Modelando las interacciones de “Juego de Tronos” con Grafos: Un estudio comparativo

Gabriela Rodríguez
Facultad de Ingeniería,
Universidad de la República
Montevideo, Uruguay
gabriela.rodriguez@fing.edu.uy

Francisco Nicolás Noya
Facultad de Ingeniería,
Universidad de la República
Montevideo, Uruguay
francisco.nicolas.noya@fing.edu.uy

Resumen

Este documento busca explorar la creación de una base de datos de grafos con la cual modelar ciertas interacciones de la popular serie de HBO “Juego de Tronos”. El objetivo principal de esta investigación consiste en desarrollar tanto una base de datos de grafos como una base de datos relacional que representen de manera adecuada la estructura y relaciones de la serie. Posteriormente, se efectuarán consultas en ambas bases de datos y se realizará una comparación de un subconjunto de los resultados obtenidos, con el fin de determinar la pertinencia de utilizar una base de datos de grafos para el modelado de la realidad planteada en este contexto específico.

I. Introducción

En este informe se busca describir el Proyecto Final del curso de Bases de Datos no Relacionales del Instituto de Computación de la Facultad de Ingeniería, dictado de marzo a julio del año 2023. Este proyecto busca modelar, mediante una base de datos de grafos, interacciones entre entidades de la serie “Juego de Tronos” . Al ser esta una serie con una trama compleja y un extenso desarrollo de los personajes y las localidades, se consideró oportuno la utilización de una base de datos de esta categoría para representar su realidad. Además se realizarán consultas sobre dicha base con la finalidad de validar la calidad de los datos obtenidos y dar respuesta a interrogantes específicas sobre la serie. Finalmente, se creó una base de datos relacional donde se llevaron a cabo consultas conceptualmente similares a las realizadas en la base de datos de grafos. Luego, se compararon los tiempos de respuesta de algunas de las consultas sobre ambas bases.

Las herramientas utilizadas para la realización de este proyecto son las siguientes:

1. Neo4j Aura DB versión 5.9.0 [1]
2. Google Colab [2]
3. Python3 [3] con librería Pandas [4].

Este proyecto se fundamenta en la filosofía de trabajo “Coast-to-Coast” (definida en el curso), por lo que su desarrollo tendrá las siguientes etapas. Primero se obtienen diversos conjuntos de datos de diversas fuentes, una vez decidido cuales utilizar se modela la base de datos de grafos a construir, luego se transforman los datos y se crean los archivos necesarios para su carga, por último se realizan consultas sobre la base.

Este documento se dividirá en cinco secciones, en la primera sección se detallarán los objetivos concretos del proyecto, en la siguiente se busca dar reconocimiento a las personas y usuarios cuyos trabajos anteriores fueron base para la realización de este proyecto, también se analizaran los distintos conjuntos de datos seleccionados. Luego se procederá con la descripción detallada de la tarea realizada, para continuar con una sección de experimentación y por último una de conclusiones.

II. Objetivos

Este proyecto tiene dos objetivos principales, el primero es determinar si realmente existe una mejora en la eficiencia de una base de datos de grafos respecto a una base de datos relacional.
Es así que se define la siguiente hipótesis a demostrar:

El conjunto de datos elegido, por su complejidad y alto grado de interrelación, favorece a una base de datos de grafos por sobre una relacional.

El otro objetivo que busca alcanzar este proyecto es realizar consultas específicas para obtener resultados sobre los datos obtenidos.

III. Trabajos relacionados

En esta sección se describirán los trabajos que fueron esenciales para la realización de este proyecto, no solo porque fueron fuente de inspiración para el modelado de la base sino también porque de ellos fue que se extrajeron los datos. En la sección de Referencias se encuentran enlaces a las distintas fuentes utilizadas.

III-A. “An illustrated guide to all 6,887 deaths in Game of Thrones”

Este conjunto de datos [5] [6] creado por Shelly Tan el 21 de mayo de 2019 y publicado en el Washington Post, fue el más utilizado para la realización de este proyecto. Este relaciona a los personajes que fueron asesinados con sus asesinos, el lugar donde fueron asesinados, a quien o que le rendían pleitesía, el método con el que fueron asesinados y el motivo

de su asesinato. Sin embargo, los datos aquí encontrados se consideraron demasiado llanos como para explotar al máximo las capacidades de una base de datos de grafos, por lo que fue necesario extraer datos adicionales de otras fuentes.

