

Proyecto NoSQL – Neo4J

Análisis de comportamiento de compra

Equipo 5

Universidad Autónoma de Yucatán
Modelos de Datos LCC

Agosto–Diciembre 2025

Profesor: M. en C. Luis R. Basto Díaz

Agenda

- 1 Dataset
- 2 Modelado en Neo4J
- 3 Importación y herramientas
- 4 CRUD y evidencias
- 5 Reproducibilidad
- 6 Conclusiones

Fuente y descripción

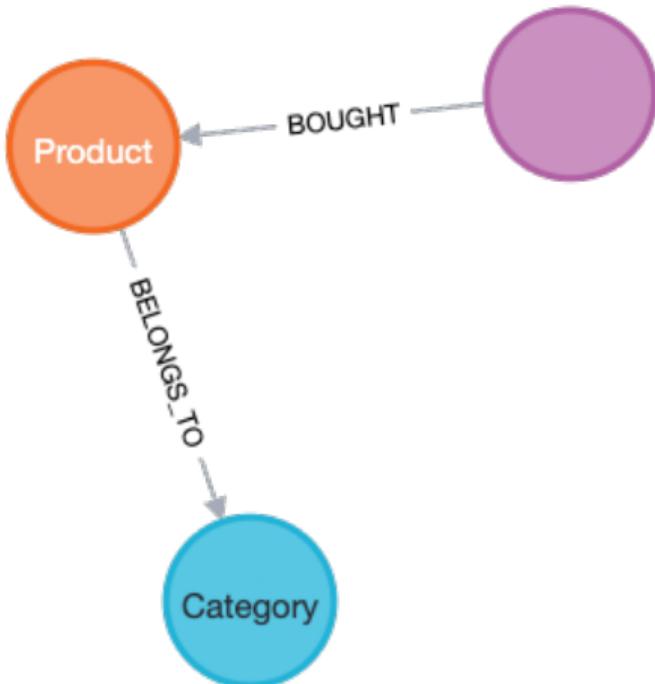
- Kaggle: *Shopping Behaviour and Product Ranking Dataset*.
- 3,900 transacciones con datos de clientes, productos, categorías, monto y rating.
- Archivo principal: data/shopping_behavior.csv.
- Adecuado para grafos: conecta clientes, productos y categorías con relaciones de compra.

Diccionario de datos (resumen)

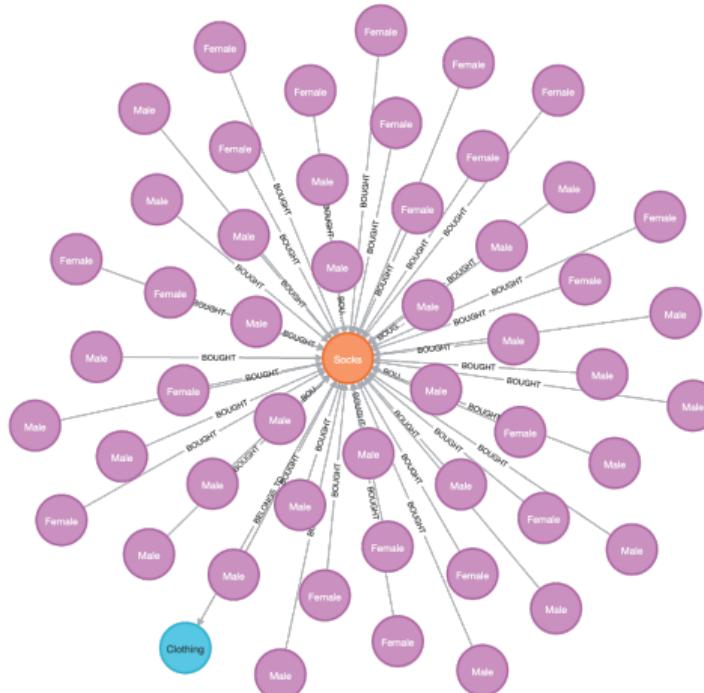
- **Customer**: customerId, edad, género, ubicación, suscripción, compras previas, frecuencia.
- **Product**: productId, nombre, talla, color, temporada, rating promedio.
- **Category**: nombre de categoría.
- **BOUGHT** (relación): monto, descuento, rating, método de pago, fecha de importación.

Esquema de grafo

- Nodos: **Customer**, **Product**, **Category**.
- Relaciones: **BOUGHT** (cliente → producto), **BELONGS_TO** (producto → categoría).
- Restricciones de unicidad: customerId, productId, Category.name.



Subgrafo con datos reales



- Ejemplo de clientes, productos y categorías conectados tras la importación.

Proceso de importación

- ① Copia del CSV a la carpeta `import` de Neo4J.
- ② Ejecución de `constraints.cypher`.
- ③ Creación de nodos con `import_nodes.cypher`.
- ④ Creación de relaciones con `import_relationships.cypher`.

Herramientas: Neo4J Browser / Desktop, Cypher, Docker Compose, GitHub.

Operaciones CRUD (5 por operación)

- **CREATE**: clientes, productos, categorías, relaciones de compra.
- **READ**: top productos, clientes por categoría, promedio por método de pago, etc.
- **UPDATE**: actualización de edad, suscripción, ratings promedios, incrementos de compras.
- **DELETE**: eliminación de clientes, relaciones de compra bajas, productos sin compras.



The screenshot shows the Neo4j browser interface with a query results table. The table has four columns: o.customerId, o.age, o.gender, and o.location. The data is as follows:

	o.customerId	o.age	o.gender	o.location
1	40	70	"Male"	"Arizona"
2	109	70	"Male"	"Montana"
3	115	70	"Male"	"Massachusetts"
4	204	70	"Male"	"Indiana"
5	230	70	"Male"	"Ohio"
6	295	70	"Male"	"New Jersey"

Started streaming 1479 records after 17 ms and completed after 34 ms, displaying first 1000 rows.

Evidencia de operaciones

```
neo4j> MERGE (c:Customer {customerId: 9999}) SET c.age = 20, c.gender = "Female", c.location = "California", c.subscriptionStatus = "Yes", c.previousPurchases = 5, c.frequency = 0
Added 1 label, created 1 node, set 7 properties, completed after 16 ms.

neo4j> MATCH (c:Customer {customerId: 1}) SET c.age = 50 RETURN c.customerId, c.age;
+-----+-----+
| customerId | age |
+-----+-----+
| 1 | 50 |
+-----+-----+

Set 1 property, matched 1 record, returned 1 records after 14 ms and completed after 10 ms.
```

```
neo4j> MATCH (c:Customer {customerId: 9999}) DETACH DELETE c;
Deleted 1 node, completed after 14 ms.
```

Ejecución con Docker

- Carpeta neo4j-docker/ con docker-compose.yml.
- Volúmenes: data, logs, import, plugins, conf.
- Montaje de neo4j/ como /scripts para ejecutar los .cypher directamente.
- Comandos principales:
 - docker compose up -d
 - cat neo4j/constraints.cypher | docker compose exec -T neo4j cypher-shell -u neo4j -p test123

Conclusiones

- El modelo de grafos captura de forma clara las relaciones cliente–producto–categoría y soporta consultas analíticas.
- Constraints e índices garantizan integridad y buen rendimiento en lecturas/actualizaciones.
- Las evidencias muestran que las operaciones CRUD funcionan con el dataset real.
- El setup Docker asegura reproducibilidad para cualquier integrante del equipo.
- El dataset es una buena base para extensiones: recomendaciones, segmentación y análisis estacional.

¡Gracias!