

INF-325 Bases de Datos Avanzadas: Laboratorio Neo4j



Grupo 4:

- ☐ Ernesto Barria Andrade – ernesto.barria@usm.cl
- ☐ Camilo Díaz Galaz – camilo.diazg@usm.cl
- ☐ Sebastián Gutiérrez Milla –
sebastian.gutierrezmi@usm.cl
- ☐ Carlos Lagos Cortes – carlos.lagosc@usm.cl
- ☐ Luis Zegarra Stuardo – Luis.zegarra@usm.cl

Repositorio:

- ☐ GitHub: <https://github.com/luphin/tareaNeo4j>

Resumen

Este documento describe la implementación de una base de datos basada en grafos utilizando Neo4J, desplegada en un entorno Docker para garantizar portabilidad y facilidad de configuración. La solución incluye una instancia de Neo4J configurada para modelar datos complejos relacionados con árboles, sectores, encargados y reportes, maximizando la eficiencia de las consultas y el análisis de relaciones entre entidades.

Además, se detalla el proceso de diseño e implementación de la base de datos, junto con la creación de las consultas necesarias para satisfacer los requisitos definidos en el problema. Se incluye una descripción de cómo esta estructura permite modelar entidades clave como árboles frutales, plagas, enfermedades y aspersores, asegurando un sistema robusto y fácil de operar.

1 Introducción

Con el paso del tiempo, el almacenamiento de estructuras de datos ha experimentado una evolución significativa, principalmente debido a la creciente cantidad de datos que es necesario procesar y gestionar en las bases de datos. Existen tres tipos principales de bases de datos:

1. SQL: Son bases de datos relacionales que permiten la gestión de datos mediante relaciones entre ellos, facilitando así las consultas. Este tipo de bases de datos tiene un escalado vertical, lo que significa que su principal limitación está en la capacidad de mejorar la máquina en la que se ejecutan.

2. NoSQL: Las bases de datos no relacionales, conocidas como NoSQL, se comenzaron a utilizar por primera vez en 1998 y fueron formalizadas en el año 2000. Estas bases de datos son más eficientes en el manejo de grandes volúmenes de datos debido a su escalado horizontal, lo que permite almacenar la información en particiones o réplicas distribuidas en diferentes nodos. Actualmente, son más utilizadas y preferidas que las bases de datos SQL, ya que, además de mejorar en aspectos como la frecuencia y latencia de las consultas, ofrecen diversas formas de almacenar la información, como pares clave-valor, bases orientadas a documentos, grafos y familias de columnas. [1,2,3,4]

3. NewSQL: Este es un nuevo sistema de gestión de bases de datos (DBMS), introducido por primera vez en 2011. NewSQL combina el enfoque relacional de SQL con la escalabilidad horizontal de NoSQL, al tiempo que mejora la consistencia, un aspecto en el que las bases de datos NoSQL suelen presentar limitaciones (según el teorema CAP). Esto se logra mediante la implementación de los enfoques BASE y ACID. [4,5]

En este informe se describe el diseño, implementación y consulta de una base de datos basada en grafos utilizando **Neo4J**. Este enfoque se seleccionó debido a las características del problema, donde se manejan relaciones complejas entre entidades como árboles, sectores, encargados y reportes. El uso de grafos permite modelar estos datos de manera natural y eficiente, aprovechando las capacidades de consulta optimizada que ofrece Neo4J.

La base de datos se ejecuta sobre una instancia de **Neo4J Docker**, proporcionando una configuración portátil y de fácil despliegue. Este entorno también asegura que las operaciones sean consistentes y repetibles, facilitando el desarrollo y la verificación de las consultas realizadas.

