Trabajo de Fin de Grado

Diseño de una base de datos orientada a grafos para el análisis de fraudes en aseguradoras

Design of a graph-oriented database for insurance fraud analysis

Adrián González Galván

La Laguna, 11 de julio de 2024

D. **Marcos Alejandro Colebrook Santamaria**, profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **José Carlos Rodríguez Palmero**, Director de Tecnologías de la Información, Mutua Tinerfeña, Seguros y Reaseguros, como cotutor

C E R T I F I C A N

Que la presente memoria titulada:

“Diseño de una base de datos orientada a grafos para el análisis de fraudes en aseguradoras”

ha sido realizada bajo su dirección por D. **Adrián González Galván**.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 11 de julio de 2024.

Agradecimientos

En primer lugar, quisiera agradecer a mi familia y en especial a mis padres. Ellos han sido una pieza clave durante el desarrollo de tanto este proyecto como de la misma formación académica. Sin toda la ayuda que me han brindado me hubiera sido imposible llegar a tener la formación, el desarrollo y el conocimiento que he ido adquiriendo.

De igual manera, agradezco todo el apoyo obtenido por parte de los compañeros y amigos que han ido surgiendo durante mi instancia universitaria. Ellos han hecho de lo más amena la convivencia y han sido más que una razón por la cual nunca he parado de aprender.

Por último, quisiera mencionar a Marcos, mi tutor de TFG, y a todos los miembros de la empresa colaboradora con este proyecto. Ellos se han asegurado de orientarme e informarme de una manera cercana y fomentando siempre la comprensión.

Licencia

Resumen

Actualmente estamos viviendo una época en la que la cantidad de datos que circulan por internet es inmensa y su crecimiento es exponencial. Pese a sus ventajas, el hecho de tratar con grandes cantidades de datos dificulta llevar a cabo procesos como la detección de fraudes. En este aspecto, las compañías aseguradoras son uno de los tipos de empresas que más sufren las consecuencias. Resulta por lo tanto de alta prioridad implementar herramientas que permitan lidiar con estos problemas. La necesidad de manejar estos datos de la manera más óptima y rápida posible ha hecho que entren en auge áreas tecnológicas como la Inteligencia Artificial, el Aprendizaje Automático o la visualización mediante Grafos.

La solución que se plantea en este proyecto es la utilización de las denominadas bases de datos orientadas a grafos, un tipo de base de datos que emplea grafos para su funcionamiento. Los grafos son estructuras matemáticas caracterizadas por su simplicidad y sus formas esquemáticas. Estas permiten detectar patrones, agrupaciones y anillos fraudulentos de una manera mucho más sencilla que empleando modelos de bases de datos relacionales.

El proyecto que se presenta a continuación explica detalladamente el proceso de conversión de un modelo de base de datos relacional proporcionado por una compañía de seguros a un modelo de grafos, la utilización del mismo para la detección de posibles casos de fraude y su posterior análisis de rendimiento con respecto al modelo original. El objetivo es la creación de una herramienta que permita a la empresa en cuestión poseer un medio óptimo de detección de fraudes con el que ahorrarse futuros costos y contribuir a la detección de evidencias criminales.

Palabras clave: Grafos, bases de datos, siniestros, fraudes, aseguradoras, Neo4j.

Abstract

Nowadays, we are living in a situation where the amount of data that is traveling around the internet is getting bigger and bigger. Actually, it's growing in an exponential way. Due to that fact, some kinds of processes like fraud detection have become more difficult to deal with. In that way, insurance companies have been one of the most affected companies. In order to solve this issue we need to create new types of tools. The need to handle these data as optimally and quickly as possible has led to a boom in technological areas such as Artificial Intelligence, Machine Learning and Graph Visualization.

The solution proposed in this project is the use of the so-called graph-oriented databases, a type of database that uses graphs for its operation. Graphs are mathematical structures characterized by their simplicity and schematic forms. They make it possible to detect patterns, clusters and fraudulent rings in a much simpler way than using traditional relational database models.

The project presented below explains in detail the process of converting a relational database model provided by an insurance company to a graph database model, the use of this model for the detection of possible fraud cases and its subsequent performance analysis with respect to the original model. The objective is the creation of a tool that will allow companies to have optimal means of fraud detection with which to save future costs and contribute to the detection of criminal evidence.

Keywords: *Graphs, databases, accidents, frauds, insurance companies, Neo4j.*

Índice General

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

En este capítulo introductorio se detalla la problemática dada a resolver en el proyecto, así como la justificación de la misma. En él se define el proyecto y lo que este abarca, se proporciona información acerca de la empresa colaboradora y se comentan las herramientas principalmente utilizadas para el desarrollo del mismo.

1.1 Contexto

Los grafos son herramientas matemáticas que sirven para representar de manera simbólica y esquemática estructuras de datos. Estas destacan principalmente por la utilización de relaciones y/o conexiones entre entidades. Sin duda, su simplicidad y su manera tan visual de trabajar con los datos son las cualidades por las que más destacan. Los grafos se han utilizado comúnmente para el modelado de problemas y han sido objeto de estudio por parte de científicos y matemáticos desde sus orígenes en el siglo XVIII. Hoy en día, marcan un papel importante dentro de muchos ámbitos de la informática como en la inteligencia artificial, la ciencia de datos, el aprendizaje automático, las redes neuronales o las bases de datos.

Este proyecto plantea concretamente el uso de grafos para la elaboración de un modelo de base de datos. Más específicamente hablando, se trata del tipo de modelo emergente y constantemente en auge de lo que se conoce como base de datos orientada a grafos. Un tipo de base de datos que ha adquirido gran relevancia durante los últimos años y que difiere notablemente de otros modelos relacionales y NoSQL presentes actualmente en el mercado.

El trabajo documentado en el siguiente informe explica un proceso llevado a cabo de transformación parcial de datos de un modelo de base de datos relacional, utilizado por una empresa, a un modelo de grafos. El principal objetivo que persigue es explotar las ventajas que ofrecen los grafos para la detección de fraudes dentro del ámbito de una compañía aseguradora. Así pues, en este informe se desglosa todo el proceso de elaboración de un modelo de base de datos de grafos, su incorporación dentro de un software que lo soporte y su posterior análisis.

1.2 Justificación

Estudios de compañías como PwC [1] estiman que cerca de un 40% de las organizaciones globales han sido víctimas de fraude, siendo además muchas del porcentaje restante las que lo han sido sin siquiera darse cuenta de ello. Aunque el fraude es tan viejo como la misma humanidad, y este puede tener numerosas formas, es algo realmente alarmante como el número está creciendo constantemente.

De igual manera, estamos experimentando un proceso tecnológico de cambios e innovaciones sin precedentes. Con la llegada de la inteligencia artificial, los datos masivos y la globalización de internet se requieren de respuestas cada vez más rápidas y fiables.

Los grafos son una herramienta perfecta para esta nueva etapa del mundo de la informática. Gracias a ellos se pueden responder preguntas complejas en segundos, se pueden detectar anomalías que humanos podrían pasar desapercibidas y, combinados con otras técnicas como el *Machine Learning,* se pueden generar predicciones sorprendentes.

Asimismo, dado a su carácter sintético y esquemático, los grafos hacen que el trabajo de identificar anillos fraudulentos, patrones y agrupaciones inusuales sea mucho más sencillo. Las ventajas de solventar estos problemas se traducirían en un gran ahorro de costes para muchas empresas, y en una contribución significativa en la identificación de evidencias criminales.

1.3 Empresa colaboradora

Este trabajo ha sido impulsado por medio de una propuesta de la empresa externa *Mutua Tinerfeña* [2]. Una empresa enfocada en satisfacer las necesidades del aseguramiento del mercado canario.

Mutua Tinerfeña cuenta con más de 150 empleados y 50 oficinas distribuidas entre las Islas Canarias. De entre estas, destaca el departamento de tecnologías de la información de la oficina ubicada en Santa Cruz de Tenerife, lugar donde se ha realizado el correspondiente proyecto. La compañía ofrece una cantidad extensa de seguros y servicios como el seguro de coche, moto, hogar o comercio. Sin embargo, el ámbito de relevancia y en el que se centra este proyecto es el que abarca todo lo relacionado con los accidentes viales.

1.4 Espacio de trabajo

Para la realización de este trabajo se ha hecho uso de un conjunto específico de herramientas, siendo algunas de ellas más relevantes que

otras. La herramienta principal y sobre la que se orienta todo el proyecto es **Neo4j** y su propio lenguaje de consultas **Cypher**. Neo4j es un software libre de base de datos orientado a grafos implementado en Java. La principal motivación a la hora de elegir el software vino dada por su actual popularidad.

Las otras herramientas manejadas durante el desarrollo del proyecto han sido:

- **Github**, como espacio de alojamiento de código fuente.

  En el repositorio asociado a este proyecto [3] se encuentran almacenados todos los scripts realizados en lenguaje Cypher para tanto la carga de datos como para la implementación de consultas.

- **Draw.io**, como herramienta de creación de bocetos, diagramas y modelos.

- **Microsoft 365**, como medio de compartición de ficheros por parte de la empresa externa.

- **Google Drive**, como espacio de almacenamiento de informes, históricos y otros archivos.

- **Google Meet**, como herramienta de comunicación entre las personas implicadas y asociadas al proyecto.

Capítulo 2.
Estado del arte

Este segundo capítulo engloba toda aquella información relacionada de manera directa o indirecta con el mundo de las bases de datos orientadas a grafos. Como en todo proyecto, y con el objetivo de poseer un mayor entendimiento del tema tratado, se detalla a continuación y de manera simplificada todos los conceptos teóricos relacionados con el ámbito de estudio. Entre estos destacan los conceptos de grafos, tecnologías de grafos, bases de datos relacionales y no relacionales, bases de datos de grafos y *Neo4j*. Este capítulo también sirve para proporcionar un contexto de actualidad sobre el mundo de los grafos.

2.1 Las bases de datos orientadas a grafos

Las bases de datos de grafos han estado presentes en el mundo desde hace casi 20 años. No obstante, no fue hasta el año 2017 que empezaron a adquirir cierta relevancia dentro del mercado. Actualmente, muchas empresas como Cisco, Walmart o Ebay dependen de sistemas en gran parte soportados por bases de datos orientadas a grafos. Asimismo, estudios proporcionados por numerosas webs han estimado un alto crecimiento de tecnologías basadas en grafos más allá del 2025.

Este tipo de bases de datos pertenecen al conjunto de bases de datos conocidas bajo el nombre de noSQL y son excelentemente óptimas cuando trabajan con datos altamente relacionados entre sí. A diferencia de las relacionales, las bases de datos de grafos no emplean tablas. Estas, en cambio, se basan en la teoría de grafos para su funcionamiento.

Hoy en día tenemos disponibles numerosas opciones de bases de datos orientadas a grafos. Entre estas destacan [4] Flock DB, Titan DB, Amazon Neptune, Azure Cosmos DB, Aerospike, TigerGraph, MemGraph, Apache AG y, por supuesto, Neo4j.

2.1.1 El mundo de las bases de datos

No obstante, antes de intentar definir de manera concreta que es una base de datos orientada a grafos deberíamos comenzar por explicar de manera clara el concepto de base de datos en sí y los tipos de bases de datos que existen.

- **El concepto de base de datos**

  Seguramente, lo primero que se nos vendría a la cabeza al pensar en una base de datos sería la idea de un sistema informático habilitado para el almacenamiento de datos. Algo así como un sistema dotado de una estructura y unas medidas de seguridad altamente complejas. No obstante, y nada más alejado de la realidad, un sistema de base de datos podría ser perfectamente un libro, una biblioteca, un documento electrónico o un conjunto de directorios. Las bases de datos destacan por tener muchas formas y estilos, por lo que su definición no debería estar ligada obligatoriamente a nada relacionado con la informática.

  Según lo define la propia RAE, una base de datos es *"un conjunto de datos organizado de tal modo que permita obtener con rapidez diversos tipos de información"* [5]. Así pues, podríamos definir el concepto de base de datos como cualquier conjunto de información ordenado, clasificado y facilitado para la obtención de datos.

  Sin embargo, pese a lo anteriormente dicho, las bases de datos actualmente más extendidas se sustentan en sistemas informáticos. El principal motivo es que estos sistemas permiten explotar las características que ofrecen las bases de datos de la manera más rápida y eficaz que cualquier otro.

  Actualmente, las bases de datos se dividen en 2 tipos principales: las bases de datos relacionales y las bases de datos no relacionales. Las definiciones de ambos grupos serán comentadas más adelante.

- **Bases de datos relacionales**

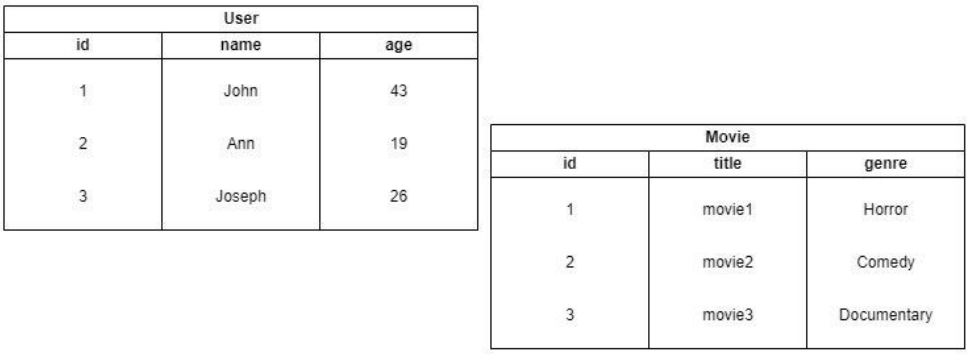
  A lo largo de la historia se ha creado una gran cantidad diferente de tipos de bases de datos. Sin embargo, entre todos ellos destacan notablemente en relevancia las bases de datos relacionales.

  Las bases de datos relacionales, también conocidas popularmente por el lenguaje de consulta que usan (SQL), han existido desde los años 70 y son, hoy en día, probablemente de las más utilizadas alrededor del mundo [6]. De hecho, si por alguna casualidad alguien ha utilizado tecnologías como Wordpress, Magento o Drupal habrá hecho uso de una base de datos del tipo SQL sin saberlo.

  Una base de datos relacional almacena información en forma de tablas, cada una de las cuales posee sus respectivas filas y columnas. En estas tablas las columnas representan los atributos o tipo de información a guardar y las filas a cada uno de los datos almacenados.

Con el objetivo de identificar cada fila, se suele emplear lo que se conoce como clave primaria. Una clave, la cual normalmente es un dígito numérico generado de manera automática e incremental, que identifica inequívocamente a cada uno de los registros almacenados en una tabla en específico.

Para ilustrar de una manera sencilla la definición de una base de datos relacional tratemos de imaginar un sistema de base de datos en el que queremos almacenar información sobre películas y sobre los usuarios que las consumen. En dicho caso, podríamos tener un modelo relacional como el que se ha elaborado en la siguiente figura.



User		
id	name	age
1	John	43
2	Ann	19
3	Joseph	26

Movie		
id	title	genre
1	movie1	Horror
2	movie2	Comedy
3	movie3	Documentary

Figura 2.1 Ejemplo simple de modelo relacional.

Como se aprecia en la figura 2.1, el modelo relacional de ejemplo cuenta con dos tablas: una tabla *User*, que almacena registros de usuario a través de los campos *id*, *name* y *age*; y una tabla *Movie* definida con los campos *id*, *title* y *genre*. Para estas tablas observamos que cada una posee tres registros, los cuales se corresponden con las tres filas que tiene cada una.

- **Bases de datos no relacionales**

  La gran relevancia que adquirieron las bases de datos relacionales hizo se que se estableciera bajo el nombre de “base de datos no relacional” a todo tipo de base de datos que difiriera con de este modelo.

  Estas llamadas bases de datos no relacionales almacenan registros sin necesidad de usar estructuras tan rígidas como lo son las tablas. También conocidas bajo el nombre de *noSQL*, se clasifican en muchos tipos diferentes y no poseen una estructura fija.

  Entre los tipos de bases de datos no relacionales que podemos encontrarnos existen algunos como:

  - Bases de datos del tipo key-value: Bases de datos que almacenan valores que luego identifican a través de una llave o *key*. Esto aporta simplicidad a la base de datos y gran flexibilidad.

Se suele emplear cuando la cantidad de datos a almacenar es pequeña.

- Bases de datos orientadas a documentos: Bases de datos que, al igual que las del tipo key-value, almacenan datos empleando una llave. Sin embargo, estos datos son documentos estructurados. El ejemplo más popular que actualmente existe de este tipo de base de datos es la base de datos *MongoDB*.
- Bases de datos orientadas a grafos: Bases de datos que conforman el tema principal del proyecto y sobre las cuáles indagaremos más adelante.

Como se puede observar, el mundo de las bases de datos no relacionales es muy amplio. Dentro de este existen numerosas clasificaciones que no hemos ni mencionado. Pese a que podríamos indagar más en él, lo que realmente nos interesa a la hora de definirlo es dar contexto y ubicación a las bases de datos orientadas a grafos.

2.1.2 Las bases de datos orientadas a grafos

Las bases de datos orientadas a grafos pueden ser un concepto nuevo para todas aquellas personas que estén estrechamente familiarizadas con las bases de datos relacionales. Estas, como hemos visto anteriormente, pertenecen a la familia de bases de datos conocidas como noSQL y utilizan grafos como elemento central para su almacenamiento de datos.

Las bases de datos de grafos son especialmente óptimas en el relacionamiento de datos. A diferencia de las relacionales, las características que más representan a las bases de datos de grafos son la flexibilidad, la consistencia en relaciones y la capacidad de resolución de problemas complejos.

