caballero ", "product_id": "2" }
{ "product_name": "una botella brandy de la marca torres 10 con la leyen da 100% de uva gran reserva con 38% alc. vol. con un contenido neto de 700ml con sello de shcp con la leyenda "el abuse en el consumo de est e producto es nocivo para la salud"", "product_category": "brandy", "prod uct_id": "3" }
{ "product_name": "envase de vidrio contenido brandy de la marca azteca de oro solera reservada con 38% alc. vol. contenido neto 700ml hecho e lmexico producto 100% de uva fabricado por industrias vinicolas pedro

### 4. Carga de productos vendidos por tienda

```
1 LOAD CSV WITH HEADERS FROM "https://raw.githubusercontent.com/nestorivanmo/iimas-data-science/master/spring-21/bne/practicas/neo4j/data/product_store.
  csv" AS row
2 CREATE (n:ProductStore)
3 SET n = row
```

```
neo4j$ MATCH (n:ProductStore) RETURN n LIMIT 25
```

n
{ "store_id": "3014", "missing": "no se detecto incumplimiento", "product_id": "6435" }
{ "store_id": "3014", "missing": "no se detecto incumplimiento", "product_id": "7144" }
{ "store_id": "3014", "missing": "no se detecto incumplimiento", "product_id": "7145" }
{ "store_id": "3014", "missing": "no se detecto incumplimiento", "product_id": "7146" }
{ "store_id": "3014", "missing": "no se detecto incumplimiento", "product_id": "7147" }
{ "store_id": "3014", "missing": "no se detecto incumplimiento", "product_id": "7148" }
{ "store_id": "3014", "missing": "no se detecto incumplimiento", "product_id": "6489" }

## 5. Carga de clientes

```
1 LOAD CSV WITH HEADERS FROM "https://raw.githubusercontent.com/nestorivanmo/iimas-data-science/master/spring-21/bne/practicas/neo4j/data/clients.csv"
  AS row
2 CREATE (n:Client)
3 SET n = row
```

```
neo4j$ MATCH (n:Client) RETURN n LIMIT 25
```

n
{ "client_id": "1" }
{ "client_id": "2" }
{ "client_id": "3" }
{ "client_id": "4" }
{ "client_id": "5" }
{ "client_id": "6" }
{ "client_id": "7" }
{ "client_id": "8" }
{ "client_id": "9" }
{ "client_id": "10" }
{ "client_id": "11" }

## 6. Carga de compras

```
1 LOAD CSV WITH HEADERS FROM "https://raw.githubusercontent.com/nestorivanmo/iimas-data-science/master/spring-21/bne/practicas/neo4j/data/purchases.csv"
  AS row
2 CREATE (n:Purchase)
3 SET n = row
```

```
neo4j$ MATCH (n:Purchase) RETURN n LIMIT 25
```

"n"
{ "store_id": "1325", "client_id": "1", "product_id": "1672" }
{ "store_id": "1325", "client_id": "1", "product_id": "1672" }
{ "store_id": "169", "client_id": "1", "product_id": "166" }
{ "store_id": "169", "client_id": "1", "product_id": "166" }
{ "store_id": "9081", "client_id": "1", "product_id": "22418" }
{ "store_id": "9081", "client_id": "1", "product_id": "22418" }
{ "store_id": "11314", "client_id": "1", "product_id": "25682" }
{ "store_id": "11314", "client_id": "1", "product_id": "25682" }
{ "store_id": "11314", "client_id": "1", "product_id": "25684" }
{ "store_id": "11314", "client_id": "1", "product_id": "25684" }
{ "store_id": "11314", "client_id": "1", "product_id": "25687" }

## 2.4. Creación del grafo

Para la creación del grafo <sup>2</sup> tuve que juntar los datos recién agregados con las relaciones previamente definidas en el modelo del grafo. A continuación muestro con mayor detalle este proceso:

### 1. State $\leftarrow$ Store

```
1 MATCH (store:Store), (state:State)
2 WHERE store.state_id = state.state_id
3 CREATE (store)-[:LOCATED_IN]->(state)
```

```
neo4j$ MATCH p=()-[r:LOCATED_IN]->() RETURN p LIMIT 25
```

