Representación realista de activos farmacéuticos en bases de datos de grafos

Ariel Malowany *Instituto de Computación*
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay
arielmalowany@gmail.com
Mathias Minacapilli *Instituto de Computación*
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay
mathiasminacapilli@gmail.com

Resumen

El presente trabajo tiene como objetivo demostrar que los datos vinculados a la industria farmacéutica pueden ser modelados en grafos resultando en un diseño intuitivo que posibilita consultas más explicativas.

Los resultados más relevantes del proyecto están vinculados al proceso de *scrapping*, diseño de una base de datos de grafos, carga en *Neo4j*, y consultas en *Cypher*.

I. Introducción

La industria de medicamentos de consumo humano es la industria más regulada y a la vez una de las más rentables. El mercado de medicamentos implica la intervención del gobierno (a través de regulación expresada principalmente por la exigencia de información en los productos), consumidores (usuarios finales que consumen un conjunto personalizado de productos), fabricantes de medicamentos (que combinan principios activos y excipientes de acuerdo con formulaciones) y fabricantes de principios activos o drogas sintéticas.

Se considera que la unidad básica de información en este sistema es el principio activo, el cual condensa información de todo tipo: efecto en el humano (aislado, combinado con otros medicamentos), formulación farmacéutica (dosificación, vida útil, activos y excipientes con los cuales deberá ser combinado), composición química (estructural), presentaciones comerciales del activo más comunes, así como presentaciones del producto terminado más dominantes en el mercado.

El presente trabajo tiene como objetivo demostrar que los datos y metadatos del activo farmacéutico tienen naturaleza de grafos. A grandes razgos, el proyecto consistió en utilizar como origen datos reales de *DrugBank* [1] obtenidos mediante *scrapping*, instanciar los datos en un motor de grafos e identificar intuitivamente: categorías y clases de medicamentos, agrupaciones por indicaciones terapéuticas, presentaciones comerciales, clusters de referencias bibliográficas.

La investigación tuvo como pauta utilizar datos reales, por lo fue necesario realizar *scrapping* para poblar la base de datos. Esta decisión permitió comprender correctamente el modelo de negocio, reflexionar extensamente sobre los principios de diseño de la base de datos, y lograr un resultado intuitivo y funcional.

El documento se estructura en las secciones: trabajos relacionados, desarrollo, experimentación y conclusiones finales.

II. Trabajos relacionados

La base de datos de referencia fue construída por el proyecto DrugBank, que al momento de este publicación de trabajo cuenta con 5 lanzamientos. Según datos de 2018, la base de datos recibe al menos 12 millones de hits por año y 30 queries de usuario por semana [1]. La primera versión de *DrugBank*(2005) buscaba crear una única implementación de base de datos que contenga secuencia, estructura, datos funcionales de las móleculas de drogas y de sus target de acción. Su objetivo era mejorar la investigación de interacciones, comparaciones y descubrimiento de drogas [2]. Esta implementación llevó 4 años de recolección, validación e integración de datos contenidos en medios físicos. El esquema consideró como unidad básica de análisis: una molécula con más de un átomo, con una estructura química conocida y que sea declarada como una droga por al menos una fuente de datos reputable [2]. En la segunda versión de *DrugBank*(2006), para las operaciones *CRUD* (Create, Read, Update, Delete) se incorporó un sistema de administración de datos científicos llamado *DrugBank-SDMS*. De ese modo, convivían el *DrugBank - SDMS* y la base de datos *MySQL*, pero las operaciones *CRUD* primero se realizaban en el *SDMS*, que se copiaba en el servidor todas las noches [3]. En esta versión se destaca la compilación de interacciones droga - droga, y comida - droga, añadidas de manera manual bajo revisión de expertos.

En el tercer lanzamiento (2010), la base de datos se convirtió a formato *XML*, el cual se mantiene hasta la publicación de este documento [4]. Una adición importante de este lanzamiento fue incorporar 5 secciones o bloques de datos: fármaco - metabolismo, fármaco - proteico, fármaco - genómico, fármaco - económico, características estructurales computadas [4]

En relación a la cuarta versión (2013), para las categorías de drogas se creó una estructura de árbol que consiste en los niveles anidados de clasificación: grupo anatómico principal, subgrupo terapéutico, subgrupo farmacológico, subgrupo químico. Dentro del subgrupo químico se encuentra la droga [5]. También se resalta la creación de categorías: kingdom,

superclass, class, subclass, basado en las características estructurales y las similitudes en estructuras. A modo de ejemplo, esto permite encontrar todas las benzodiacepinas de manera sencilla [5].

