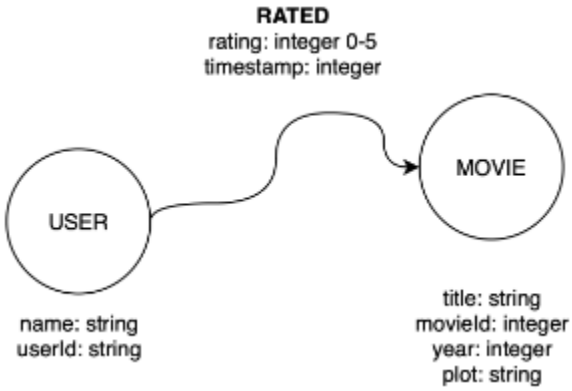
UVG | UNIVERSIDAD DEL VALLE DE GUATEMALA

CC3089, Base de datos 2
Sección 10

LABORATORIO 06
The Game of Nodes - Casa Lannister

Ejercicio 1 - Arquitectura de grafo



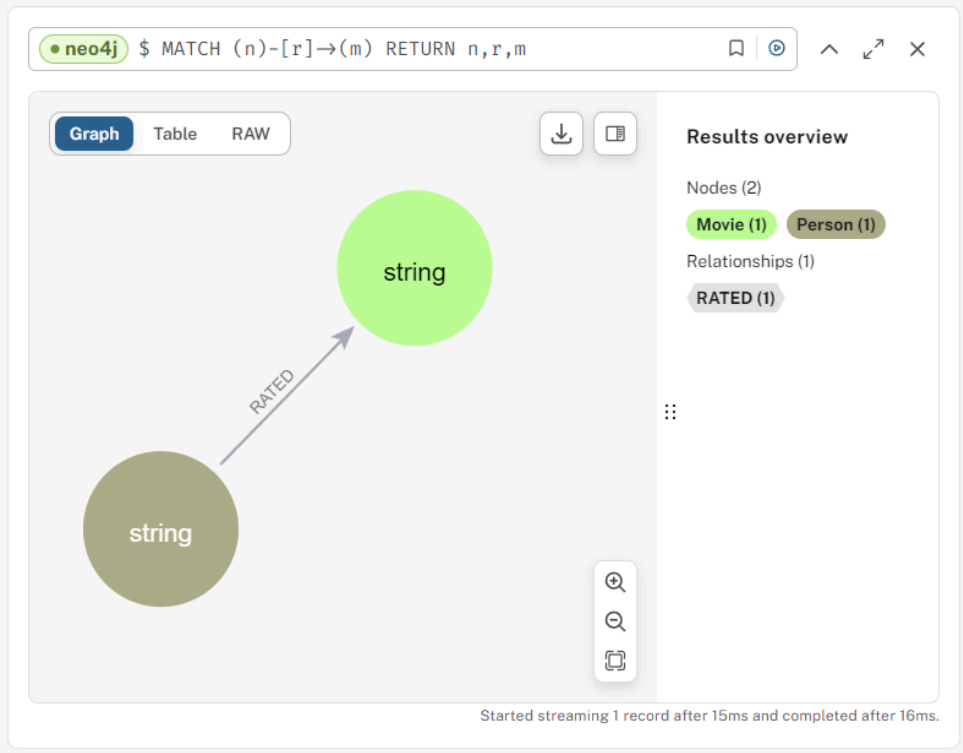
1. En el lenguaje de su preferencia, implementar funciones para crear el grafo (la idea es que el grafo se cree a partir de funciones que reciban parámetros)
   Funciones:

```python
def create_node(tx, node: Neo4jNode):
    query = f"""
        CREATE (n:{':'.join(node.labels)} $properties)
        RETURN n
        """
    result = tx.run(query, properties=node.properties)
    print(f'SUCCES: Node with labels: {node.labels} and properties: {node.properties} created')

    return result.consume()

def create_relationship(tx, relationship: Neo4jRelationship):
    # Extract the first property's key and value for both nodes
    node1_key, node1_value = next(iter(relationship.node1.properties.items()))
    node2_key, node2_value = next(iter(relationship.node2.properties.items()))

    query = (
        f"MATCH (a:{':'.join(relationship.node1.labels)}), "
        f"(b:{':'.join(relationship.node2.labels)}) "
        f"WHERE a.{node1_key} = $node1_value AND b.{node2_key} = $node2_value "
        f"CREATE (a)-[r:{relationship.type} $rel_properties]->(b) "
        f"RETURN type(r)"
    )

    # Execute the query, passing the first property's value for each node
    result = tx.run(query,
                    node1_value=node1_value,
                    node2_value=node2_value,
                    rel_properties=relationship.properties or {})
    print(f'SUCCESS: Relationship {relationship.type} created between {node1_value} and {node2_value}')

    return result.consume()
```