Para explicar con mayor detalle este tipo de base de datos resulta necesario definir previamente el concepto de grafo, la teoría de grafos y el origen de la misma.

- **Los grafos, la teoría de grafos y su historia**

  La historia nos cuenta que el **primer modelo de grafos** fue creado en 1736 por el físico y matemático suizo Leonhard Euler, quien en su época había realizado ya numerosas contribuciones científicas,

  Este primer modelo surgió tras tratar de resolver el problema conocido como *Los 7 puentes de Konigsberg*. Un problema ubicado en una ciudad de Prusia (actualmente Rusia) cuyo contexto presentaba a dos islas interconectadas a través de siete puentes al resto del territorio. La incógnita que surgió con respecto a este problema era la de si cabía la posibilidad de que se pudiese visitar todas las partes de la ciudad cruzando cada puente una única vez.

Figura 2.2 Modelado del problema de los puentes de Konigsberg [6]

En la figura inmediatamente anterior podemos observar la abstracción que hizo Euler de dicho problema, convirtiéndolo en un modelo de grafos. En este diseño, cada área de la ciudad se “convirtió” en lo que hoy definimos como nodo y cada puente en una arista. A través de esto se pudo llegar a la conclusión de que el problema era irresoluble dado al inferir un teorema que afirmaba que debían existir un número par de aristas.

Independientemente de la resolución del problema, lo que realmente nos interesa es comprender que esta ideación fue lo que dió pie a lo que hoy conocemos como **teoría de grafos**. Una teoría basada en el estudio de las estructuras matemáticas que denominamos grafos.

Los grafos cuentan principalmente con dos componentes: los nodos y las aristas. Los nodos usualmente representan objetos, estructuras de datos o tipos de entidades, mientras que las aristas representan los distintos tipos de relaciones que pueden existir entre los datos.

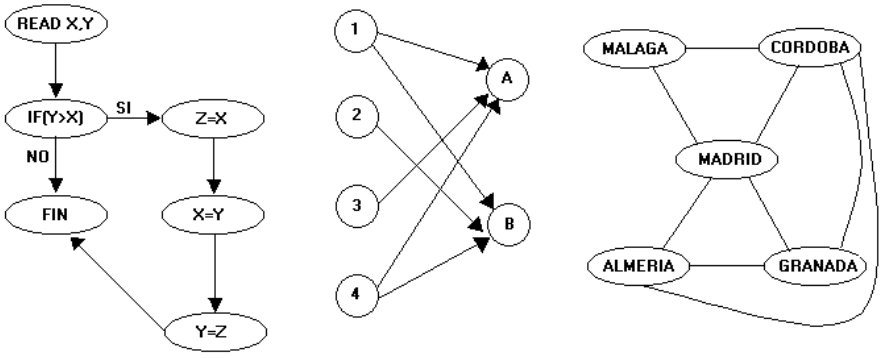


Figura 2.3 Distintos tipos de grafos [7]

En la *Figura 2.4* podemos observar ejemplos de distintos tipos de grafos que podemos encontrarnos. Estos tienden a representarse, como observamos, de una manera muy esquemática y visual.

Dentro de los grafos podemos encontrarnos con diversos tipos. Entre ellos destacamos a los grafos dirigidos y los grafos no dirigidos. Los grafos dirigidos tienen una dirección marcada en sus aristas. Los grafos no dirigidos, en cambio, poseen relaciones bidireccionales.

- **Definición del concepto de base de datos orientada a grafos**

  Tras haber comprendido lo que es un grafo resulta sencillo explicar que es una base de datos orientada a grafos. Una base de datos orientada a grafos es aquella que emplea grafos para almacenar la información. Realmente podríamos imaginarnos a cualquier base de datos orientada a grafos existente como un gran grafo lleno de miles de conexiones. En este, cada nodo representaría una entidad de la base de datos; y cada arista, las relaciones que existen entre los datos registrados.

  Es usual que nos encontremos con mucho de lo que parecen ser esquemas de bases de datos orientadas a grafos en películas o series. En numerosas series policiacas destacan la aparición de mapas que correlacionan variables a fin de poder hallar relaciones que clarifiquen la resolución de ciertos crímenes.

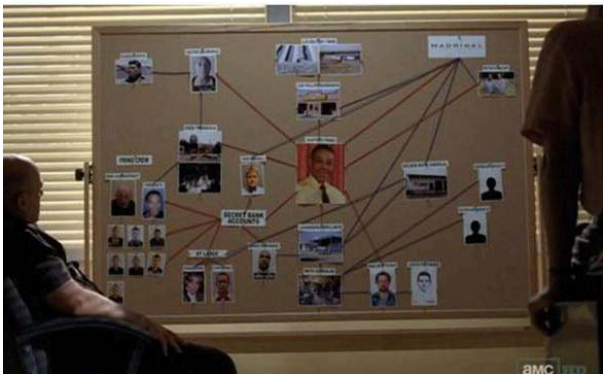


Figura 2.4 Esquema de grafo empleado en la serie *Breaking Bad* [6]

Pese a que la terminología es distinta dependiendo de la base de datos que estemos usando, el significado de sus consultas suele ser similar. En el caso en el que nos centraremos nosotros, que será el de describir la base de datos de *Neo4j*, se destaca el uso de lo que se conoce como *Labels* (en inglés) o etiquetas. Las etiquetas tienen como funcionalidad la de agrupar nodos con el objetivo de poder realizar a posteriori indexaciones mucho más eficientes.

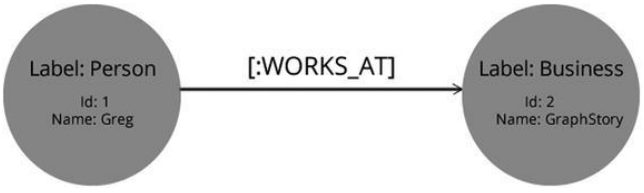


Figura 2.5 Relación de 2 nodos en una base de datos de grafos [8]

En la figura anterior podemos observar una relación entre dos nodos. Uno de ellos tiene la etiqueta *“Person”*, mientras que la del otro se denomina *“Business”*. Por otro lado, la relación que se establece entre

estos dos nodos se denomina “*WORKS_AT”* y se dirige desde el nodo con etiqueta “*Person”* al nodo con etiqueta “*Business”*. De todo esto, podemos inferir un significado realmente sencillo: una determinada persona trabaja para una compañía.

- **Las tecnologías de grafos en la actualidad**

  Actualmente, tal y como comentamos en la introducción de este informe, prácticamente todas las tecnologías basadas en grafos están sufriendo un crecimiento exponencial en popularidad. Estamos viviendo épocas en las que el tamaño y la complejidad de los datos está en constante crecimiento. Esto supone problemas para las organizaciones que desean realizar predicciones y manejar la incertidumbre y el cambio. Todo ello ha hecho que el mundo de los grafos y en especial áreas como *Graph Data Science* hayan adquirido gran relevancia por las capacidades que estas ofrecen de conectar datos y de proporcionar respuestas rápidas.

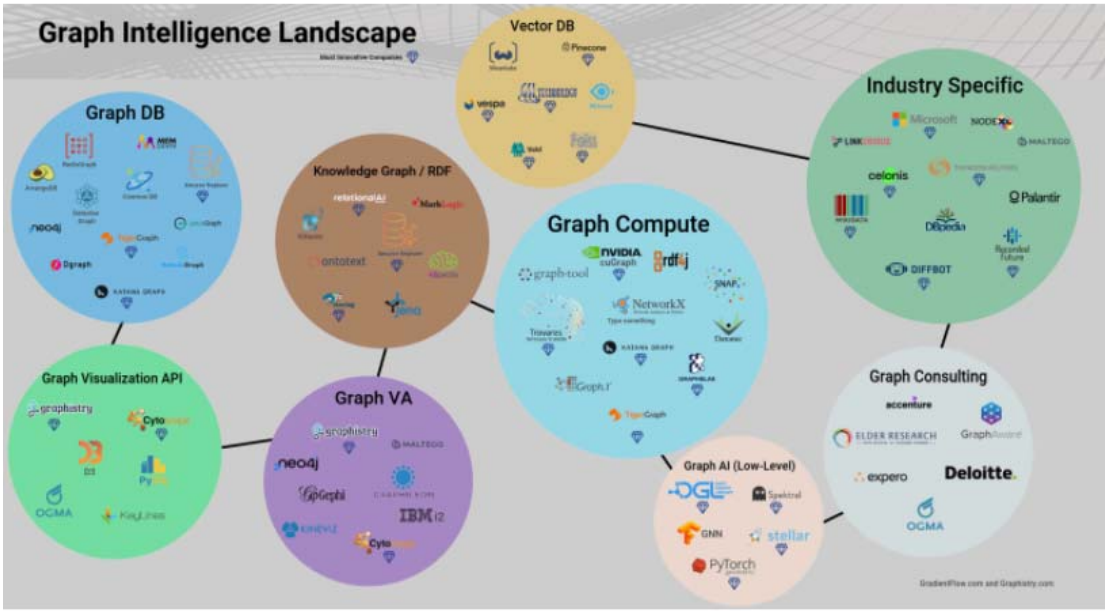


Figura 2.6 Áreas de relevancia dentro del mundo de los grafos y la tecnología [9]

En la figura 2.7 observamos que la cantidad de áreas que actualmente abarcan las tecnologías orientadas a grafos es inmensa. De entre estas destacamos con especial énfasis a la visualización y análisis de grafos o *Graph VA*, a los modelos predictivos con grafos o *Graph AI* y a las bases de datos de grafos o *Graph DBs*.

De entre estas tecnologías, es sin duda destacable la denominada *Graph VA*. Esta es una gran alternativa frente al uso de bases de datos, pues proporciona herramientas para visualizar información en forma de grafos independientemente del modo en el que esta se encuentre almacenada. Así pues, nos permite convertir modelos que pudieran ser por ejemplo relacionales en grafos.

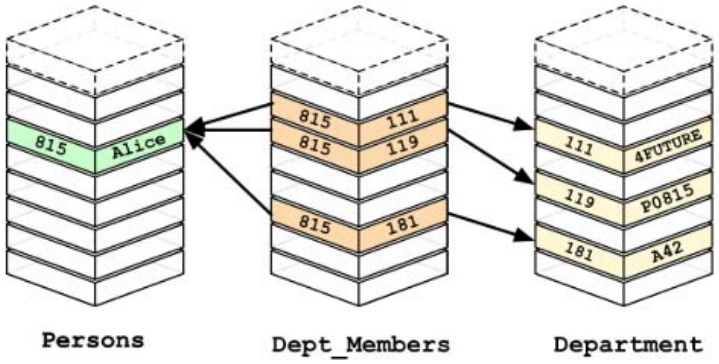


Figura 2.7 Ejemplo de modelo de datos relacional [10]

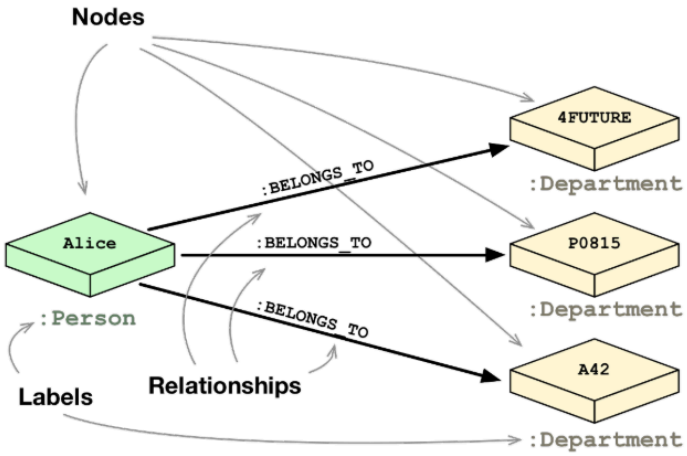


Figura 2.8 Ejemplo de modelo de datos relacional convertido a grafo a través de Graph VA [10]

Por otro lado, en el caso de los modelos predictivos se aprecia que la reciente gran evolución de la inteligencia artificial ha beneficiado notablemente el uso de grafos. Por lo que se observa, grafos e inteligencia artificial van de la mano.

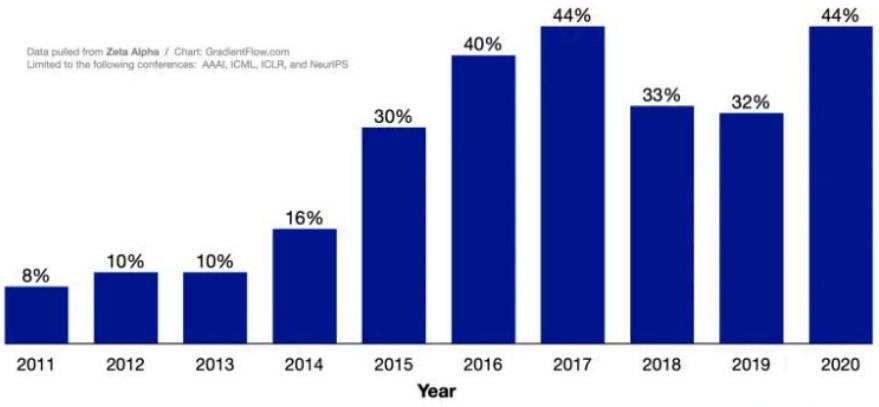


Figura 2.9 Evolución de la presencia de redes neuronales de grafos en conferencias de *machine learning* [9]

Por último, y el que conforma nuestro tema principal de estudio, las bases de datos orientadas a grafos han sido actualmente adoptadas por empresas multinacionales como *Google*, *Facebook*, *Cisco* o *Twitter*.

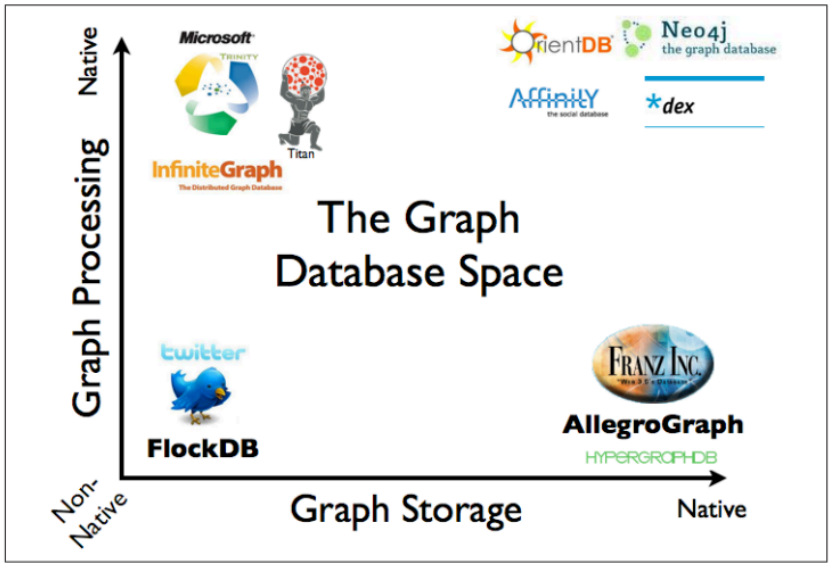


Figura 2.10 Ejemplos de algunas bases de datos de grafos clasificadas según procesamiento y almacenamiento [11]

Como se observa en la figura 2.11 contamos actualmente en el mercado con el auge de modelos como *FlockDB*, *InfiniteGraph*, *AllegroGraph* o *Neo4j*.

2.1.3 Bases de Datos Relacionales vs. Bases de Datos de Grafos

Es evidente que si simplemente queremos almacenar grandes cantidades de registros no tiene sentido emplear grafos. Para estos casos, existen las bases de datos tradicionales: las bases de datos relacionales. La utilidad que nos presentan los grafos es la de suplir la necesidad del relacionamiento

masivo de datos. Una necesidad que, como hemos visto, que está actualmente en auge.

A continuación, se recogen de manera resumida las diferencias que existen entre las bases de datos relacionales y las de grafos.

Ámbito	Bases de datos relacionales	Bases de datos orientadas a grafos
Origen	Concepto clásico. Extendidas desde hace años.	Concepto actualmente emergente.
Tipo de almacenamiento	Estructurado	No estructurado
Tipo de instrucciones	Lenguaje de alto nivel	Almacenamiento a nivel de registro
Eficiencia	Eficiente manejando grandes cantidades de datos	Eficiente manejando datos altamente relacionados
Gestión del almacenamiento	Uso de espacio de manera óptima	Requiere más espacio dado a los datos implícitos en las relaciones
Entornos de trabajo	Óptimas en entornos simples y ordenados.	Óptimas en entornos altamente complejos y con datos dispersos.

Tabla 2.1. Bases de datos relacionales y bases de datos de grafos.

En resumen, los grafos vienen a suplir todo ese uso innecesario de recursos computacionales y memoria utilizado al relacionar numerosos datos a través de tablas.

2.2 Neo4j y su lenguaje de consultas Cypher

Neo4j está actualmente reconocido por muchas páginas web como una de las mejores opciones para trabajar con bases de datos orientadas a grafos. Este, a su vez, conforma el foco principal del proyecto. En los apartados de esta sección indagaremos sobre esta herramienta.

2.2.1 Neo4j

Neo4j es un proyecto de código abierto implementado en el lenguaje *Java* que empezó a desarrollarse en el año 2003 para posteriormente ser publicado en 2007. Este se describe a sí mismo como el modelo de base de datos orientado a grafos líder mundial. Su eslogan es esta misma afirmación en inglés (*“The world 's leading graph database”*). Por otro lado, su funcionamiento está basado en una API REST con protocolo HTTP, su cliente se encuentra disponible para prácticamente cualquier sistema operativo, y su instalación no precisa de grandes requerimientos.