Finalmente, la quinta versión (2018) incorporó datos de efectos en cascada de drogas en genes, proteínas, metabolitos. Como consecuencia, se agregaron interacciones droga - metabolito, droga - proteína, droga - gen. Cada una de estas relaciones se clasifica como incrementa / disminuye el efecto en el metabolito, proteína o gen, y tiene incorporadas referencias a literatura científica [1]. Además, se agregaron los datos de indicaciones de drogas, una serie de tags extraídos de la indicación escrita en párrafos de texto [1]

Como críticas a la plataforma, se destaca que los datos están en *XML*, son de tamaño elevado y requieren aplicaciones especiales de software para leerlos. En segundo lugar, el esquema de la base de datos es difícil de entender. En tercer lugar, el esquema no es consistente con *SQL* y no puede ser importado a un *DBMS* (SQL database management system). En cuarto lugar, los datos en el XML varían de droga en droga, y además hay datos duplicados [6].

Es posible implementar los datos de *DrugBank* en una base de datos relacional. Esto se realizó en una investigación que tenía como objetivo desarrollar una base de datos que permitiera investigar adecuadamente el reaprovechamiento de drogas, actividad que consiste en reutilizar drogas aprobadas para terapias diferentes de las originales, aprovechando que las etapas de desarrollo y aprobación ya se realizaron, por lo tanto se ahorran costos y tiempo [6].

El desarrollo de la nueva base de datos relacional se realizó utilizando una técnica *Map-Reduce* mediante los siguientes pasos. El primer paso consistió en conseguir el archivo *XML*, luego se dividió en archivos más pequeños. Para cada droga se creó un archivo mediante un script de *Python*, que escanea el *XML* original línea por línea y almacena los datos de la droga en un archivo separado. Luego se analizaron los archivos de los pasos anteriores para determinar entidades y relaciones (modelado conceptual). Seguidamente se realizó la normalización de la base de datos en la tercera forma normal, y se implementaron las entidades en *SQL DBMS*. El esquema es: una tabla por entidad, tablas de relaciones según la cardinalidad, índices en los atributos. El último paso consistió en insertar los datos de los archivos anteriormente particionados en las tablas usando un método de procesamiento paralelo *Map-Reduce*, luego se validaron los datos y se subió la base de datos al servidor [6].

III. Desarrollo

III-A. Obtención de datos

Para la obtención de datos, se optó por crear un proceso automatizado de descarga de los datos de la página web https://go.drugbank.com/. El mismo consistió en los siguientes pasos:

1. Curaduría de la base de datos. En primer lugar se realizó una selección de 100 drogas legales e ilegales disponibles en Uruguay. Para agregar aleatoriedad y complejidad al conjunto de datos, se consultó a un large language model [1] por 50 drogas adicionales según los prompts: 100 activos más comunes, 50 activos relacionados a enfermedades respiratorias, 50 activos incorporados en productos *OTC* (over the counter, venta libre), 20 activos relacionados a enfermedades sexuales, y 50 activos más recientes o novedosos.
2. *Scrapping* de los datos. Se realizó el *scrapping* de la página web utilizando la librería *bsf* de *Python* con el componente *BeautifulSoup*.

El proceso de obtención de los datos tuvo como resultado 5 conjuntos de datos con la siguiente estructura.

Cuadro I: Estructura del conjunto de datos

Datos	Cantidad
Drogas	300
Información de drogas	300
Usos de drogas	3150
Categorías de drogas	7694
Formas farmacéuticas de drogas	11917

El código de la obtención de los datos, como el posprocesamiento, la carga en la base de datos, y las consultas realizadas sobre la misma, se encuentra en un repositorio de Gitlab[2].

III-B. Preanálisis de los datos

Se procedió a realizar pre-análisis de los datos para entender la estructura del conjunto de datos.