El problema abordado consiste en gestionar los datos de AgroUSM, una empresa agrícola que monitorea su producción y estado de salud de los árboles frutales. La base de datos debe ser capaz de almacenar y consultar información sobre:

- Árboles: Identificados por un código único, tipo, nombre científico y ubicación geográfica.
- Sectores: Áreas delimitadas donde se ubican los árboles, cada una asignada a un encargado.
- Encargados: Personas responsables de visitar y registrar el estado de los árboles semanalmente.
- Reportes: Documentos generados por los encargados que contienen datos sobre conteo de frutas, plagas, enfermedades y estado de los aspersores.
- Plagas y enfermedades: Factores que afectan la producción y requieren ser monitoreados.
- Aspersores: Equipos asociados a los árboles para el riego, cuyo estado debe mantenerse operativo.

El uso de una base de datos basada en grafos permite modelar estas entidades y sus relaciones de manera directa, mejorando la capacidad de realizar análisis complejos sobre los datos.

2. Desarrollo

2.1 Desarrollo Requisito 1

Nos reunimos todos los integrantes mediante Discord y uno compartía pantalla con la herramienta [Excalidraw](#), logramos formar el esquema en la figura 1 al aplicar whiteboard-friendly, luego lo trasparamos a Latex y definimos los tributos para cada nodo representado en la figura 2.

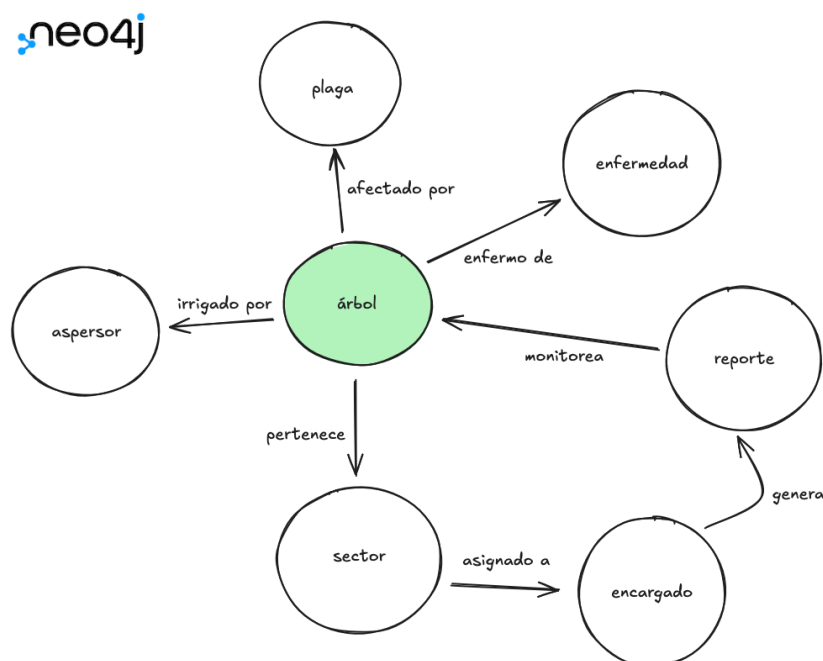


figura 1. Esquema inicial en Excalidraw



figura 2. Esquema con atributos en Latex

2.2 Desarrollo Requisito 2

Se utilizó un archivo docker-compose.yml para desplegar una instancia de Neo4J localmente. La configuración incluye mapeos de puertos y volúmenes para garantizar persistencia de datos y accesibilidad mediante la interfaz web de Neo4J. La guía para poder hacer este compose, fue sacada de la [documentación de Neo4](#) y la [imagen oficial en Docker](#).

```
Code Blame 21 lines (20 loc) · 561 Bytes Code 55% faster with GitHub Copilot
1  services:
2    neo4j:
3      image: neo4j:5.8 # Versión de Neo4J
4      container_name: neo4j-container
5      ports:
6        - "7474:7474" # Puerto para la interfaz web
7        - "7687:7687" # Puerto para el protocolo Bolt
8      volumes:
9        - neo4j_data:/data
10       - neo4j_logs:/logs
11       - neo4j_import:/import
12       - neo4j_plugins:/plugins
13     environment:
14       - NEO4J_AUTH=neo4j/test123456789 # Usuario y contraseña
15       - NEO4J_ACCEPT_LICENSE_AGREEMENT=yes # Aceptar el acuerdo de licencia
16   volumes:
17     neo4j_data:
18     neo4j_logs:
19     neo4j_import:
20     neo4j_plugins:
21
```

Figura 3. `docker-compose.yml`

Se puede descargar del repositorio de GitHub, [link](#). Al descargarlo se debe abrir Docker y ejecutar los siguientes comandos para levantar la instancia.

