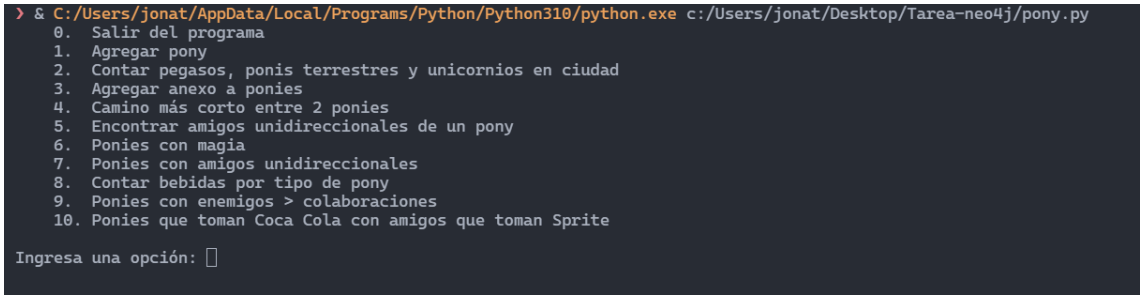
Bases de datos Avanzadas - Tarea 2

Alvaro Opazo - Jonathan Olivares

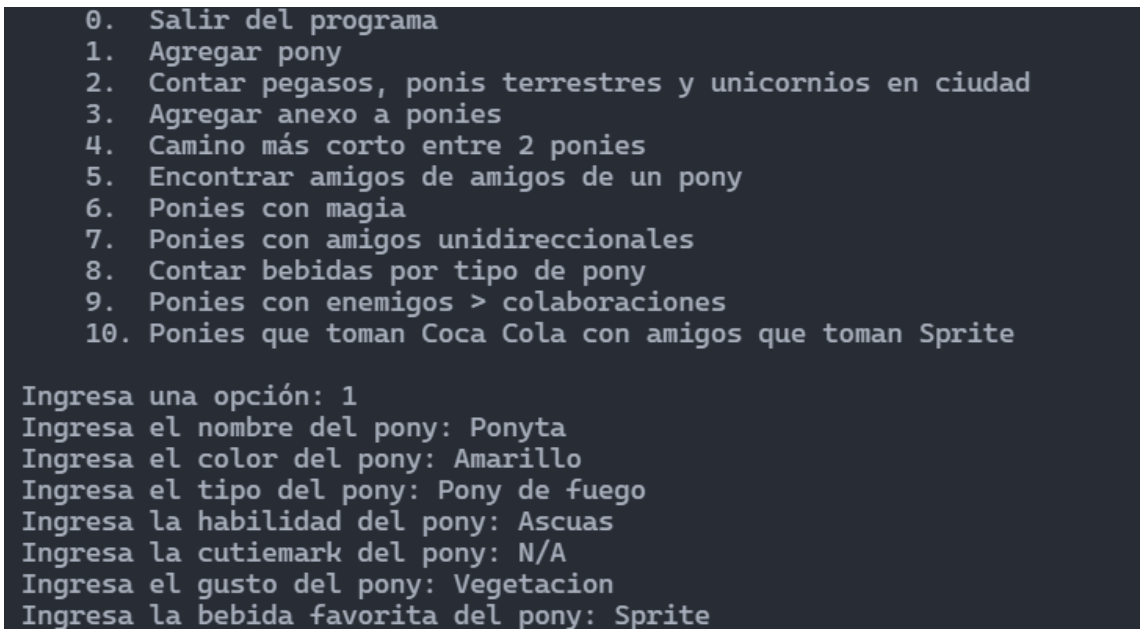
0 Menú de opciones

Para facilitar la interacción con la base de datos Neo4j, desarrollamos un menú de opciones que permite seleccionar diversas funcionalidades según sus necesidades.