III-B. "Network of thrones"

Este conjunto de datos [7] creado por Andrew Beveridge especifica las interacciones que los personajes tuvieron entre ellos a lo largo de cada temporada.

III-C. "neo4j-got"

Este conjunto de datos [8] creado por Mark Needham, es sumamente rico en datos y relaciones, sin embargo del mismo se extrajo únicamente los nombres de los episodios.

III-D. "GoTGraphGist"

Este conjunto de datos [9] creado por el usuario de GitHub "bladow", fue utilizada principalmente para extraer los honores y título nobiliarios que poseen los personajes.

III-E. "AnApiOfIceAndFire"

Este conjunto de datos [10] creado por el usuario de GitHub "joakimskoog", fue utilizada para extraer información acerca de las casas.

IV. Desarrollo del Proyecto

En esta sección se detallará cómo el proyecto fue desarrollado, para esto se procederá de la siguiente manera: Primero se hará una descripción de los problemas encontrados y cómo estos se abordaron, luego se ahondará en el modelado de la base de datos, justificando las decisiones tomadas. Por último se detallará la creación de una base de datos relacional para comparar la eficiencia de las consultas con la base de datos de grafos construida.

IV-A. Problema

Durante la etapa de investigación surgieron gran cantidad de fuentes de datos y sitios en los que era posible extraerlos, sin embargo luego de un análisis meticuloso de los mismos se observó que, en ciertos casos eran inconsistentes o expresaban distintos valores para los mismos atributos de los objeto de la realidad. Por lo que fue necesario solucionar estos dos problemas.

La inconsistencia de datos fue abordada mediante un proceso que constó de dos etapas, primero se verificaron las fuentes de los datos, en caso de que estos proviniesen de sitios poco confiables, eran descartados o examinados meticulosamente, de esta forma se evitó que los datos obtenidos representaran de una forma incorrecta la realidad de la serie. Sin embargo, incluso los conjuntos de datos cuyas fuentes fueron consideras confiables, tenían sus defectos, existían tuplas con todos los atributos iguales menos uno, este generalmente consistía de un valor numérico, o tuplas que se encontraban repetidas. Por fortuna estos pares de tuplas inconsistentes eran reducidos por lo que con una pequeña modificación manual estos problemas fueron solucionados.

Otro problema inherente a la utilización de varias fuentes para la creación de la base fue la diferencia en representaciones de los mismos atributos con distintos orígenes. Existía el caso en el que un mismo personaje en una fuente se identificaba con su nombre y apellido mientras que en otra poseía únicamente su nombre y un identificador. Para abordar esta problemática fue necesario realizar un programa en python que tomara los archivos en formato Comma-Separated Values (csv) obtenidos de las fuentes y normalizara los nombres, considerando la forma normal de un nombre como nombre y apellido. El código que permitió el preprocesamiento de los datos se encuentra en un notebook entregado en el repositorio GitHub asociado al proyecto [12].

Además, es relevante destacar la conexión entre las "alianzas" y los "lugares" mediante una arista. Para establecer esta conexión, se requirió combinar dos conjuntos de datos provenientes de distintos proveedores, donde los nombres de las alianzas presentaban diferencias. Como resultado, fue necesario realizar un proceso de normalización de los nombres para lograr la coherencia necesaria.

IV-B. Modelado de la Base

El modelado de la base consistió de varias etapas, primero se determinaron los aspectos de la realidad que se pretendían modelar con mayor énfasis, esto surgió naturalmente de los conjuntos de datos en donde se decidió modelar las muertes ocurridas en la serie como tema central de la base, en conjunto con información relevante de los personajes. Es así que se procedió a determinar cómo sería la representación de las muertes. Al comienzo se consideró modelar a los personajes como nodos y a las muertes como relaciones entre los personajes, sin embargo se hizo evidente instantáneamente que este modelo no iba a resultar debido a la gran cantidad de atributos que posee una muerte y a que estas son centrales en la base. Por lo que se decidió que tanto las muertes como los personajes fueran nodos independientes.

Otro punto en el que fue necesario un análisis más meticuloso fue en el caso de los episodios en los que habían ocurrido las muertes, era posible modelarlos como un simple atributo extra en el nodo de las muertes, sin embargo esto recargaría aún más al nodo e introduciría la posibilidad de que la base de datos tuviese errores de consistencia, en donde un episodio tuviese el mismo número en la misma temporada pero con nombres distintos. Por lo que por temas de memoria, consistencia y simplicidad, se decidió que los episodios fueran considerados como nodos independientes.