1. Comando para ejecutar docker compose
``` docker-compose up -d ```
2. Luego abrir en el buscador  
``` Localhost:7474 ```
3. Para detener la instancia, colocar.
``` docker-compose down ```

## 2.3 Desarrollo Requisito 3

### 1. Crear Nodos y relaciones

Los datos para el dataset fueron creados con la herramienta ChatGPT que es desarrollada por la empresa [OpenAI](#), nosotros le dijimos como era la estructura, la cantidad de nodos que queríamos y como se debían relacionar. El dataset que retorno se puede encontrar en el repositorio del grupo para la tarea ([link directo al archivo](#)).

### 2. Visualizar los datos

Comandos para visualizar los nodos y relaciones como se ve en la figura 4.

```
``` MATCH (p) RETURN p ```
```

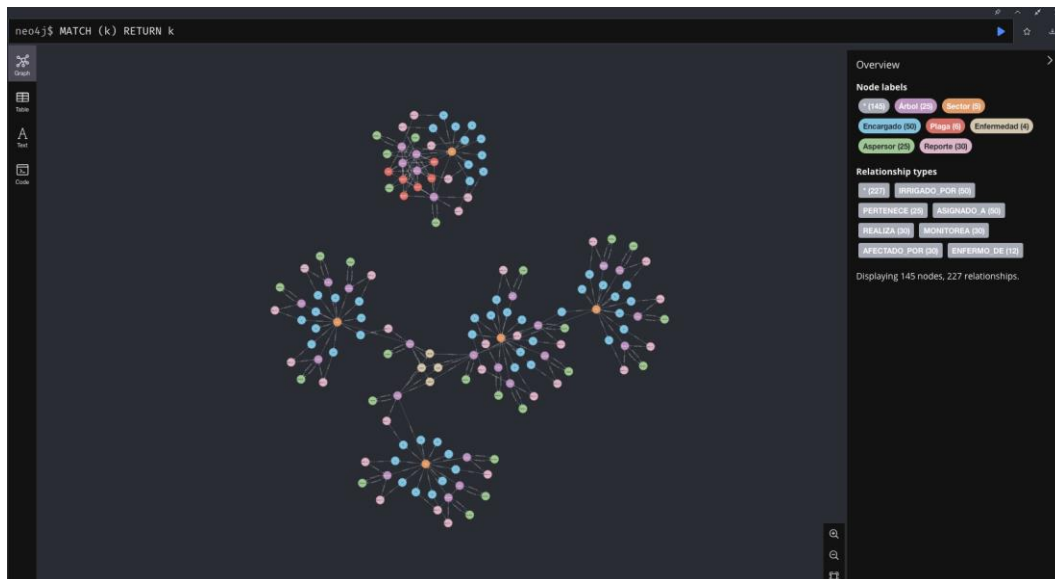


Figura 4. Ocurrencias cargadas

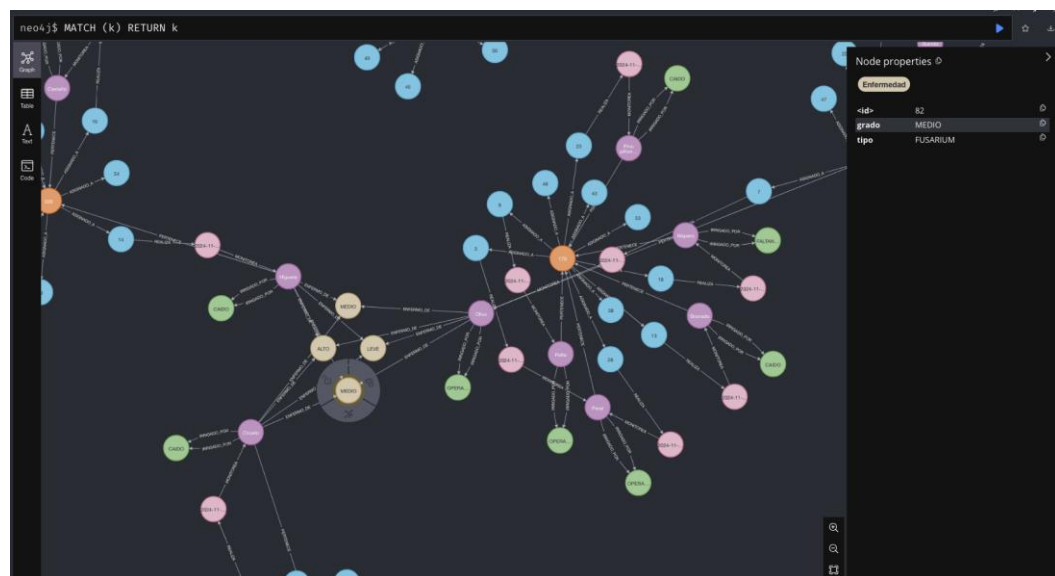


Figura 5. Sector y relaciones

2.4 Desarrollo Requisito 3

1. Consulta: Determinar la producción total de frutas y kilos cosechados por sector

Esta consulta calcula el total de frutas y kilos cosechados por cada sector según los reportes realizados.

```

```
MATCH (s:Sector)-[:PERTENECE]-(a:Árbol)-[:MONITOREA]-(r:Reporte) RETURN s.nombre