Figura 1: Principales 20 áreas terapéuticas

[1] OpenAI. (2023). ChatGPT (24 Mayo) [Large language model]. https://chat.openai.com/

[2] https://gitlab.fing.edu.uy/mathias.minacapilli/bdnr2023_gr35_proyectofinal

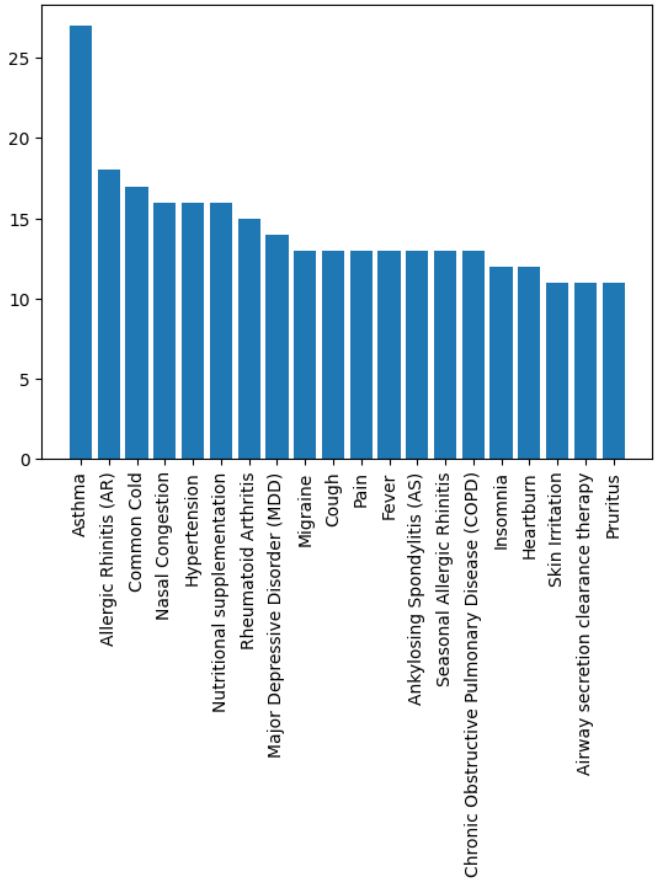


Figura 2: Principales 20 clases de activos farmacéuticos

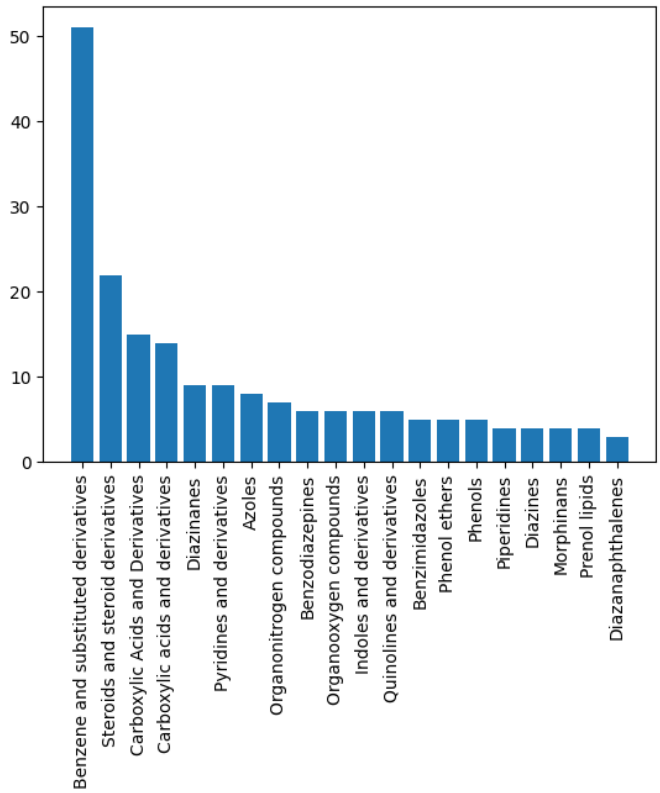
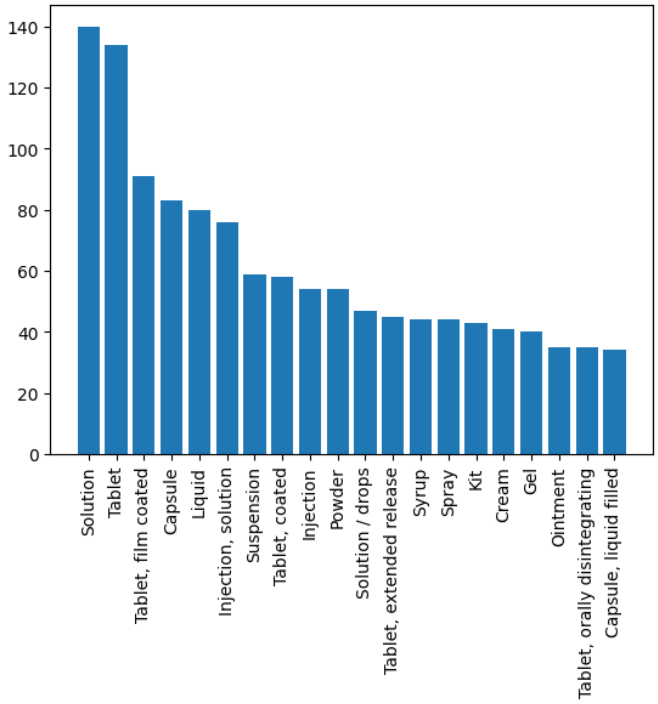


