EXAMEN MINERÍA DE GRAFOS

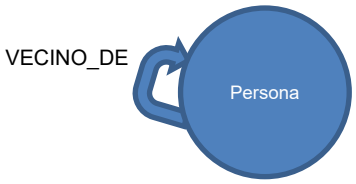
Nombre	Cordero Hernández Marco Ricardo	Expediente	727272	Carrera	Ingeniería en Sistemas Computacionales	Fecha	30 de noviembre del 2023

Yo **Marco Ricardo Cordero Hernández** al participar en este examen me comprometo a que lo entregado sea fruto solamente de mi trabajo, como una oportunidad de demostrar lo que he aprendido hasta el día de hoy. Además, reconozco que la copia es una falta a la honestidad académica y personal. Por lo que me comprometo a no incurrir en acciones que impliquen copiar o dejar que alguien me copie, ni participar en cualquier tipo de intercambio de información por cualquier medio con los estudiantes inscritos a este curso o personas ajenas al mismo.

Responde las preguntas dejando la evidencia solicitada en este documento

Modela un grafo que incluya la información de los archivos Persona.csv y Relaciones.csv

1. Puedes hacer tu diagrama a mano, o en cualquier otra herramienta que elijas. **Evidencia un diagrama del grafo [10 puntos]**



Implementa el grafo utilizando el archivo los archivos Persona.csv y Relaciones.csv
Evidencia archivo cql con el código cypher y la captura del esquema general del grafo [20 puntos]

2. En el archivo Persona.csv, tiene los datos de la persona
3. El archivo Relaciones.csv muestra las relaciones de amistad y el peso entre las personas

CÓDIGO

```
// Importar nodos e índices
LOAD CSV WITH HEADERS FROM "file:///Persona.csv" AS row
CREATE (p:Persona)
SET p.id = toInteger(row.IdPersona),
    p.nombre = row.Persona,
    p.colonia = row.Colonia;


CREATE INDEX index_p_name FOR (p:Persona) ON (p.nombre);
CREATE INDEX index_p_neigh FOR (p:Persona) ON (p.colonia);


// Importar relaciones
LOAD CSV WITH HEADERS FROM "file:///Relaciones.csv" AS row
MATCH (pr1:Persona), (pr2:Persona)
WHERE pr1.id = toInteger(row.IdPersona1)
AND pr2.id = toInteger(row.IdPersona2)
MERGE (pr1)-[:VECINO_DE {weight: toInteger(row.Peso)}]->(pr2);
```

GRAFO CREADO

Analiza el grafo

4. Realiza una consulta en cypher que muestre el nombre de las personas que viven en las colonias MODERNA y MONRAZ. Los datos para mostrar son la colonia, el total de personas y como lista el nombre de las personas. **Evidencia código cypher y la captura de pantalla de la respuesta [10 puntos]**

```
MATCH (p:Persona)
WHERE p.colonia = 'MODERNA' OR p.colonia = 'MONRAZ'
RETURN p.colonia AS Colonia,
       COUNT(DISTINCT p.nombre) AS TotalPersonas,
       COLLECT(DISTINCT p.nombre) AS Personas;
```

Colonia	TotalPersonas	Personas
"MODERNA"	3	["FLIX SASTRE ROLDAN", "NOELIA DE OTERO", "ERASMO DE PUGA"]

5. ¿Cuál es el número de saltos que deben darse de GALA PERALTA con las demás personas alcanzables?, ¿Cómo puedes validar tu respuesta? **Evidencia código cypher y la captura de pantalla de la respuesta [10 puntos]** *(CAMBIADA A CUÁNTOS NODOS ALCANZA A 3 SALTOS)*

```
MATCH (p:Persona)
MATCH r=(p)-[*1..3]->()
WHERE p.nombre = 'GALA PERALTA'
RETURN COUNT(DISTINCT r) AS CantidadAlcanzables;
```