Por otro lado las temporadas consistían únicamente de un número, no poseen nombre ni información relevante, es por esto que se decidió modelarlas simplemente como un atributo numérico más dentro del episodio.

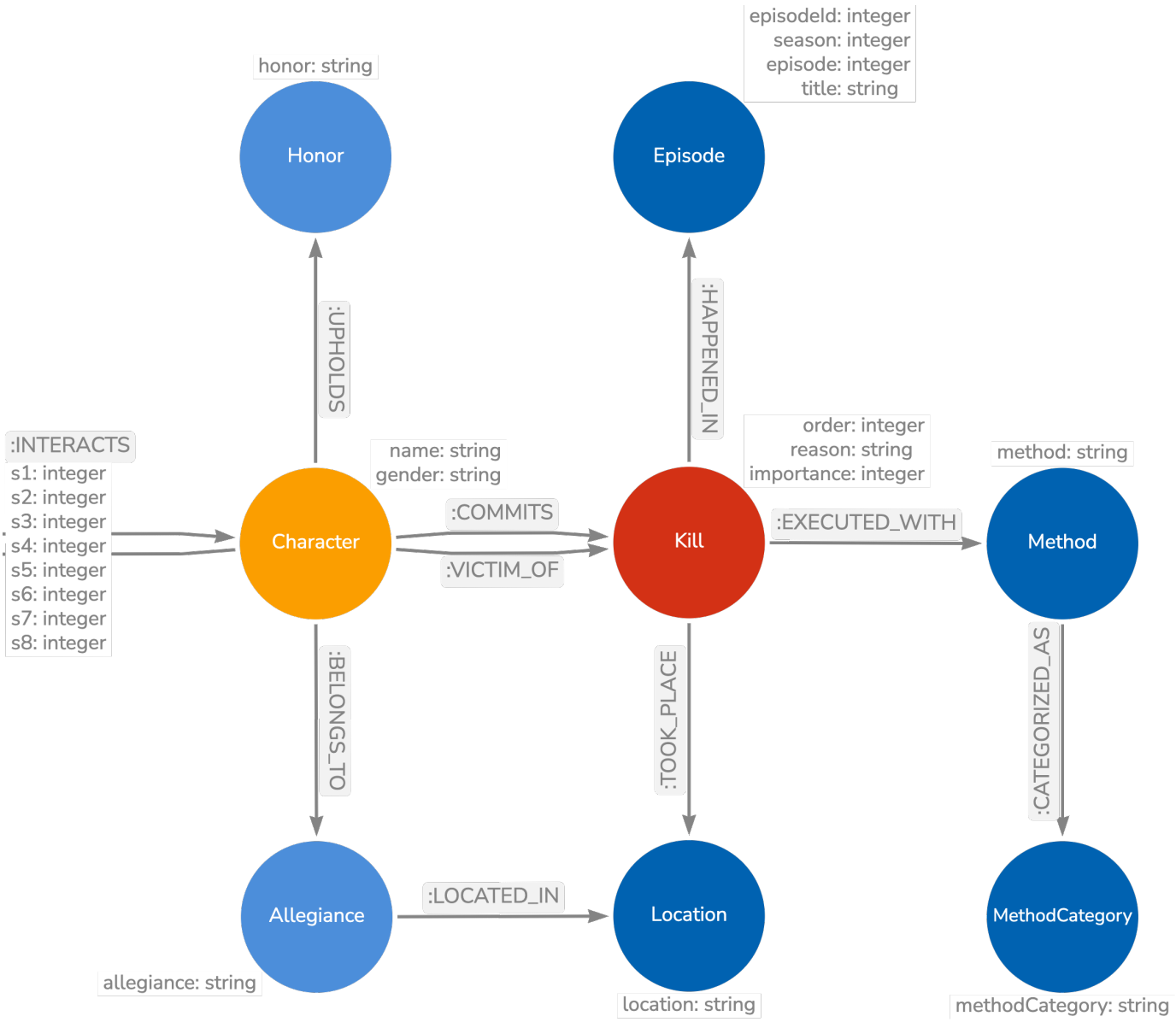

Figura 1: Modelo de la base de datos de grafos.