Figura 3: Principales 20 formas farmacéuticas



Como esperado, las afecciones principales son las más comunes en la población: asma, rinitis alérgica, congestión nasal, resfriado, hipertensión, etc. Se observa que con excepción de la afección asma, el resto de las principales categorías está igualmente representada en el conjunto de datos.

La clase más presente en el conjunto de datos es benceno y derivados, seguida de esteroides y derivados, ácidos carboxílicos y derivados.

Las principales formas farmacéuticas son soluciones y comprimidos. El resultado es esperable ya que la mayoría de los principios activos son solubles en agua y/o compresibles.

III-C. Preprocesamiento de los datos

El paso siguiente a obtener los datos del *scrapping* fue preparación, limpieza y conversión de datos para cargar en el motor de grafos.

El *pipeline* fue el siguiente:

Listado 1: Pipeline procesamiento datos

```
cleaned_df = (
    load_drugs
            .pipe(map_column_names)
            .pipe(remove_commas_from_list_elements)
            .pipe(set_dummy_values)
            .pipe(map_lists_to_string_separated_by_comma)
)
```

El detalle de las funciones se aclara a continuación.

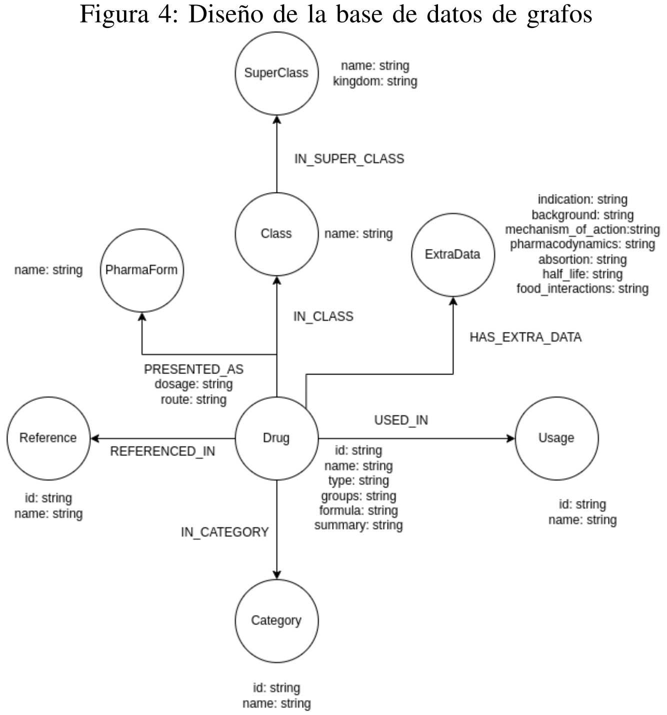
1. Función map_column_names: cambiar los nombres de las columnas para estandarizarlos.
2. Función remove_commas_from_list_elements: eliminar comas de los elementos de listas para así no tener inconsistencias en la consulta de carga de la base de datos.

3. Función set_dummy_values: asignar valores ficticios a los valores vacíos para poder cargar estos datos.
4. Función map_lists_to_string_separated_by_comma: convertir las columnas de datos que son de tipo lista a un texto donde se separa cada elemento de la lista con comas.