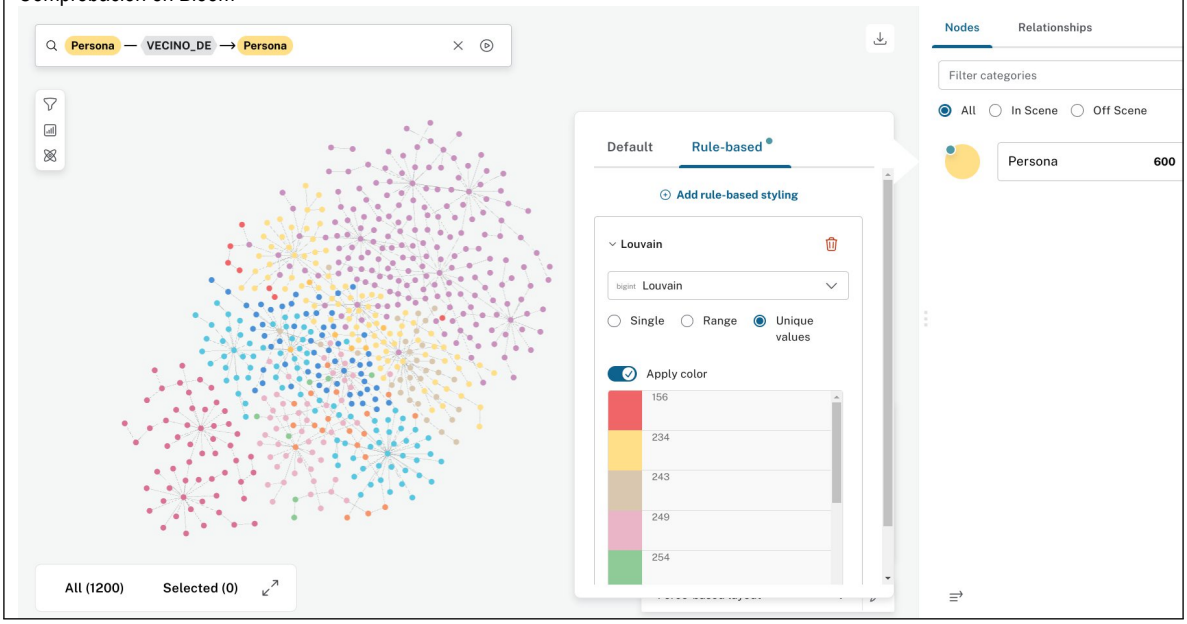
CantidadAlcanzables
3

6. Usando el algoritmo de comunidad *Louvain*, modifica la estructura del grafo para que implementes un nuevo atributo de comunidad con el nombre del algoritmo. **Evidencia código cypher y la captura de pantalla de la respuesta [10 puntos]**

```
// Creación del subgrafo
CALL gds.graph.project(
    'LOUVAIN_DIR',
    'Persona',
    {
        VECINO_DE: {
            orientation: 'NATURAL'
        }
    }
);


// Ejecución del algoritmo
CALL gds.louvain.stream('LOUVAIN_DIR')
YIELD nodeId, communityId, intermediateCommunityIds
RETURN gds.util.asNode(nodeId).nombre AS Persona, communityId, intermediateCommunityIds
ORDER BY communityId DESC;


// Escritura de los resultados
CALL gds.louvain.write(
    'LOUVAIN_DIR',
    { writeProperty: 'Louvain' }
)
YIELD communityCount, modularity, modularities;

// Agrupación (Miembros completos omitidos por brevedad)
MATCH (p:Persona)
RETURN p.Louvain AS Grupo, COLLECT(p.nombre)[1..4] AS Miembros;
```

Grupo	Miembros
588	["GRACIANO PINA CRESPO", "VALERIO SERNA AMAT", "EPIFANIO MADRID ABRIL"]
355	["ROLANDO BUSTOS CARVAJAL", "MARTIN DE POLO", "XIOMARA OJEDA-NARANJO"]
234	["PÁNFILO PONCIO REY ARRIBAS", "MAR CAZORLA HUGUET", "HILDA DUARTE-MATEO"]
517	["VICTORIA BARRIGA MORILLO", "IRMA DE CORTÉS", "CONSUELO VALENCIA DELGADO"]
249	["JOSÉ LUIS MIRANDA GUAL", "ANDREA PEDRAZA VILANOVA", "JEREMÍAS VALLEJO MALDONADO"]
416	["VALENTINA AZORIN MONTSERRAT", "DANIEL MADRIGAL CARBONELL", "ELEUTERIO PERALTA PAZ"]

