6th Lab: Graph Databases (Neo4J). Document Explicatiu

Per aquesta pràctica de *graph databases*, un cop ens hem familiaritzat amb l'entorn de Neo4J, la seva sintaxis i les diferents eines que proporciona, el que fem es donat l'esquema del TPC-H benchmark dissenyem la base de dades a Neo4J, és a dir, dissenyem la base de dades en termes de graphs, nodes i edges.

Com es comú en el disseny d'una base de dades NoSQL necessitem el workload al que aquesta se sotmetrà. Aquest workload està format per 4 queries diferents que se'ns proporcionen en l'enunciat i que hem analitzat per tal de poder estructurar el graph de manera que aquestes queries s'executin de la forma més òptima posible només utilitzant un graph dissenyat.

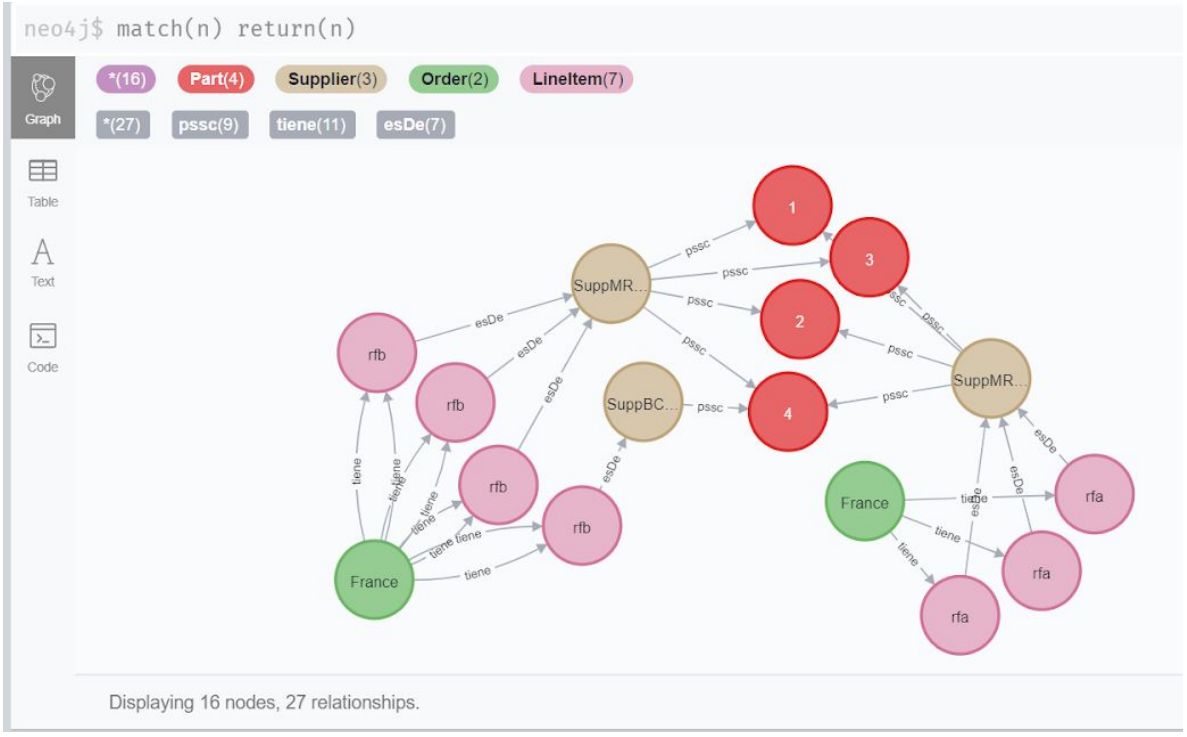

fig1. esquema de tot el graph database generat

Primer de tot hem decidit de crear un node **Order**, amb els atributs demandats per les diferents queries, que son o_orderkey, o_orderdate i o_shippriority. Aquest node conceptualment també va força per bé ja que com veurem més endavant ens servirà com a punt d'entrada al graph en les queries en que necessitem atributs d'Order i ens conectarà amb els següents nodes tot seguint les relacions (1-*) de l'esquema TPC-H benchmark proporcionat.

A més, al poder unir atributs de diferents entitats en un mateix node, hem decidit d'agrupar-hi també dins del node **Order** el atributs c_marketsegment (de **Customer**) i n_name (de **Nation**) ja que a més de que sovint es consulten aquests atributs a la vegada veiem que a partir de l'esquema relacional donat, veiem que la relació entre Orders i

Customers es de (1-*) respectivament, llavors només podrem tenir dins el node Order amb uns valors dels seus atributs determinats un posible valor per a l'atribut c_marketsegment. El mateix passa amb Nation i n_name (ja que un Customer només pot tenir un n_name associat).

A partir del node Order, com ja hem comentat, podrem navegar fins al node **LineItem** creat ja que la relació es de (1-*). Aquest node es un node que ens pot estalviar entrar al graph pel node Order en cas de que en la query demanada no s'utilitzi cap atribut d'Order. Tindrà únicament tots els atributs de LineItem que s'utilitzin en les 4 queries.
Aquest node per exemple és ideal per a l'execució de la query 1, ja que tot i que el major cost en una consulta en una graph database es troba en trobar el node pel qual començar a navegar en aquest cas, el de la query 1, ens estalviariem el petit cost de navegar (comparat amb el de la búsqueda del primer node), ja que només consulta atributs de LineItem.

Finalment hem creat els node vinculats amb les entitats de **Supplier** i **Part**. Aquests dos nodes els hem unit amb un edge que conté l'atribut **ps_supplycost**. En aquest aspecte determinat de la creació de l'edge entre Supplier i Part veiem una de les avantatges que pot tenir una graph database, ens estalviem les anomenades "bridge table" que usariem en una base de dades relacional, i que ara el joins més selectius tenen una millora en rendiment. A més, veiem que ara ja no necessitem un edge entre LineItem i Part ja que ara ens queda un graf connex i hi podem navegar per arribar a un o l'altre.
Els atributs atribuïts al node Part son els que requereixen les queries de l'entitat Part, en el node Supplier hi fem el mateix però afegint-hi igual que fem en el cas del node Order uns atributs addicionals, en concret el **n_name** i el **r_name**, ens estalviem edges i nodes addicionals i ho podem fer gràcies a que la relació entre Supplier, Nation i Region es una concatenació de (1-*), per tant, donat un Supplier només podrem tenir 1 n_name i 1 r_name. Per altra banda, encara que no sembli important, estem estalviant espai (Neo4j no fa swapping) i hem d'anar amb compte també amb la memòria, a la vegada, aquest mateix factor ens dona rapidesa al trobar els nodes en qüestió.

Finalment un cop decidit el model de graph database a realitzar hem decidit utilitzar dos indexs.
En concret un per al node **LineItem** al seu atribut **shipdate** i l'altre per al node **Order** en l'atribut **orderdate**. Tots dos atributs s'utilitzen com a filtratge de les diferents quereis en la sentencia WHERE d'aquestes.
Aquesta decisió ve presa a partir del model de graph database dissenyat i a partir de les 4 queries que formen el workload. Les 4 queries o bé inicien la seva consulta i navegació pel graph a partir del node Order o pel node LineItem; hem de tenir en compte que en una graph database el major cost d'una query vindrà donat per la localització del node d'entrada al graph, per tant, aquests son els que fer un ús de índexs es el més indicat així millorant el rendiment.