Figura 2.11 Logotipo de Neo4j

Una de las características a tener más en cuenta de Neo4j es su gran popularidad. El haberse posicionado en el podio de las bases de datos de grafos ha hecho que se cree una gran comunidad en torno al mismo. Una comunidad que cuenta con más de 500 eventos al año y de reuniones con cifras de más de 20.000 miembros.

Otras singularidades que nos ofrece Neo4j son su alta capacidad de lectura y escritura, su gran escalabilidad, y la facilidad de su aprendizaje.

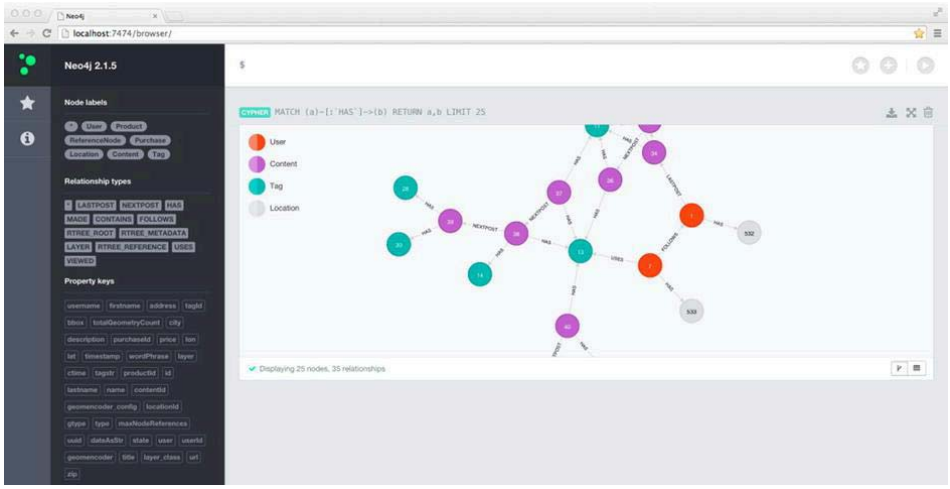


Figura 2.12 Interfaz gráfica de Neo4j [8]

2.2.2 Cypher

Las siguientes cuestiones que nos generan curiosidad o dudas cuando hablamos de Neo4j es preguntarnos qué lenguaje de consultas utiliza. Aquí es donde entra Cypher, un lenguaje de consultas declarativo y textual diseñado únicamente para Neo4j.

Cypher fue generado con la principal idea de que sea fácil de leer y manejar. Este comparte numerosas similitudes con SQL, pues la manera que tiene de realizar consultas es muy parecida. Su sintaxis maneja principalmente 4 tipos de entidades: Nodos, propiedades, relaciones y etiquetas.

1. Nodos

   Los nodos en Cypher se simbolizan empleando paréntesis. Estos representan las principales entidades de datos con las que trabaja Neo4j. Un ejemplo de nodo podría ser: *“(n)”*.

2. Propiedades

   Las propiedades se representan con llaves y se asemejan en su definición a las típicas definiciones de datos hechas en archivos con extensión *".json"*. Estas definen los datos que pueden almacenar tanto nodos como relaciones. Un ejemplo de propiedad sería *"{name: Adrián}"*.

3. Relaciones

   Las relaciones se representan con corchetes y son las que permiten establecer conexiones entre nodos. Un ejemplo de relación podría ser *"[VIVE_EN]"*.

4. Etiquetas

   La funcionalidad de las etiquetas es la de agrupar nodos como ya anteriormente comentamos. Estas no tienen una representación como tal dado a que son simplemente marcas que podemos realizar a los nodos y se representan con nombres. Un ejemplo de etiqueta sobre un nodo "n" podría ser: *"n:Persona"*.

Es importante remarcar que las propiedades pueden estar presentes tanto en los nodos como en las relaciones, pues recordemos que esa es una de las características más relevantes de Neo4j.

Dicho esto, un ejemplo de sentencia que nos permitiría crear dos nodos unidos por una relación sería la siguiente:

```
Unset

CREATE (n:Business { name : 'GraphStory', description : 'Graph as a Service' })-[r:LOCATED_IN {numLocalizations: 45}]-(m:Country {name: 'Spain'})
```

En el bloque de código inmediatamente anterior podemos ver que se han creado dos nodos, uno de tipo *'Business'* y otro de tipo *'Country'*; y una relación denominada *'LOCATED_IN'*. Todos estos elementos, cada uno de los nodos y la relación, tienen propiedades. La interpretación de esta sentencia no podría ser más clara: una determinada empresa con nombre y descripción tiene 45 localizaciones en España.

2.2.3 Sintaxis básica de Cypher

Cypher maneja una cantidad considerable de cláusulas. Al igual que cualquier otro lenguaje de consultas, Cypher debe proveer una sintaxis que permita realizar diversos tipos de operaciones con los datos.

Entre el tipo de cláusulas que podemos encontrar destacan las cláusulas de lectura, que nos permiten capturar ciertos datos; las cláusulas de proyección, que nos permiten obtener los datos capturados; las cláusulas de escritura, que nos permiten crear, eliminar y modificar datos; las cláusulas de lectura/escritura, que nos permiten modificar datos al mismo tiempo que los leemos; las cláusulas de importación de datos, que permiten extraer datos de ficheros; las cláusulas restrictivas, que nos permiten crear restricciones; y/o las subcláusulas, que nos permiten realizar operaciones más complejas en todo el tipo restante de cláusulas.

Cláusula	Definición
CREATE	Para crear nodos, relaciones y propiedades
MATCH	Para recuperar datos acerca de los nodos, las relaciones y las propiedades
RETURN	Para devolver los resultados de una consulta
WHERE	Para proporcionar las condiciones para filtrar los datos de recuperación
DELETE	Para eliminar nodos y relaciones
SET	Para modificar las propiedades de los nodos y de las relaciones
REMOVE	Para eliminar las propiedades de los nodos y de las relaciones

Tabla 2.2 Cláusulas básicas de Cypher

En la tabla 2.2 podemos observar algunas cláusulas con las que trabaja Cypher. Si bien podríamos realizar un estudio exhaustivo de cada una de ellas y ver varios ejemplos, ese no es el objetivo principal de este trabajo. En siguientes apartados, cuando se comenten los scripts de cargas de datos y de creación de consultas elaborados, se hará una mayor indagación en las cláusulas de Cypher según se considere necesario.

2.3 Aplicaciones al uso de bases de datos orientadas a grafos

En la actualidad el uso de grafos y, en particular, de bases de datos orientadas a grafos abarca numerosas y variadas áreas. No son únicamente las redes sociales las que se benefician de esto. También entornos como el marketing, el servicio al cliente, las cadenas de distribución o las finanzas pueden obtener notables beneficios al usar grafos.

Como ya hemos mencionado, hoy en día el uso de datos se encuentra masificado. Esto propicia que los grafos puedan ajustarse casi que a cualquier tipo de aplicación.

El área de estudio en la que más nos interesa indagar es en la detección de fraudes. Sin embargo, resulta de gran interés prestar aunque sea un mínimo de atención a las otras áreas de aplicación.

2.3.1 Áreas de influencia

Algunas de las áreas y más concretamente secciones en las que los grafos tienen actualmente más influencia son las siguientes [12]:

1. El alojamiento de recursos

   Uno de los problemas que enfrentan muchas empresas es el de determinar dónde exactamente alojar sus recursos con tal de optimizar al máximo sus gastos. A través del uso de grafos se pueden encontrar correlaciones y realizar predicciones de manera dinámica sobre distintas ubicaciones que determinen cuales son los sitios que por estadística tendrían mejor rendimiento. Además, la información generada puede ser fácilmente expuesta a otras personas dado al propio carácter visual y de sencillo entendimiento de los grafos.

2. La cancelación de clientes

   Los grafos permiten identificar qué clientes son más propensos a cancelar sus suscripciones en determinadas plataformas a través de establecer correlaciones con otros clientes de características similares. Tener este tipo de datos ofrecería a las empresas la posibilidad de salvar innumerables relaciones con sus clientes si se toman a tiempo las medidas adecuadas.

3. La recomendación de artículos

   Los grafos son excelentes relacionando datos como ya sabemos. Estos datos podrían ser perfectamente artículos, los cuales estuviesen relacionados con otros artículos de características similares. A la hora de analizar las compras de los usuarios los grafos pueden ser de gran ayuda dado a que son capaces de inferir que futuras compras podrían realizarse.

4. La identificación y optimización de rutas y conexiones entre puntos

   En cualquier sistema de navegación siempre interesa encontrar de alguna manera las rutas más cortas u óptimas, así como los puntos críticos que podrían generarse en las mismas. Para áreas como las de distribución de artículos este tipo de tecnologías resulta de gran utilidad, y cabe destacar que los grafos aquí también ejercen un gran impacto.

2.3.2 Detección de fraudes

Como hemos visto, la cantidad de fraudes registrados está subiendo cada vez más y más con el pasar de los años. Estos pueden tener tanto un carácter organizacional interno como externo y son debidos en gran parte a la gran cantidad de datos que manejamos hoy en día. Tener cantidades de datos masivos hace que la tarea de realizar análisis con exactitud sea algo casi imposible.

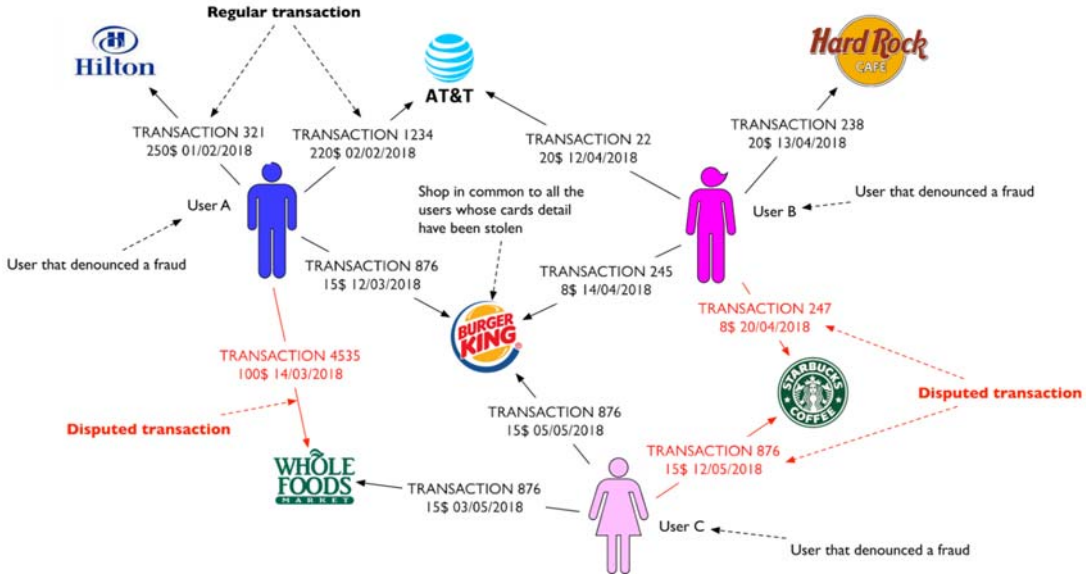


Figura 2.13 Transacciones entre compañías analizadas con grafos [13]

Los grafos permiten que las tareas de detección de anomalías, patrones inusuales y anillos fraudulentos sean algo al alcance de más empresas. Un ejemplo claro de esto podría ser el problema que supone visualizar el flujo de movimiento del dinero a través de una red. Los grafos permiten que la visualización de los seguimientos del desplazamiento del dinero sea mucho más sencilla.

2.3.3 Grafos, *Machine Learning* e Inteligencia Artificial

Si bien los grafos por sí solos ya son suficientes para solventar muchos y variados tipos de problemas, estos pueden tener un potencial enormemente mayor si se combinan con *Machine Learning* o Aprendizaje Automático e Inteligencia Artificial. Como sabemos, la IA es algo actualmente emergente, por lo que resulta casi impensable que su utilización no se vincule de alguna manera también con las tecnologías de grafos.

A través de inteligencia artificial se pueden detectar patrones y anomalías que podrían pasar desapercibidos ante el ojo humano. La manera de funcionamiento que tienen los sistemas de grafos que emplean inteligencia

artificial suele ser la de señalar y/o avisar cuando se detectan ciertos patrones.

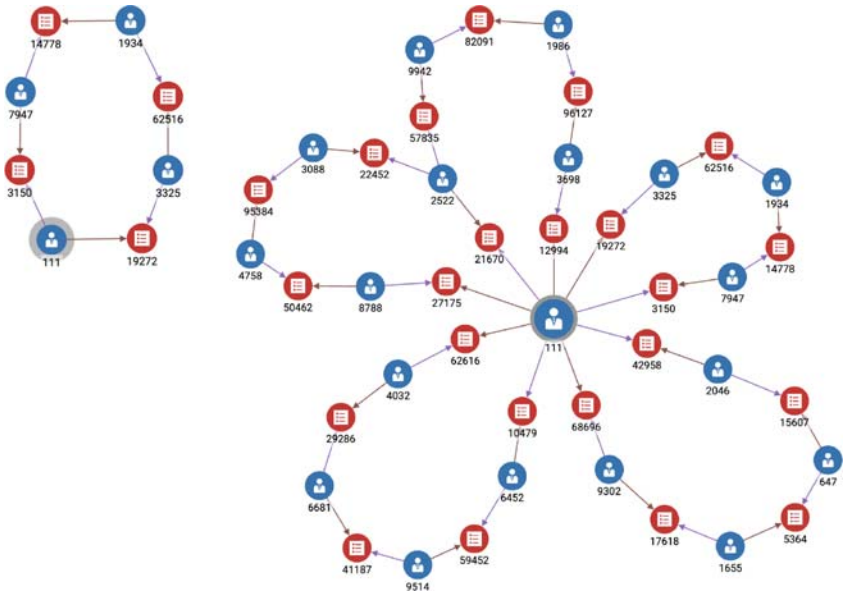


Figura 2.14 Ejemplo de patrones circulares visualizados a través de grafos [14]

El uso de *Machine Learning* e Inteligencia Artificial suele proporcionar resultados muy buenos. Sin embargo, como cualquier otro sistema de su misma índole se requiere de un entrenamiento previo que emplee los datos con los que se vaya a trabajar.

Capítulo 3.
Puesta en marcha

A través de lo visto en los otros capítulos hemos podido deducir que este trabajo tiene tres temáticas principales: las bases de datos orientadas a grafos, los siniestros y la detección de fraudes. Hemos observado que estos son temas muy vinculados entre sí, hemos analizado prácticamente cada uno de ellos y, además, hemos visto su relevancia en el mundo actual.

Este tercer apartado deja de lado los conceptos teóricos para centrarse en los primeros pasos que se han dado para el desarrollo del proyecto, los cuáles se han centrado sobre todo en el análisis del modelo relacional.

3.1 Primeros pasos: definición de conceptos

En este proyecto se trabajó con conceptos relacionados con el mundo de las aseguradoras, los siniestros y los fraudes. Estos conceptos tienen la misma relevancia para el proyecto que aquellos relacionados con el mundo de los grafos. Esto implica que sea necesario comprenderlos para poder entender posteriormente el modelo de base de datos realizada.

Los conceptos con los que principalmente se trabajó fueron los siguientes:

1. **Pólizas y contratos**

   Un contrato se define como un acuerdo entre una compañía aseguradora y un cliente. En este, la entidad aseguradora se compromete a ofrecer cobertura frente a una serie de riesgos a cambio de una compensación económica por parte del asegurado.

   La póliza, por otro lado, vendría a ser el documento escrito que registra las estipulaciones del contrato realizado. A través de este escrito se prueba la existencia de un acuerdo de seguro entre las partes vinculadas.

2. **Siniestros**

   Se define como siniestro a cualquier tipo de acontecimiento que produce daños garantizados en una póliza. En otras palabras, es el evento que hace que el asegurado tenga derecho a recibir una compensación o indemnización por parte de la aseguradora. En nuestro caso, los siniestros con los que trabajaremos se corresponden con accidentes automovilísticos.

3. **Agentes de póliza**

   Los agentes de póliza son aquellas personas o entidades que actúan como intermediarios entre los clientes y las aseguradoras. Su función principal es la de ayudar a los clientes a elegir una póliza acorde a sus necesidades.

4. **Vehículos asegurados y vehículos contrarios**

   Dentro de un siniestro, los vehículos pueden intervenir bajo el rol de asegurados o de contrarios, según estén vinculados o no a una póliza. Un siniestro, en nuestro caso, siempre tendrá asociado uno y solo un vehículo asegurado. No obstante, puede no tener asociado ningún vehículo asegurado, así como uno o inclusive muchos.

5. **Intervinientes en un siniestro**

   En un siniestro de ámbito automovilístico podemos tener diferentes tipos de intervinientes o, lo que es lo mismo, diferentes tipos de personas implicadas.

   En nuestro ámbito de estudio, las personas implicadas en un siniestro pueden ocupar los roles de peatón, ciclista, lesionado, ocupante del vehículo asegurado u ocupante de un vehículo contrario.

Si bien el mundo de las aseguradoras y concretamente el de los accidentes viales es amplio y engloba muchos más conceptos, para este proyecto solo utilizaremos los expuestos anteriormente.

3.2 Estudio del modelo relacional

Como ya hemos comentado, la empresa colaboradora utiliza una base de datos del tipo relacional para almacenar sus datos. Sin embargo, de este amplio modelo relacional se extrajo un pequeño segmento. La cuestión de porqué se ha empleado un extracto y no todo el modelo relacional atiende a las siguientes dos razones:

- El ámbito de estudio del problema se centra en la detección de fraudes en siniestros automovilísticos. Resulta por lo tanto irrelevante emplear datos sobre los otros sectores en los que está involucrada la empresa.
- El proyecto se impulsó con un propósito de desarrollo incremental e iterativo. Esto es debido a que resulta más eficiente desarrollar un trabajo cuya complejidad vaya aumentando con el tiempo que comenzar directamente con un sistema excesivamente amplio.