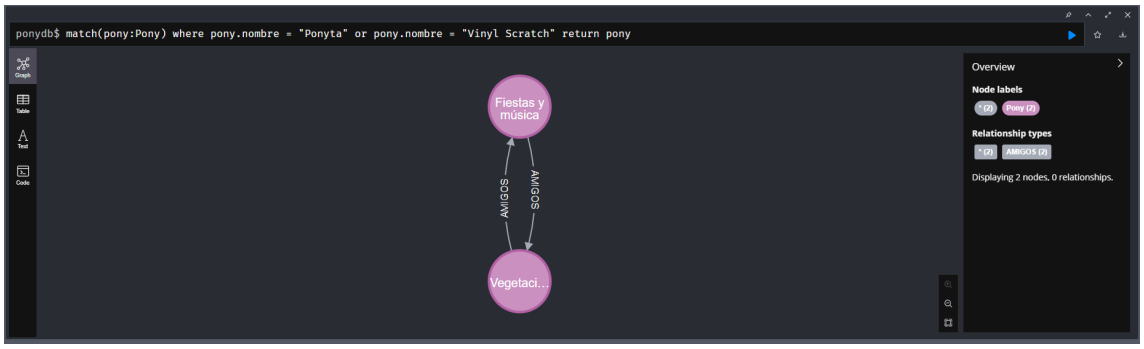


1 Agregar pony

Ingresamos la opción de agregar pony en el menú.



Aquí podemos notar su relación basada en grafos del explorador de Neo4j.



Esta fue la query utilizada.

```
query = """
MATCH (vinyl:Pony {nombre: 'Vinyl Scratch'})
CREATE (pony:Pony {
    nombre: $name, color: $color, tipo: $tipo,
    habilidad: $habilidad, cutiemark: $cutiemark,
    gusto: $gusto, bebida: $bebida}),
(pony)←[:AMIGOS]-(vinyl),
(pony)-[:AMIGOS]→(vinyl)
"""
```

2 Ciudad - Cantidad

Ingresamos la opción de contar pegasos, ponies terrestres y unicornios junto a la ciudad Canterlot, que nos entrega un conteo de 5.

```
    0.  Salir del programa
    1.  Agregar pony
    2.  Contar pegasos, ponis terrestres y unicornios en ciudad
    3.  Agregar anexo a ponies
    4.  Camino más corto entre 2 ponies
    5.  Encontrar amigos de amigos de un pony
    6.  Ponies con magia
    7.  Ponies con amigos unidireccionales
    8.  Contar bebidas por tipo de pony
    9.  Ponies con enemigos > colaboraciones
    10. Ponies que toman Coca Cola con amigos que toman Sprite

Ingresa una opción: 2
Ingresa el nombre de la ciudad: Canterlot
5
```

Esta fue la query utilizada.

```
query = """
MATCH (ciudad:Ciudad {nombre: $ciudad})←[r:VIVE_EN]-(p:Pony)
WHERE p.tipo IN ["Pony terrestre", "Pegaso", "Unicornio"]
RETURN count(*) AS total
"""
```

3 Campo anexo

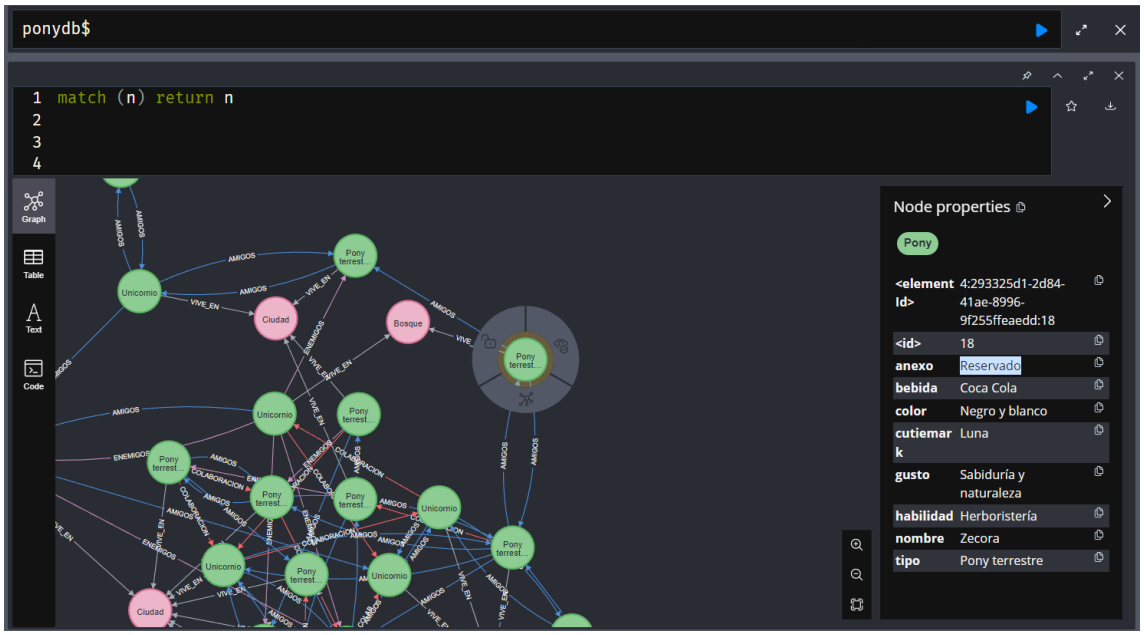
Ingresamos la opción para agregar el anexo a los ponies.

```
Ingresa una opción: 3
    0.  Salir del programa
    1.  Agregar pony
    2.  Contar pegasos, ponis terrestres y unicornios en ciudad
    3.  Agregar anexo a ponies
    4.  Camino más corto entre 2 ponies
    5.  Encontrar amigos de amigos de un pony
    6.  Ponies con magia
    7.  Ponies con amigos unidireccionales
    8.  Contar bebidas por tipo de pony
    9.  Ponies con enemigos > colaboraciones
    10. Ponies que toman Coca Cola con amigos que toman Sprite
```