"p"
[{"store_id": "71", "store_county": "aguascalientes", "store_name": "dulceria la chachita / hector javier de alba perez", "state_id": "1", "store_purpose": "dulceria"}, {"state_name": "aguascalientes", "state_id": "1"}]
[{"store_id": "8", "store_county": "aguascalientes", "store_name": "vicky boutique- rocio esparza ortiz", "state_id": "1", "store_purpose": "ropa para dama"}, {"state_name": "aguascalientes", "state_id": "1"}]
[{"store_id": "82", "store_county": "aguascalientes", "store_name": "office depot de mexico s.a. de c.v.", "state_id": "1", "store_purpose": "articulos de oficina"}, {"state_name": "aguascalientes", "state_id": "1"}]
[{"store_id": "91", "store_county": "aguascalientes", "store_name": "joyas neri - maria guadalupe neri santos", "state_id": "1", "store_purpose": "joyeria"}, {"state_name": "aguascalientes", "state_id": "1"}]
[{"store_id": "5", "store_county": "aguascalientes", "store_name": "motel luna- compa0ia hotelera pescadores s.a. de c.v.", "state_id": "1", "store_purpose": "motel"}, {"state_name": "aguascalientes", "state_id": "1"}]
[{"store_id": "63", "store_county": "aguascalientes", "store_name": "sinatra

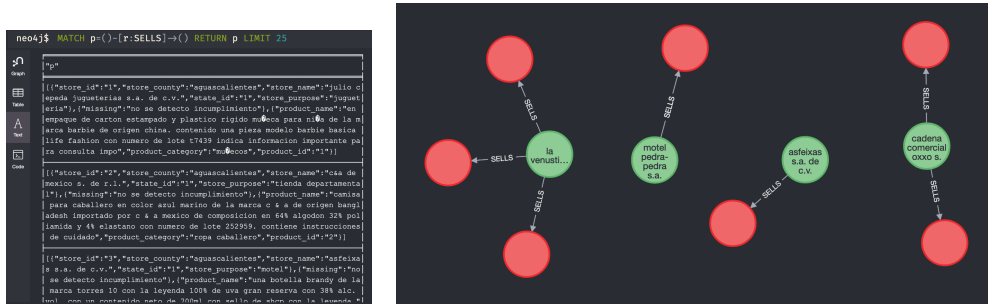


### 2. Store $\rightarrow$ Product

```
1 MATCH (s:Store), (p:Product), (ps:ProductStore)
2 WHERE ps.store_id = s.store_id AND ps.product_id = p.product_id
3 CREATE (s)-[:SELLS {missing: ps.missing}]->(p)
```

<sup>2</sup>s-02-relationships.cql



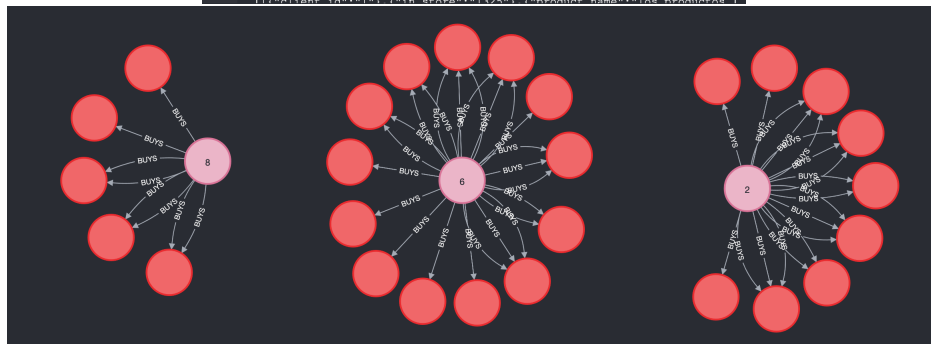
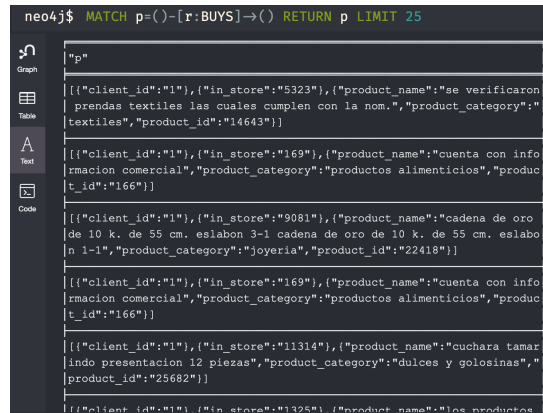