III-D. Diseño

El diseño fue pensado con el objetivo de representar las entidades de la realidad del negocio de los fármacos, además de permitir realizar consultas sobre los datos de forma eficiente.

Figura 4: Diseño de la base de datos de grafos



La entidad principal de nuestro modelo es *Drug*, representa el principio activo. Las drogas pueden ser: moléculas pequeñas aprobadas por la *FDA*[3], proteínas y péptidos aprobados por la *FDA* (*biotech*), nutracéuticos o micronutrientes (tales como vitaminas y metabolitos), drogas experimentales (pendientes de aprobación), drogas ilícitas y drogas discontinuadas por cuestiones de seguridad médica [2] [3].

La entidad *Drug* se relaciona con: *Category, Reference, Usage*. Como principio de diseño se decidió otorgar nombres descriptivos a las relaciones.

La entidad *Category* agrupa las drogas tanto según su composición química como por su acción terapéutica y efecto sobre receptores en el cuerpo humano. Se relaciona con *Drug* mediante *BELONGS_TO*. En el modelo original, los datos de categorías tienen una estructura de árbol que no fue posible replicar en la etapa de *scrapping*. Los datos tienen estructura de árbol que consiste en los niveles anidados de clasificación: grupo anatómico principal, subgrupo terapéutico, subgrupo farmacológico, subgrupo químico. Dentro del subgrupo químico se encuentra la entidad droga [5].

[3]Administración de Alimentos y Medicamentos de Estados Unidos

En tanto que la entidad *Reference* representa referencias de la droga artículos científicos incluidos en *DrugBank*. Se relaciona con *Drug* mediante *REFERENCED_IN*.

Por otra parte, la entidad *Usage* simboliza los usos que tienen las drogas, es decir, sobre cuáles condiciones clínicas es posible utilizar drogas. Son etiquetas curadas manualmente por el equipo de *DrugBank*, asociados a los párrafos de prescripciones médicas [1]. Se relaciona con *Drug* mediante *USED_IN*.

Adicionalmente, con el objetivo de evitar la saturación del nodo de droga se decidió crear un nuevo nodo que contenga la información extra de la droga. Es decir, se separaron los datos de relación 1 a 1 con la droga en datos consultados frecuentemente y de manera conjunta, y datos que no cumplen la condición anterior. Esta información extra se cargó en un nodo con nombre *ExtraData* y se relacionó a las drogas con la relación *HAS_EXTRA_DATA*.

Las drogas son parte de clases que describen su composición química, a su vez estas clases están contenidas en superclases y reinos. Esto se diseñó cómo un nodo Class que está relacionado con *Drug* a través de la relación *IN_SUPER_CLASS* y luego, el nodo *SuperClass* que contiene el dato del reino, relacionado con el nodo Class a través de la relación *IN_CLASS*.

Por último, para representar las formas farmacéuticas en las cuales se presentan las drogas, se creó una entidad *PharmaForm* que contiene la forma en la cual se presenta. Luego se relaciona la droga con la forma farmacéutica a través de la relación *PRESENTED_AS*, que almacena como valores de la relación tanto la dosis como la ruta de acción. En este caso el objetivo es no repetir las formas farmaceuticas, tener un único nodo por forma farmacéutica, y conceptualizar los datos de dosis y ruta de acción como metadatos de la forma farmacéutica.

Listado 2: Consulta para crear índice por nombre de droga

```
CREATE INDEX drug_name_index FOR (d:Drug) ON (d.name)
```

Debido a que la mayoría de consultas son centradas en una droga en específico, y en particular, en su nombre, se decidió también como parte del diseño crear un índice por el nombre de la droga cómo se muestra en la consulta previa.

