Trabajo Práctico Especial

El paradigma NoSQL - Bases de datos de Grafos

Alumno:

Wischñevsky, David 62494

Profesores:

Gomez, Leticia Irene

Vaisman, Alejandro Ariel

Índice

Setup

Para compilar el proyecto usando maven:

$ *mvn clean install*

Para correr el proyecto dentro del node 1:

$ spark-submit --master yarn --deploy-mode=cluster --class org.itba.tpe.Main --packages com.tinkerpop.blueprints:blueprints-core:2.6.0 --jars hdfs://node1/user/dwischnevsky/graphframes-0.8.0-spark2.4-s_2.11.jar hdfs://node1/user/dwischnevsky/original-tpe-1.jar hdfs:///user/dwischnevsky/my-routes-2.graphml

Consultas de Janusgraph

b.1) Indicar para aquellos aeropuertos que tengan valores de latitud y longitud negativos, cuáles van al aeropuerto SEA (Seattle) usando a lo sumo una escala, y cuál es esa forma de llegar.

graph.traversal().V().as('V1').hasLabel('airport').has('code', neq('SEA')).has('lat', lt(0)).has('lon', lt(0)).outE('route').inV().as('V2').hasLabel('airport').where('V1', neq('V2')).or(__.has('code', 'SEA'), __.outE('route').inV().hasLabel('airport').has('code', 'SEA')).path().map{ p -> def path = p.get(); def v1 = path.V1.values('code').next(); def v2 = path.V2.values('code').next(); v2.equals("SEA") ? v1 + " " + "No stop" + " " + v2 : v1 + " " + v2 + " " + "SEA" }.dedup()

b.2) Listar por cada continente y país, la lista de valores de las elevaciones de sus aeropuertos. Debe aparecer una sola tupla por cada continente y país con la agrupación de los valores de las elevaciones registradas.

graph.traversal().V().as('continent').hasLabel('continent').outE('contains').inV().as('a').inE('contains').outV().as('country').hasLabel('country').group().by(select('continent').values('desc')).by(group().by(select('country').valueMap('desc', 'code')).by(select('a').values('elev').order().fold())).unfold().map { def continent = it.get().key; def countries = it.get().value; countries.collect { country, elevations -> "${continent}\t${country['code'][0]} (${country['desc'][0]})\t[${elevations.join(', ')}]" } }.unfold()

Comparación resultados Graphframes y Janusgraph

Una forma sencilla de comparar los resultados (además de las triviales como contar las cantidades o comparar a ojo con archivos pequeños), es escribir los resultados de Janusgraph en un archivo de texto y luego simplemente correr un **diff** con los dos archivos generados por Graphframes.

La creación de archivos en Janushgraph se puede hacer wrappeando las queries con el método **withWriter** de la clase **File** y luego escribiendo cada resultado en una linea. Por ejemplo para la query 1:

gremlin> ***(new File("/home/dwischnevsky/output-1.txt")).withWriter { writer ->*** *graph.traversal().V().as('V1').hasLabel('airport').has('code', neq('SEA')).has('lat', lt(0)).has('lon', lt(0)).outE('route').inV().as('V2').hasLabel('airport').or(__.has('code', 'SEA'), __.outE('route').inV().hasLabel('airport').has('code', 'SEA')).path().map { p -> def path = p.get(); def v1 = path.V1.values('code').next(); def v2 = path.V2.values('code').next(); v2.equals("SEA") ? v1 + " " + "No stop" + " " + v2 : v1 + " " + v2 + " " + "SEA" }.dedup()****.each { writer.writeLine(it.toString()) } }***

Aclaración: es importante tener los permisos adecuados para la escritura del archivo. Estos pueden agregarse con el comando **chmod**.