### 3. Client → Product

```

1 MATCH (c:Client),(p:Product),(s:Store),(pu:Purchase)
2 WHERE pu.client_id = c.client_id AND
3     pu.product_id = p.product_id AND
4     pu.store_id = s.store_id
5 CREATE (c)-[:BUYS {in_store: s.store_id}]->(p)

```



## 2.5. Consultas

El desarrollo de las consultas <sup>3</sup> se muestra a continuación:

<sup>3</sup>s-03-queries.cql

## 1. Búsqueda de productos

```

1 // 1. Dado un estado y un producto, buscar lugares donde pueda encontrarlo
2 // Nota: en lugar de usar producto estoy usando la categoria de producto,
  ofrece
3 // mejores resultados
4
5 MATCH (p:Product {product_category:"ropa"})
6   <-[r:SELLS]-(s:Store)--(st:State {state_name:'distrito federal'})
7 RETURN st.state_name, s.store_county, s.store_name, p.product_name, r.
  description

```



```

1 MATCH (p:Product {product_category:"ropa"})
2   <-[r:SELLS]-(s:Store)--(st:State {state_name:'distrito
  federal'})
3 RETURN st.state_name, s.store_county, s.store_name, p.product_name

```

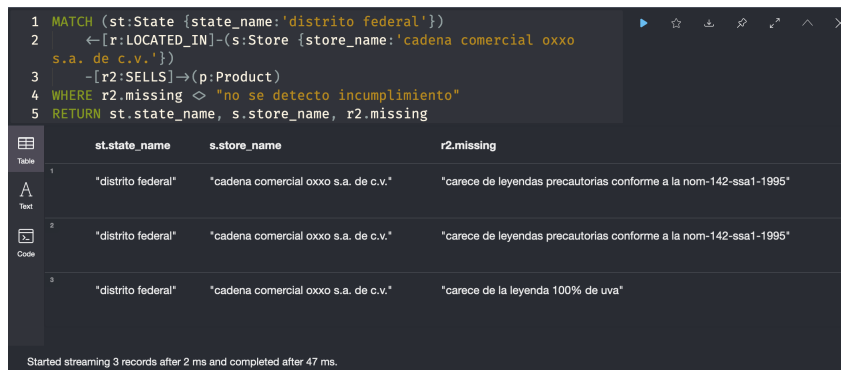
	st.state_name	s.store_county	s.store_name	p.product_name
1	"distrito federal"	"iztapalapa"	"robustae mexico s.a. de c.v."	"camiseta para dama marca lift color baige rayas negras"
2	"distrito federal"	"iztapalapa"	"robustae mexico s.a. de c.v."	"shor para dama marca lift"
3	"distrito federal"	"iztapalapa"	"robustae mexico s.a. de c.v."	"vestido para dama amrca lift 45 % algodón"
4	"distrito federal"	"iztapalapa"	"robustae mexico s.a. de c.v."	"pantalon para dama marca lift"
5	"distrito federal"	"cuauhtemoc"	"casa rivera"	"vestido de presentacion marca rivero hecho en mexico."
6	"distrito federal"	"italpan"	"deportes marti s.a. de c.v. suc. nike paseo acoxpa"	"short para dama"

## 2. Verificación de incumplimiento en tiendas

```

1 // 2. Dado un estado y una tienda, verificar si tiene algun incumplimiento con
2 // un producto
3
4 MATCH (st:State {state_name:'distrito federal'})
5   <-[r:LOCATED_IN]-(s:Store {store_name:'cadena comercial oxxo s.a. de c.v.'})
6   -[r2:SELLS]->(p:Product)
7 WHERE r2.missing <> "no se detecto incumplimiento"
8 RETURN st.state_name, s.store_name, r2.missing

```



```

1 MATCH (st:State {state_name:'distrito federal'})
2   <-[r:LOCATED_IN]-(s:Store {store_name:'cadena comercial oxxo
  s.a. de c.v.'})
3   -[r2:SELLS]->(p:Product)
4 WHERE r2.missing <> "no se detecto incumplimiento"
5 RETURN st.state_name, s.store_name, r2.missing