Esta fue la query utilizada.

```
query = """
MATCH (p:Pony)-[r:AMIGOS]→(a:Pony)
WITH count(r) AS num_amigos, p
SET p.anexo = CASE
    WHEN p.tipo = "Unicornio" AND num_amigos ⩾ 3 THEN "Sociable"
    WHEN p.tipo = "Alicornio" THEN "Realeza"
    WHEN p.tipo = "Unicornio" AND num_amigos = 2 THEN "Reservado"
    WHEN p.tipo = "Unicornio" AND num_amigos = 1 THEN "Solitario"
    WHEN p.tipo = "Pony terrestre" AND num_amigos ⩾ 4 THEN "Hipersociable"
    WHEN p.tipo = "Pony terrestre" AND num_amigos ⩽ 2 THEN "Reservado"
    ELSE "Por completar"
END
"""
```

Aquí podemos ver un pony con su campo anexo ya agregado.



4 Camino más corto

Ingresamos la opción para conocer el camino más corto entre 2 ponies junto a sus nombres.

```
    0.  Salir del programa
    1.  Agregar pony
    2.  Contar pegasos, ponis terrestres y unicornios en ciudad
    3.  Agregar anexo a ponies
    4.  Camino más corto entre 2 ponies
    5.  Encontrar amigos de amigos de un pony
    6.  Ponies con magia
    7.  Ponies con amigos unidireccionales
    8.  Contar bebidas por tipo de pony
    9.  Ponies con enemigos > colaboraciones
    10. Ponies que toman Coca Cola con amigos que toman Sprite

Ingresa una opción: 4
Ingresa el nombre del primer pony: Tempest Shadow
Ingresa el nombre del segundo pony: Octavia Melody
Tempest Shadow -> Trixie Lulamoon -> Starlight Glimmer -> Sunset Shimmer -> Applejack -> Rarity -> Sour Sweet -> Zecora -> Octavia Melody
```

Comparamos el caso de cuando no hay un camino entre ellos.

```
    0.  Salir del programa
    1.  Agregar pony
    2.  Contar pegasos, ponis terrestres y unicornios en ciudad
    3.  Agregar anexo a ponies
    4.  Camino más corto entre 2 ponies
    5.  Encontrar amigos de amigos de un pony
    6.  Ponies con magia
    7.  Ponies con amigos unidireccionales
    8.  Contar bebidas por tipo de pony
    9.  Ponies con enemigos > colaboraciones
    10. Ponies que toman Coca Cola con amigos que toman Sprite

Ingresa una opción: 4
Ingresa el nombre del primer pony: Silver Spoon
Ingresa el nombre del segundo pony: Ponyta
No existe relación
```

Esta fue la query utilizada.

```
query = """
MATCH p = shortestPath((a:Pony {nombre: $name1})-[:AMIGOS*0..]→(b:Pony {nombre: $name2}))
RETURN p
"""
```

5 Amigos de amigos

Ingresamos la opción para ver los amigos de amigos de un pony junto al nombre del pony.

```
    0.  Salir del programa
    1.  Agregar pony
    2.  Contar pegasos, ponis terrestres y unicornios en ciudad
    3.  Agregar anexo a ponies
    4.  Camino más corto entre 2 ponies
    5.  Encontrar amigos de amigos de un pony
    6.  Ponies con magia
    7.  Ponies con amigos unidireccionales
    8.  Contar bebidas por tipo de pony
    9.  Ponies con enemigos > colaboraciones
    10. Ponies que toman Coca Cola con amigos que toman Sprite

Ingresa una opción: 5
Ingresa el nombre del pony: Moondancer
Applejack
Starlight Glimmer
```