Para representar las interacciones entre los personajes, se optó por utilizar lazos, esto es debido a que las interacciones no agregan una cantidad de información importante que no pueda ser representada de esta forma, lo central en las interacciones entre los personajes son precisamente los personajes, luego la solución de representar las interacciones como aristas entre los mismos se dio muy natural.

La etapa de modelado fue la más creativa del proyecto, en ella fue necesario idear la base deseada utilizando los conjuntos de datos disponibles, un diagrama con el resultado de esta etapa se encuentra detallado en la Figura 1.

IV-C. Creación de la base de datos relacional

Para realizar la base de datos relacional se utilizó la herramienta SQLite [11] pues esta brinda un entorno de desarrollo sumamente sencillo de utilizar junto con Google Colab [2] y es lo suficientemente poderosa para modelar la base deseada y extraer la información requerida.

El modelo de la base de datos relacional fue derivado casi en su totalidad del modelo de la base de datos de grafos, este modelo respeta la forma normal 3NF. Esto con el objetivo de que las propiedades que cumpliese la base de datos de grafos, la cumpliese también la base de datos relacional. En la Figura 2, se presenta el modelo entidad-relación (MER) que ha sido utilizado para representar las entidades y relaciones en la base de datos.

El código de creación y carga de esta base fue entregado en un notebook junto con este informe y subido al GitHub correspondiente [12], la creación y carga de esta base se realizó procesando los mismos archivos .csv utilizados para cargar la base de datos de grafos.

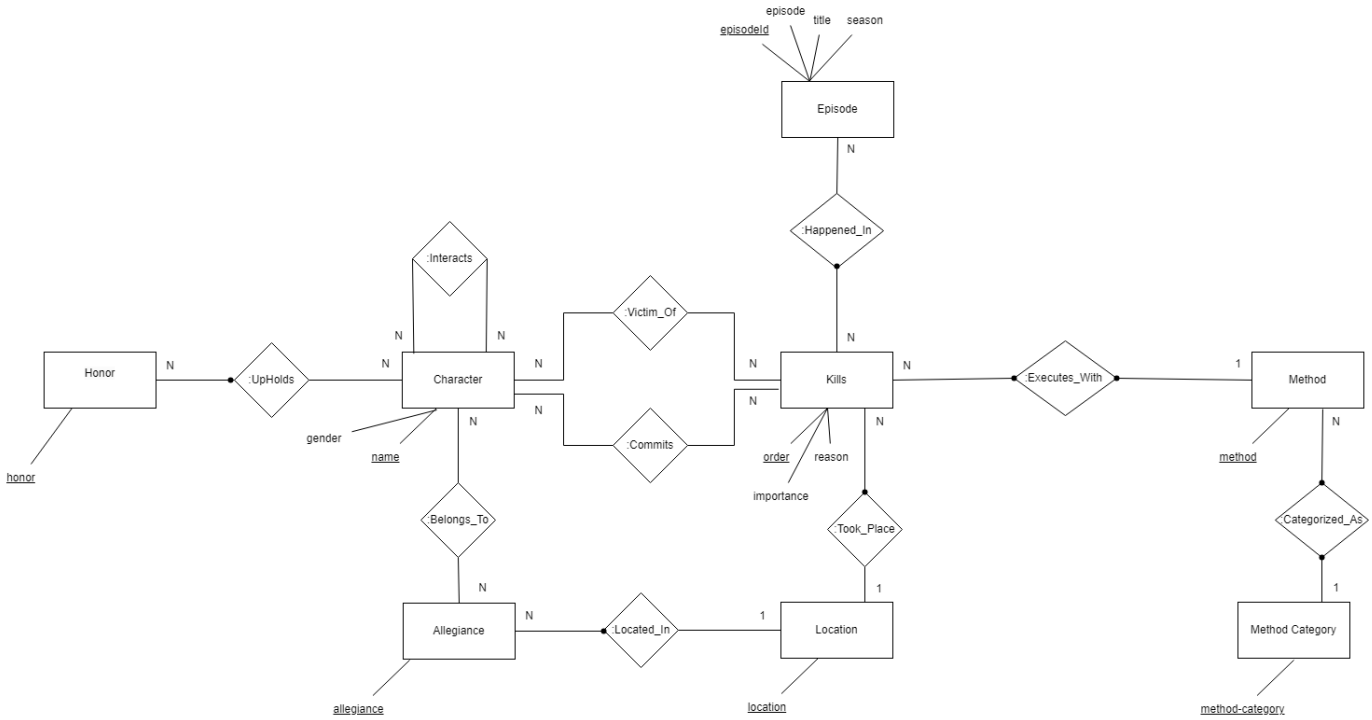

Figura 2: Modelo Entidad Relación base de datos relacional.