```

	st.state_name	s.store_name	r2.missing
1	"distrito federal"	"cadena comercial oxxo s.a. de c.v."	"carece de leyendas precautorias conforme a la nom-142-ssa1-1995"
2	"distrito federal"	"cadena comercial oxxo s.a. de c.v."	"carece de leyendas precautorias conforme a la nom-142-ssa1-1995"
3	"distrito federal"	"cadena comercial oxxo s.a. de c.v."	"carece de la leyenda 100% de uva"

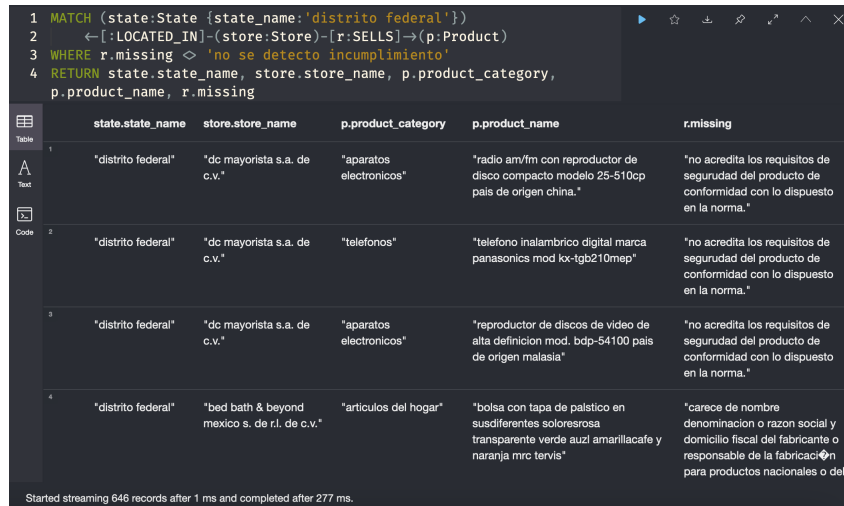
Started streaming 3 records after 2 ms and completed after 47 ms.

### 3. Buscador de productos o categorías alternativas que no tengan incumplimiento

```

1 // 3. Dado un estado y un producto, buscar alternativas sin incumplimiento de
  ese
2 // producto o categoría.
3
4 // Primero busco productos sin cumplimiento
5 MATCH (state:State {state_name:'distrito federal'})
6   <-[:LOCATED_IN]-(store:Store)-[r:SELLS]->(p:Product)
7 WHERE r.missing <> 'no se detecto incumplimiento'
8 RETURN state.state_name, store.store_name, p.product_category, p.product_name,
   r.missing

```



The screenshot shows a query execution interface with a code editor at the top and a table view below. The code editor contains the same query as above. The table view shows the results of the query, with columns: state.state\_name, store.store\_name, p.product\_category, p.product\_name, and r.missing. The table contains 4 rows of data.

	state.state_name	store.store_name	p.product_category	p.product_name	r.missing
1	"distrito federal"	"dc mayorista s.a. de c.v."	"aparatos electronicos"	"radio am/fm con reproductor de disco compacto modelo 25-510cp pais de origen china."	"no acredita los requisitos de seguridad del producto de conformidad con lo dispuesto en la norma."
2	"distrito federal"	"dc mayorista s.a. de c.v."	"telefonos"	"telefono inalambrico digital marca panasonics mod lx-tgb210mep"	"no acredita los requisitos de seguridad del producto de conformidad con lo dispuesto en la norma."
3	"distrito federal"	"dc mayorista s.a. de c.v."	"aparatos electronicos"	"reproductor de discos de video de alta definicion mod. bdp-54100 pais de origen malasia"	"no acredita los requisitos de seguridad del producto de conformidad con lo dispuesto en la norma."
4	"distrito federal"	"bed bath & beyond mexico s. de r.l. de c.v."	"articulos del hogar"	"bolsa con tapa de plastico en sus diferentes colores rosa transparente verde azul amarillacafe y naranja mirc tervis"	"carece de nombre denominacion o razon social y domicilio fiscal del fabricante o responsable de la fabrica" para productos nacionales o del