En los siguientes apartados se describe en detalle tanto el modelo extraído como cada una de las entidades que lo forman.

3.2.1 Modelo relacional

El modelo estudiado, analizado y a partir del cual se ha trabajado está conformado por un conjunto total de 6 tablas. Los datos que recogen son, entre otros, los contratos realizados, las personas implicadas en estos, los siniestros ocurridos, los datos de los lesionados de cada siniestro y los vehículos tanto asegurados como contrarios implicados de igual manera en los distintos siniestros.

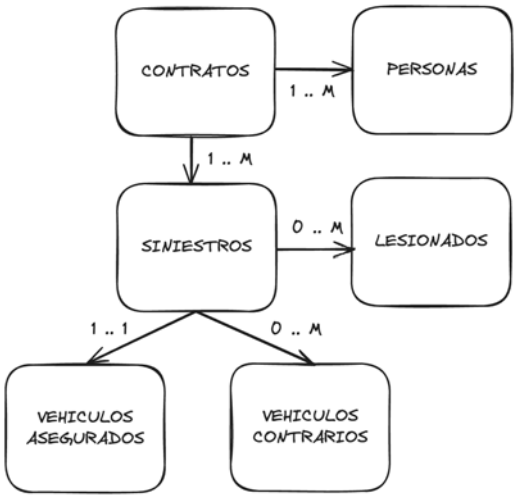


Figura 3.1 Modelo relacional del proyecto (esquema)

En la figura inmediatamente superior pueden observarse de manera clara las tablas o entidades del modelo y las relaciones que existen entre ellas.

El modelo tiene como elemento central a los siniestros. Así pues, un siniestro está compuesto por cuatro elementos relevantes: la póliza y el contrato a los que pertenece, los distintos lesionados que posee, el vehículo asegurado que está implicado en el mismo y los posibles vehículos contrarios que también podrían estar implicados.

3.2.1 Modelo relacional: entidades que lo componen

Como ya hemos estudiado, al tratarse de un modelo relacional cada entidad que lo compone se trata de una tabla. A continuación se describen los atributos que conforman cada una de estas entidades.

- **Contratos**

  La primera tabla es la llamada *CONTRATOS* y como su nombre indica recoge información relacionada con las características de cada contrato definido.

CONTRATOS	
PK	ID_POLIZA
PK	ID_CONTRATO
	CODIGO_AGENTE
	DANYOS_PROPIOS
	PERDIDA_TOTAL
	LUNAS
	INCENDIO
	ROBO
	DAP_FRANQUICIA

Figura 3.2 Modelo relacional: entidad “CONTRATOS”

Según se observa en la figura superior, cada contrato se identifica inequívocamente a través de su identificador de contrato y el identificador de póliza al que pertenece. Así mismo, en esta tabla es donde se refleja el agente vinculado al contrato y las características que puede cubrir.

- **Personas**

  El propósito de la tabla *PERSONAS* es identificar a cada una de las personas asociadas con los diferentes contratos registrados.

CONTRATOS_PERSONAS	
FK	ID_POLIZA
FK	ID_CONTRATO
	ID_INVERVINIENTE
	COD_ROL
	ROL
	NOMBRE
	APELLIDO1
	APELLIDO2
	DIRECCION
	MUNICIPIO
	PROVINCIA
	COD_POSTAL
	FCH_CARNET
	FCH_NACIMIENTO
	SEXO
	TELEFONO1
	TELEFONO2
	MOVIL
	FAX
	EMAIL

Figura 3.3 Modelo relacional: entidad “PERSONAS”

Como observamos, la tabla recoge los identificadores de las pólizas y los contratos y a continuación los datos de una persona asociada a cada uno de ellos.

- **Siniestros**

  Para definir los siniestros registrados también se requiere de utilizar los identificadores de las pólizas y los contratos. En esta tabla se describen las características de cada siniestro registrado. Entre estas características destacan la fecha y hora en la que ocurrió el siniestro, así como en la que se registró, los tipos de daños ocasionados, la posible existencia de lesionados, el lugar donde ocurrió y si dicho siniestro resultó ser fraudulento.

CONTRATOS_SINIESTROS	
PK	ID_SINIESTRO
FK	ID_POLIZA
FK	ID_CONTRATO
	FCH_OCURRENCIA
	HORA_OCURRENCIA
	FCH_DECLARACION
	RESPONSABILIDAD_CIVIL
	ROBO
	INCENDIO
	LUNAS
	PERDIDA_TOTAL
	DANYOS_PROPIOS
	DEFENSA
	LESIONADOS
	IND_INTERVIENE_AUTORIDAD
	IND_ATESTADO
	IND_ALCOHOL_DROGA
	VIA_PUBLICA
	NUMERO
	ENTIDAD
	MUNICIPIO
	IND_INDICIO_FRAUDE
	IND_FRAUDE_CONFIRMADO
	IND_ASISTENCIA_VIAJE

Figura 3.4 Modelo relacional: entidad “SINIESTROS”

La clave primaria de esta tabla es el identificador del siniestro, pues cada siniestro tiene un identificador único e inequívoco.

- **Lesionados**

  Esta tabla recoge información acerca de cada persona lesionada en un determinado siniestro.

CONTRATOS_SINIESTROS_LESIONADOS	
FK	ID_SINIESTRO
	ROL
	VEHICULO_VIAJA
	NOMBRE
	APELLIDO1
	APELLIDO2
	EDAD
	VIA_PUBLICA
	NUMERO
	PISO
	MUNICIPIO
	PROVINCIA
	COD_POSTAL
	DAÑOS
	TELEFONO1
	TELEFONO2
	MOVIL
	FAX
	EMAIL

Figura 3.5 Modelo relacional: entidad “LESIONADOS”

Los datos de esta tabla son muy similares a los almacenados en la tabla *PERSONAS*. Su principal diferencia radica en que están vinculados a un siniestro y a que el atributo *ROL* toma los valores “ocupante del vehículo asegurado”, “peatón” o “ciclista”, entre otros.

- **Vehículos asegurados**

  Esta tabla recoge información sobre los vehículos asegurados asociados a cada siniestro.

CONTRATOS_SINIESTROS_VEHICULOS_ASEGURADOS	
PK FK	ID_SINIESTRO
	MATRICULA
	MARCA
	MODELO
	IND_FLOTA
	TIPO
	USO
	DANYOS
	CTOR_NOMBRE
	CTOR_APELLIDO1
	CTOR_APELLIDO2
	CTOR_T_SEXO
	CTOR_SEXO
	CTOR_FCH_NACIMIENTO
	CTOR_FCH_CARNET
	CTOR_SIGLA
	CTOR_VIA_PUBLICA
	CTOR_NUMERO
	CTOR_ESCALERA
	CTOR_PISO
	CTOR_MUNICIPIO
	PROVINCIA
	CTOR_ENTIDAD
	CTOR_COD_POSTAL
	CTOR_MOVIL
	CTOR_FAX
	CTOR_EMAIL

Figura 3.6 Modelo relacional: entidad “VEHICULOS_ASEGURADOS”

Cada vehículo se identifica inequívocamente por su matrícula. Los otros tipos de datos que almacena la tabla son aquellos que identifican rasgos del vehículo y del conductor del mismo. Los datos que se recogen para identificar al conductor son los mismos que para los otros modelos en los que trabajamos con datos de personas.

- **Vehículos contrarios**

  Esta tabla recoge los datos de los vehículos contrarios asociados a cada siniestro, así como los datos de los conductores de los mismos.

Figura 3.7 Modelo relacional: entidad “VEHICULOS_CONTRARIOS”

Esta tabla, a diferencia de la anterior y como ya hemos mencionado anteriormente, permite que existan varias entradas para un mismo siniestro según los múltiples vehículos contrarios que pueda tener asociado.

3.3 Metodología y plan de trabajo

A expensas de seguir un determinado orden en todo el proceso de elaboración del proyecto se ha optado por emplear una metodología en específico y por adaptarse a un plan de actividades.

La **metodología** empleada siguió lo que se conoce como modelo de metodología ágil. Este modelo se caracteriza por la mejora continua e incremental del proyecto mediante la realización de publicaciones pequeñas y frecuentes. El objetivo era plantear un modelo de trabajo que fuera creciendo en complejidad a medida que se fuese iterando sobre el mismo. Así pues, se organizaron reuniones de manera semanal en las que se evaluaba el estado actual del proyecto y se comentaba con otras personas

las dudas o bloqueos de desarrollo que iban surgiendo. La mayor parte de estas reuniones se hicieron con un único trabajador de la empresa externa, el cual era el que estaba principalmente vinculado con el manejo de las bases de datos de la organización, y se realizaron principalmente vía *Google Meet*. El modelo de trabajo que se presentó fue semipresencial. Esto significa que se dió la posiblidad de trabajar en el proyecto tanto desde la oficina como de manera remota a través de recursos proporcionados por la empresa. Esta última opción fue la que se consideró utilizar la mayor parte del tiempo.

El marco de trabajo que siguió esta metodología se apoyó en los modelos SCRUM y KANBAN. Por un lado, siguiendo el modelo SCRUM, se realizaron sprints de trabajo para gestionar el tiempo, se fomentó la adaptabilidad ante posibles cambios imprevistos y destacó ante todo la colaboración recurrente entre miembros. Por otro lado, siguiendo el modelo KANBAN, se realizó una segmentación de cada una de las tareas en otras más pequeñas y estas se etiquetaron atendiendo al estado de desarrollo en el que se encontraban en cada momento.

El **plan de trabajo** se fue organizado durante la etapa de desarrollo del modelo de anteproyecto y se compuso inicialmente de 6 actividades. Estas actividades se muestran a continuación de manera ordenada y acompañadas del tiempo de desarrollo estimado de cada una:

Tarea a realizar	Tiempo de estimación
Entendimiento de conceptos con los que trabaja la organización	3 semanas
Estudio del modelo de datos relacional utilizado	2 semanas
Planteamiento de un modelo de datos orientado a grafos correspondiente con el anterior modelo	4 semanas
Transformación total o parcial de los datos a una base de datos orientada a grafos	3 semanas
Estudio de las diversas casuísticas a resolver	1 semana
Evaluación de los resultados y comparación del modelo con respecto al anterior modelo	2 semanas

Tabla 3.1 Plan de trabajo

Podemos observar que las 2 primeras actividades han sido ya estudiadas en los apartados precedentes.

Capítulo 4.
Desarrollo

Los capítulos anteriores sirvieron para cubrir los aspectos relacionados con la descripción del problema a resolver, los conceptos que abarca el mismo, los objetivos que se plantearon y la metodología de análisis y ejecución.

Este capítulo número 4 explica en profundidad el desarrollo central del proyecto, el diseño de la base de datos orientada a grafos. En este capítulo haremos hincapié en el proceso de modelado de la base de datos, la carga de esta en Neo4j, y la implementación de consultas para poner a prueba su eficiencia.

4.1 Generación de modelos: proceso de iteraciones

El proceso de creación de la base de datos orientada a grafos comenzó con la elaboración de distintos diagramas o modelos. Como ya se comentó, se siguió un proceso de trabajo iterativo e incremental, lo que significa que se fueron elaborando distintos bocetos e iterando sobre los mismos. En total se elaboraron más de diez modelos diferentes, de los cuáles todos compartieron características similares.

Durante las **primeras iteraciones** se realizaron modelos extremadamente complejos. Esto es debido a que en estas iteraciones se intentó abarcar la mayor cantidad de entidades posibles. Asimismo, se optó por utilizar datos reales para implementar dichos esquemas. La razón de esta decisión radica en hacer que el modelo incluyera todo tipo de dato existente en el modelo relacional. Sin embargo, estos modelos resultaron difíciles de analizar por las otras personas implicadas en el proyecto, debido fundamentalmente a su complejidad. No obstante, parte del trabajo se aprovechó, puesto que se definieron algunos tipos de nodos que fueron finalmente utilizados en el modelo elegido.

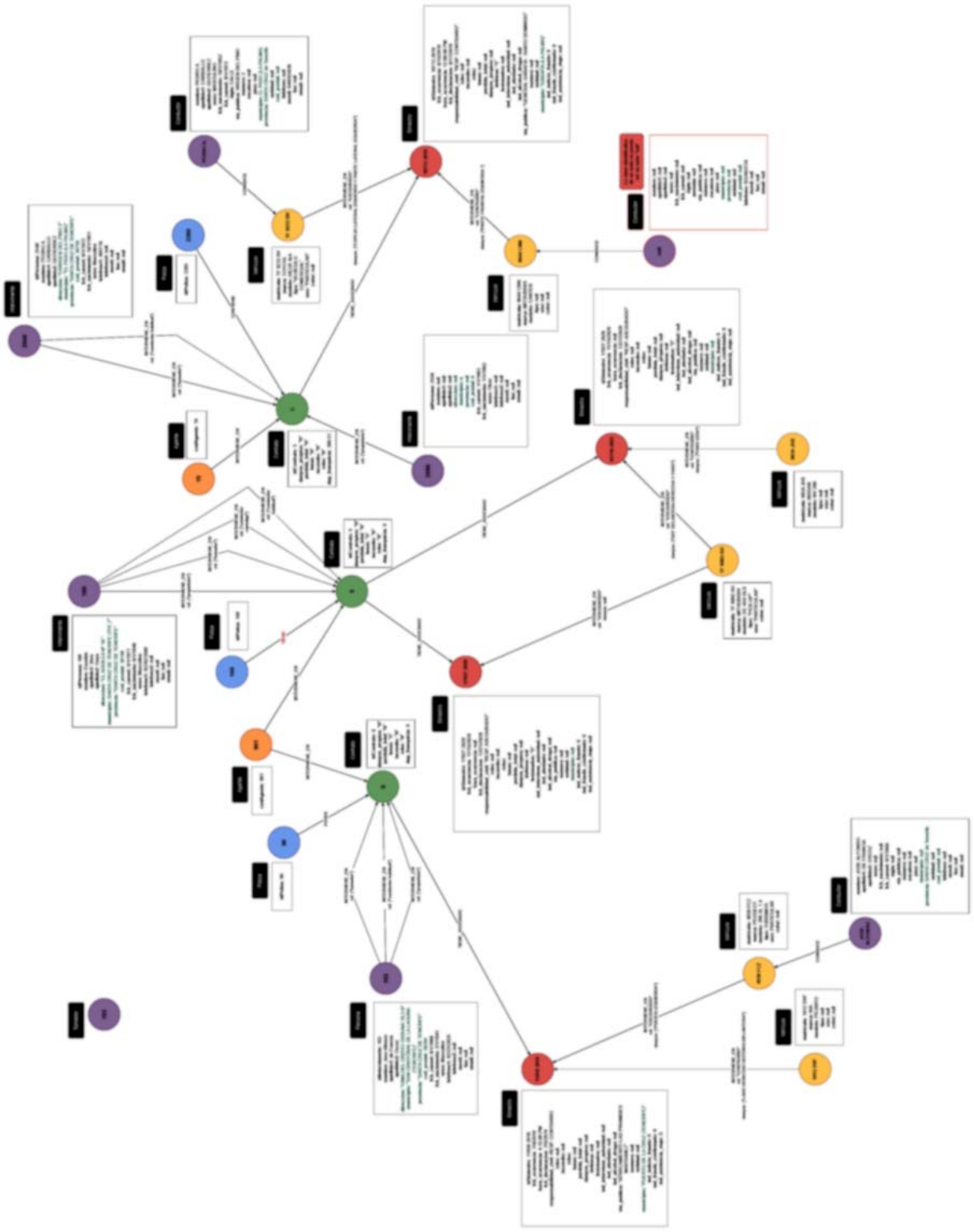


Figura 4.1 Modelo orientado a grafos: primera iteración

En **posteriores iteraciones** y, debido a lo sucedido, se optó por representar los datos de una manera más simplificada. Se eliminaron datos reales para el modelaje y, en su lugar, se optó por un modelo lo más descriptivo posible. En este último, para cada nodo se marcaba su tipo o

etiqueta. Es decir, si se trataba de una póliza, de un contrato o de un agente, entre otros.