Figura 5: Explicación de consulta en el motor previo agregado índice

```
Planner COST

Runtime PIPELINED

Runtime version 5.9

Batch size 128

+-----------------+----+------------------------+----------------+---------------------+
| Operator        | Id | Details                | Estimated Rows | Pipeline            |
+-----------------+----+------------------------+----------------+---------------------+
| +ProduceResults |  0 | d, r, c                |            100 |                     |
| |               +----+------------------------+----------------+                     |
| +Limit          |  1 | 100                    |            100 |                     |
| |               +----+------------------------+----------------+                     |
| +Filter         |  2 | c:Category             |            100 |                     |
| |               +----+------------------------+----------------+                     |
| +Expand(All)    |  3 | (d)-[r]-(c)            |            375 |                     |
| |               +----+------------------------+----------------+                     |
| +Filter         |  4 | d.name = $autostring_0 |              4 |                     |
| |               +----+------------------------+----------------+                     |
| +NodeByLabelScan|  5 | d:Drug                 |             78 | Fused in Pipeline 0 |
+-----------------+----+------------------------+----------------+---------------------+

Total database accesses: ?
```

Figura 6: Explicación de consulta en el motor luego de agregado índice

```
Planner COST

Runtime PIPELINED

Runtime version 5.9

Batch size 128

+-----------------+----+------------------------------------------------+----------------+---------------------+
| Operator        | Id | Details                                        | Estimated Rows | Pipeline            |
+-----------------+----+------------------------------------------------+----------------+---------------------+
| +ProduceResults |  0 | d, r, c                                        |             26 |                     |
| |               +----+------------------------------------------------+----------------+                     |
| +Limit          |  1 | 100                                            |             26 |                     |
| |               +----+------------------------------------------------+----------------+                     |
| +Filter         |  2 | c:Category                                     |             26 |                     |
| |               +----+------------------------------------------------+----------------+                     |
| +Expand(All)    |  3 | (d)-[r]-(c)                                    |             96 |                     |
| |               +----+------------------------------------------------+----------------+                     |
| +NodeIndexSeek  |  4 | RANGE INDEX d:Drug(name) WHERE name = $autostring_0 |          1 | Fused in Pipeline 0 |
+-----------------+----+------------------------------------------------+----------------+---------------------+

Total database accesses: ?
```

Cómo se puede ver en las imágenes anteriores, el agregar el índice por nombre de droga optimiza la búsqueda reduciendo el costo de consultar por nombre de droga, ya que el primer paso de la consulta es filtrar las drogas y en el caso del índice es costo uno. Esto mejorando la experiencia de las consultas, la carga sobre la base de datos, y los tiempos de respuesta.

III-E. *Carga en la base de datos de grafos.*

Para la carga de datos en la base de datos se optó por realizar una consulta, a partir de un *dataframe* de la librería *pandas* de Python.

Se cargaron los datos en una instancia de *Neo4j Auradb* creando la instancia en la plataforma web de *Neo4j*.

Se dividió la consulta de carga de datos en múltiples consultas para no sobrepasar la memoria dada de la instancia y así poder finalizar exitosamente el proceso de carga. Por ejemplo, se implementó una consulta para cargar las drogas y datos de drogas, y otra consulta para cargar las referencias bibliográficas de las drogas.

Se utilizó la librería *py2neo* para conectarse al grafo y ejecutar las consultas.

Luego, se ejecutó el proceso de carga de a lotes, también con el fin de no saturar la instancia, y excedernos del uso de memoria a la hora de ejecutar. Así fue posible cargar todos los datos satisfactoriamente.

Las consultas para la carga de datos se ejecutaron satisfactoriamente en los siguientes tiempos:

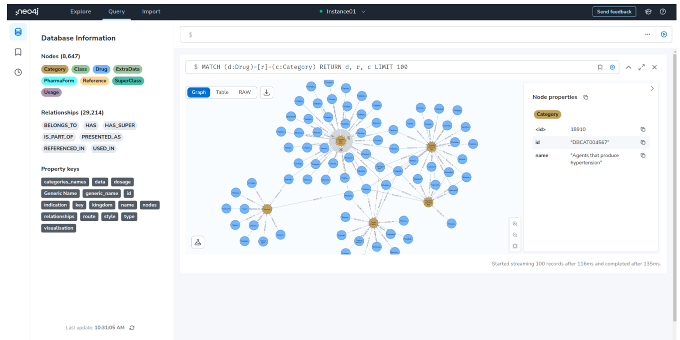
Cuadro II: Tiempo de carga de consultas

Consulta	Tiempo
Consulta principal de drogas	13 segundos
Usos de drogas	6 segundos
Referencias a artículos científicos	16 segundos
Formas farmaceuticas	56 segundos

III-F. *Resultado de la carga*

Se utilizó la plataforma web de *Neo4j Auradb* para visualizar los datos cargados.