Started streaming 646 records after 1 ms and completed after 277 ms.

```

1 MATCH (state:State {state_name:'distrito federal'})
2   <-[:LOCATED_IN]-(store:Store)-[r:SELLS]->
3   (p:Product {product_category:'telefonos'})
4 WHERE r.missing = 'no se detecto incumplimiento'
5 RETURN state.state_name, store.store_name, p.product_category, p.product_name,
6   r.missing

```

```

1 MATCH (state:State {state_name:'distrito federal'})
2   ←[:LOCATED_IN]-(store:Store)-[:SELLS]→
3   (p:Product {product_category:'telefonos'})
4 WHERE r.missing = 'no se detecto incumplimiento'
5 RETURN state.state_name, store.store_name, p.product_category,
6        p.product_name,
7        r.missing

```

	state.state_name	store.store_name	p.product_category	p.product_name	r.missing
1	"distrito federal"	"aldan electronica s.a. de c.v. y/o steren shop perisur"	"telefonos"	"telefono inalambrico marca steren modelo tel-2480 en caja de carton"	"no se detecto incumplimiento"
2	"distrito federal"	"aldan electronica s.a. de c.v. y/o steren shop perisur"	"telefonos"	"telefono inalambrico marca steren modelo tel-605 en caja de carton"	"no se detecto incumplimiento"
3	"distrito federal"	"lumen y/o abastecedora lumen s.a. de c.v."	"telefonos"	"caja de carton con telefono inalambrico hecho en china imp. por servicios de comercio exterior myme sc contiene poliza de garantia lista de centros de servicio y guia de uso en idioma espaol mod. g280"	"no se detecto incumplimiento"
4	"distrito federal"	"elekra del milenio s.a de c.v."	"telefonos"	"1 telefono inalambrico digital empaques instructivos y garantias hecho en china exhibe leyenda modelo gate4800r auricular adaptador entrada"	"no se detecto incumplimiento"

#### 4. Registro de compras y lugares visitados por usuario

```

1 // Registro de compras y lugares visitados por usuario
2
3 MATCH p=()-[r:BUYS]->() RETURN p LIMIT 25

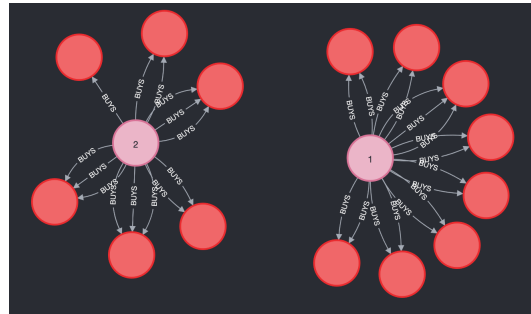
```

```

neo4j$ MATCH p=()-[r:BUYS]->() RETURN p LIMIT 25

```

"p"
[{"client_id":"1"}, {"in_store":"5323"}, {"product_name":"se verificaron prendas textiles las cuales cumplen con la nom.", "product_category":"textiles", "product_id":"14643"}]
[{"client_id":"1"}, {"in_store":"169"}, {"product_name":"cuenta con informacion comercial", "product_category":"productos alimenticios", "product_id":"166"}]
[{"client_id":"1"}, {"in_store":"9081"}, {"product_name":"cadena de oro de 10 k. de 55 cm. eslabon 3-1 cadena de oro de 10 k. de 55 cm. eslabon 1-1", "product_category":"joyeria", "product_id":"22418"}]
[{"client_id":"1"}, {"in_store":"169"}, {"product_name":"cuenta con informacion comercial", "product_category":"productos alimenticios", "product_id":"166"}]
[{"client_id":"1"}, {"in_store":"11314"}, {"product_name":"cuchara tamarindo presentacion 12 piezas", "product_category":"dulces y golosinas", "product_id":"25682"}]
[{"client_id":"1"}, {"in_store":"1325"}, {"product_name":"los productos verificados cumplen con la norma", "product_category":"juguetes", "product_id":"1672"}]
[{"client_id":"1"}, {"in_store":"11314"}, {"product_name":"chocolate con fitado presentacion 6 paquetes", "product_category":"chocolate", "product_id":"25687"}]