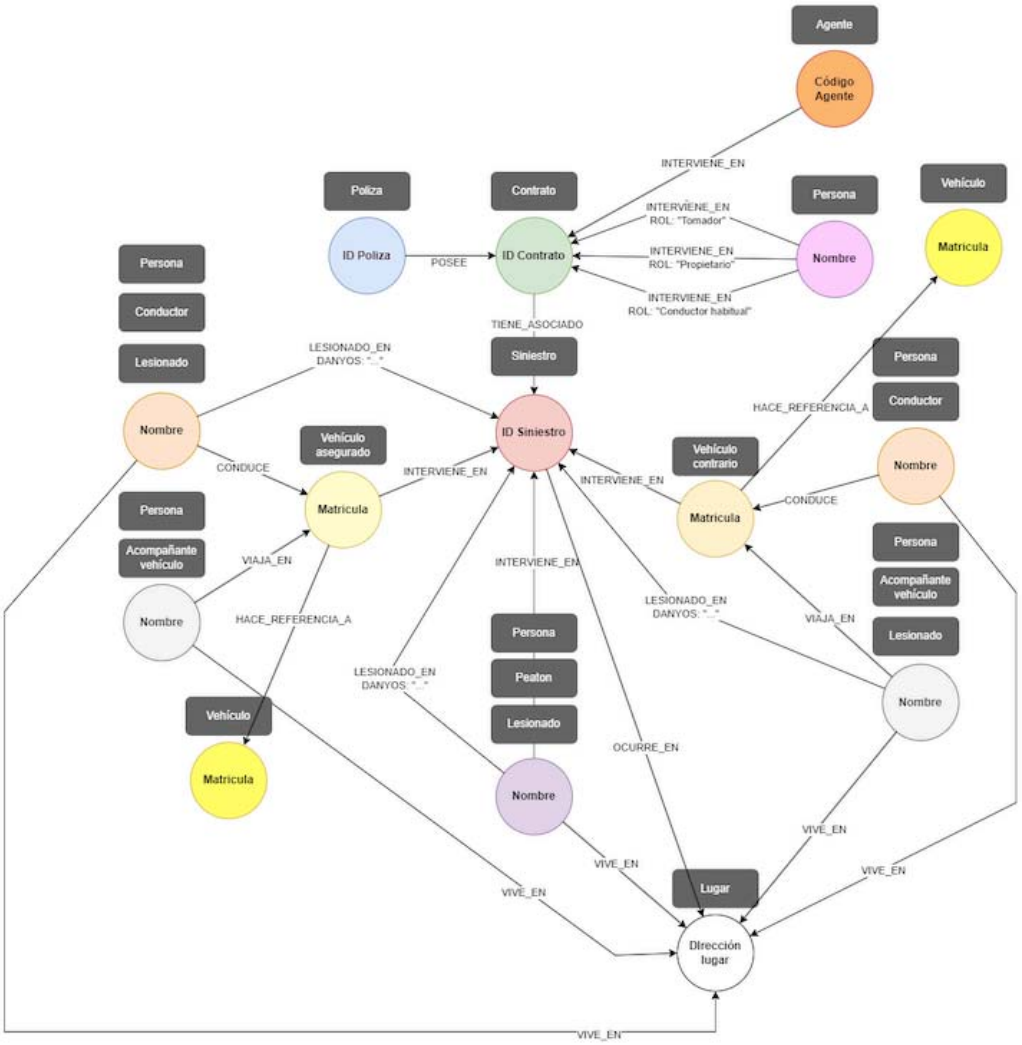


Figura 4.2 Modelo orientado a grafos: cuarta iteración

Por último, el proceso de generación de las **últimas iteraciones** tuvo como principal foco a las personas y, más específicamente, su rol de 'lesionadas'. Esto fue debido a la peculiaridad de este rol, pues una persona puede estar vinculada a un siniestro ya sea como peatón u ocupante de algún vehículo y, a la vez, no encontrarse lesionada en el mismo. Esto generó varias ideas acerca de cómo se deberían implementar correctamente este tipo de relaciones.

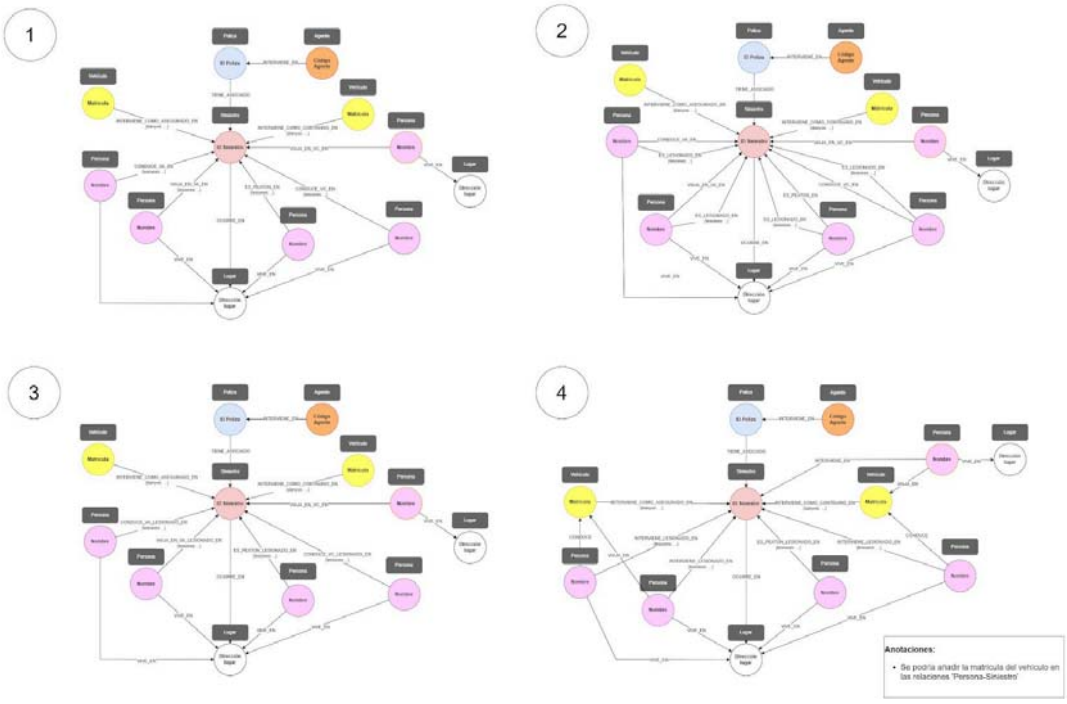


Figura 4.3 Modelo orientado a grafos: Novena iteración

Todo este proceso de iteraciones terminó desembocando en un modelo final mucho más simple que el que se planteó inicialmente.

4.2 Modelo de grafos seleccionado

El diseño del modelo de base de datos orientada a grafos finalmente escogido posee un total de 6 tipos distintos de nodos y más de 12 tipos de relaciones. Tras los análisis inferidos a partir del proceso iterativo anterior, se llegó a la conclusión de que lo más eficiente era generar un modelo sencillo. Dicho modelo se observa a continuación.

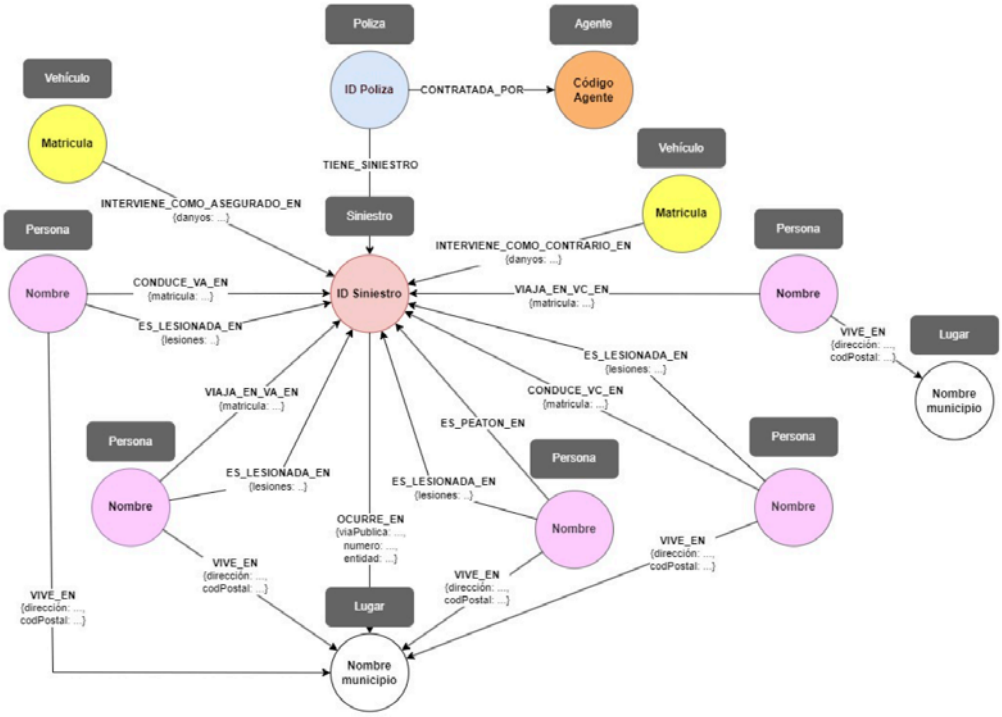


Figura 4.4 Modelo de base de datos orientada a grafos escogido

Este modelo se compone de las siguientes entidades, *"etiquetas"* o tipos de nodos: *Póliza*, *Agente*, *Siniestro*, *Vehículo*, *Persona* y *Lugar*. De entre todos ellos, el nodo que representa a los "*Siniestros"* es el más importante, pues sobre este inciden la mayoría de las relaciones presentes. Esta es en sí la pieza central del diseño.

Además, las otras **características más relevantes** de este modelo son las siguientes:

- Las personas se vinculan con los siniestros a través de dos relaciones. Una que determina qué rol cumplen ante el siniestro, y otra que determina si dicha persona sufrió lesiones o no en el mismo.
- Se ha eliminado del diseño el tipo de entidad que representa a los correspondientes contratos. En su lugar, vinculamos directamente cada siniestro con sus respectivas pólizas. Por consecuencia, esto ha provocado que se eliminen del modelo aquellas relaciones que asociaban a las personas con los contratos.
- Se ha extraído un nuevo tipo de nodo denominado *"Lugar"*, el cual no estaba a priori definido como una tabla en el modelo relacional. Dicha extracción se ha realizado a través de los datos de lugar de ocurrencia y localización de los siniestros, y de las personas, respectivamente. Este tipo de nodo es de gran importancia para la implementación de casos de uso que veremos a posteriori.
- El nodo *"Lugar"* describe un municipio de Canarias. Para poder inferir de manera más precisa la localización donde ocurrieron los siniestros o donde viven las personas se emplean las propiedades de las

relaciones *“VIVE_EN”* y *“OCURRE_EN”*. Esto se hizo así para poder centralizar más los datos de las localizaciones y generar la menor cantidad de nodos de lugar posibles.

Como se ve en la Figura 4.4, el modelo es por sí mismo bastante descriptivo. Para cada siniestro podemos inferir las personas y los vehículos que estuvieron implicados, el lugar de ocurrencia del mismo y la póliza a la que pertenece. Por otro lado, para cada póliza tenemos a los agentes que están vinculados a las mismas.

4.3 Carga de datos en Neo4j

En el anterior apartado hemos descrito el proceso que se ha llevado a cabo de creación de distintos modelos y hemos definido el modelo que finalmente tendrá nuestra base de datos orientada a grafos. No obstante, este modelo es solo un esbozo. El siguiente paso a dar es realizar su implementación a través de *Neo4j*. En este paso de carga de datos se tienen en cuenta dos temas principales: el formato de lectura de los datos del modelo relacional y los *scripts* de carga de datos realizados.

4.3.1 Formato de los datos de lectura

Antes de analizar cada uno de los scripts creados, es importante resaltar la manera en la que se extrajeron los datos contenidos en la base de datos relacional. Durante la realización de este proyecto no se trabajó directamente con la base de datos relacional, sino que se trasladó el conjunto de los datos a un formato de tablas *Excel.*

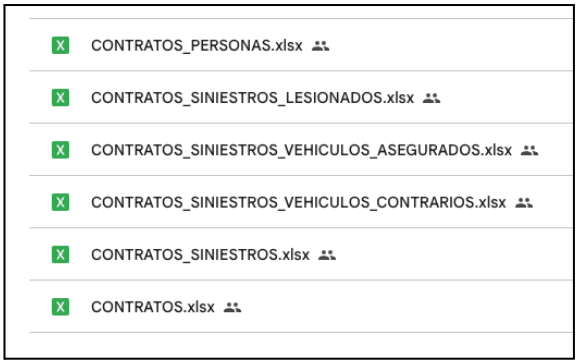


Figura 4.5 Modelo de base de datos relacional en formato de tablas *Excel*

El motivo de esta decisión surgió a raíz de seguir el mismo patrón o metodología iterativa e incremental que definimos al inicio. El usar ficheros ya predefinidos para cargar los datos facilitó posteriormente el traspaso de datos de un modelo a otro.

4.3.2 Scripts de carga de datos

Para llevar a cabo este proceso de implementación se crearon una serie de scripts en el lenguaje de consultas *Cypher*. En total se generaron un **conjunto de 7 scripts**, donde seis de estos scripts se corresponden con la carga de datos de cada una de las tablas o entidades de la base de datos relacional, y otro con la carga de los datos relacionados con los lugares. Para este último se hizo uso de los datos ya almacenados en las personas y en los siniestros.

Tal y como comentamos al inicio de este documento, se utilizó *Github* como herramienta de manejo de código. Así pues, todos los *scripts* mencionados se almacenaron en un repositorio de *Github* sobre el que se realizan actualizaciones iterativamente. Este repositorio se puede encontrar en [3] y, a continuación, se muestra una vista previa del mismo.

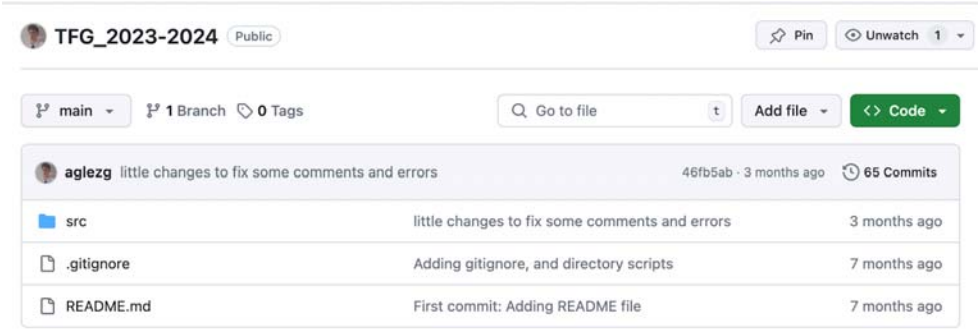


Figura 4.6 Visualización del repositorio de trabajo

El conjunto de scripts desarrollados los podemos encontrar bajo la ruta *"/src/cypher/loads"* del repositorio y todos comparten una estructura similar.

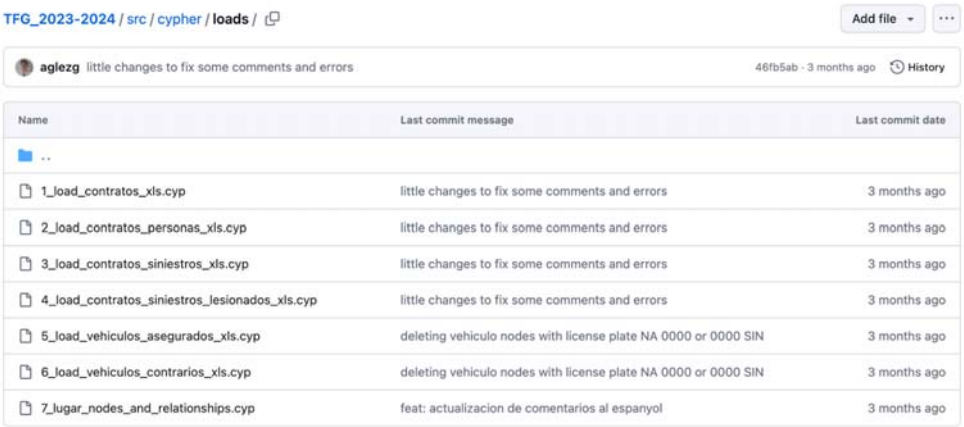


Figura 4.7 Repositorio de trabajo: carpeta de scripts de carga de datos

Cuando hablamos de estructura nos referimos a que cada uno de los scripts sigue una serie de pasos muy marcados. Pasos que se encuentran señalados por medio de comentarios y que normalmente se resumen en la eliminación de nodos y restricciones previamente ya existentes, la definición de restricciones nuevas, la creación de nodos y la creación de relaciones.

Para entender mejor esto utilicemos como ejemplo el fichero de carga de datos de la tabla *CONTRATOS*.

Como podemos observar en el bloque de código del subapartado [A.3] del Apéndice A, y como ya comentamos anteriormente, el código se encuentra segmentado en distintos bloques. Estos se encuentran todos señalados cada uno por comentarios.

El objetivo de este script es cargar los nodos correspondientes con las *Pólizas* y con los *Agentes* y crear el tipo de relación *"CONTRATADA_POR"* que une a ambos. Para realizar esto se siguen los siguientes pasos:

1. Se elimina cualquier nodo con la etiqueta *"Poliza"* o *"Agente"* ya previamente existente. Nótese que para este paso se emplea la orden denominada *"DETACH DELETE"*, la cual elimina, además de los nodos, cualquier tipo de relación asociada a ellos.
2. Se eliminan las restricciones que crearemos en el paso posterior. Este paso se hizo únicamente para no tener errores en la creación de restricciones en el caso de que se ejecutara el script repetidamente en *Neo4j*.
3. Se crea un tipo de restricción para los nodos del tipo *Poliza*. Esta define que cada identificador de este tipo de nodos debe ser único e irrepetible.
4. Se cargan los nodos *Poliza*, los cuales solo tienen una única propiedad y es su identificador.
5. Se crea otra restricción similar para los nodos del tipo *Agente*.
6. Se crean los nodos *Agente*, los cuales tienen del mismo modo un único atributo y es su código.
7. Se crea la relación mencionada que asocia a los agentes con sus respectivas pólizas.

Este código tiene algunas características a mencionar bastante interesantes, las cuales también se repiten en el resto de scripts. Estas características son:

- Se emplea la sentencia *MERGE* para crear los nodos. A diferencia de la sentencia *CREATE*, esta no genera errores si el nodo existe. Si esto sucede actúa como un *MATCH* en su lugar.
- El fichero *Excel* desde el que se cargan los datos se recorre numerosas veces. Esto se hace para segmentar la lectura de datos en secciones y, además, para evitar errores relacionados con tener datos no creados a los que estemos intentando acceder. Errores tales como crear relaciones con nodos que aún no están cargados.
- Se utiliza la orden *"apoc.load.xls"* para cargar los ficheros, la cual no está incluida de manera predeterminada en *Neo4j*. Esta es propia de un paquete externo que podemos encontrar en [15].

