Concepts for Specialised Databases

Graph Databases: Neo4j

Anna Llanza Carmona

Javier Llobet Rodriguez

Primero de todo hemos descargado e instalado Neo4j. Una vez ya disponíamos de todos los elementos para poder trabajar, empezamos a familiarizarnos con el entorno, creando nodos y relaciones, haciendo consultas y eliminando los nodos. Para ello hemos seguido un tutorial del propio Neo4j para así poder descubrir todas las funcionalidades que nos proporciona.

Después de esto, decidimos utilizar Python para realizar esta práctica y Cypher para hacer las consultas a la base de datos. A partir de este momento, empezamos a analizar cómo modelar las tablas en nodos y aristas para así poder decidir una estructura adecuada y eficiente para utilizar en las 4 queries proporcionadas. Para ello, vamos a comentar query por query, las decisiones que hemos ido tomando:

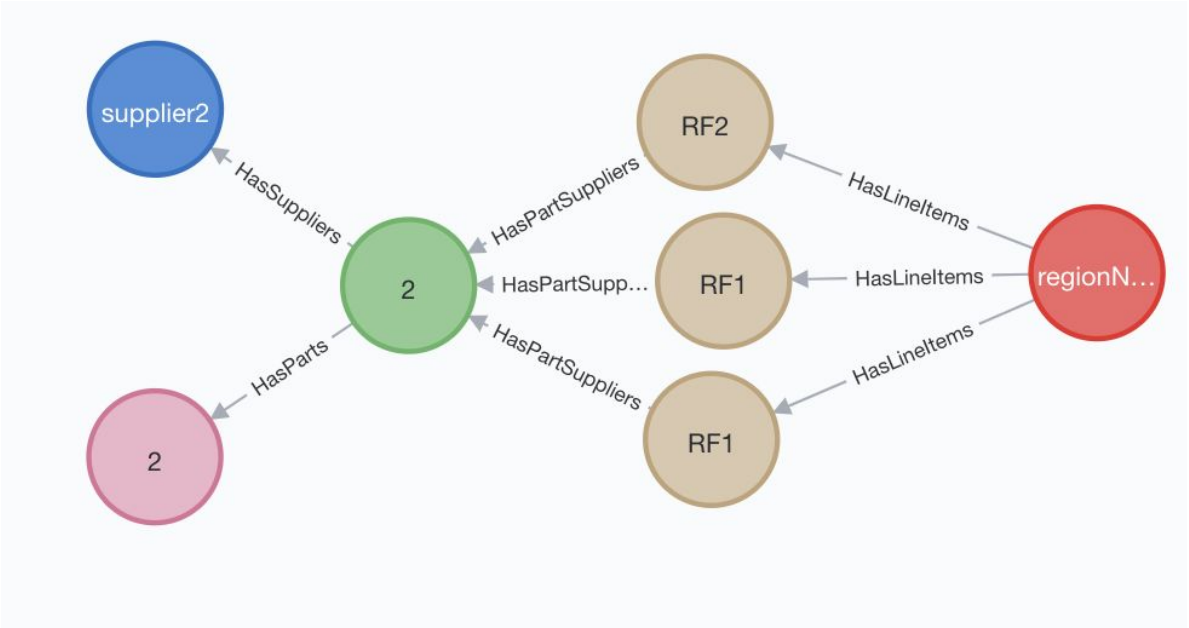
Para la primera query como solo utiliza la tabla LineItem, pensamos que la mejor manera de representarla con Neo4j era modelando con el nodo *LineItem y* con las propiedades necesarias para satisfacer la consulta.

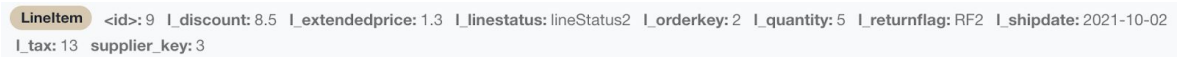
Para la segunda query, al utilizar las tablas Part, Supplier, PartSupplier, Nation y Region, pensamos que en vez de hacer 2 nodos para Nation y Region, añadiendo los atributos name de cada uno en Supplier ya era suficiente. Por lo tanto, hemos identificado los nodos *PartSupplier*, *Part* y *Supplier*. Cada nodo solamente incluye los atributos necesarios para satisfacer la consulta. Además, para modelarlos correctamente hemos creado las relaciones (aristas) siguientes:
(PartSupplier) - [:HasParts]-> (Part)
(PartSupplier) - [:HasSuppliers]-> (Supplier)

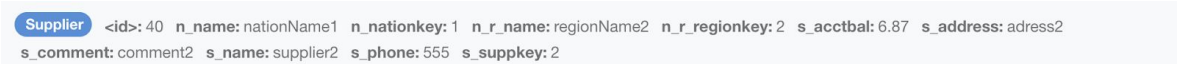
En la tercera query, se trabaja con las tablas Customer, Order i Lineitem. Analizando detalladamente diversas opciones, hemos acabado decidiendo que tendremos el nodo *Order* con todos los atributos necesarios. También, hemos sopesado la opción de crear un nodo para Customer, pero esta idea la hemos descartado ya que hemos considerado que añadiendo los atributos mk_segment, nation_name y region_name en *Order* sería suficiente para satisfacer las consultas. Además, hemos visto necesaria la creación de la relación HasLineItems para poder representar que LineItems contiene cada Order:
(Order) → [:HasLineItems]-> (LineItems)

