6 LAB SESSION ON Graph Database

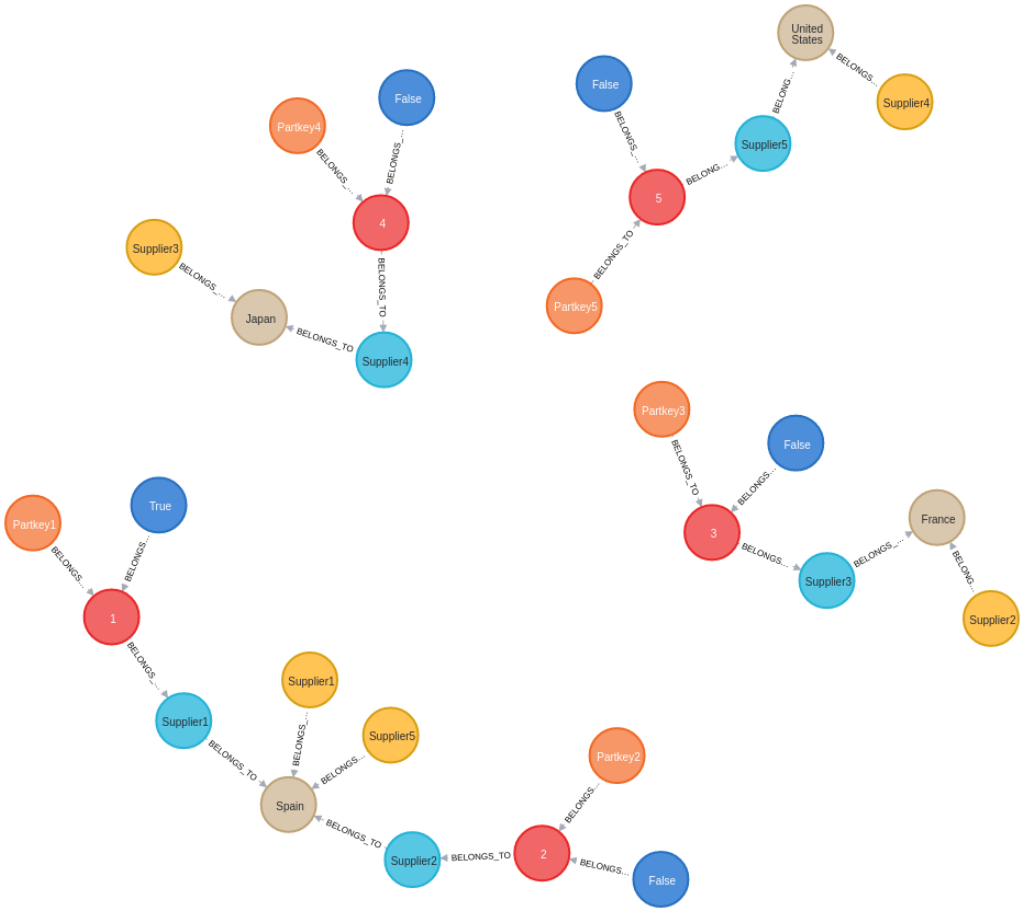
Given Name: Miquel **Family name:** Perelló Rodríguez
Given Name: Laia **Family name:** Ondoño Pujol

Estructura

Per la base de dades en Neo4j, hem decidit fer servir la següent estructura:

- Creació de nodes per a: Customer, Lineitem, Nation, Order, Part, PartSupp, Region i Supplier.
- Creació de relacions entre:
  - P → PS → S → N → R
  - C → O → L
  - C → N
  - L → PS

La representació gràfica d'aquest disseny és el següent:



On els colors representen:

Els índexs creats han sigut respecte als atributs l_shipdate de Lineitem, o_orderdate d'Order i ps_supplycost de PartSupp.

Explicació de l'estructura dissenyada

En aquest apartat veurem l'explicació detallada dels motius per a dissenyar el model anteriorment explicat.

Query 1

La primera query se centra exclusivament a la taula Lineitem. Fa un group by i order by pels atributs l_returnflag i l_linestatus, i busca valors per a instàncies de l_shipdate més petites o iguals que la data introduïda. Per aquest motiu, hem decidit crear un node per a lineitem, doncs no cal accedir a altres taules. A més, donat que l_shipdate s'utilitza constantment tant a aquesta query (és l'única comprovació del WHERE), com a la query 3, s'ha decidit crear un índex per a aquest atribut.

Query 2

Aquesta query combina 5 classes diferents: Part, Supplier, PartSupp, Nation i Region. En aquest cas, la clau està en veure com podem maximitzar el flux entre classes. Partint de la classe Part, podem anar navegant cap a la resta de nodes, fins a arribar a les regions corresponents. La query selecciona diversos atributs dels nodes que tenen el p_size, p_type y r_name dels valors introduïts per l'usuari. A més, també comprova que el valor de ps_suplycost sigui el mínim tots els de la regió escollida.

A la base de dades hi ha un node creat per cada classe utilitzada aquesta query. Creiem que aquesta és una bona opció perquè, tot i que podríem haver fet que els atributs de PartSupp es trobessin a la relació entre Part i Supp, amb la nostra estructura tenim un accés més directe als atributs de la query 4.

Hem considerat que la solució més òptima és tenir un índex per a l'atribut ps_supplycost, ja que com la query agafa el node amb el valor mínim per a una regió, tenir un índex sobre aquest atribut presenta un elevat factor de selectivitat.

Query 3

En aquesta tercera query accedim a les taules customer, orders i lineitem. Es filtra per el mktsegment de customer i es mira que la primera data introduïda sigui més gran que l'orderdate, i la segona més petita que el shipdate. Per aconseguir la màxima eficiència, ens hem d'adonar del Scale Factor. Si primer filtrem per mktsegment, estem partim de la base d'un scale factor *150.000. En canvi, si primer filtrem per orderdate, ens quedem amb un scale factor *1.500.000. El scale factor indica el nombre de files d'aquella taula que pot haver. Per tant, l'opció més eficient és crear tres nodes, un per cada taula, i crear relacions de Customer a Order i d'aquest a Lineitem. D'aquesta manera aconseguim filtrar per mktsegment, i de primeres treballem amb sf*150.000.

Query 4

Finalment, tenim la quarta query, que combina gairebé totes les classes que es troben a la base de dades. La seva estructura és similar a la query 3 però, en aquest cas, també té en compte alguns atribut de les classes Supplier, Nation i Region. Per a aquesta query, hem trobat adient començar des dels nodes de la classe Customer, ja que partint d'aquesta classe es pot navegar i accedir a la resta de classes involucrades en la query. Com a condicions, podem destacar que torna a aparèixer l'atribut o_orderdate, sobre el qual hi ha un índex creat, ja que també s'utilitza a la query 2. Amb aquest índex augmenta el rendiment de la query perquè evitem haver d'accedir a tots els nodes Order i, en canvi, només recorrerem (accedint directament i en ordre) els nodes que es troben en l'interval del WHERE.