Como comentamos, el resto de scripts resultan ser muy similares a este. Las principales diferencias radican en que los nodos a crear pueden tener más o menos atributos y que la lógica mediante la cual se crean las relaciones puede ser más o menos compleja.

4.4 Casos de uso e implementación de consultas

Una vez ejecutado cada uno de los *scripts* de carga de datos correspondientes habremos generado el modelo orientado a grafos que estábamos buscando dentro del cliente de *Neo4j*. A través de las herramientas que nos ofrece, podremos ver todos los tipos de nodos, relaciones y restricciones creadas. Sin embargo, el contenido de la base de datos no puede visualizarse a menos que realicemos alguna consulta.

En este apartado nos centraremos en comprobar la correcta instancia de todos los datos introducidos, en comentar los casos de uso que se han estudiado y en visualizar la implementación de algunos de ellos por medio de consultas.

4.4.1 Visualización del modelo de datos generado en *Neo4j*

Pese a que *Neo4j* te permite trabajar con varias al mismo tiempo, este proyecto cuenta con una única base de datos para su funcionamiento. La configuración establecida para la misma es la que viene por defecto cuando se crea una nueva base de datos.

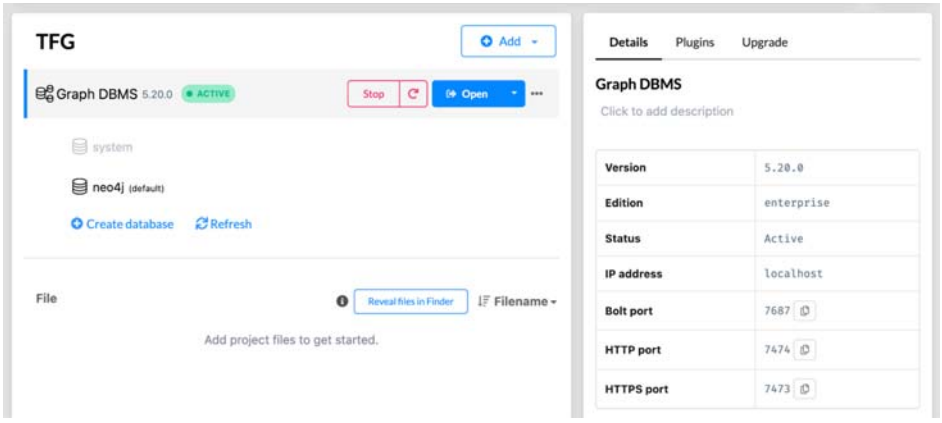


Figura 4.8 Configuración de base de datos en *Neo4j*

Una vez cargados correctamente los *scripts* en la terminal que nos ofrece *Neo4j* observaremos a partir de la siguiente figura que, entre otras cosas, se muestra esta leyenda en la pestaña de información de la base de datos.

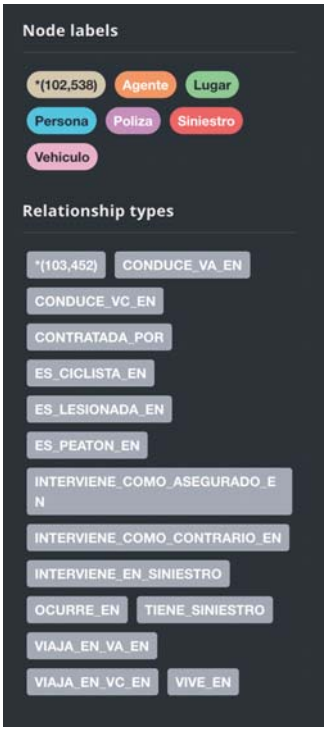


Figura 4.9 Leyenda de la base de datos creada en *Neo4j*: nodos y relaciones

A partir de la información expuesta en la Figura 4.9 podemos extraer todos los tipos de nodos con los que cuenta nuestra base de datos, así como también todos los tipos de relaciones. Asimismo, podemos también observar la cantidad de nodos y relaciones que presenta nuestra base de datos. Según se observa en la imagen hemos cargado un total de 102.538 nodos y 103.452 relaciones.

Figura 4.10 Leyenda de la base de datos creada en *Neo4j*: propiedades

Por otro lado, *Neo4j* también permite visualizar el conjunto total de propiedades cargadas en nuestra base de datos. Justo debajo de la información de nodos y relaciones podemos observar otra sección donde se visualizan estas propiedades.

4.4.2 Casos de uso

Una vez comprobado que se han instanciado correctamente todos los datos a través de los *scripts* mencionados anteriormente podemos asegurarnos de que hemos creado correctamente la base de datos en *Neo4j*. Sin embargo, y como ya hemos comentado, crear la base de datos no es el fin único de este proyecto. La utilidad de este trabajo es asegurarnos de que la base de datos creada nos permite obtener ciertos datos con una mayor facilidad que la de usando su correspondiente modelo relacional. Para ello es necesario crear consultas que nos permitan verificar esto mismo.

Las consultas que se han generado se basan en diversos casos de uso que queremos estudiar aprovechando las utilidades que nos proporcionan los grafos. Para este estudio de nuevo nos remitimos a aquellos casos que nos resultan interesantes estudiar en torno a la detección de fraudes ocurridos en siniestros.

Los casos de uso estudiados en la base de datos generada se pueden apreciar en la siguiente tabla:

Caso de uso	Descripción
[1] Vehículos reincidentes	Vehículos que tienen un siniestro con otro vehículo más de una ocasión.
[2] Lesionados reincidentes	Personas que han sido lesionadas en más de 1 siniestro.
[3] Vehículos que participan en siniestros ocurridos en el mismo lugar	Vehículos que participan en siniestros ocurridos en una misma zona más de 1 vez.
[4] Personas partícipes de un siniestro residentes en la misma localidad.	Personas que participan en un siniestro y comparten la localidad de residencia.
[5] No tercería.	Personas familiares intervinientes en un siniestro.
[6] Reclamación tardía	Siniestros cuya fecha de declaración es mayor a 60 días en comparación a su fecha de ocurrencia.
[7] Siniestros próximos	Siniestros pertenecientes a una misma póliza cuyas fechas de ocurrencia son menores o iguales a 30 días.
[8] Siniestros ocurridos durante la madrugada.	Siniestros ocurridos entre las 00:00 y las 08:00 horas de la madrugada.
[9] Exceso de lesionados	Siniestros que cuentan con un número de lesionados mayor a 3.
[10] Exceso de jóvenes implicados.	Siniestros en los que todas las personas que intervienen tienen entre 18 y 30 años.

Tabla 4.1 Casos de uso

Es importante aclarar que los resultados recogidos de aplicar estos casos de uso no implican necesariamente que encontremos siniestros fraudulentos. El proceso de detección de fraudes es algo más complejo que simplemente analizar ciertos datos dentro de una base de datos. Este requiere de un proceso de investigación y análisis más profundo de tanto la situación ocurrida como de las personas implicadas. Si bien los casos de uso anteriormente expuestos no pueden asegurarnos que lo que encontremos

sean fraudes, estos nos sirven como señalizadores. Aún así, esto resulta de igual manera muy útil ya que es una herramienta más que podemos aplicar.

Como se observa en la Tabla 4.1 hemos generado un total de 10 casos de uso distintos para comprobar la eficiencia de nuestra base de datos.

4.4.3 Implementación de casos de uso mediante consultas

El siguiente paso a realizar es analizar los casos de uso anteriormente expuestos mediante consultas a la base de datos creada. Para ello hemos hecho uso nuevamente de una serie de *scripts* similares a los ya utilizados cuando cargamos los ficheros *excels*. Si nos remitimos otra vez al repositorio, podemos encontrar este otro conjunto de scripts bajo la ruta *"/src/cypher/queries"*.

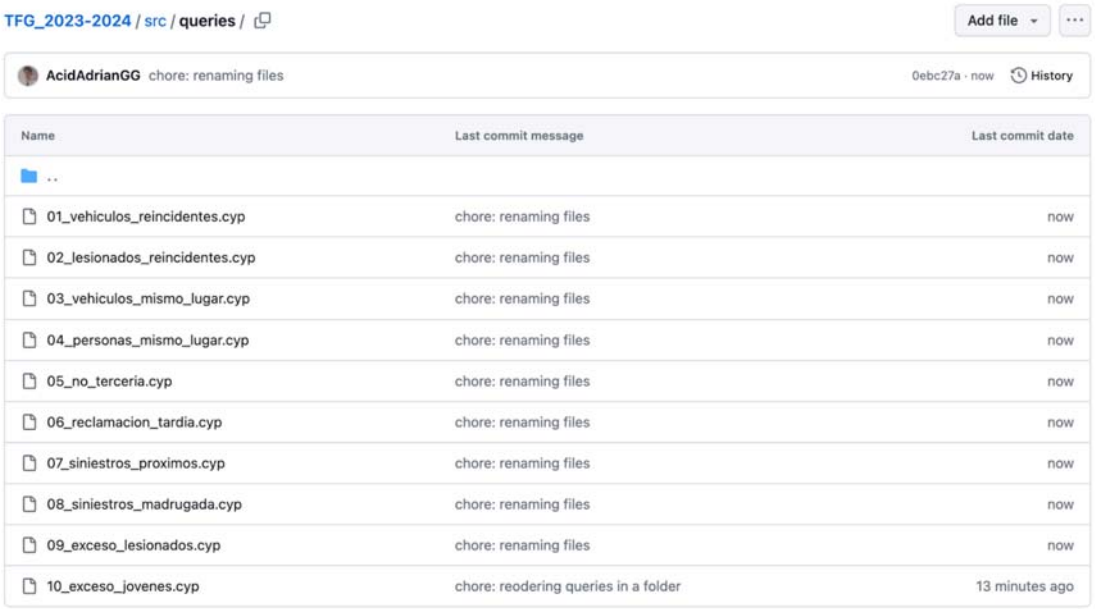


Figura 4.11 Repositorio de trabajo: carpeta de scripts de consultas

Bajo este directorio, y como se ve en la Figura 4.10, observamos que tenemos 10 ficheros de código con extensión "*.cyp"* correspondientes con los 10 casos de uso mencionados. Resulta significativo mencionar que, a la hora de realizar cada *script*, se ha intentado que los resultados proporcionados se devuelvan en forma de grafos. Esto se debe a que *Neo4j* también permite devolver resultados en forma de tablas. Algo lógico cuando queremos obtener solamente algún atributo de algún tipo de nodo en específico y no todo el nodo al completo.

Comentemos a continuación los *scripts* que resultan más significativos.

- ***02_lesionados_reincidentes.cyp***

A través de este *script* queremos comprobar casos de lesionados reincidentes en nuestra base de datos. Estos son casos en los que encontremos a personas que han sido lesionadas en más de un siniestro.

```
Unset
MATCH (p:Persona)-[:ES_LESIONADA_EN]->(s:Siniestro)
WITH
  p.nombre + ' ' + p.apellido1 + ' ' +  p.apellido2 as nombrePersona,
  COUNT(DISTINCT s) as numeroDeSiniestrosLesionado
WHERE
  nombrePersona IS NOT NULL AND
  numeroDeSiniestrosLesionado > 1
RETURN
  nombrePersona,
  numeroDeSiniestrosLesionado
ORDER BY
  numeroDeSiniestrosLesionado DESC;
```

Como observamos, este script es realmente simple. En él utilizamos varias cláusulas siguiendo un orden en específico con las que vamos estructurando la petición que queremos realizar. Estas cláusulas y su definición son:

1. Cláusula *MATCH*: a través de ella establecemos los nodos y las relaciones que nos interesa seleccionar. En nuestro caso queremos encontrar personas lesionadas en muchos siniestros. Sabiendo esto, los nodos que buscaremos serán los de *Persona* y *Siniestro* y la relación será la de *ES_LESIONADA_EN*.
2. Cláusula *WITH*: esta cláusula permite definir variables que podremos utilizar más adelante. En este caso, nos interesa recoger el nombre completo de la persona y el número de siniestros en los que ha sido lesionado.
3. Cláusula *WHERE*: nos permite establecer condiciones que han de cumplirse en la consulta. Aquí es donde establecemos que el número de siniestros contados sea mayor a 1.
4. Cláusula *RETURN*: en ella escribimos los valores que queremos retornar como resultado de la consulta.
5. Cláusula *ORDER BY*: esta última cláusula nos permite, tal y como su nombre indica, ordenar los resultados.

Ejecutando este *script* en nuestra base de datos obtenemos un total de 18 resultados. Es decir, que en nuestra base de datos contamos con exactamente 18 personas que han sido lesionadas en más de un siniestro.

Figura 4.12 Resultado de ejecutar el script "*02_lesionados_reincidentes.cyp*"

La información que hemos deseado obtener ha sido el nombre de la persona y el número total de siniestros en los que ha sido lesionado. En este caso, ninguno de los datos supera el valor de lesionado en dos siniestros.

- ***04_personas_mismo_lugar.cyp***

Otro de los *scripts* que resulta relevante comentar es el que determina aquellas personas partícipes de un siniestro que a su vez residen en la misma localidad.

```
Unset
  MATCH (s:Siniestro)<-[rel1]-(p1:Persona)-[:VIVE_EN]->(l:Lugar)
  MATCH (s)<-[rel2]-(p2:Persona)-[:VIVE_EN]->(l)
  WHERE
    p1 <> p2 AND
    type(rel1) CONTAINS 'VA' AND
    type(rel2) <> "ES_LESIONADA_EN" AND
    NOT (type(rel2) CONTAINS 'VA')
  RETURN
    s,
    p1,
    p2,
    l;
```

Este *script*, aunque corto, es ligeramente más complejo que el anterior. Para elaborarlo se han tenido en cuenta las múltiples situaciones que

podemos encontrarnos en las relaciones de las personas con los siniestros. Estas son:

1. Las relaciones tipo *LESIONADA_EN* no nos aportan ninguna información relevante en este caso.
2. Una de las personas puede ser o bien conductor o bien acompañante en el vehículo asegurado y otra de las personas debe encontrarse en el vehículo contrario o bien figurar como peatón o ciclista.

En resumen, debemos asegurarnos de que las personas implicadas hayan sido las causantes del siniestro.

Los resultados que obtenemos para este script son un total de 3 y podemos visualizarlos con un grafo que nos proporciona *Neo4j*.

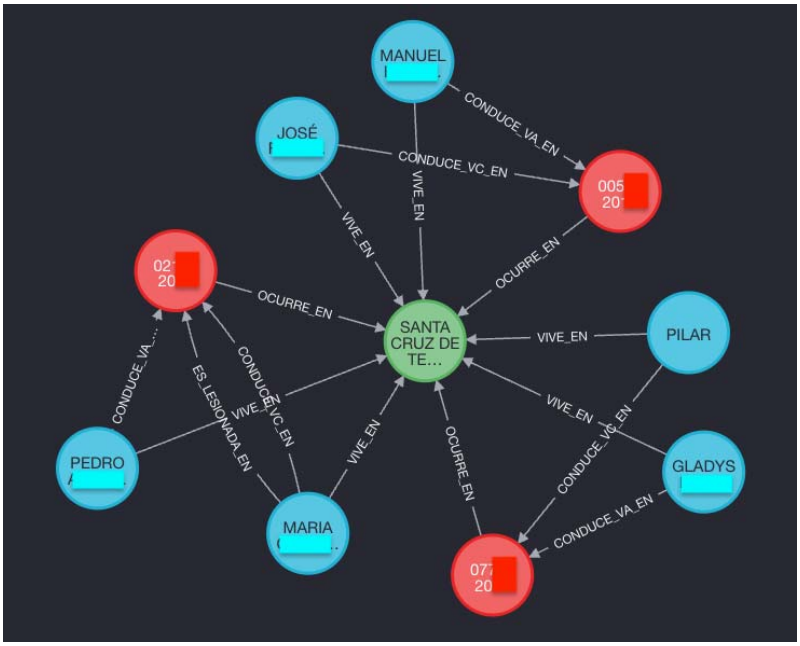


Figura 4.13 Resultado de ejecutar el script "*04_personas_mismo_lugar.cyp*"

A través de la figura inmediatamente anterior observamos que el caso de uso que estudiamos se aplica en 3 siniestros, y que cada una de las parejas de personas implicadas tiene residencia en Santa Cruz de Tenerife. El siguiente paso en el estudio es identificar la ubicación exacta de cada persona estudiada con tal de comprobar si se encuentra lo más cerca de su pareja.

- ***07_siniestros_proximos.cyp***

Por último, otro de los scripts que comentaremos en este apartado será en el que se calculan aquellos siniestros que se encuentran próximos entre sí para una misma póliza.

```
Unset
  MATCH (p:Poliza)-[:TIENE_SINIESTRO]->(s:Siniestro)
  MATCH (p)-[:TIENE_SINIESTRO]->(s2:Siniestro)
  WHERE
    s.idSiniestro <> s2.idSiniestro AND
    s2.fchOcurrencia - Duration({days: 30}) <=
    s.fchOcurrencia <= s2.fchOcurrencia + Duration({days: 30})
  RETURN
   p, s, s2;
```

Este *script* de nuevo vuelve a tener una estructura sencilla. Lo que más destaca es el uso de la orden *Duration*, la cual permite establecer periodos de tiempo. Nótese también que, a pesar de no haberlo comentado en otros *scripts*, para indicar que queremos encontrar siniestros pertenecientes a una misma póliza no necesitamos igualar los identificadores de las mismas. En su lugar *Neo4j* nos permite utilizar una sintaxis diferente. Como se puede observar en la primera cláusula *MATCH*, marcamos la póliza que queremos señalar bajo el nombre *"p"* y luego utilizamos esta variable en el *MATCH* siguiente. Con esto indicamos que se trata de la misma póliza.