V. Experimentación

En esta sección se detallará el proceso de experimentación, se explicaran los objetivos, las hipótesis y cómo se llevó a cabo.

La tarea de experimentación se realizó apuntando a dos objetivos principales. Demostrar la hipótesis planteada en la sección de "objetivos" y obtener información relevante sobre los datos de la base. Para cumplirlos se efectuaron consultas sobre la misma. Estas consultas se dividieron en tres categorías distintas.

1. Consultas para verificar correctitud de datos: Estas consultas se realizaron con el objetivo de verificar que los datos fueron correctamente transferidos a la base, para esto se compararon los resultados obtenidos de las consultas con los valores publicados [5] por las fuentes, sin embargo esto no fue posible debido a que los valores publicados no estaban actualizados respecto al repositorio [6] en GitHub, por lo tanto se contrastaron los resultados con los del archivo .csv obtenido del repositorio.
2. Consultas para comparar la eficiencia de la base respecto a una base de datos relacional: Con el objetivo de evidenciar las diferencias en eficiencia de una base respecto a la otra, se realizaron consultas que utilizaran una gran cantidad de relaciones o que generaran una gran cantidad de tuplas. Cada consulta de esta categoría deberá realizarse tanto para la base de datos de grafos como para la base de datos relacional, para luego comparar las eficiencias entre los dos modelos.
3. Consultas complejas: Estas consultas tendrán el objetivo de explotar al máximo la representación de la base, por lo que buscarán ser específicas para grafos e imposibles (o difíciles) de realizar en relacional.

V-A. *Consultas para Verificar Correctitud de Datos*

En este apartado se describirán las consultas realizadas con el fin de verificar la correctitud de los datos. Para esto se dará una breve descripción de la consulta, luego los resultados esperados de las mismas y por último los resultados reales arrojados por la base.

V-A1. Muertes por Temporada: Descripción: Devuelve la cantidad de muertes ocurridas en cada temporada.
Resultado esperado: Season 1: 59, Season 2: 130, Season 3: 87, Season 4: 181, Season 5: 246, Season 6: 540, Season 7: 1096, Season 8: 4548
Resultado obtenido: Season 1: 59, Season 2: 130, Season 3: 87, Season 4: 181, Season 5: 246, Season 6: 540, Season 7: 1096, Season 8: 4548

V-A2. Locaciones más Mortíferas: Descripción: Devuelve los 5 lugares en donde ocurrieron más asesinatos junto con la cantidad de asesinatos cometidos en esos lugares, ordenadas en orden descendente sobre la cantidad de muertes.
Resultado esperado: Winterfell 3,709, King's Landing 1,357, Beyond the Wall 993, Meereen 154, Goldroad 116
Resultado obtenido: Winterfell 3,709, King's Landing 1,357, Beyond the Wall 993, Meereen 154, Goldroad 116

V-A3. Armas más Utilizadas: Descripción: Devuelve las 5 armas con las que se cometieron la mayor cantidad de asesinatos y cuantos fueron cometidos con cada una, ordenadas en orden descendente sobre la cantidad de muertes.
Resultado esperado: Animal 1,756, Magic 1,244, Blade 803, Fire/Burning 335, Arrow/Bolt 209

Resultado obtenido: Animal 1,756, Magic 1,244, Blade 803, Fire/Burning 335, Arrow/Bolt 209

V-A4. Asesinos más Mortíferos : Descripción: Devuelve los 5 personajes que hayan cometido la mayor cantidad de asesinatos junto con el arma más utilizada por los mismos, ordenados en forma descendente sobre la cantidad de asesinatos cometidos.
Resultado esperado: Drogon 1,376 kills Deadliest weapon: Dragonfire, Arya Stark 1,191 kills Deadliest weapon: Valyrian steel dagger, Rhaegal 273 kills Deadliest weapon: Dragonfire, Cersei Lannister 197 kills Deadliest weapon: Wildfire, Jon Snow 101 kills Deadliest weapon: Sword (Longclaw)
Resultado obtenido: Drogon 1,376 kills Deadliest weapon: Dragonfire, Arya Stark 1,191 kills Deadliest weapon: Valyrian steel dagger, Rhaegal 273 kills Deadliest weapon: Dragonfire, Cersei Lannister 197 kills Deadliest weapon: Wildfire, Jon Snow 101 kills Deadliest weapon: Sword (Longclaw)

V-B. Consultas para comparar la eficiencia de las bases

En este apartado se describirán las consultas realizadas con el objetivo de comparar la eficiencia entre la base de datos relacional y la base de datos de grafos. Cada consulta tendrá una descripción y el tiempo requerido para realizarse (este tiempo fue calculado utilizando la biblioteca time de python).