AS Sector, SUM(r.frutas_cosechadas) AS TotalFrutas, SUM(r.kilos_cosechados) AS
TotalKilos ORDER BY TotalKilos DESC;
```

```



```
neo4j$ MATCH (s:Sector)←[:PERTENECE]-(a:Árbol)←[:MONITOREA]-(r:Reporte) RETURN s.nombre AS Sector, SUM(r.frutas...
```

	Sector	TotalFrutas	TotalKilos
1	"Sector Este"	1280	1040
2	"Sector Sur"	1240	1010
3	"Sector Oeste"	1230	970
4	"Sector Norte"	1240	960
5	"Sector Central"	1050	800

Started streaming 5 records after 16 ms and completed after 31 ms.

2. Consulta: Identificar los árboles con problemas de salud (plagas o enfermedades)

Esta consulta lista los árboles que están afectados por plagas o enfermedades, junto con los detalles de estas afectaciones.

```
...  
MATCH (a:Árbol)  
OPTIONAL MATCH (a)-[:AFECTADO_POR]->(p:Plaga)  
OPTIONAL MATCH (a)-[:ENFERMO_DE]->(e:Enfermedad)  
RETURN  
    a.id AS ArbolID,  
    a.tipo AS Tipo,  
    COLLECT(DISTINCT p.tipo) AS Plagas,  
    COLLECT(DISTINCT e.tipo) AS Enfermedades  
ORDER BY size(Plagas) + size(Enfermedades) DESC;  
...
```

```
neo4j$ MATCH (a:Árbol) OPTIONAL MATCH (a)-[:AFECTADO_POR]->(p:Plaga) OPTIONAL MATCH (a)-[:ENFERMO_DE]->(e:Enfermedad) RETURN a.id AS ArbolID, a.tipo AS Tipo, COLLECT(DISTI...
```