El número de resultados que obtenemos al ejecutar este *script* es un total de 560. Estos son muchos más nodos de los que habíamos esperado inicialmente, lo que por consiguiente provoca que *Neo4j* tenga ciertas dificultades para mostrarlos. Para acortar el conjunto total de resultados haremos uso de otra sentencia adicional. Esta será una nueva condición dentro de la cláusula *WHERE*.

```
Unset
  (s.indFraudeConfirmado = 1 OR s2.indFraudeConfirmado = 1)
```

Con esta filtramos aquellos siniestros que hayan sido detectados ya previamente como fraudes para así poder determinar cuán cerca nos encontrábamos de detectar algún caso fraudulento. Los resultados sin embargo son muy distintos. Para este caso contamos con un total de 30 y estos si podemos observarlos a través de grafos.

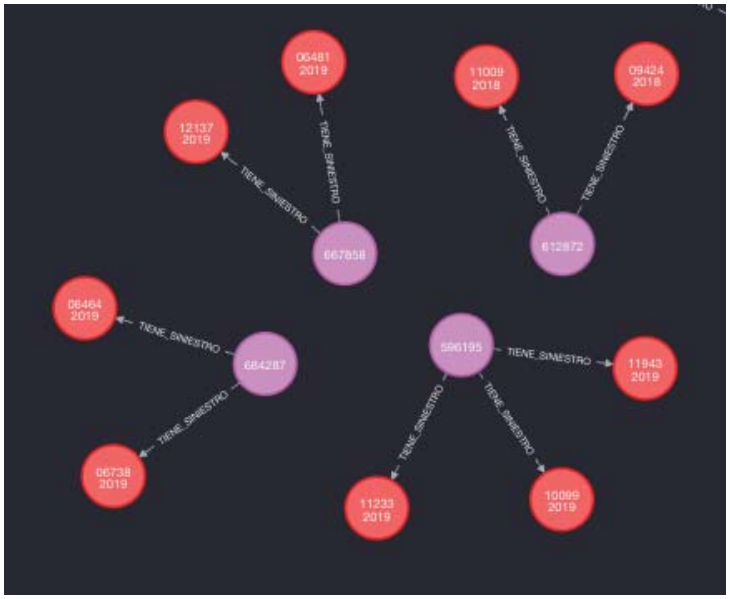


Figura 4.14 Resultado de ejecutar el script "*07_siniestros_proximos.cyp*"

En la *Figura 4.13* podemos observar algunos de los nodos resultantes de realizar la consulta. En ellos encontramos pólizas que tienen tanto 2 como hasta 3 siniestros próximos entre sí.

Capítulo 5.
Análisis y comparación de resultados

En el capítulo 4 detallamos todo el proceso de desarrollo del proyecto. Explicamos todo lo relacionado con la elaboración de los *scripts* necesarios para crear la base de datos y para realizar pruebas en la misma. Pese a que esta parte es fundamental y necesaria, resulta evidente que no obtendremos ninguna conclusión si no realizamos el análisis y las comparaciones pertinentes.

En este capítulo 5 se abordará un análisis de la base de datos desarrollada, tanto los problemas encontrados durante su desarrollo e implementación, como aquellos resultados encontrados sospechosos a través de la ejecución de los casos de uso creados, la comparación con respecto al modelo relacional y, finalmente, la determinación de si el proyecto es o no viable en cuanto a su aplicabilidad.

5.1 Análisis inferido de la base de datos desarrollada

La base de datos finalmente desarrollada destaca por su **simplicidad, optimalidad, cohesión y rapidez**. El diseño es sin duda fácil de entender y a la vez eficiente. Al tener prácticamente todas las relaciones unidas al tipo de nodo *Siniestro* se ha conseguido un diseño centralizado, lo cual nos permite ser capaces de relacionar mejor todos los datos. No obstante, este diseño también presenta algunas desventajas. Entre estas destacan la reducción de la cantidad de información almacenada en comparación con la base de datos relacional original, la no existencia de relaciones que conecten directamente a las personas con sus vehículos o pólizas asociadas, y la simplificación de los detalles en la información de los lugares.

Cabe destacar que la creación de este tipo de base de datos no ha resultado ser exactamente un proceso sencillo y lineal. La conversión de un modelo relacional a un modelo orientado a grafos acarrea como consecuencia lidiar con **ciertos inconvenientes**, y más si se trata de una base de datos que maneja datos reales propios de una empresa. Algunos de los problemas encontrados han sido, entre otros, los siguientes:

- La utilización de datos privados.

La base de datos creada ha sido alimentada con datos reales propios de *Mutua Tinerfeña*. Si bien esto no genera ningún problema a la hora de trabajar con ellos, si lo hace a la hora de exponer de cara al público el trabajo desarrollado. Así pues, este hecho limita la cantidad de información que se puede mostrar. Es por ello que en este informe se recogen simplemente el modelo desarrollado, los *scripts* generados y algún que otro resultado obtenido de consultas.

- Los datos en las tablas *Excel*.

  Como ya hemos comentado, los datos de un modelo a otro se han proporcionado empleando ficheros *Excel* de por medio. El no tener una herramienta que de alguna manera nos permita trabajar con ambos modelos al mismo tiempo nos obliga a emplear algún tipo de intermediario pese a los inconvenientes que esto conlleva. Al recorrer estos ficheros de datos hemos localizado ciertos datos requeridos que se encontraban vacíos, y hemos recogido también datos que se encontraban desplazados con respecto a la columna original de su tabla.

- La configuración de *Neo4j*.

  Por último, destacar los ajustes realizados dentro de los archivos de configuración de *Neo4j*. Según se comenta en la propia documentación del programa, el cliente de *Neo4j* necesita una serie de especificaciones de configuración distintas dependiendo de la máquina que esté utilizando. Una de estas configuraciones es la de las recomendaciones del uso de la memoria, para la cual *Neo4j* te proporciona un comando cuya documentación puede encontrarse en [16]. La resolución de este problema resultó ser de alta prioridad dado a que muchos de los documentos *Excel* contenían numerosas ristras de datos, las cuales tardaban en procesarse y en ocasiones producían errores.

  Por otro lado, destacamos el uso de paquetes externos en la realización de este proyecto. En concreto la instalación de la librería *APOC* y del paquete en específico que nos permitía cargar ficheros *Excel*. La configuración e instalación de estos paquetes en el entorno de trabajo se encuentra igualmente documentada por *Neo4j* y se puede localizar en [15].

A continuación, sobre los casos de uso elaborados y sus respectivos *scripts* de consultas podemos extraer algunas conclusiones. El número de resultados que nos proporcionan los diversos *scripts* desarrollados es muy variado. Algunos de ellos nos generan cientos de nodos y relaciones, mientras que otros nos reportan menos de 10 casos. El estudio que podemos llevar a cabo para aquellos casos de resultados masivos es mínimo. Apenas

podemos extraer conclusiones dado que resulta inviable considerar como potenciales fraudes a cientos de siniestros. Una solución por la que se podría optar sería la de crear en un futuro consultas más complejas que acotaran aún más el rango de opciones que se consideran. Sin embargo, dentro de los *scripts* que devuelven pocos **resultados** se han encontrado algunos datos y relaciones que resultan curiosos de mencionar.

Estos remarcables resultados obtenidos son los siguientes:

- **La misma persona interviene en un siniestro con roles contrarios entre sí.**

  A través del análisis de los datos mediante el uso de grafos se han encontrado algunos datos que parecen ser inconexos. Uno de estos es el caso de un siniestro en particular que posee como conductora del vehículo asegurado y conductora del vehículo contrario a la misma persona. Una situación imposible de darse.

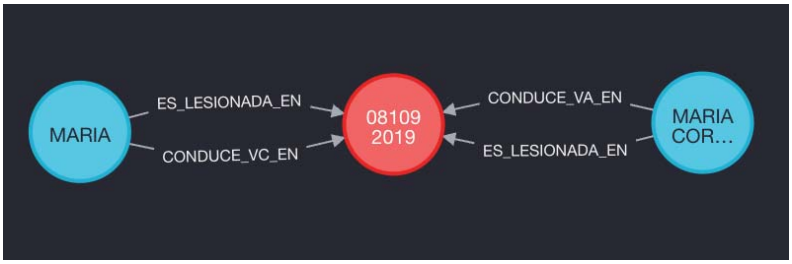


Figura 5.1 Misma persona interviene en un siniestro con roles contrarios.

  Aún así, esto no indica que nos encontremos delante de un fraude. Podría ocurrir que estos datos hayan surgido simplemente de un error generado al momento de realizar el almacenamiento.

- **Un mismo vehículo figura como asegurado y contrario para un mismo siniestro**

  De igual manera, encontramos un caso similar al anterior para los nodos tipo *Vehiculo*. En esta ocasión nos encontramos ante un vehículo que actúa bajo roles contrarios ante un mismo siniestro.

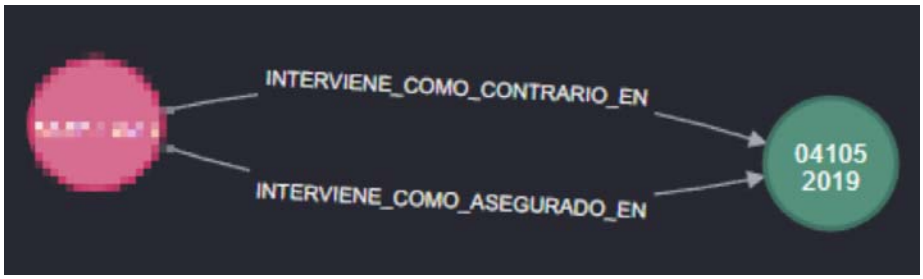


Figura 5.2 Mismo vehículo interviene en un siniestro con roles contrarios.

  La situación que puede haber generado esto puede deberse a lo mismo que se comentó para el caso anterior.

- **Un siniestro detectado por el *script* que detectaba siniestros con un exceso de lesionados resultó ser fraudulento.**

  A través de uno de los casos de uso utilizados se pudo localizar un siniestro que resultó ser fraudulento. Este caso de uso específico es el que detectaba siniestros con un exceso de lesionados.

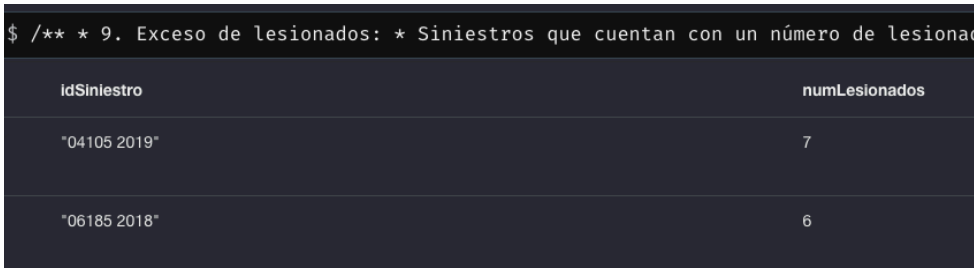


Figura 5.3 Un siniestro que cuenta con un total de 7 personas lesionadas.

En concreto se localizó un siniestro que contaba con un número total de 7 lesionados. Un número increíblemente alto e inusual, lo cual levantó sospechas de que este pudiera tratarse de un posible fraude.

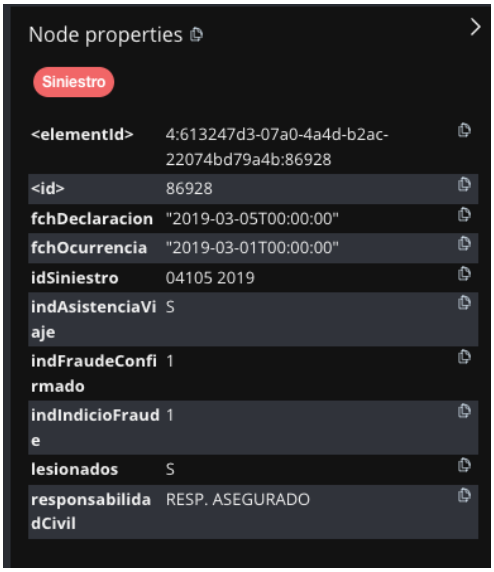


Figura 5.4 Detalles de un siniestro fraudulento

Analizando las propiedades del nodo en específico se pudo observar que efectivamente se trataba de un fraude. El atributo "*indFraudeConfirmado*" con valor "1" indica exactamente esto.

La base de datos desarrollada nos permite realizar análisis considerablemente óptimos y nos sirve como modelo del cuál partir para abarcar en un futuro muchos más datos. No obstante, el modelo desarrollado nos han impedido realizar la elaboración de ciertos casos de uso actualmente muy importantes. Casos como aquellos involucrados en la

detección de los denominados “anillos de fraude” o “anillos fraudulentos”. Un tipo de caso de fraude que se centra en la detección de ciclos.

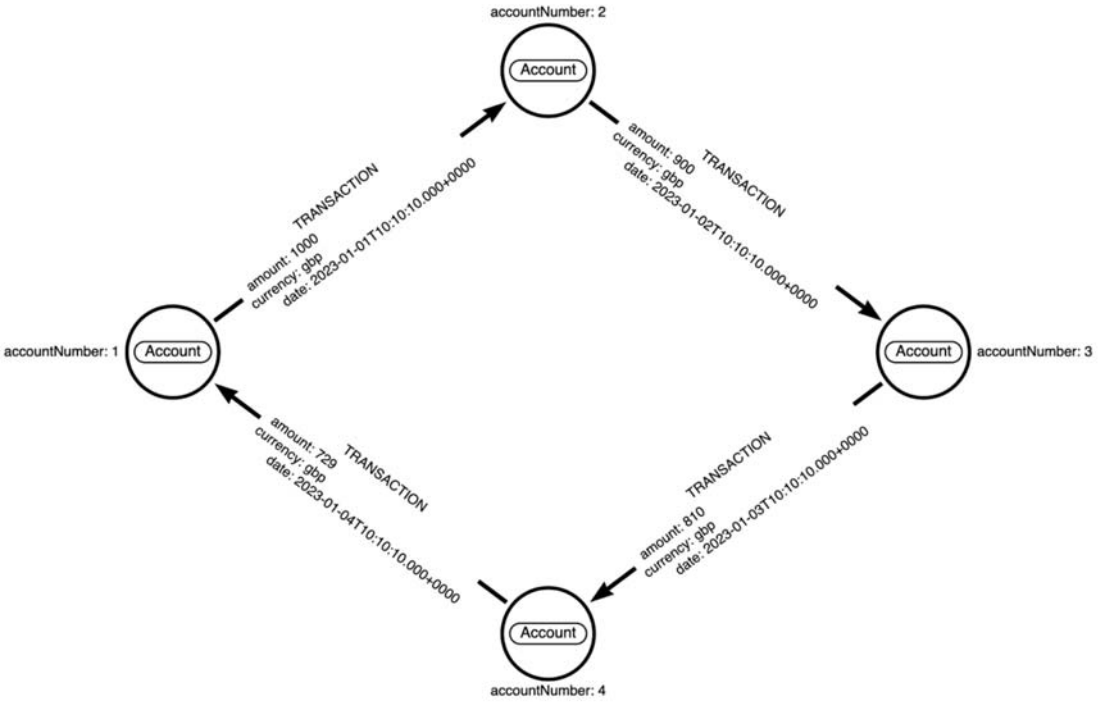


Figura 5.5 Ejemplo de anillo de fraude

A través de la *Figura 5.5* podemos apreciar un ejemplo de lo que comentábamos. Un conjunto de nodos y relaciones que finalizan su recorrido en el mismo nodo de partida.

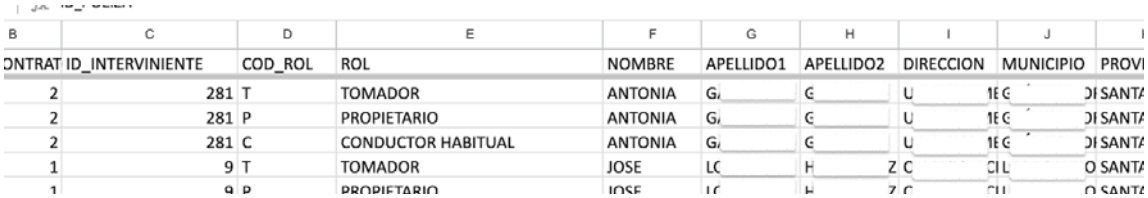
Al no contar nuestra base de datos con ningún nodo que tenga tanto relaciones salientes como entrantes, no cabe la posibilidad de encontrar ningún ciclo.

5.2 Modelo Relacional vs Modelo Orientado a Grafos

Resulta evidente que la utilización del modelo de base de datos de grafos creado nos reporta tanto una serie de ventajas como desventajas con respecto al modelo relacional. Muchas de estas ya han sido mencionadas en el apartado 2 cuando comentamos las diferencias entre estos dos tipos de modelos, y mencionamos para qué casos específicos estos resultaban más eficientes. Sabemos pues que las bases de datos de grafos destacan en cuanto a realizar consultas que impliquen mucho relacionamiento de datos, mientras que las relacionales, entre otras cosas, sobresalen por ser una muy buena opción de almacenamiento de ristras largas de información. No obstante, para el modelo de grafos generado, también existen otras particularidades que cabe destacar.

Por un lado, una de las **ventajas** que se han obtenido con la correspondiente conversión fue la simplificación de numerosa información

repetida. Para algunas tablas como la de *PERSONAS* o ambas de *VEHICULOS* tanto asegurados como contrarios se repetían para algunos casos los datos asociados a una persona.