Resultado de consola:

```
(venv) C:\Users\diego\Documents\UVG\7mo Semestre\Base de Datos 2\Laboratorio6_DB>"c:/Users/diego/Documents/UVG/7mo Semestre/Base de Datos 2/
ratorio6_DB/main.py"
SUCCES: Connection to Neo4j instance established
SUCCES: Node with labels: ['Person'] and properties: {'name': 'string', 'userID': 'string'} created
SUCCES: Node with labels: ['Movie'] and properties: {'title': 'string', 'movieID': 'string', 'year': 'integer', 'plot': 'string'} created
SUCCESS: Relationship RATED created between string and string
```

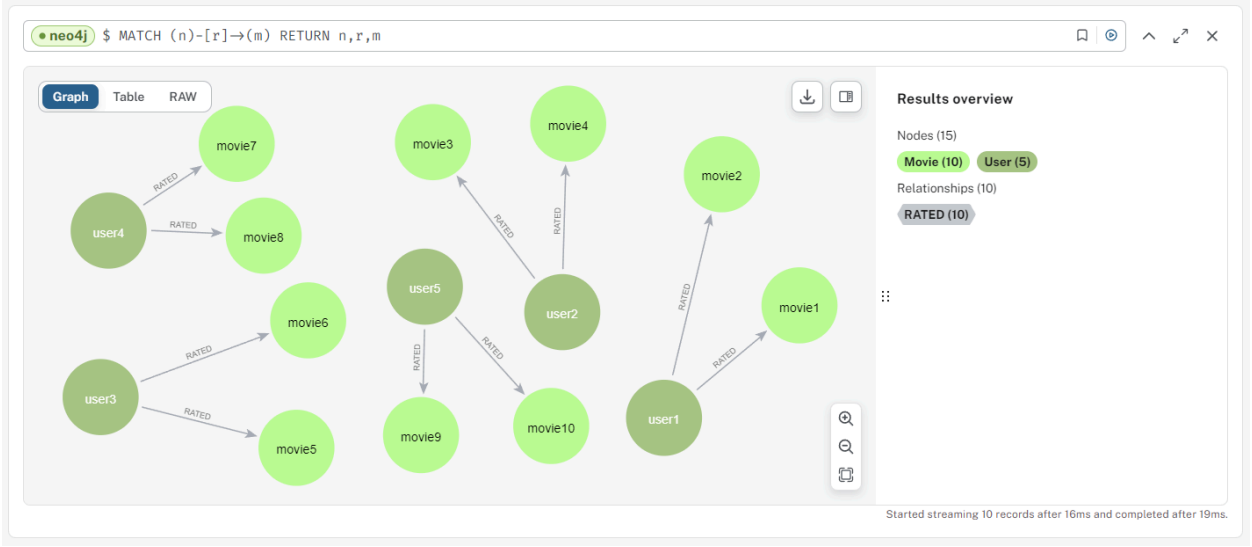
Resultado en Aura:



2. Popular el grafo con 5 registros (5 usuarios que tengan un rate hacia como mínimo dos películas distintas.)
   Output Consola:

```
(venv) C:\Users\diego\Documents\UVG\7mo Semestre\Base de Datos 2\Laboratorio6_DB>"c:/Users/diego/Documents/UVG/7mo Semestre/Base de Datos 2/Labo
ratorio6_DB/main.py"
SUCCES: Connection to Neo4j instance established
SUCCES: Node with labels: ['User'] and properties: {'name': 'user1', 'userID': 1} created
SUCCES: Node with labels: ['User'] and properties: {'name': 'user2', 'userID': 2} created
SUCCES: Node with labels: ['User'] and properties: {'name': 'user3', 'userID': 3} created
SUCCES: Node with labels: ['User'] and properties: {'name': 'user4', 'userID': 4} created
SUCCES: Node with labels: ['User'] and properties: {'name': 'user5', 'userID': 5} created
SUCCES: Node with labels: ['Movie'] and properties: {'title': 'movie1', 'movieID': 1, 'year': 2000, 'plot': 'plot of the movie 1'} created
SUCCES: Node with labels: ['Movie'] and properties: {'title': 'movie2', 'movieID': 2, 'year': 2004, 'plot': 'plot of the movie 2'} created
SUCCES: Node with labels: ['Movie'] and properties: {'title': 'movie3', 'movieID': 3, 'year': 2021, 'plot': 'plot of the movie 3'} created
SUCCES: Node with labels: ['Movie'] and properties: {'title': 'movie4', 'movieID': 4, 'year': 2001, 'plot': 'plot of the movie 4'} created
SUCCES: Node with labels: ['Movie'] and properties: {'title': 'movie5', 'movieID': 5, 'year': 2003, 'plot': 'plot of the movie 5'} created
SUCCES: Node with labels: ['Movie'] and properties: {'title': 'movie6', 'movieID': 6, 'year': 2015, 'plot': 'plot of the movie 6'} created
SUCCES: Node with labels: ['Movie'] and properties: {'title': 'movie7', 'movieID': 7, 'year': 2004, 'plot': 'plot of the movie 7'} created
SUCCES: Node with labels: ['Movie'] and properties: {'title': 'movie8', 'movieID': 8, 'year': 2017, 'plot': 'plot of the movie 8'} created
SUCCES: Node with labels: ['Movie'] and properties: {'title': 'movie9', 'movieID': 9, 'year': 2001, 'plot': 'plot of the movie 9'} created
SUCCES: Node with labels: ['Movie'] and properties: {'title': 'movie10', 'movieID': 10, 'year': 2004, 'plot': 'plot of the movie 10'} created
SUCCESS: Relationship RATED created between user1 and movie1
SUCCESS: Relationship RATED created between user1 and movie2
SUCCESS: Relationship RATED created between user2 and movie3
SUCCESS: Relationship RATED created between user2 and movie4
SUCCESS: Relationship RATED created between user3 and movie5
SUCCESS: Relationship RATED created between user3 and movie6
SUCCESS: Relationship RATED created between user4 and movie7
SUCCESS: Relationship RATED created between user4 and movie8
SUCCESS: Relationship RATED created between user5 and movie9
SUCCESS: Relationship RATED created between user5 and movie10
```

Resultado en Aura:



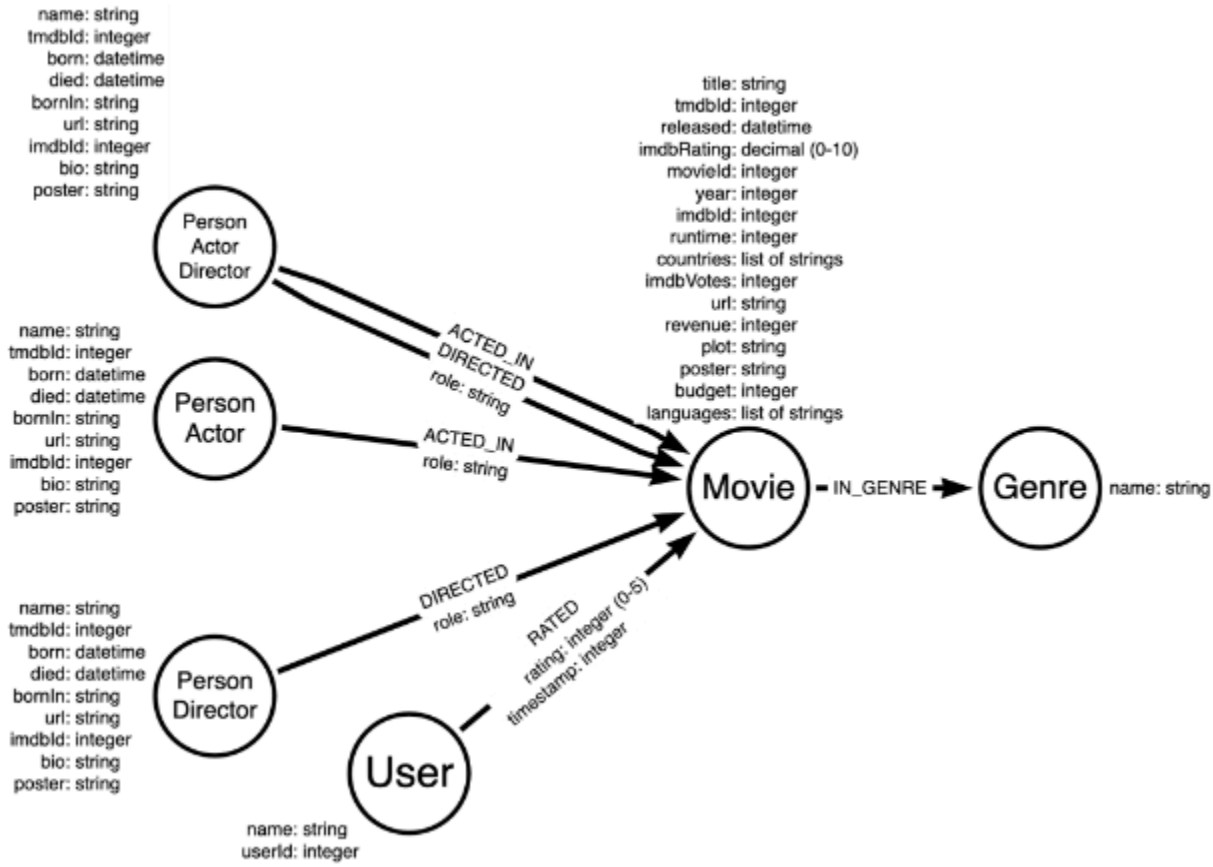
3. Implementar función para encontrar un usuario, una película y un usuario con su relación rate a película.
   Funciones de búsqueda:

```python
def find_user(tx, user_id=None, user_name=None):
    if user_id:
        query = "MATCH (u:User) WHERE u.userID = $user_id RETURN u"
        result = tx.run(query, user_id=user_id)
    elif user_name:
        query = "MATCH (u:User) WHERE u.name = $user_name RETURN u"
        result = tx.run(query, user_name=user_name)
    else:
        raise ValueError('User ID or User Name is required')

    users = [record["u"] for record in result]
    if not users:
        return f"No user found with {'ID' if user_id else 'name'}: {user_id if user_id else user_name}"
    return f"User(s) found: {', '.join([str(user['name']) for user in users])}"


def find_movie(tx, movie_id=None, movie_title=None):
    if movie_id:
        query = "MATCH (m:Movie) WHERE m.movieID = $movie_id RETURN m"
        result = tx.run(query, movie_id=movie_id)
    elif movie_title:
        query = "MATCH (m:Movie) WHERE m.title = $movie_title RETURN m"
        result = tx.run(query, movie_title=movie_title)
    else:
        raise ValueError('Movie ID or Movie Title is required')

    movies = [record["m"] for record in result]
    if not movies:
        return f"No movie found with {'ID' if movie_id else 'title'}: {movie_id if movie_id else movie_title}"
    return f"Movie(s) found: {', '.join([str(movie['title']) for movie in movies])}"


def find_user_rating(tx, user_id, movie_id):
    query = """
    MATCH (u:User)-[r:RATED]->(m:Movie)
    WHERE u.userID = $user_id AND m.movieID = $movie_id
    RETURN u, r, m
    """
    result = tx.run(query, user_id=user_id, movie_id=movie_id)
    ratings = [{"user": record["u"], "rating": record["r"], "movie": record["m"]} for record in result]
    if not ratings:
        return f"No rating found for user ID {user_id} on movie ID {movie_id}"
    return f"Rating found: User {ratings[0]['user']['name']} rated movie {ratings[0]['movie']['title']} with {ratings[0]['rating']['rating']} stars"
```

   Output de Consola:

```
(venv) C:\Users\diego\Documents\UVG\7mo Semestre\Base de Datos 2\Labor
SUCCES: Connection to Neo4j instance established
SUCCES: Node with labels: ['User'] and properties: {'name': 'user1', '
SUCCES: Node with labels: ['Movie'] and properties: {'title': 'movie1'
SUCCESS: Relationship RATED created between user1 and movie1
User(s) found: user1

Movie(s) found: movie1

Rating found: User user1 rated movie movie1 with 3 stars
```

4. Ahora, con las funciones previamente creadas, deberá generar el siguiente grafo. Es importante que tome en cuenta todas las propiedades de cada label y de aquellas relaciones que tengan alguna