V-B1. Método más Utilizado por el Honor: Descripción: Devuelve el nombre del honor y el método que fue más utilizado por asesinos que poseían este honor.
Tiempo utilizado por la base de datos de grafos: $5{,}79357 \times 10^{-6} s$
Tiempo utilizado por la base de datos relacional: $0{,}00542302 s$

V-B2. Interacciones más significativas: Descripción: Para cada par de personajes que hayan interactuado, devuelve el peso y la temporada de la interacción con mayor peso.
Tiempo utilizado por la base de datos de grafos: $3{,}743171 \times 10^{-6} s$
Tiempo utilizado por la base de datos relacional: $4{,}35948 \times 10^{-5} s$

V-B3. Cantidad de Veces Asesinado y con qué Método: Descripción: Para cada personaje asesinado, devuelve su nombre, la cantidad de veces que fue asesinado y una lista con los métodos.
Tiempo utilizado por la base de datos de grafos: $5{,}06639 \times 10^{-6} s$
Tiempo utilizado por la base de datos relacional: $0{,}0130381 s$

V-B4. La categoría más utilizada en un episodio: Descripción: Para cada episodio, devuelve la categoría más utilizada para matar en ese episodio.
Tiempo utilizado por la base de datos de grafos: $6{,}36577 \times 10^{-6} s$
Tiempo utilizado por la base de datos relacional: $0{,}0159626 s$

V-C. Consultas Complejas

En este apartado se describen las consultas que se favorecen de la representación de la base en forma de grafos. Aquí se mostrará simplemente una descripción de la consulta, sin embargo se encuentra publicado en el repositorio del proyecto [12] un notebook con el código utilizado.

V-C1. Distancia mínima entre Ayra Stark y Daenerys Targaryen: Descripción: Devuelve la distancia mínima entre los personajes Ayra Stark y Daenerys Targaryen, utilizando las relaciones vinculadas al nodo que representa la muerte (Kill) siendo estas: "$Victim\_Of$" y "$Commits$". No se utilizara la relación de interacción entre los personajes.

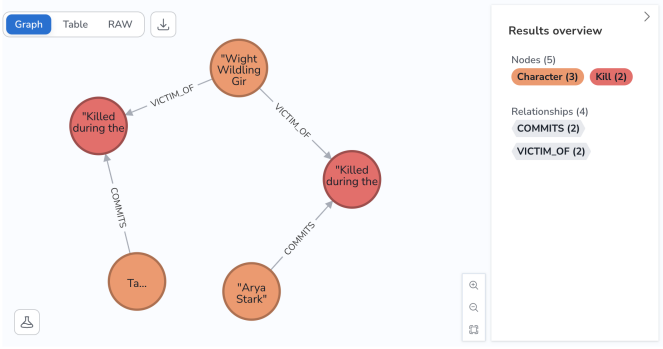

Figura 3: Grafo resultado de la consulta V-C1

V-C2. Cantidad de Caminos de distancia 4 entre Ayra Stark y Daenerys Targaryen: Descripción: Devuelve la cantidad de caminos de largo 4 que conectan a Ayra Stark con Daenerys Targaryen a través de las relaciones "$Victim\_Of$" y "$Commits$" que las vinculan con alguna muerte.

V-C3. Asesino y ubicaciones donde comete sus asesinatos: Descripción: Devuelve el nombre de los personajes asesinos junto con un listado de los lugares donde cometieron los asesinatos.

V-C4. Las Muertes más Importantes: Descripción: Devuelve el nombre de los personajes que fueron asesinados en las muertes con valor 4 de importancia (valor máximo) y el asesino correspondiente a dicha muerte.

V-C5. Personajes que mueren más de una vez: Devuelve el nombre de los personajes que han sido víctimas de más de una muerte. Esto puede ocurrir en el caso de que el personaje haya revivido o si el nodo personaje representa a un grupo genérico, este caso ocurre si el personaje es un animal o no posee un nombre único y es representado por su función (ej: soldier). Un personaje conocido que muere más de una vez es "Beric Dondarion", en el resultado de la consulta se pudo corroborar dicha información.