Figura 7: Visualización de la base de datos de grafos cargada



Cuadro III: Cantidad de nodos por entidad

Entidad	Cantidad
Category	1303
Class	71
Drug	300
ExtraData	300
PharmaForm	374
Reference	4637
SuperClass	14
Usage	1648

Cuadro IV: Cantidad de relaciones

Relación	Cantidad
HAS_EXTRA_DATA	300
IN_CATEGORY	7700
IN_CLASS	300
IN_SUPER_CLASS	300
PRESENTED_AS	11953
REFERENCED_IN	5486
USED_IN	3173

El resultado de la carga generó un total de 8647 nodos y 29212 relaciones.

IV. Experimentación

En este apartado se presentan las consultas más relevantes que se escribieron en el lenguaje *Cypher* para consultar la instancia de *Neo4j*. Se tuvo en consideración tres usuarios: desarrollador de medicamentos, médico recomendador de medicamentos, redactor de información legal de medicamentos.

IV-A. *Obtener medicamentos similares para un médico*

Listado 3: Consulta para obtener medicamentos con indicaciones similares

```
MATCH (d1) - [:USED_IN] -> (t) <- [:USED_IN] - (d2)
WHERE t.name <> ""
AND d1 <> d2
WITH d1.name AS droga1,
d2.name AS droga2,
count(t) AS uso_compartido,
collect(t.name) AS indicaciones
WITH droga1,
[droga2, indicaciones] AS droga2 ORDER BY
uso_compartido DESC
RETURN droga1,
```

```
head(collect(droga2))[0] AS droga2,
head(collect(droga2))[1] AS indicaciones_comunes
ORDER BY droga1 ASC
"""
```

Se considera que la implementación en *Neo4j* logra hacer más intuitiva esta consulta. El resultado es una tabla con una droga en la primera columna, la droga más similar en cuanto a indicaciones médicas compartidas en la segunda columna, y las indicaciones en común en la tercera columna. Se observa que si la droga tiene un único substituto entonces la dupla de drogas aparecerá 2 veces en el listado pero en columnas cambiadas.

IV-B. Literatura relevante para un escritor de información legal de medicamentos

Listado 4: Consulta para obtener literatura relevante para investigar

```
MATCH (r) <- [:REFERENCED_IN] - (d) - [:IN_CLASS] -> (c)
WHERE c.name <> "Empty Class" AND r.name <> ""
WITH c.name AS clase,
r.name AS referencia,
count(c.name) AS freq_referencia
WITH clase,
freq_referencia,
referencia AS ref ORDER BY freq_referencia DESC
RETURN clase, collect(ref) AS referencias
"""
```

La intención de la consulta es obtener para cada clase de drogas la literatura relacionada, ordenada por más frecuente a menos frecuente en la clase. Se define clase como grupo de drogas con similares características estructurales [5].

Listado 5: Indicaciones de una droga

```
MATCH (d1) - [:USED_IN] -> (u) <- [:USED_IN] - (d2)
WHERE (u.name <> "" AND u.name IS NOT NULL AND d1 <> d2)
WITH d1.name AS droga,
u.name AS uso,
count(d2.name) AS freq_usos
WITH droga,
freq_usos,
uso AS usos_frec ORDER BY freq_usos DESC
RETURN droga, collect(usos_frec) AS usos
"""
```

En esta consulta se listan los tags de indicaciones de una droga, siendo posible agregar una cláusula en el paso *WHERE* que permita filtrar una droga en particular.

IV-C. Información de la droga para desarrollador de medicamentos

Listado 6: Consulta para obtener la forma farmacéutica más frecuente dentro de cada clase

```
MATCH (c) <- [:IN_CLASS] - (d1) - [r:PRESENTED_AS] -> (f)
<- [r2:PRESENTED_AS] - (d2) - [:IN_CLASS] -> (c)
WHERE (
f.name <> ""
AND f.name IS NOT NULL
AND c.name <> "Empty Class"
AND d1 <> d2
)
WITH c.name AS clase,
f.name AS forma,
count(DISTINCT d2.name) AS freq_formas
WITH clase,
freq_formas,
forma AS forma_freq ORDER BY freq_formas DESC
RETURN clase, collect([forma_freq, freq_formas]) AS formas
"""
```