	ArbolID	Tipo	Plagas	Enfermedades
1	10	"Mandarino"	["BURRITO DE LA VID", "CONCHUELA NEGRA DEL OLIVO", "ARAÑITA ROJA", "TRIPS", "ESCAMA BLANCA", "CHANCHITO BLANCO"]	[]
2	15	"Papayo"	["CHANCHITO BLANCO", "BURRITO DE LA VID", "TRIPS", "ARAÑITA ROJA", "ESCAMA BLANCA", "CONCHUELA NEGRA DEL OLIVO"]	[]
3	20	"Anacardo"	["CONCHUELA NEGRA DEL OLIVO", "ESCAMA BLANCA", "BURRITO DE LA VID", "CHANCHITO BLANCO", "TRIPS", "ARAÑITA ROJA"]	[]
4	25	"Arándano"	["ARAÑITA ROJA", "TRIPS", "ESCAMA BLANCA", "CONCHUELA NEGRA DEL OLIVO", "CHANCHITO BLANCO", "BURRITO DE LA VID"]	[]
5	5	"Duraznero"	["ESCAMA BLANCA", "BURRITO DE LA VID", "TRIPS", "CONCHUELA NEGRA DEL OLIVO", "ARAÑITA ROJA", "CHANCHITO BLANCO"]	[]
6	7	"Olivo"	[]	["VERTICILLOSIS", "HONGO DE LA MADERA", "FUSARIUM", "PHYTOPHTHORA"]
7	14	"Higuera"	[]	["HONGO DE LA MADERA", "FUSARIUM", "VERTICILLOSIS", "PHYTOPHTHORA"]
8	21	"Ciruelo"	[]	["VERTICILLOSIS", "PHYTOPHTHORA", "FUSARIUM", "HONGO DE LA MADERA"]
9	6	"Almendro"	[]	[]
10	8	"Palto"	[]	[]
11	9	"Limón"	[]	[]
12	11	"Nogal"	[]	[]
13	12	"Mango"	[]	[]

Started streaming 25 records after 20 ms and completed after 23 ms.

3. Consulta: Evaluar el estado de los aspersores en cada sector

Esta consulta identifica el número total de aspersores, su estado operativo y los

defectuosos en cada sector para priorizar intervenciones técnicas.

```

```
MATCH (s:Sector) <-[:PERTENECE]-(a:Árbol)-[:IRRIGADO_POR]->(asp:Aspersor)
RETURN
 s.nombre AS Sector,
 COUNT(asp) AS TotalAspersores,
 COUNT(CASE WHEN asp.estado = "OPERATIVO" THEN 1 END) AS AspersoresOperativos,
 COUNT(CASE WHEN asp.estado <> "OPERATIVO" THEN 1 END) AS AspersoresDefectuosos
ORDER BY AspersoresDefectuosos DESC;
```
```

neo4j\$ MATCH (s:Sector) <-[:PERTENECE]-(a:Árbol)-[:IRRIGADO_POR]->(asp:Aspersor) RETURN s.nombre AS Sector, COUNT(asp) AS TotalAspersores, COUNT(CASE WHEN asp.estado = "OPERATIVO" THEN 1 END) AS AspersoresOperativos, COUNT(CASE WHEN asp.estado <> "OPERATIVO" THEN 1 END) AS AspersoresDefectuosos ORDER BY AspersoresDefectuosos DESC;

	Sector	TotalAspersores	AspersoresOperativos	AspersoresDefectuosos
1	"Sector Sur"	10	0	10
2	"Sector Este"	10	2	8
3	"Sector Oeste"	10	4	6
4	"Sector Central"	10	4	6
5	"Sector Norte"	10	10	0

Started streaming 5 records after 16 ms and completed after 16 ms.

