In [1]:

```python
import csv
from neo4j import GraphDatabase
```

In [2]:

```python
#CLASE PAR CREAR NODO CARROS
class CLASE_NODO(object):
    def __init__(self):
        self._driver = GraphDatabase.driver("bolt:neo4j://localhost:7687", auth=("neo4j
    def close(self):
        self._driver.close()
    def CREAR_NODO(self, message,make,model,fuel_type,aspiration,num_of_doors,body_styl
                   engine_location,wheel_base,length,width,height,curb_weight,engine
                   engine_size,fuel_system,compression_ratio,horsepower,peak_rpm,cit
        with self._driver.session() as session:
            greeting = session.write_transaction(self._VALIDAR_NODO, message,make,model
                        engine_location,wheel_base,length,width,height,curb_weight,engine
                        engine_size,fuel_system,compression_ratio,horsepower,peak_rpm,cit
    #METODO PARA CREAR LOS NODOS CARROS
    @staticmethod
    def _VALIDAR_NODO(tx, message,make,model,fuel_type,aspiration,num_of_doors,body_sty
                      engine_location,wheel_base,length,width,height,curb_weight,engine
                      engine_size,fuel_system,compression_ratio,horsepower,peak_rpm,cit
        result = tx.run("CREATE("+make+":Carros {make:'"+make+"',model:'"+model+"',fuel
                        +"',num_of_doors:'"+num_of_doors+"',body_style:'"+body_style+"
                        +"',engine_location:'"+engine_location+"',wheel_base:'"+wheel_b
                        +"',height:'"+height+"',curb_weight:'"+curb_weight+"',engine_ty
                        "',engine_size:'"+engine_size+"',fuel_system:'"+fuel_system+"'
                        +"',horsepower:'"+horsepower+"',peak_rpm:'"+peak_rpm+"',city_mp
                        "SET "+make+".message = $message "
                        "RETURN "+make+".message + ', from node ' + id("+make+")", mess
```

In [3]:

```python
#SE INICIALIZA LA CLASE DE LOS METODOS DE NEO4J
grafo=CLASE_NODO()
with open('C:/Users/ADMINX/Documents/9ciclo/MATERIAS/DeberesQuisi/Tareas/SE/car_dataset
    reader = csv.reader(File)
    for row in reader:
        if str(row[1])=='make':
            pass
        elif str(row[23])!='?':
            grafo.CREAR_NODO("Se crea el nod0 "+row[0],str(row[1]).replace("-","_"),str
        elif str(row[23]=='?'):
            v = str(row[23]).replace("?","0")
            grafo.CREAR_NODO("Se crea el nod0 "+row[0],str(row[1]).replace("-","_"),str
```

In [39]:

```
1   # MATCH (n) OPTIONAL MATCH (n)-[r]-() DELETE n,r
2
3   #SE APLICA EL METODO DE KNN
4   #SE CREA EL GRAF DE LOS NODOS Y EL ATRIBUTO PARA LA SIMILTUD
5   CALL gds.graph.create(
6       'GrafoC',
7       {
8           Carros: {
9               label: 'Carros',
10              properties: 'price'
11          }
12      },
13      '*'
14  );
15  #CUANTO SE REQUERIERE DE MEMORIA
16  CALL gds.beta.knn.write.estimate('GrafoC', {
17    nodeWeightProperty: 'price',
18    writeRelationshipType: 'SIMILAR',
19    writeProperty: 'score',
20    topK: 1
21  })
22  YIELD nodeCount, bytesMin, bytesMax, requiredMemory
23
24  #SE EJECUTA EL ALGORITMO
25  CALL gds.beta.knn.stream('GrafoC', {
26      topK: 1,
27      nodeWeightProperty: 'price',
28      randomSeed: 42,
29      concurrency: 1,
30      sampleRate: 1.0,
31      deltaThreshold: 0.0
32  })
33  YIELD node1, node2, similarity
34  RETURN gds.util.asNode(node1).make AS Carros1, gds.util.asNode(node2).make AS Carros2,
35  ORDER BY similarity DESCENDING, Carros1, Carros2
36
37  #
38  CALL gds.beta.knn.write('GrafoC', {
39      writeRelationshipType: 'SIMILAR',
40      writeProperty: 'score',
41      topK: 1,
42      randomSeed: 42,
43      nodeWeightProperty: 'price'
44  })
45  YIELD nodesCompared, relationshipsWritten
```

```
  File "<ipython-input-39-bad8d3e794bf>", line 4
    CALL gds.beta.knn.write(
            ^
SyntaxError: invalid syntax
```