Creación de grafo en python:

```
with GraphDatabase.driver(URI, auth=AUTH) as driver:
    driver.verify_connectivity()
    print('SUCCES: Connection to Neo4j instance established')
    with driver.session() as session:
        # Creacion de nodo persona actor director (PAD)
        node_PAD = Neo4jNode(labels=['Person', 'Actor', 'Director'],
                             properties={'name': 'Tom Hanks',
                                         'tmdbId': 31,
                                         'born': datetime.strptime('01-01-1956', '%d-%m-%Y'),
                                         'died': 'Not Dead',
                                         'bornIn': 'Concord, California, USA',
                                         'url:': 'tomhanks.com',
                                         'imdbId': 158,
                                         'bio':'Tom Hanks bio Here',
                                         'poster': 'tomhanks.jpg'})
        create_node(session, node_PAD)
```

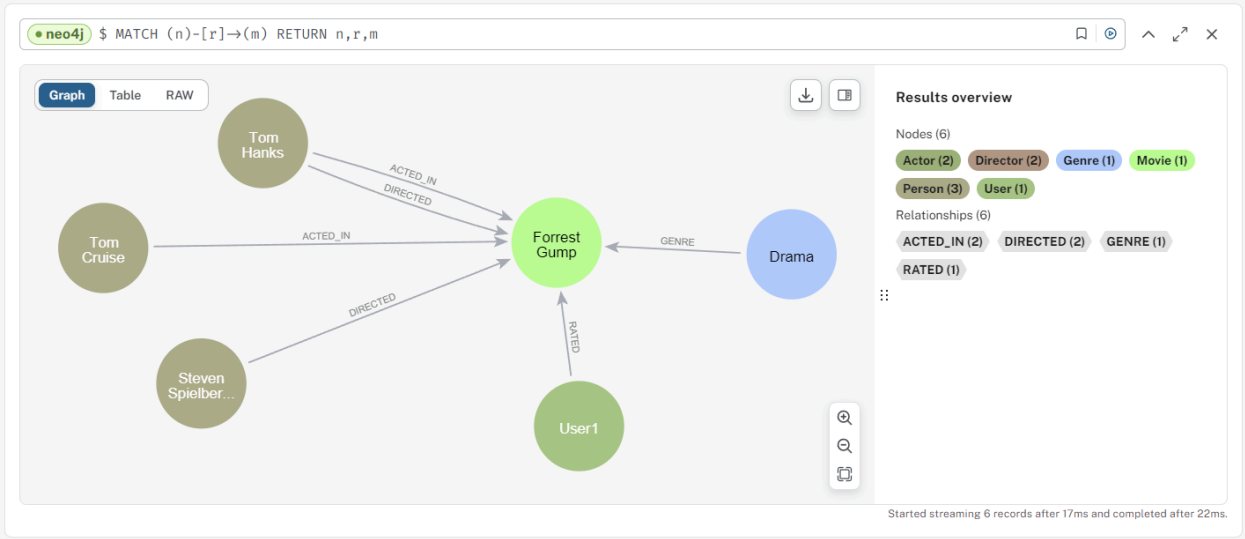
```python
# Creacion de nodo persona actor (PA)
node_PA = Neo4jNode(labels=['Person', 'Actor'],
                    properties={'name': 'Tom Cruise',
                                'tmdbId': 500,
                                'born': datetime.strptime('01-01-1962', '%d-%m-%Y'),
                                'died': 'Not Dead',
                                'bornIn': 'Syracuse, New York, USA',
                                'url:': 'tomcruise.com',
                                'imdbId': 500,
                                'bio':'Tom Cruise bio Here',
                                'poster': 'tomcruise.jpg'})
create_node(session, node_PA)

# Creacion de nodo persona director (PD)
node_PD = Neo4jNode(labels=['Person', 'Director'],
                    properties={'name': 'Steven Spielberg',
                                'tmdbId': 488,
                                'born': datetime.strptime('01-01-1946', '%d-%m-%Y'),
                                'died': 'Not Dead',
                                'bornIn': 'Cincinnati, Ohio, USA',
                                'url:': 'stevenspielberg.com',
                                'imdbId': 488,
                                'bio':'Steven Spielberg bio Here',
                                'poster': 'stevenspielberg.jpg'})
create_node(session, node_PD)

# Creacion de nodo usuario (User)
node_user = Neo4jNode(labels=['User'],
                      properties={'name': 'User1',
                                  'userID': 1})
create_node(session, node_user)

# Creacion de nodo pelicula (Movie)
node_movie = Neo4jNode(labels=['Movie'],
                       properties={'title': 'Forrest Gump',
                                   'tmdbId': 13,
                                   'released': datetime.strptime('01-01-1994', '%d-%m-%Y'),
                                   'imdbRating': 8,
                                   'movieID': 1,
                                   'year': 1994,
                                   'imdbId': 13,
                                   'runtime': 142,
                                   'countries': ['USA', 'India'],
                                   'imdbVotes': 1800,
                                   'url': 'forrestgump.com',
                                   'revenue': 1000000,
                                   'plot': 'Forrest Gump plot Here',
                                   'poster': 'forrestgump.jpg',
                                   'budget': 900000,
                                   'languages': ['English', 'Hindi']})
create_node(session, node_movie)

# Creacion de nodo genero (Genre)
node_genre = Neo4jNode(labels=['Genre'],
                       properties={'name': 'Drama'})
create_node(session, node_genre)
```

```python
# Creacion de relaciones entre nodos
# Relacion PAD con pelicula
relationship1 = Neo4jRelationship(node1=node_PAD,
                                  node2=node_movie,
                                  relationship='ACTED_IN',
                                  properties={'role': 'Forrest Gump'})
create_relationship(session, relationship1)
relationship2 = Neo4jRelationship(node1=node_PAD,
                                  node2=node_movie,
                                  relationship='DIRECTED',
                                  properties={'role': 'Visual Director'})
create_relationship(session, relationship2)

# Relaciones PA con pelicula
relationship3 = Neo4jRelationship(node1=node_PA,
                                  node2=node_movie,
                                  relationship='ACTED_IN',
                                  properties={'role': 'Forrest Gump'})
create_relationship(session, relationship3)

# Relaciones PD con pelicula
relationship4 = Neo4jRelationship(node1=node_PD,
                                  node2=node_movie,
                                  relationship='DIRECTED',
                                  properties={'role': 'General Director'})
create_relationship(session, relationship4)

# Relacion USER con pelicula
relationship5 = Neo4jRelationship(node1=node_user,
                                  node2=node_movie,
                                  relationship='RATED',
                                  properties={'rating': 5,
                                              'timestamp': 105})
create_relationship(session, relationship5)

# Relacion GENRE con pelicula
relationship6 = Neo4jRelationship(node1=node_genre,
                                  node2=node_movie,
                                  relationship='GENRE',
                                  properties={})
create_relationship(session, relationship6)
```