Esta fue la query utilizada.

```
query = """
MATCH (p:Pony {nombre: $name}) -[r:AMIGOS*2]→(a:Pony)
WHERE NOT (p)-[:AMIGOS]→(a) AND p <> a
RETURN DISTINCT a.nombre
"""
```

6 Habilidad mágica

Ingresamos la opción para obtener los ponies con habilidades mágicas.

```
    0.  Salir del programa
    1.  Agregar pony
    2.  Contar pegasos, ponis terrestres y unicornios en ciudad
    3.  Agregar anexo a ponies
    4.  Camino más corto entre 2 ponies
    5.  Encontrar amigos de amigos de un pony
    6.  Ponies con magia
    7.  Ponies con amigos unidireccionales
    8.  Contar bebidas por tipo de pony
    9.  Ponies con enemigos > colaboraciones
    10. Ponies que toman Coca Cola con amigos que toman Sprite

Ingresa una opción: 6
Shining Armor
Starlight Glimmer
Tempest Shadow
Sunset Shimmer
Moondancer
Twilight Sparkle
```

Esta fue la query utilizada.

```
query = """
MATCH (p:Pony)
WHERE toLower(p.habilidad) =~ ".*magia.*"
RETURN p.nombre
"""
```

7 Amistad unidireccional

Ingresamos la opción que nos muestra los nombres de los ponies con amistades unidereccionales y con quién.

```
    0.  Salir del programa
    1.  Agregar pony
    2.  Contar pegasos, ponis terrestres y unicornios en ciudad
    3.  Agregar anexo a ponies
    4.  Camino más corto entre 2 ponies
    5.  Encontrar amigos de amigos de un pony
    6.  Ponies con magia
    7.  Ponies con amigos unidireccionales
    8.  Contar bebidas por tipo de pony
    9.  Ponies con enemigos > colaboraciones
    10. Ponies que toman Coca Cola con amigos que toman Sprite

Ingresa una opción: 7
Sunset Shimmer -> Applejack
Twirly Treats -> Applejack
Sweetie Belle -> Fluttershy
Silver Spoon -> Princess Celestia
Princess Cadance -> Shining Armor
Tempest Shadow -> Trixie Lulamoon
Vinyl Scratch -> Trixie Lulamoon
Trixie Lulamoon -> Starlight Glimmer
Princess Cadance -> Sour Sweet
Apple Bloom -> Sweetie Belle
Diamond Tiara -> Silver Spoon
Zecora -> Octavia Melody
Sour Sweet -> Twilight Sparkle
```

Esta fue la query utilizada.

```
query = """
    MATCH (a:Pony)-[c:AMIGOS]→(b:Pony)
    WHERE NOT (b)-[:AMIGOS]→(a)
    RETURN a.nombre,b.nombre
    """
```

8 Coca-Cola o Sprite

Ingresamos la opción para contar la cantidad de ponies que prefieren Coca-Cola o Sprite y el tipo de pony.

```
    0.  Salir del programa
    1.  Agregar pony
    2.  Contar pegasos, ponis terrestres y unicornios en ciudad
    3.  Agregar anexo a ponies
    4.  Camino más corto entre 2 ponies
    5.  Encontrar amigos de amigos de un pony
    6.  Ponies con magia
    7.  Ponies con amigos unidireccionales
    8.  Contar bebidas por tipo de pony
    9.  Ponies con enemigos > colaboraciones
    10. Ponies que toman Coca Cola con amigos que toman Sprite

Ingresa una opción: 8
Ingresa el tipo de pony: Alicornio
Coca Cola: 2, Sprite: 2
```

Comparamos ahora ingresando el tipo del pony agregado en el punto 1.

```
    0.  Salir del programa
    1.  Agregar pony
    2.  Contar pegasos, ponis terrestres y unicornios en ciudad
    3.  Agregar anexo a ponies
    4.  Camino más corto entre 2 ponies
    5.  Encontrar amigos de amigos de un pony
    6.  Ponies con magia
    7.  Ponies con amigos unidireccionales
    8.  Contar bebidas por tipo de pony
    9.  Ponies con enemigos > colaboraciones
    10. Ponies que toman Coca Cola con amigos que toman Sprite

Ingresa una opción: 8
Ingresa el tipo de pony: Pony de fuego
Coca Cola: 0, Sprite: 1
```