Comprobación en Bloom



7. Usando el algoritmo de centralidad *PageRank*, modifica la estructura del grafo para que implementes un atributo de centralidad con el nombre del algoritmo. **Evidencia código cypher y la captura de pantalla de la respuesta [10 puntos]**

```
// Creación del subgrafo
CALL gds.graph.project(
    'PageRank',
    'Persona',
    'VECINO_DE',
    {relationshipProperties: 'weight'}
);

// Ejecución del algoritmo
CALL gds.pageRank.stream('PageRank')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).nombre AS Persona, score
ORDER BY score DESC, Persona ASC;

// Escritura de los resultados (PageRank tradicional)
CALL gds.pageRank.write(
    'PageRank',
    {
        maxIterations: 20,
        dampingFactor: 0.85,
        writeProperty: 'pageRank'
    }
```

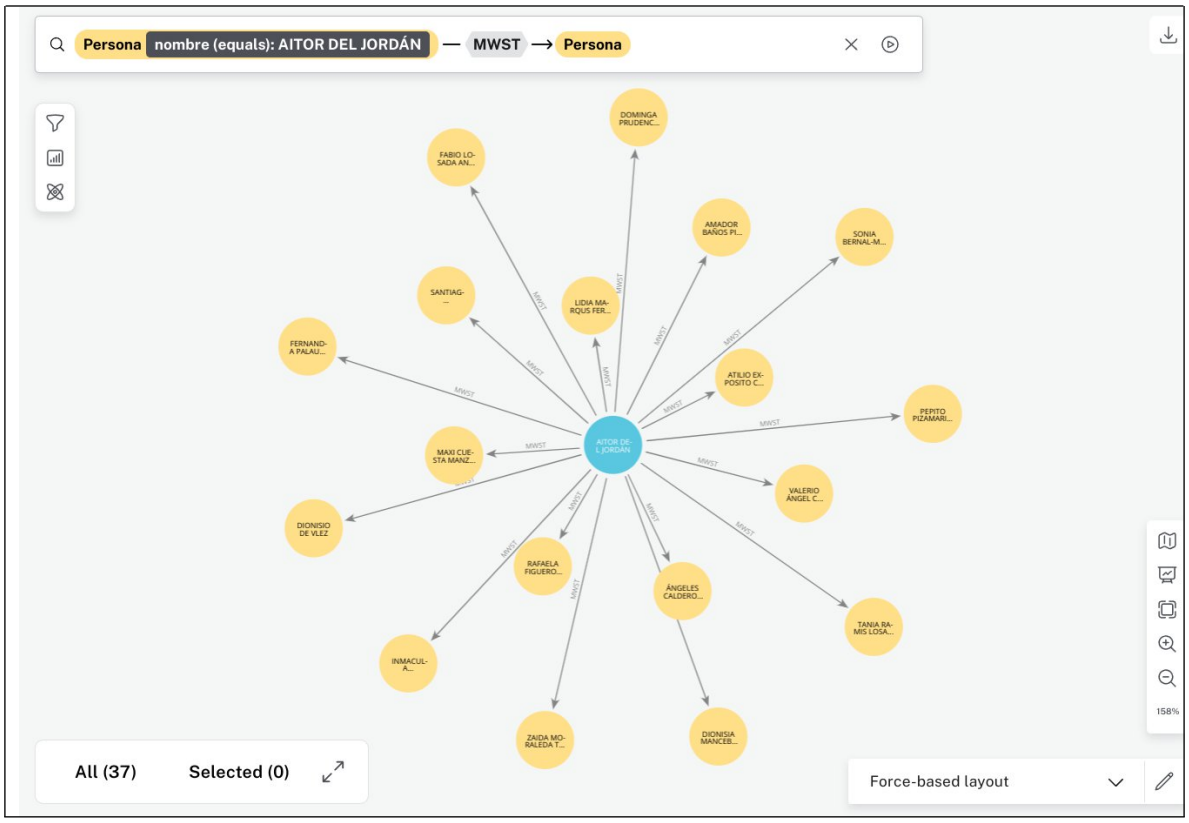
```
)
YIELD nodePropertiesWritten, ranIterations;

// Muestra de los resultados
MATCH (p:Persona)
RETURN p.nombre AS Persona, p.pageRank AS Influencia
ORDER BY Influencia DESC;
```