Output Consola:

```
(venv) C:\Users\diego\Documents\UVG\7mo Semestre\Base de Datos 2\Laboratorio6_DB>"c:/Users/diego/Documents/UVG/7mo Semestre/Base de Datos 2/Laboratorio6_DB/venv/Scripts/python.exe" "c:/Users/diego/Documents/UVG/7mo Semestre/Base de Datos 2/Laboratorio6_DB/main.py"
SUCCES: Connection to Neo4j instance established
SUCCES: Node with labels: ['Person', 'Actor', 'Director'] and properties: {'name': 'Tom Hanks', 'tmdbId': 31, 'born': datetime.datetime(1956, 1, 1, 0, 0), 'died': 'Not Dead', 'bornIn': 'Concord, California, USA', 'url:': 'tomhanks.com', 'imdbId': 158, 'bio': 'Tom Hanks bio Here', 'poster': 'tomhanks.jpg'} created
SUCCES: Node with labels: ['Person', 'Actor'] and properties: {'name': 'Tom Cruise', 'tmdbId': 500, 'born': datetime.datetime(1962, 1, 1, 0, 0), 'died': 'Not Dead', 'bornIn': 'Syracuse, New York, USA', 'url:': 'tomcruise.com', 'imdbId': 500, 'bio': 'Tom Cruise bio Here', 'poster': 'tomcruise.jpg'} created
SUCCES: Node with labels: ['Person', 'Director'] and properties: {'name': 'Steven Spielberg', 'tmdbId': 488, 'born': datetime.datetime(1946, 1, 1, 0, 0), 'died': 'Not Dead', 'bornIn': 'Cincinnati, Ohio, USA', 'url:': 'stevenspielberg.com', 'imdbId': 488, 'bio': 'Steven Spielberg bio Here', 'poster': 'stevenspielberg.jpg'} created
SUCCES: Node with labels: ['User'] and properties: {'name': 'User1', 'userID': 1} created
SUCCES: Node with labels: ['Movie'] and properties: {'title': 'Forrest Gump', 'tmdbId': 13, 'released': datetime.datetime(1994, 1, 1, 0, 0), 'imdbRating': 8, 'movieID': 1, 'year': 1994, 'imdbId': 13, 'runtime': 142, 'countries': ['USA', 'India'], 'imdbVotes': 1800, 'url': 'forrestgump.com', 'revenue': 1000000, 'plot': 'Forrest Gump plot Here', 'poster': 'forrestgump.jpg', 'budget': 900000, 'languages': ['English', 'Hindi']} created
SUCCES: Node with labels: ['Genre'] and properties: {'name': 'Drama'} created
SUCCESS: Relationship ACTED_IN created between Tom Hanks and Forrest Gump
SUCCESS: Relationship DIRECTED created between Tom Hanks and Forrest Gump
SUCCESS: Relationship ACTED_IN created between Tom Cruise and Forrest Gump
SUCCESS: Relationship DIRECTED created between Steven Spielberg and Forrest Gump
SUCCESS: Relationship RATED created between User1 and Forrest Gump
SUCCESS: Relationship GENRE created between Drama and Forrest Gump
```