B	C	D	E	F	G	H	I	J	
ONTRAT	ID_INTERVINIENTE	COD_ROL	ROL	NOMBRE	APELLIDO1	APELLIDO2	DIRECCION	MUNICIPIO	PROVI
2	281	T	TOMADOR	ANTONIA	G	G	U		SANTA
2	281	P	PROPIETARIO	ANTONIA	G	G	U		SANTA
2	281	C	CONDUCTOR HABITUAL	ANTONIA	G	G	U		SANTA
1	9	T	TOMADOR	JOSE	L	H	Z		SANTA
1	9	P	PROPIETARIO	JOSE	L	H	Z		SANTA

Figura 5.6 Tabla *CONTRATO_PERSONAS*: datos repetidos

Como se aprecia en la Figura 5.6, los datos de una persona podían repetirse numerosas veces en la tabla *CONTRATOS_PERSONAS* si es que esta poseía numerosos roles para un mismo contrato.

Cuando se creó la base de datos orientada a grafos se centralizaron todos estos datos de las personas en un mismo y único nodo. Lo que podría corresponder a tres filas prácticamente idénticas de datos en el modelo relacional se transforma en un único nodo en el modelo de grafos.

Por otro lado, la principal **desventaja** a mencionar es la de que la base de datos generada no incluye tanta información como incluía el anterior modelo. Recordemos que para simplificar el modelado de esta nueva base de datos se obvió el tipo de dato que identificaba a los contratos y sus respectivas relaciones. Si bien estos datos podrían no ser necesarios en una primera iteración de análisis de fraude, convendría tener en cuenta su posible inclusión en futuras iteraciones.

5.3 Aplicabilidad del proyecto

Tras todo el estudio y análisis realizado con respecto al proyecto, nos surge tratar uno de los temas más importantes. Este tema es: *¿es aplicable este proyecto? ¿Podría realizarse una sustitución de la base de datos relacional por la base de datos de grafos creada? ¿Este cambio beneficiaría a la empresa?*

La respuesta a todas estas cuestiones no es determinista. Con esto queremos decir que no podemos asegurar que la base de datos generada sea mejor o peor que la anterior en todos sus sentidos. Por el contrario, esta resulta muy eficiente para algunos casos y bastante ineficiente para otros.

Como respuesta ante este problema se considera que la **solución** más óptima sería la de utilizar **ambos modelos** según proceda. Como ya hemos comentado, el modelo generado no es más que un prototipo. Este cuenta con muchos menos datos que el modelo original. Abarca solo una parte de la información manejada dentro de la organización, por lo que sería recomendable que se siguiera iterando sobre él según se considere

necesario. Aún así, este resulta muy útil según el tipo de consultas que se quieran realizar sobre los datos. Consultas que, como ya hemos dicho, conlleven mucho relacionamiento de datos.

La decisión de instalar o no este nuevo modelo dentro de la organización no es precisamente sencilla, pues esta siempre va a depender de los intereses de la empresa, el marco situacional en la que se encuentre, y la visión estratégica que se quiera plantear, entre otros factores. Sin embargo, la combinación de ambos modelos da la posibilidad de tener estudios y análisis de los datos empresariales mucho más ricos y fiables. Un primer acercamiento a la utilización de ambos modelos pudiera basarse en mantener el modelo relacional vigente en todos sus aspectos y utilizar los *scripts* desarrollados para generar el modelo de grafos a la hora de querer tener más información en el estudio de los casos fraudulentos.

Los *scripts* desarrollados tienen la característica de generar la correspondiente base de datos orientada a grafos en menos de un par de segundos. Esto hace que no necesariamente se deba tener en un primer momento una base de datos de grafos, sino que esta pueda generarse en sólo aquellos casos en los que pueda ser necesaria.

Lo que es cierto es que la implementación de este nuevo modelo a priori no ofrece ningún inconveniente. Al contrario, enriquece el sistema actualmente implementado. De cara al futuro, y viendo los avances que están en auge con respecto a la inteligencia artificial, parece prácticamente necesario contar con cualquier tipo de modelo de base de datos orientada a grafos.

Capítulo 6.
Conclusiones y líneas futuras

Analizando lo visto en este documento, podemos sacar en conclusión que las bases de datos orientadas a grafos y ,en general, las tecnologías de grafos son ámbitos que prometen tener bastante relevancia de cara al futuro. Las facilidades que nos reportan para realizar rápidos análisis, detectar anomalías y patrones inusuales y construir estimaciones resultan más que necesarias para los tiempos actuales donde la cantidad de datos a tratar es masiva. De igual manera, en lo que respecta al ámbito de las aseguradoras, los grafos son prácticamente la solución más eficiente para tratar con la detección de fraudes.

A lo largo de la elaboración de este proyecto se ha podido indagar en el trabajo que supone la elaboración de una base de datos de este tipo, las ventajas que proporciona frente a otros modelos y las nuevas posibilidades que nos ofrece para el manejo de datos. Si bien el producto desarrollado se trata de un prototipo, este no deja de ser menos útil. Ya hemos visto que con solo un modelo simple podemos extraer otros tipos de análisis que difieren notablemente de los proporcionados por una base de datos relacional convencional.

Como se remarcó en el capítulo 5, la base de datos creada es solo el comienzo. De cara al futuro, si se quiere seguir con el desarrollo de un proyecto de ámbito similar, este trabajo puede sentar las bases del mismo. Asimismo, la combinación de una base de datos de este tipo con inteligencia artificial y/o aprendizaje automático podría generar resultados inimaginables.

Capítulo 7.
Summary and Conclusions

Analyzing what we have seen in this paper, we can conclude that graph databases and, in general, graph technologies are areas that promise to be very relevant in the future. The facilities they provide for fast analysis, detecting anomalies and unusual patterns and building estimates are more than necessary for the current times where the amount of data to be processed is massive. Similarly, as far as the insurance industry is concerned, graphs are practically the most efficient solution for dealing with fraud detection.

Throughout the development of this project it has been possible to investigate the work involved in the development of a database of this type, the advantages it provides over other models and the new possibilities it offers for data management. Although the product developed is a prototype, it is no less useful. We have already seen that with just a simple model we can extract other types of analysis that differ significantly from those provided by a conventional relational database.

As noted in Chapter 5, the database created is only the beginning. For the future, if the development of a project of similar scope is to be pursued, this work can lay the groundwork for it. Also, the combination of such a database with artificial intelligence and/or machine learning could generate amazing results.

Capítulo 8. Presupuesto

Este proyecto, como todo trabajo de fin de grado, dispone de un presupuesto. Para la creación de este presupuesto se han tenido en cuenta 2 factores: los recursos materiales utilizados y el tiempo de trabajo desempeñado.

8.1 Costes de recursos empleados

Recurso	Descripción	Coste asociado aproximado
MacBook Pro M1, 2020	Portátil personal de trabajo	2000 €
Licencia de Neo4j	Licencia de desarrollo gratuita de Neo4j	0 €
Licencia de VSCode	Licencia gratuita de Visual Studio Code	0 €
Licencia de Github	Licencia gratuita de GitHub	0 €

Tabla 7.1 Tabla resumen de los Tipos.

8.2 Costes de tiempo de desarrollo

Tarea	Duración	Coste aproximado
Búsqueda de información e investigación	45 horas	20 €/h
Estudio y análisis del modelo relacional	30 horas	20 €/h
Diseño de la base de datos orientada a grafos	65 horas	30 €/h
Creación de la base de datos en *Neo4j*	85 horas	30 €/h
Elaboración de casos de uso y consultas	65 horas	25 €/h
Evaluación de resultados y comparación de modelos	35 horas	20 €/h
Documentación	35 horas	20 €/h

Tabla 7.2 Tabla resumen de los Tipos.

A través de estos datos determinamos que el trabajo desarrollado ha comprendido una duración y costes totales de **360 horas** y **11.025 €**.

Apéndice A.
Elementos adicionales

A.1. Repositorio de trabajo

- GitHub - aglezg/TFG_2023-2024

A.2. Ejemplos de registros del modelo relacional

- Entidad "*CONTRATOS*"

```
Unset

(ID_POLIZA, ID_CONTRATO, CODIGO_AGENTE, DANYOS_PROPIOS, LUNAS, INCENDIO,
ROBO, DAP_FRANQUICIA)=(129, 30, 90, "S", "N", "S", "N", 300.0)
```

- Entidad "*CONTRATOS_PERSONAS*"

```
Unset

(ID_POLIZA, ID_CONTRATO, ID_INTERVINIENTE, COD_ROL, ROL, NOMBRE,
APELLIDO1, APELLIDO2, DIRECCION, MUNICIPIO, PROVINCIA, COD_POSTAL,
FCH_CARNET, FCH_NACIMIENTO, SEXO, TELEFONO1, TELEFONO2, MOVIL, FAX,
EMAIL)=(180, 45, 301, "T", "TOMADOR", "John", "Doe", "Doe", "Calle Viera y
Clavijo", "La Laguna", "Santa Cruz de Tenerife", 34510, "07/12/1978",
"01/02/1944", "Femenino", "922134507", "", "615345677", "",
"john@example.com")
```

- Entidad "*CONTRATOS_SINIESTROS*"

```
Unset

(ID_POLIZA, ID_CONTRATO, ID_SINIESTRO, FCH_OCURRENCIA, HORA_OCURRENCIA,
FCH_DECLARACION, RESPONSABILIDAD_CIVIL, ROBO, INCENDIO, LUNAS,
PERDIDA_TOTAL, DANYOS_PROPIOS, DEFENSA, LESIONADOS,
IND_INTERVIENE_AUTORIDAD, IND_ATESTADO, IND_ALCOHOL_DROGA, VIA_PUBLICA,
NUMERO, ENTIDAD, MUNICIPIO, IND_INDICIO_FRAUDE, IND_FRAUDE_CONFIRMADO,
IND_ASISTENCIA_VIAJE)=('1456', '9874', '00108 2018', '15/05/2023',
```

```
'14:30', '2023-05-16', 'S', 'N', 'N', 'S', 'N', 'S', 'S', 'N', 'S', 'S',
'N', 'Calle Principal', '10', 'Entidad A', 'Santa Cruz de Tenerife', 'N',
'N', 'S')
```

- Entidad “*CONTRATOS_SINIESTROS_LESIONADOS*”

```
Unset

(ID_SINIESTRO, ROL, VEHICULO_VIAJA, NOMBRE, APELLIDO1, APELLIDO2, EDAD,
VIA_PUBLICA, NUMERO, PISO, MUNICIPIO, PROVINCIA, COD_POSTAL, DAÑOS,
TELEFONO1, TELEFONO2, MOVIL, FAX, EMAIL)=('00108 2018', 'OCUP V/A', '3012
VHY', 'John', 'Doe', 'Doe', 35, 'Calle Falsa', '123', '2B', 'Santa Cruz de
Tenerife', 'Santa Cruz de Tenerife', '08001', 'Cervical daños',
'931234567', '934567890', '612345678', '934567891',
'jhon.doe@example.com')
```

- Entidad “*CONTRATOS_SINIESTROS_VEHICULOS_ASEGURADOS*”

```
Unset

(ID_SINIESTRO, MATRICULA, MARCA, MODELO, IND_FLOTA, TIPO, USO, DANYOS,
CTOR_NOMBRE, CTOR_APELLIDO1, CTOR_APELLIDO2, CTOR_T_SEXO, CTOR_SEXO,
CTOR_FCH_NACIMIENTO, CTOR_FCH_CARNET, CTOR_SIGLA, CTOR_VIA_PUBLICA,
CTOR_NUMERO, CTOR_ESCALERA, CTOR_PISO, CTOR_MUNICIPIO, PROVINCIA,
CTOR_ENTIDAD, CTOR_COD_POSTAL, CTOR_MOVIL, CTOR_FAX, CTOR_EMAIL)=('00108
2018', '5678DEF', 'Honda', 'Civic', 'S', 'Sedán', 'Particular', 'Graves',
'Ana', 'López', 'García', 'F', 'F', '22/03/1975', '18/07/1993', 'A',
'Calle Imaginaria', '789', '', '4B', 'Valencia', 'Valencia', 'Entidad C',
'46001', '612345678', '961234567', 'ana.lopez@example.com')
```

- Entidad “*CONTRATOS_SINIESTROS_VEHICULOS_CONTRARIOS*”

```
Unset

(ID_SINIESTRO, MATRICULA, MARCA, MODELO, COLOR, USO, TIPO, DANYOS,
CTOR_NOMBRE, CTOR_APELLIDO1, CTOR_APELLIDO2, CTOR_FCH_NACIMIENTO,
CTOR_FCH_CARNET, CTOR_CLASE_CARNET, CTOR_VIA_PUBLICA, CTOR_NUMERO,
CTOR_ESCALERA, CTOR_PISO, CTOR_COD_MUNICIPIO, CTOR_MUNICIPIO,
CTOR_ENTIDAD, CTOR_COD_POSTAL, CTOR_TELEFONO1, CTOR_TELEFONO2, CTOR_MOVIL,
```

```
CTOR_EMAIL)=('00108  2018',  '1234ABC',  'Toyota',  'Corolla',  'Rojo',
'Particular',  'Sedán',  'Moderados',  'Carlos',  'Pérez',  'Sánchez',
'1980-01-01',  '1998-05-15',  'B',  'Calle  Verdadera',  '456',  '3',  '1A',
'08002',  'Barcelona',  'Entidad  B',  '08002',  '931234567',  '934567890',
'612345678', 'carlos.perez@example.com')
```

A.3. Script de carga de la tabla *CONTRATOS*

Unset

```
// 1.1 Eliminar nodos 'Poliza' y 'Agente'
MATCH (p:Poliza) DETACH DELETE p;
MATCH (a:Agente) DETACH DELETE a;

// 1.2 Eliminar restricciones en nodos 'Poliza' y 'Agente'
DROP CONSTRAINT Poliza_idPoliza IF EXISTS;
DROP CONSTRAINT Agente_codAgente IF EXISTS;

// 1.3 Crear restricciones en nodos 'Poliza'
CREATE CONSTRAINT Poliza_idPoliza IF NOT EXISTS
FOR (p:Poliza)
REQUIRE p.idPoliza IS UNIQUE;

// 1.4 Crear nodos 'Poliza'
CALL apoc.load.xls(
    'file:///1_CONTRATOS.xlsx',
    'Hoja1',
    {
        header: true
    }
) YIELD map as row
WHERE row.ID_POLIZA IS NOT NULL
MERGE (p:Poliza {idPoliza: row.ID_POLIZA});
```

```
// 1.5 Crear restricciones en nodos 'Agente'
CREATE CONSTRAINT Agente_codAgente IF NOT EXISTS
FOR (a:Agente)
REQUIRE a.codAgente IS UNIQUE;

// 1.6 Crear nodos 'Agente'
CALL apoc.load.xls(
    'file:///1_CONTRATOS.xlsx',
    'Hoja1',
    {
        header: true
    }
) YIELD map as row
WHERE row.COD_AGENTE IS NOT NULL
MERGE (a:Agente {codAgente: row.COD_AGENTE});

// 1.7 Crear relaciones (Agente)-[:CONTRATADA_POR]->(Poliza)
CALL apoc.load.xls(
    'file:///1_CONTRATOS.xlsx',
    'Hoja1',
    {
        header: true
    }
) YIELD map as row
MATCH (p:Poliza {idPoliza: row.ID_POLIZA})
MATCH (a:Agente {codAgente: row.COD_AGENTE})
MERGE (p)-[:CONTRATADA_POR]->(a);
```

Bibliografía

[1] PWC, *Embracing Risk Intelligently* https://www.pwc.com/gx/en/services/forensics/economic-crime-survey.html

[2] Mutua Tinerfeña https://mutuatfe.es/

[3] Github Repository https://github.com/aglezg/TFG_2023-2024

[4] DB-Engines Graph Dbms Ranking https://db-engines.com/en/ranking/graph+dbms

[5] RAE “Base” Definition https://www.rae.es/drae2001/base

[6] Chris Kemper. *Beginning Neo4j*, 9781484212271. January 2016. Disponible en https://www.oreilly.com/library/view/beginning-neo4j/9781484212271/

[7] Grafos, Departamento de Ciencias de la Computación e Inteligencia Artificial , Universidad de Granada https://ccia.ugr.es/~jfv/ed1/tedi/cdrom/docs/grafos.htm

[8] Gregory Jordan. *Practical Neo4j*, 9781484200223. January 2015. Disponible en https://www.oreilly.com/library/view/practical-neo4j/9781484200223/

[9] Gradient Flow, *What is Graph Intelligence?* https://gradientflow.com/what-is-graph-intelligence/

[10] Jorge Ramirez Carrasco, *Modelado y Visualización de Grafos con Datos Masivos*. 2015

[11] O’Reilly Media Inc. *Graph Databases*, 9781491930892. June 2015. Disponible en https://www.oreilly.com/library/view/graph-databases-2nd/9781491930885/

[12] Neo4j, *Graph Data Science, Use Case Selection Guide*. https://go.neo4j.com/rs/710-RRC-335/images/Neo4j-Graph-Data-Science-Use-Case-Selection-Guide.pdf

[13] Alessandro Negro. *Graph Powered Machine Learning*, 9781617295645. September 2021. Disponible en https://www.oreilly.com/library/view/graph-powered-machine-learning/9781617295645/

[14] O’Reilly Media Inc. *Graph-Powered Analytics and Machine Learning with TigerGraph*, 9781098106652. July 2023. Disponible en https://www.oreilly.com/library/view/graph-powered-analytics-and/9781098106645/

[15] Neo4j Documentation, APOC.load.xls. https://neo4j.com/labs/apoc/4.4/overview/apoc.load/apoc.load.xls/

[16] Neo4j Operations Manual, Memory Recommendations https://neo4j.com/docs/operations-manual/current/tools/neo4j-admin/neo4j-admin-memrec/