Luego, alcanza con correr:

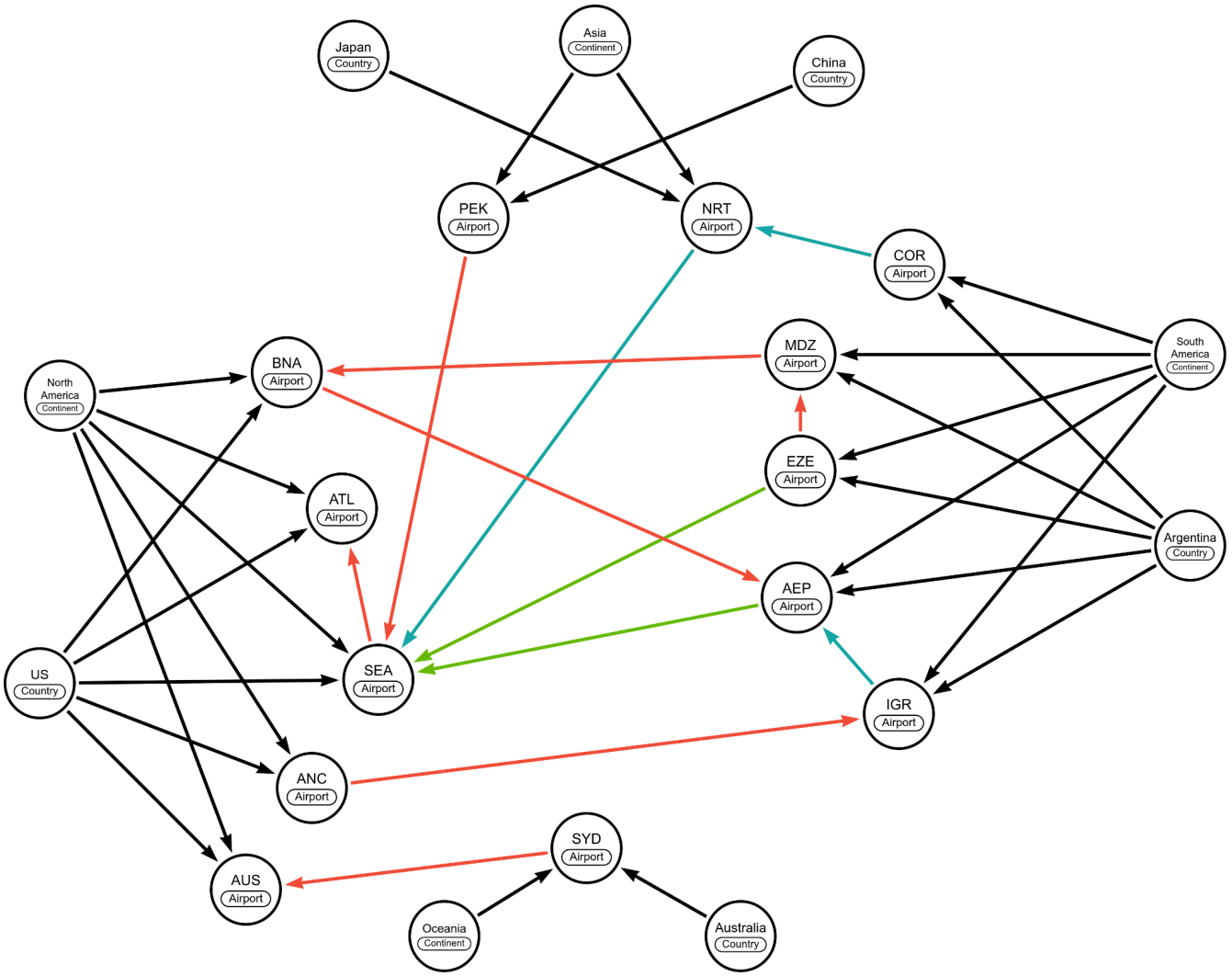
$ diff <(sort query1.txt) <(sort output-1.txt)

Donde *query1.txt* fue generado a partir del output del ejercicio 1 de Graphframes.

Grafos propuestos

Se proponen dos grafos:

1. **my-routes-1.graphml**: que cuenta con 50 aristas, pero pocos aeropuertos. Pensado para plantear un escenario similar a air-routes original en densidad de rutas (aunque con cierto bias a SEA como destino, para hacer la respuesta más interesante). Esto evidentemente hace que no sea fácil de interpretar, aunque sigue valiendo el criterio de comparación de resultados entre Janusgraph y Graphframes.
2. **my-routes-2.graphml**: que tiene solo 11 aristas, pero más aeropuertos en más países y continentes. Es más útil para poder verificar visualmente la correctitud de las consultas.



my-routes-2.graphml

En rojo se pueden ver las rutas que no son relevantes para la consulta 1, en verde las que llegan directamente, y en celeste las que llegan con una escala