En la cuarta query, se utilizan las tablas Customer, Orders, Lineitem, Supplier, Nation y Region. Al disponer de todos los nodos que hemos decidido crear, para poder conseguir

modelar esta query, hemos concluido que con definir las relaciones entre los nodos es suficiente. Con lo cual, hemos creado la relación entre *LineItem* con *PartSupplier*.
(LineItem) - [:HasPartSuppliers]-> (PartSupplier)

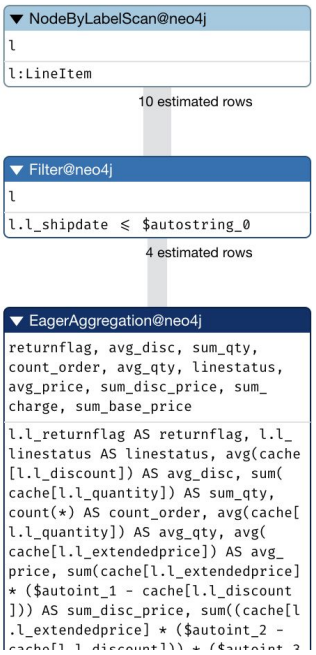




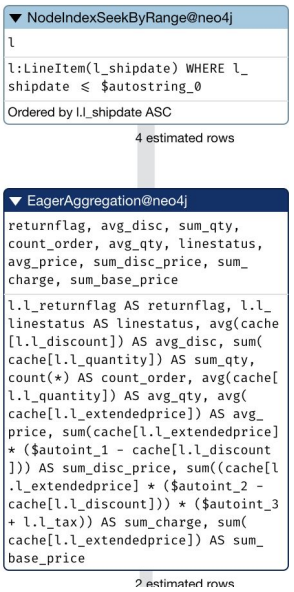


Consideraciones Índices

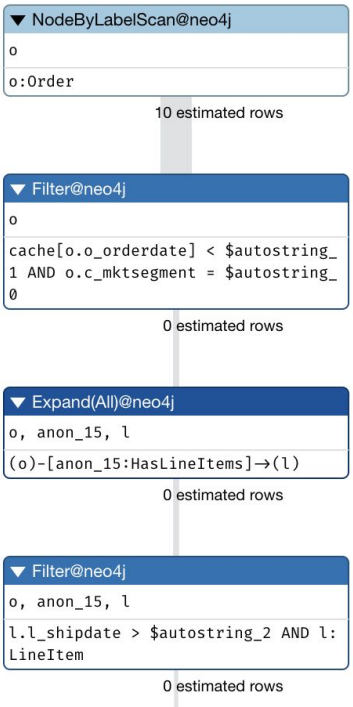
Hemos creado índices sobre los campos:
Order.c_mktsegment y Order.o_orderdate para mejorar la query 3 y 4 debido a que *Orders* tiene buen SF. También hemos creado otro índice sobre *LineItem.l_shipdate* para mejorar la query 1 y 3 debido a que Orders tiene buen SF.



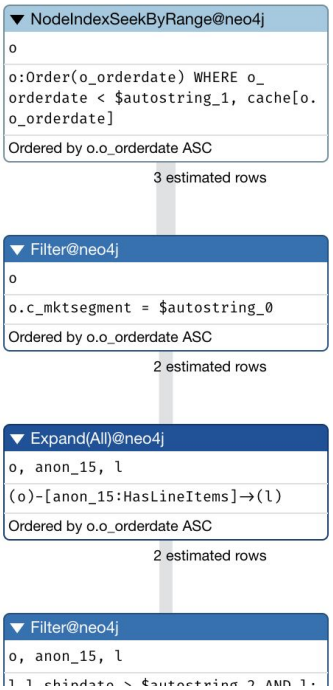
Plan para query1 sin indice



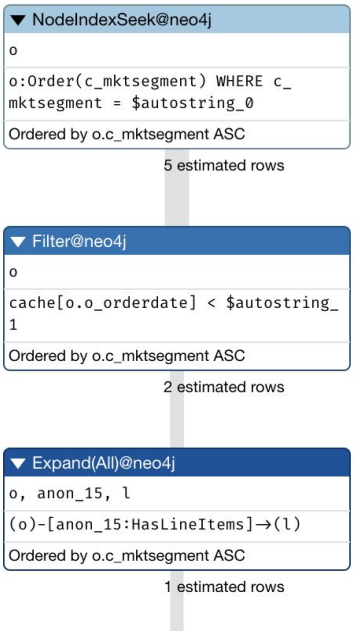
Plan para query1 con índice sobre *l_shipdate*



Query plan para query 3 sin indice



Query 3 con usando orderdate index



Query 3 con usando mktsegment index