La intención de la consulta es obtener las formas farmacéuticas presentes en la clase, y la cantidad de drogas que tienen esa presentación, ordenada por forma más frecuente a menos frecuente en la clase. Se define clase como grupo de drogas con similares características estructurales [5]. Esta consulta permite orientar la producción de una planta de medicamentos en función de la maquinaria presente y las características del mercado. Por ejemplo, una planta que solo tenga máquinas para fabricar líquidos deberá posicionarse en la formas farmacéuticas suspensiones, spray, soluciones y buscar las clases que hagan más diverso el portfolio de productos de la empresa. Por el contrario, una empresa que tenga pocos activos farmacéuticos pero muchas máquinas deberá diversificar las formas farmacéuticas que adoptan sus activos.

V. Conclusiones y trabajo futuro

El proyecto tuvo como objetivo demostrar que el modelado en grafos de la entidad drogas junto con sus datos y metadatos era factible y resultaba más intuitivo que un modelo relacional.

La investigación tuvo como pauta utilizar datos reales, por lo fue necesario realizar *scrapping* para poblar la base de datos. Esta decisión permitió comprender correctamente el modelo de negocio, reflexionar extensamente sobre los principios de diseño de la base de datos, y lograr un resultado intuitivo y funcional.

El proceso de carga en *Neo4j* fue satisfactorio y consistió en las siguientes etapas. En primer lugar se creó una instancia de *Neo4j Auradb* en la plataforma web de *Neo4j*. Se dividió la consulta de carga de datos en múltiples consultas para no sobrepasar la memoria dada de la instancia y así poder finalizar exitosamente el proceso de carga. Además, se ejecutaron las consultas de a lotes, y así fue posible cargar todos los datos sin problemas.

Si bien el objetivo del proyecto se cumplió de manera satisfactoria, se considera que es necesaria una demostración más formal de la eficiencia de la implementación en grafos respecto a una instancia en una base de datos relacional. Dicho análisis consistiría en comparar la eficiencia de las consultas en ambas bases, así como en ejecutar algoritmos de análisis de datos sobre ambas implementaciones.

Por motivos de cumplimiento de los términos de uso no se pudo utilizar la base de datos completa de *DrugBank* en *XML*. Este hecho sumado con la dificultad de extraer en la etapa de *scrapping* las interacciones droga - droga no permitió incorporar a la base de datos la dimensión de interacciones. La implementación de estas interacciones en la base de datos de grafos también es un proyecto interesante a realizar en el futuro, pues se pueden explotar las funcionalidades de caminos y utilizar algoritmos para identificar nodos relevantes. Debe tenerse en cuenta que los caminos pueden incluir drogas y targets de acción, transporte (mueve drogas dentro y alrededor de la célula), *carrier* (mueve drogas en el cuerpo), enzimas pro - conversión de drogas [4].

Referencias

[1] D. S. Wishart, Y. D. Feunang, A. C. Guo et al., "DrugBank 5.0: a major update to the DrugBank database for 2018," *Nucleic acids research*, vol. 46, n.º D1, págs. D1074-D1082, 2018.

[2] D. S. Wishart, C. Knox, A. C. Guo et al., "DrugBank: a comprehensive resource for in silico drug discovery and exploration," *Nucleic acids research*, vol. 34, n.º suppl_1, págs. D668-D672, 2006.

[3] D. S. Wishart, C. Knox, A. C. Guo et al., "DrugBank: a knowledgebase for drugs, drug actions and drug targets," *Nucleic acids research*, vol. 36, n.º suppl_1, págs. D901-D906, 2008.

[4] C. Knox, V. Law, T. Jewison et al., "DrugBank 3.0: a comprehensive resource for 'omics' research on drugs," *Nucleic acids research*, vol. 39, n.º suppl_1, págs. D1035-D1041, 2010.

[5] V. Law, C. Knox, Y. Djoumbou et al., "DrugBank 4.0: shedding new light on drug metabolism," *Nucleic acids research*, vol. 42, n.º D1, págs. D1091-D1097, 2014.

[6] Y. Masoudi-Sobhanzadeh, Y. Omidi, M. Amanlou y A. Masoudi-Nejad, "DrugR+: a comprehensive relational database for drug repurposing, combination therapy, and replacement therapy," *Computers in biology and medicine*, vol. 109, págs. 254-262, 2019.