4. Consulta: Identificar el sector con mayor afectación por plagas

Esta consulta busca el sector con la mayor cantidad de árboles afectados por plagas y el tipo de plagas que predominan.

```

```
MATCH (s:Sector) <-[:PERTENECE]-(a:Árbol)-[:AFECTADO_POR]->(p:Plaga) RETURN
 s.nombre AS Sector, COUNT(DISTINCT a) AS ArbolesAfectados, COLLECT(DISTINCT
 p.tipo) AS TiposDePlagas ORDER BY ArbolesAfectados DESC LIMIT 1;
```
```

neo4j\$ MATCH (s:Sector) <-[:PERTENECE]-(a:Árbol)-[:AFECTADO_POR]->(p:Plaga) RETURN s.nombre AS Sector, COUNT(DISTINCT a) AS ArbolesAfectados, COLLECT(DISTINCT p.tipo) AS TiposDePlagas ORDER BY ArbolesAfectados DESC LIMIT 1;

	Sector	ArbolesAfectados	TiposDePlagas
1	"Sector Norte"	5	["CONCHUELA NEGRA DEL OLIVO", "ESCAMA BLANCA", "BURRITO DE LA VID", "CHANCHITO BLANCO", "TRIPS", "ARAÑITA ROJA"]

Started streaming 1 records after 14 ms and completed after 14 ms.

5. Consulta Calcular el porcentaje de árboles revisados por encargado y los árboles faltantes en un sector

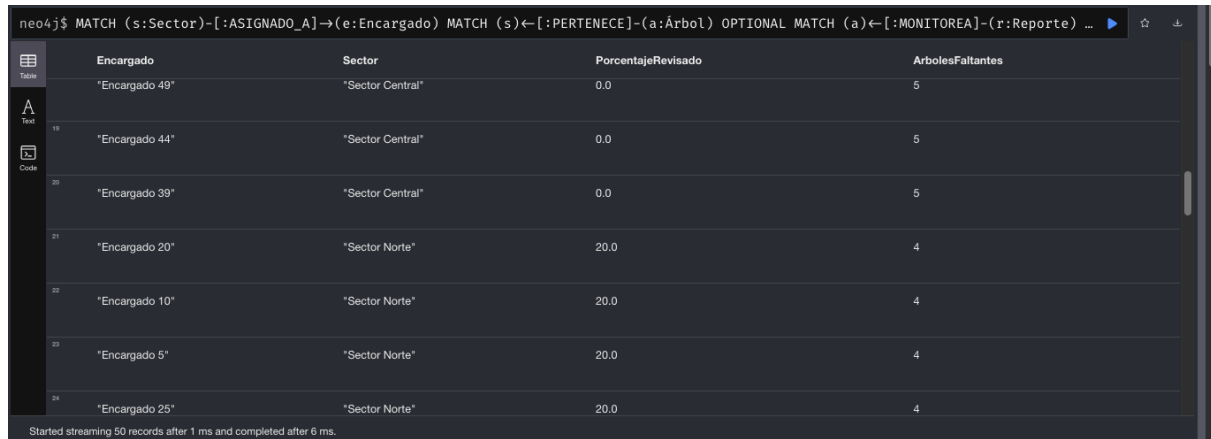
Esta consulta determina el porcentaje de árboles revisados por cada encargado en su sector asignado, identifica cuántos árboles faltan por revisar en el sector y ordena los resultados por el menor porcentaje de revisión.

```

```
MATCH (s:Sector)-[:ASIGNADO_A]->(e:Encargado) MATCH (s) <-[:PERTENECE]-(a:Árbol)
OPTIONAL MATCH (a) <-[:MONITOREA]-(r:Reporte) <-[:REALIZA]-(e) WITH e, s,
COUNT(DISTINCT a) AS TotalArbolesSector, COUNT(DISTINCT r) AS ArbolesRevisados
```