Esta fue la query utilizada.

```
query = """
MATCH (p:Pony {tipo: $tipo})
WITH
    SUM(CASE WHEN p.bebida = "Coca Cola" THEN 1 ELSE 0 END) AS cocacola,
    SUM(CASE WHEN p.bebida = "Sprite" THEN 1 ELSE 0 END) AS sprite
RETURN cocacola, sprite
"""
```

9 Más enemigos

Ingresamos la opción que muestra los ponies con más enemigos que colaboraciones con otros ponies.

```
    0.  Salir del programa
    1.  Agregar pony
    2.  Contar pegasos, ponis terrestres y unicornios en ciudad
    3.  Agregar anexo a ponies
    4.  Camino más corto entre 2 ponies
    5.  Encontrar amigos de amigos de un pony
    6.  Ponies con magia
    7.  Ponies con amigos unidireccionales
    8.  Contar bebidas por tipo de pony
    9.  Ponies con enemigos > colaboraciones
    10. Ponies que toman Coca Cola con amigos que toman Sprite

Ingresa una opción: 9
Trixie Lulamoon
Tempest Shadow
Diamond Tiara
```

Esta fue la query utilizada.

```
query = """
MATCH (b:Pony)←[e:ENEMIGOS]-(p:Pony)
WITH p, COUNT(e) AS enemigos
OPTIONAL MATCH (p)-[c:COLABORACION]→(a:Pony)
WITH p, enemigos, SUM(
    CASE
        WHEN c IS NULL THEN 0
        ELSE 1
    END) AS colaboraciones
WHERE enemigos > colaboraciones
RETURN p.nombre
"""
```

10 Coca-Cola y Sprite

Ingresamos la opción que nos muestra los ponies que gustan de la Coca-Cola y tienen algún amigo pony terrestre que guste de la Sprite.

```
    0.  Salir del programa
    1.  Agregar pony
    2.  Contar pegasos, ponis terrestres y unicornios en ciudad
    3.  Agregar anexo a ponies
    4.  Camino más corto entre 2 ponies
    5.  Encontrar amigos de amigos de un pony
    6.  Ponies con magia
    7.  Ponies con amigos unidireccionales
    8.  Contar bebidas por tipo de pony
    9.  Ponies con enemigos > colaboraciones
    10. Ponies que toman Coca Cola con amigos que toman Sprite

Ingresa una opción: 10
Rainbow Dash
Princess Cadance
Scootaloo
Zecora
Twilight Sparkle
```

Esta fue la query utilizada.

```
query = """
MATCH (p:Pony {bebida: "Coca Cola"}) -[r:AMIGOS]→(a:Pony {tipo:"Pony terrestre", bebida: "Sprite"})
RETURN DISTINCT p.nombre
"""
```

11 Trade-offs

El uso de índices en bases de datos mejora la velocidad de las consultas, pero mantenerlos tiene un costo. Cada vez que se añade o modifica un dato los índices deben actualizarse, lo que consume recursos y tiempo. Por eso, no siempre es conveniente crear índices para todo.

El equilibrio se encuentra evaluando qué atributos se consultan con más frecuencia y asegurándonos de que los índices solo se apliquen a esos casos. Si el sistema se utiliza más para lecturas que para escrituras, se pueden usar más índices para optimizar las consultas. Sin embargo, si hay muchas actualizaciones y escrituras, es mejor ser más selectivo y evitar agregar índices innecesarios.

La clave estaría en monitorear el desempeño y ajustar los índices según las necesidades del momento, enfocándose en aquellos que realmente impactan en el rendimiento. Debemos priorizar aquellos índices que aporten más valor a las consultas críticas, manteniendo un número mínimo de índices necesarios para evitar una sobrecarga innecesaria.

12 EXPLAIN o PROFILE

No siempre es adecuado usar EXPLAIN en lugar de PROFILE en Neo4j, ya que ambos comandos tienen propósitos distintos. EXPLAIN permite ver cómo se planearía ejecutar una consulta, mostrando su estructura sin ejecutarla realmente. Esto es útil para verificar el plan sin alterar la base de datos.

Sin embargo, PROFILE se utiliza para ver el plan de ejecución real y analizar en detalle el consumo de recursos. Es ideal cuando queremos entender el rendimiento real de una consulta, ya que muestra métricas como el tiempo y la cantidad de registros procesados. Por ejemplo, si una consulta parece ser lenta, PROFILE nos permite detectar cuellos de botella y entender mejor qué partes del plan de ejecución son las más costosas.