**CODICE UTILIZZATO SIA DA SCRIPT CHE DA SHELL:

```
package edu.unict.BigData

import org.apache.spark.{SparkConf,SparkContext}
import scala.math.random
import org.apache.spark.SparkContext._
import org.apache.spark.sql.{SparkSession, DataFrame}
import org.opencypher.morpheus.api.MorpheusSession
import org.opencypher.okapi.api.io.conversion.{NodeMappingBuilder,
RelationshipMappingBuilder}
import org.opencypher.okapi.api.value.CypherValue
import org.opencypher.morpheus.api.io.{MorpheusNodeTable,
MorpheusRelationshipTable, MorpheusElementTable}


object App {
  def main(args: Array[String]): Unit = {
    val spark = SparkSession
            .builder()
            .appName( name = "meyloma")
            .config("spark.master","local[*]")
            .getOrCreate()

    implicit val morpheus: MorpheusSession = MorpheusSession.local()
    import spark.sqlContext.implicits._

    val csvOptions = Map("header"->"true", "delimiter" -> “;”, "inferSchema" -> "true")

    val nodesDF = spark.read.options(csvOptions).csv("nodes_elab.csv")
    val edgesDF = spark.read.options(csvOptions).csv("edges_elab.csv")

    val NodeMapping =
NodeMappingBuilder.withSourceIdKey("id:ID").withImpliedLabel("Node").withPropert
yKey("names", "names").withPropertyKey("size", "size").withPropertyKey("labels",
"labels").withPropertyKey("rho", "rho").build
    val RelationMapping =
RelationshipMappingBuilder.withSourceIdKey("identific").withSourceStartNodeKey("
src:START_ID").withSourceEndNodeKey("dst:END_ID").withRelType("type").withPro
pertyKey("labels", "labels").withPropertyKey("mrho",
"mrho").withPropertyKey("tf_idf", "tf_idf").build
    val Node = MorpheusElementTable.create(NodeMapping, nodesDF)
    val Relation = MorpheusElementTable.create(RelationMapping, edgesDF)

    val Graph = morpheus.readFrom(Node, Relation)

    val results = Graph.cypher("MATCH (dis:disease {names:'multiple myeloma'})-
[r:BIO_VALUE_HIGH]->(drg:drug) RETURN dis, r, drg LIMIT 10;")
```

```
    results.show

    spark.stop()
  }
}
```

**QUERY TESTATE:

1) val results = Graph.cypher("MATCH (dis:disease {names:'multiple myeloma'})-[r:BIO_VALUE_HIGH]->(drg:drug) RETURN dis, r, drg LIMIT 10”) —> **OTTIENE RISULTATI**

2) MATCH p=(dis:disease)-[:BIO_VALUE_HIGH]->(gen:gene) WHERE dis.names =~ '(?i)my.*' RETURN p LIMIT 10; —> **Not yet able to convert expression: PathExpression(NodePathStep(Variable(dis),SingleRelationshipPathStep(Variable(UNNAMED22),OUTGOING,Some(Variable(gen)),NilPathStep)))**

3) MATCH p=(dis:disease)-[:BIO_VALUE_HIGH]->(gen:gene) WHERE dis.names CONTAINS 'io' RETURN p LIMIT 10; —> **Not yet able to convert expression: PathExpression(NodePathStep(Variable(dis),SingleRelationshipPathStep(Variable(UNNAMED22),OUTGOING,Some(Variable(gen)),NilPathStep)))**

4) WITH [‘multiple myeloma’] AS diseases, [‘ixazomib’, ‘auranofin’] AS drugs MATCH (dis:disease)<-[r:BIO_VALUE_HIGH]-(drg:drug) WHERE toLower(dis.names) IN diseases AND toLower(drg.names) IN drugs RETURN dis, r, drg; —> **OTTIENE RISULTATI**

5) MATCH p=(dis:disease {names:’multiple myeloma’})-[:BIO_VALUE_HIGH *1..3]->(gp:`gene:protein`) RETURN p LIMIT 10 —> **Not yet able to convert expression: PathExpression(NodePathStep(Variable(dis),MultiRelationshipPathStep(Variable(UNNAMED49),OUTGOING,Some(Variable(gp)),NilPathStep)))**

6) MATCH p=(dis1:disease {names:’multiple myeloma’})-[:BIO_VALUE_HIGH *1..3]->(drg1:drug), q=(dis1)-[:BIO_VALUE_HIGH *1..2]->(dis2:disease), r=(dis2)-[:BIO_VALUE_HIGH *1..3]->(drg2:drug), t=(drg2)-[:BIO_VALUE_HIGH *2..3]->(drg1) RETURN p,q,r,t LIMIT 1; —> **Not yet able to convert expression: NoneIterablePredicate(FilterScope(Variable(UNNAMED148),Some(AnyIterablePredicate(FilterScope(Variable(UNNAMED195),Some(Equals(Variable(UNNAMED148),Variable(UNNAMED195)))),Variable(UNNAMED195)))),Variable(UNNAMED148))**

7) MATCH (dis:disease {names:’multiple myeloma’}), (gen:gene {names: ‘PARTICL’}), p= shortestPath((dis)-[:BIO_VALUE_HIGH *]->(gen)) RETURN p; —> **Support for pattern conversion of RelationshipChain(NodePattern(Some(Variable(dis)),Vector(LabelName(diseas**

e)),None,None),RelationshipPattern(Some(Variable(UNNAMED101)),List(RelTypeName(BIO_VALUE_HIGH)),Some(None),None,OUTGOING,false,None),NodePattern(Some(Variable(gen)),Vector(LabelName(gene)),None,None)) not yet implemented

8) MATCH (dis1:disease)<-[r1:BIO_VALUE_HIGH]-(gen1:gene)<-[r2:BIO_VALUE_HIGH]-(gen2:gene)<-[r3:BIO_VALUE_HIGH]-(dis1) WHERE gen1.names <> gen2.names WITH dis1,r1,gen1,r2,gen2,r3 LIMIT 5 MATCH (gen1)-[r4:BIO_VALUE_HIGH]-(drg:drug)-[r5:BIO_VALUE_HIGH]-(gen2) RETURN dis1,r1,gen1,r2,gen2,r3,drg,r4,r5 —> **OTTIENE RISULTATI**