	Persona	Influencia
1	"AITOR DEL JORDÁN"	58.2337160965981
2	"TANIA RAMIS LOSADA"	56.433299880734396
3	"MARGARITA MARIN OLIVA"	18.732890128178855
4	"MODESTO VILALTA MUOZ"	15.828771563119583
5	"ODALIS POMBO"	15.200405360475433

8. Usando el algoritmo de búsqueda de caminos *Minimum Weight Spanning Tree*, modifica la estructura del grafo para que muestre el camino desde la persona con mayor *PageRank* a las demás personas alcanzables. **Evidencia código cypher y la captura de pantalla de la ruta [10 puntos]**

Dato preliminar —> Persona con mayor PageRank = 'AITOR DEL JORDÁN' (Obtenido en tiempo de ejecución)

```
// Creación del subgrafo
CALL gds.graph.project(
    'MWST',
    'Persona',
    {
        VECINO_DE: {
            orientation: 'UNDIRECTED',
            Properties: 'weight'
        }
    }
);

// Escritura de los resultados
CALL {
    MATCH (p:Persona)
    RETURN p.nombre AS n, p.pageRank AS pageRank
    ORDER BY pageRank DESC
    LIMIT 1
} WITH n AS name
MATCH (source:Persona {nombre: name})
CALL gds.spanningTree.write(
    'MWST',
    {
        sourceNode: ID(source),
        relationshipWeightProperty: 'weight',
        writeProperty: 'writeCost',
        writeRelationshipType: 'MWST'
    }
)
YIELD preProcessingMillis, computeMillis, writeMillis, effectiveNodeCount
RETURN preProcessingMillis, computeMillis, writeMillis, effectiveNodeCount;
```



9. Si elegimos a HUMBERTO PERA, ¿cuáles serían los tres amigos para recomendar (que actualmente no tengan la relación amigo) por similaridad? **Evidencia código cypher y la captura de pantalla de la respuesta [10 puntos]**

```
// Creación del subgrafo
CALL gds.graph.project(
    'Similarity',
    ['Persona', 'Persona'],
    {
        VECINO_DE: {
            properties: 'weight'
        }
    }
);


// Ejecución del algoritmo
CALL gds.nodeSimilarity.stream('Similarity')
YIELD node1, node2, similarity
RETURN
    gds.util.asNode(node1).nombre AS Persona1,
    gds.util.asNode(node2).nombre AS Persona2,
    similarity
ORDER BY similarity DESC, Persona1, Persona2;


// Escritura de los resultados
CALL gds.nodeSimilarity.write(
    'Similarity',
    {
        writeRelationshipType: 'SIMILAR',
        writeProperty: 'score'
    }
)
YIELD nodesCompared, relationshipsWritten;

// Sugerencia de amistad
MATCH (pr1:Persona)-[:VECINO_DE]->(pi:Persona)<--[s:SIMILAR]-(pr2:Persona)
```

```
WHERE pr1.nombre = 'HUMBERTO PERA'
AND s.score ≥ 0.75
AND pr1 <> pr2
AND pi <> pr2
RETURN pr1.nombre AS PersonaInicial,
       COLLECT(DISTINCT pr2.nombre)[0..3] AS Sugerencias;
```

PersonaInicial	Sugerencias
"HUMBERTO PERA"	["FABIO LOSADA ANGUITA", "AMADOR BAÑOS PIQUER", "FERNANDA PALAU ARAGÓN"]

10. Si elegimos a FLORENCIA ARMAS, ¿cuáles serían los tres amigos para recomendar (que actualmente no tengan la relación amigo) por comunidad? **Evidencia código cypher y la captura de pantalla de la respuesta [10 puntos]**

```
MATCH (pr1:Persona)-[:VECINO_DE]->(pi:Persona)-[:VECINO_DE]-(pr2:Persona)
WHERE pr1.nombre = 'FLORENCIA ARMAS'
AND pr1.Louvain = pr2.Louvain
AND pr1 <> pr2
AND pi <> pr2
RETURN pr1.nombre AS PersonaInicial,
       COLLECT(DISTINCT pr2.nombre)[0..3] AS Sugerencias;
```

PersonaInicial	Sugerencias
"FLORENCIA ARMAS"	["EUFEMIA ESPINOSA GUARDIOLA", "ELIGIO DOMINGUEZ ZABALA", "HERNN CARB BERNAT"]