Resultado en Aura:



Ejercicio 2 - Exploración de librerías y utilidades de Neo4j

- Agregaciones en Neo4j (Cypher) usando Count

  Documentación de la agregación COUNT:
  https://neo4j.com/docs/cypher-manual/current/functions/aggregating/#functions-count
  https://neo4j.com/developer/kb/fast-counts-using-the-count-store/

  En Cypher, count permite contar la cantidad de nodos que cumplen ciertas condiciones de un grafo.

  Utilicemos la siguiente metáfora para poder comprender la agregación de manera creativa: Imagina que estás organizando una fiesta con invitados y distintas actividades. Quieres analizar cuántos invitados asisten a cada actividad para asegurarte de que todos se están divirtiendo.

  Con lo anterior, tuviste la grandiosa idea de utilizar Cypher para contar cuántos invitados participan en cada actividad.

  La fiesta tiene las siguientes actividades:
  - Baile
  - Juegos de mesa
  - Charla en grupo

Por lo que quieres contar cuántos invitados están involucrados en cada una de las actividades y desarrollas los siguientes queries:

```
MATCH (i:Invitado)-[:PARTICIPA_EN]->(:Baile)
RETURN COUNT(i) AS InvitadosBailando

MATCH (i:Invitado)-[:PARTICIPA_EN]->(:JuegosDeMesa)
RETURN COUNT(i) AS InvitadosJugando

MATCH (i:Invitado)-[:PARTICIPA_EN]->(:CharlaEnGrupo)
RETURN COUNT(i) AS InvitadosCharlando
```

La agregación COUNT en Neo4j nos permite contar la cantidad de nodos (en este caso, invitados) que están relacionados con otros nodos (actividades) de una manera eficiente y poderosa. Así podemos asegurarnos de que todos los invitados están teniendo una experiencia satisfactoria en la fiesta.

Apliquemos los conceptos:

- La fiesta es un grafo en Neo4j, donde los nodos representan a los invitados y las relaciones entre los nodos representan las diferentes actividades en las que participan los invitados.
- Se usa la función COUNT para contar cuántos invitados están participando en cada actividad. Por ejemplo, podemos contar cuántos están bailando, cuántos están jugando juegos de mesa y cuántos están charlando en grupos.

En resumen, la función COUNT en Neo4j nos permite realizar un análisis resumido de nuestros datos, contando la cantidad de elementos que cumplen ciertas condiciones en nuestro grafo. Esto nos ayuda a obtener información clave sobre la distribución y participación de nuestros datos, como en el caso de la fiesta, donde queremos asegurarnos de que todos los invitados estén en las diferentes actividades.

Enlace a presentación:
https://www.canva.com/design/DAGB6Ln2v2s/PosGoF2cmeHcKLtfDfiFZQ/view?utm_content=DAGB6Ln2v2s&utm_campaign=designshare&utm_medium=link&utm_source=editor