V-C6. Víctimas que mataron a alguien luego de morir: Devuelve el nombre de los personajes que han muerto en determinado momento y posteriormente a su muerte cometen algún asesinato. Esto puede ocurrir en el caso de que el personaje haya revivido o si en la serie se muestra posteriormente al personaje cometiendo un asesinato en una escena que refiere a su pasado, como es el caso del personaje "Ned Stark". En esta consulta se espera que el resultado contenga al personaje principal "Jon Snow" el cual revive y se encuentra en las condiciones de la consulta.

V-D. Resultados de las etapas de experimentación

Aquí se detallarán los resultados de las distintas etapas de la experimentación.

V-D1. Resultados Verificación de Datos: Como se puede apreciar en la sección V-A, los resultados de las consultas verifican la correctitud de los datos, es decir, que el proceso de carga y depuración de estos transcurrió sin perdida de información.

V-D2. Resultados Comparación Base de Datos Relacional y de Grafos: Hacer una comparación de eficiencia entre una base de datos relacional y una de grafos no es algo trivial, las primitivas que obtienen las eficiencias de las bases están diseñadas con el objetivo de detectar problemas y analizar su comportamiento, no están pensadas para hacer una comparativa entre dos bases de datos de distintos proveedores. Por lo tanto el método que se encontró viable para comparar la eficiencia de las bases fue realizar una medida del tiempo promedio requerido por consulta, el siguiente pseudocódigo muestra cómo se realizó esta medida.

Listado 1 Pseudo código obtención de los tiempos promedio de las consultas

```
start_time = time.time()
for i in range(n):
  CONSULTA

end_time = time.time()
execution_time_pro = (end_time - start_time)/n

print(execution_time_pro)
```

durante el análisis, se pudo observar que al utilizar n=20 se obtuvieron resultados consistentes y estables. Estos resultados se consideran satisfactorios, ya que no mostraron cambios significativos después de múltiples ejecuciones. Esta metodología que puede parecer imprecisa a la hora de comparar bases de datos, fue la utilizada para realizar esta comparación en la bibliografía científicas consultadas [15]. Por otro lado su propósito es establecer un orden de magnitud, no una comparación con un nivel alto de precisión, por lo que esta técnica será de utilidad en estas circunstancias.

La figura 4 muestra una gráfica que contrasta los tiempos de respuesta de las diferentes bases de datos para cada consulta realizada. Notar que para construir esta gráfica fue necesario escalar los valores ya que, debido a la cantidad de ordenes de magnitud de diferencia, era imposible obtener una gráfica útil graficando linealmente sobre el eje horizontal. Además se observa que el eje vertical corresponde a uno, esto es debido a que los logaritmos de los resultados, son negativos, ya que los tiempos son menores a uno, luego cuanto menor sea el tiempo mayor será el tamaño de la barra, se observa que la barra correspondiente a la base de datos de grafos es significativamente más grande, lo que indica que las consultas se ejecutaron de manera más rápida en comparación con la otra base.

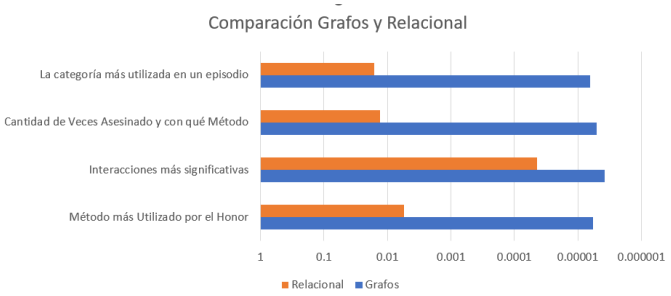

Figura 4: Gráfica comparación Relacional y Grafos.

V-D3. Resultados de las Consultas Complejas: En este tipo de consultas, se encontró una dificultad al enfrentarse a cálculos de caminos que requerían un tiempo de ejecución elevado y, en algunos casos, no se obtenía una respuesta utilizando AuraDB. Se realizó una investigación para mejorar estos tiempos mediante el uso de plugins como Neo4j Graph Data Science Library [14] y APOC (Awesome Procedures on Cypher) [13]. No obstante, se descubrió que estos no eran compatibles con la versión gratuita de AuraDB. A pesar de ello, se llevaron a cabo diversas pruebas en la versión de escritorio de Neo4j con ambos plugins instalados, pero no se logró apreciar su potencial, por lo que se considerará como trabajo a futuro.