#### 5. Búsqueda de estados con mayor y menor incumplimiento relativo al número de tiendas en ese estado

```

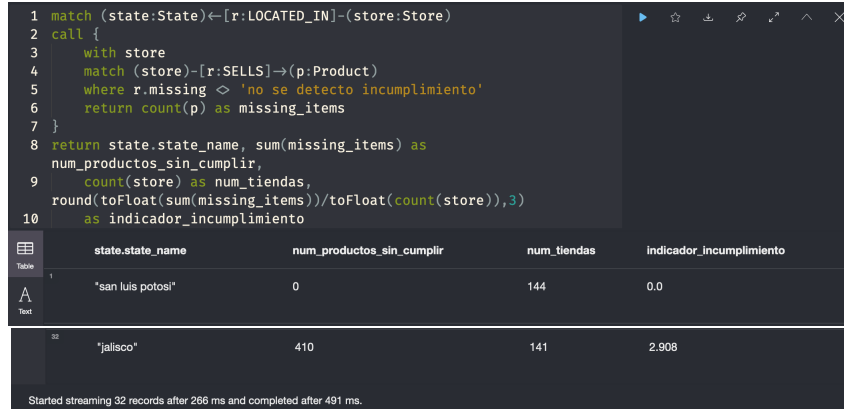
1 // 5. Encontrar los estados con mayor y menor incumplimiento relativo al numero
2 // de tiendas que tiene.
3
4 match (state:State) <-[:LOCATED_IN]-(store:Store)
5 call {

```

```

6   with store
7   match (store)-[r:SELLS]->(p:Product)
8   where r.missing <> 'no se detecto incumplimiento'
9   return count(p) as missing_items
10 }
11 return state.state_name, sum(missing_items) as num_productos_sin_cumplir,
12        count(store) as num_tiendas, round(toFloat(sum(missing_items))/toFloat(
13        count(store)),3)
14        as indicador_incumplimiento
15 order by indicador_incumplimiento

```



```

1 match (state:State)←[r:LOCATED_IN]-(store:Store)
2 call {
3   with store
4   match (store)-[r:SELLS]->(p:Product)
5   where r.missing <> 'no se detecto incumplimiento'
6   return count(p) as missing_items
7 }
8 return state.state_name, sum(missing_items) as
num_productos_sin_cumplir,
9 count(store) as num_tiendas,
round(toFloat(sum(missing_items))/toFloat(count(store)),3)
10 as indicador_incumplimiento

```

	state.state_name	num_productos_sin_cumplir	num_tiendas	indicador_incumplimiento
1	"san luis potosi"	0	144	0.0
32	"jalisco"	410	141	2.908

Started streaming 32 records after 266 ms and completed after 491 ms.

### 3. Conclusiones

Sin lugar a dudas que Neo4j es una base de datos muy versátil. A lo largo de esta práctica pude implementar preguntas que son difíciles de responder desde el punto de vista computacional de manera muy sencilla mediante un comando en Cypher. Sin embargo, algo a notar es que cree aproximadamente 1.2 millones de registros de compras y la herramienta Neo4j Desktop decrementó su rendimiento sustancialmente. Las consultas tardaban más tiempo en ejecutar de lo normal.

Aunado a esto, algo que me encantó de Neo4j es la facilidad con la que se pueden resolver problemas de gráficos muy complejos manteniendo un modelado extremadamente sencillo si vienes de un panorama relacional. La transición entre el modelado relacional al modelado en grafo es sumamente sencillo y me atrevería a decir que incluso más sencillo que el relacional.

### Referencias

- [1] Pimentel. *Apuntes de Bases de Datos No Estructuradas: Neo4j*. Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas. UNAM. México. 2021. Revisadas el 18 de junio del 2021
- [2] Vukotic A., Watt N., et. al. *Neo4j in Action*. Manning. 2015
- [3] Neo4j Docs. *The Neo4j Cypher Manual v4.3*. Revisado el 18 de junio del 2021 en: <https://neo4j.com/docs/cypher-manual/current/>

- [4] Neo4j Docs. *Graph database concepts*. Revisado el 18 de junio del 2021 en: <https://neo4j.com/docs/getting-started/current/graphdb-concepts/>