```
RETURN e.nombre AS Encargado, s.nombre AS Sector, (ArbolesRevisados * 1.0 /
TotalArbolesSector) * 100 AS PorcentajeRevisado, TotalArbolesSector -
ArbolesRevisados AS ArbolesFaltantes ORDER BY PorcentajeRevisado ASC;
````
```



| Encargado | Sector | PorcentajeRevisado | ArbolesFaltantes |
|----------------|------------------|--------------------|------------------|
| "Encargado 49" | "Sector Central" | 0.0 | 5 |
| "Encargado 44" | "Sector Central" | 0.0 | 5 |
| "Encargado 39" | "Sector Central" | 0.0 | 5 |
| "Encargado 20" | "Sector Norte" | 20.0 | 4 |
| "Encargado 10" | "Sector Norte" | 20.0 | 4 |
| "Encargado 5" | "Sector Norte" | 20.0 | 4 |
| "Encargado 25" | "Sector Norte" | 20.0 | 4 |

Started streaming 50 records after 1 ms and completed after 6 ms.

Esta consulta tiene un **defecto**, ya que, no logramos hacer que se registrara en la tabla que los árboles faltantes contaran todos los faltantes en el sector, lo que está haciendo ahora es solo contar los árboles faltantes para cada encargado.

3 Conclusiones

Este informe presenta el diseño, implementación y consulta de una base de datos en Neo4J para AgroUSM, una empresa agrícola que se encarga de gestionar y monitorear árboles frutales. Utilizando una base de datos basada en grafos, se logró modelar de manera eficiente y clara las relaciones entre árboles, sectores, encargados, plagas, enfermedades y otros elementos clave en el proceso de monitoreo agrícola.

La combinación de Neo4J y Docker permitió crear un sistema robusto, portátil y escalable, facilitando tanto el desarrollo como el despliegue de la solución. La creación del esquema nos dio un mejor entendimiento de cómo se tenían que hacer las relaciones entre los objetos, y este es más intuitivo por la forma en la que se organizan los datos. Tuvimos problemas con hacer las consultas, debido al bajo dominio que poseíamos dentro de la herramienta, lo que desembocó en que la última consulta no pudiéramos desarrollarla como nosotros queríamos y dejando la con datos que, si bien son de importancia, queremos comprimir para tener un mejor análisis de lo que se está midiendo.

Las consultas en Cypher demuestran ser una herramienta eficaz para extraer información valiosa que apoya la toma de decisiones y optimiza el monitoreo de la producción y el estado de los árboles, es un sistema en el que desarrollar las consultas es intuitivo debido al patrón de las consultas y como estas se deben ejecutar. La flexibilidad del sistema y su capacidad para adaptarse a futuros requerimientos garantizan que la solución sea adecuada para manejar grandes volúmenes de datos relacionados con la agricultura.

En resumen, este proyecto evidencia cómo las bases de datos basadas en grafos, como Neo4J, son altamente eficaces para gestionar relaciones complejas, facilitando un análisis detallado que mejora la eficiencia operativa y la toma de decisiones dentro de AgroUSM.

4 Referencias Bibliográficas

1. Figueroa Colarte, M. (n.d.). *Bases de datos Avanzadas*. INF-325 Bases de Datos



Avanzadas. Universidad Técnica Federico Santa María.

2. Meier, A., Kaufmann, M., Meier, A., & Kaufmann, M. (2019). Nosql databases. *SQL & NoSQL Databases: Models, Languages, Consistency Options and Architectures for Big Data Management*, 201-218.
3. Chen, J. K., & Lee, W. Z. (2018, December). A study of NoSQL Database for enterprises. In *2018 International Symposium on Computer, Consumer and Control (IS3C)* (pp. 436-440). IEEE.
4. Praschl, C., Pritz, S., Krauss, O., & Harrer, M. (2022, November). A comparison of relational, NoSQL and NewSQL database management systems for the persistence of time series data. In *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)* (pp. 1-6). IEEE.
5. Pavlo, A., & Aslett, M. (2016). *What's really new with NewSQL?*. *ACM Sigmod Record*, 45(2), 45-55.
6. Neo4j (2024). Docker. Neo4j. <https://neo4j.com/docs/operations-manual/current/docker/>