Debido a lo anterior, se realizaron consultas limitadas en cuanto a la longitud de los caminos a buscar, con el fin de evitar un procesamiento extenso que se requeriría en caso de no acotar este valor. Estas consultas se ejecutaron exitosamente y se lograron visualizar los resultados en tiempos cortos gracias a la limitación establecida.

VI. Conclusiones y Trabajos Futuros

Como se puede observar en la Figura 4 y en los resultados de la sección de experimentación, los tiempos de ejecución que se obtuvieron en las consultas realizadas con la base de datos de grafos fueron muy inferiores a los que se obtuvieron en las mismas consultas pero para la base de datos relacional (llegando en algunos casos a ser hasta mil veces más rápidas en promedio), esto cumple la hipótesis propuesta al comienzo de este trabajo y muestra como, en ciertos casos, es conveniente utilizar una base de datos de grafos por sobre una base de datos relacional, lo que comprueba la superioridad de este enfoque en determinados escenarios.

Por otro lado, como se puede concluir de la sección “Consultas Complejas”, existen consultas posibles de realizarse en la base de datos de grafos que serían imposibles (o muy difíciles) de realizar en una base de datos relacional, lo que muestra un beneficio más de utilizar bases de datos de grafos: su mayor flexibilidad a la hora de realizar consultas. Además, se puede plantear como posible extensión a este proyecto, la ampliación de la base para ser utilizada con distintas bibliotecas y así poder realizar consultas más complejas.

No obstante, es importante destacar que estas conclusiones tan favorables no implican la sustitución total del

paradigma de las bases de datos relacionales por el de las bases de datos de grafos u otros tipos de bases de datos no relacionales. Si bien es cierto que en algunas circunstancias las bases de datos no relacionales pueden ofrecer un rendimiento superior, esto no significa que dicha propiedad se cumpla en todos los escenarios. La elección entre ambos paradigmas dependerá de la naturaleza de los datos y las consultas objetivo. Es preciso reconocer que existen situaciones en las que una base de datos relacional puede ser más eficiente que una no relacional. No obstante, los resultados obtenidos en este estudio demuestran que hay casos en los que el uso de bases de datos no relacionales es conveniente. Por tanto, es fundamental comprender la realidad planteada y los objetivos de la base de datos para tomar decisiones adecuadas sobre la representación y la persistencia de los datos.

REFERENCIAS

[1] Neo4j AuraDB. URL: https://neo4j.com/docs/aura/auradb/

[2] Google Colab. URL: https://colab.research.google.com/

[3] Python 3. URL: https://docs.python.org/3/

[4] Librería Pandas. URL: https://pandas.pydata.org/

[5] Shelly Tan. *An illustrated guide to all 6,887 deaths in 'Game of Thrones'* URL:https://www.washingtonpost.com/graphics/entertainment/game-of-thrones/
2019

[6] Shelly Tan. *data-game-of-thrones-deaths* URL:https://github.com/washingtonpost/data-game-of-thrones-deaths
2019

[7] Andrew Beveridge. *Network of thrones* URL:https://github.com/mathbeveridge/gameofthrones/tree/master/data
2021

[8] Mark Needham. *neo4j-got* URL:https://github.com/mneedham/neo4j-got/tree/master/data
2016

[9] Usuario: bladow. *GoTGraphGist* URL:https://github.com/bladow/GoTGraphGist
2016

[10] Usuario: joakimskoog. *AnApiOfIceAndFire* URL:https://github.com/joakimskoog/AnApiOfIceAndFire
2022

[11] Librería Sqlite3. URL: https://docs.python.org/3/library/sqlite3.html

[12] Repositorio GitHub del Proyecto
URL: https://gitlab.fing.edu.uy/bdnrgrupo11/bdnrGOT
2023

[13] Librería APOC (Awesome Procedures on Cypher).
URL: https://neo4j.com/developer/neo4j-apoc/

[14] Librería Neo4j Graph Data Science.
URL: https://neo4j.com/docs/graph-data-science/current/installation/

[15] Kotiranta, Petri and Junkkari, Marko and Nummenmaa, Jyrki. *Performance of graph and relational databases in complex queries*. Applied Sciences.
URL: https://doi.org/10.3390/app12136490
2022