

Desarrollo dirigido por modelos de políticas de seguridad en bases de datos orientadas a grafos

Carlos Blanco^{1,2}, David G. Rosado², and Eduardo Fernández-Medina²

¹ Grupo ISTR. Departamento de Ingeniería Informática y Electrónica.
Universidad de Cantabria. España
`carlos.blanco@unican.es`

² Grupo GSyA. Instituto de Tecnologías y Sistemas de Información.
Departamento de Tecnologías y Sistemas de Información.
Universidad de Castilla-La Mancha. España
`David.GRosado@uclm.es`, `Eduardo.Fdezmedina@uclm.es`

Abstract. La importancia de la seguridad y privacidad de los datos está aumentando actualmente en todo el mundo debido al enorme número de transacciones de información que se producen continuamente. Cantidades cada vez mayores de datos, incluyendo información sensible y personal, se cargan en NoSQL y otras tecnologías de Big Data para su análisis y procesamiento. Sin embargo, los enfoques de seguridad y privacidad actuales no tienen en cuenta las características especiales de estas tecnologías, dejando los datos sensibles a nivel de negocio y los datos personales sin protección, arriesgándose así a sufrir graves pérdidas monetarias o daños a la marca, así como incurrir en el incumplimiento de normativas de protección de datos. En este artículo, nos centramos en garantizar la seguridad de las bases de datos NoSQL orientadas a grafos siguiendo paradigmas como la "seguridad por diseño" y la "ingeniería dirigida por modelos". Para ello, proponemos un meta-modelo que permite al diseñador modelar las políticas de seguridad sobre las estructuras específicas de las bases de datos orientadas a grafos y una serie de transformaciones que permiten obtener de forma automatizada su correspondiente implementación en un gestor de bases de datos orientadas a grafos concreto, Neo4J. Por último, aplicamos el marco propuesto a un caso de estudio del ámbito sanitario.

Keywords: Modelado conceptual · Big Data · Bases de datos orientadas a grafos · Seguridad.

1 Introducción

Los científicos, médicos, empresas, organismos públicos y el público en general transfieren habitualmente grandes cantidades de datos, pero las herramientas y tecnologías existentes aún no son capaces de manejar la escala, la velocidad, la variedad y la complejidad de estos conjuntos de datos masivos [21]. Tampoco son capaces de incorporar medidas de seguridad o privacidad adecuadas [8, 17], debiéndose, principalmente, a la falta de formación suficiente y de conocimientos

fundamentales sobre cómo garantizar la seguridad y la privacidad de los datos a gran escala [12, 14].

Una de las tecnologías de bases de datos actuales es la de los almacenes de datos NoSQL o datastores, que se utilizan actualmente en el back-end de gestión de datos de varias aplicaciones TIC como el Big Data debido a sus altos niveles de escalabilidad y rendimiento [19]. Sin embargo, a pesar de sus innegables ventajas, presentan ciertas vulnerabilidades, y existen algunos obstáculos (granularidad fina, protección de datos, cifrado, control de acceso, etc.) para su adopción generalizada, sobre todo en materia de seguridad [3, 20]. De hecho, cuando se diseñaron inicialmente las bases de datos NoSQL, la seguridad no se consideraba una característica importante [16], por lo que la seguridad se proporciona a nivel de middleware y sistema operativo [10].

En la última década, los investigadores y los profesionales han llegado a reconocer que las características de seguridad deben incorporarse desde las primeras fases del desarrollo, mediante un enfoque estructurado y sistemático, combinando principios tanto de la ingeniería del software como de la seguridad [8, 13], en lo que se conoce como seguridad por diseño (security-by-design). El enfoque del desarrollo dirigido por modelos se ha aplicado con éxito durante el desarrollo de sistemas seguros, lo que permite integrar los aspectos de seguridad en todos los niveles de modelado y transformarlos automáticamente [4]. Estas ideas también deberían utilizarse en las bases de datos NoSQL seguras y en los sistemas que hacen uso de esta tecnología NoSQL, como el Big Data, en los que los esfuerzos de seguridad se están centrando en el nivel de implementación, cuando una identificación temprana de los requisitos de seguridad generaría soluciones más robustas y de mayor calidad [7].

En este trabajo nos centramos en las bases de datos orientadas a grafos, las cuales han cobrado un mayor interés en los últimos años, debido a sus aplicaciones en áreas como la web semántica y el análisis de redes sociales. Estas bases de datos proporcionan una solución eficaz y eficiente para el almacenamiento y la consulta de datos en estos escenarios, en los que los datos son ricos en relaciones y permiten la aplicación de numerosos algoritmos de recorrido, útiles para encontrar patrones [6]. Los principales componentes de una base de datos orientadas a grafos son los nodos y las aristas, que representan las relaciones y suelen ser dirigidas (una arista tiene un nodo inicial y otro final). Los nodos y las aristas tienen etiquetas y pueden tener propiedades. Las bases de datos orientadas a grafos no tienen esquema, que aunque proporciona más flexibilidad [18], tiene inconvenientes ya que, sin un esquema, es difícil que el diseñador tenga una visión completa del sistema y pueda definir correctamente las restricciones de seguridad necesarias.

En este trabajo presentamos una propuesta para el diseño e implementación de políticas de seguridad en bases de datos orientadas a grafos, siguiendo lo que se conoce como "Seguridad por diseño" para incorporar la seguridad en las primeras fases de su desarrollo, y "desarrollo dirigido por modelos" para automatizar su implementación final. Más concretamente, las principales aportaciones de este trabajo son las siguientes:

- La definición de un metamodelo que permite definir las políticas de seguridad necesarias sobre estructuras específicas de bases de datos orientadas a grafos.
- El planteamiento de las transformaciones necesarias para procesar de forma automatizada las políticas de seguridad modeladas a nivel de diseño y generar su correspondiente implementación en un gestor concreto, en este caso, Neo4J.
- La presentación de un caso en el ámbito sanitario donde se muestra la aplicabilidad de nuestra propuesta, modelando las políticas de seguridad a nivel de diseño y obteniendo su correspondiente implementación en Neo4J.

El resto del artículo está estructurado de la siguiente forma. En la Sección 2 se discuten los trabajos relacionados. La Sección 3 presenta la propuesta junto al metamodelo para representar políticas de seguridad en bases de datos orientadas a grafos, un análisis de las capacidades para implementar mecanismos de seguridad que ofrecen las herramientas de destino y se definen las transformaciones de las reglas de seguridad a su implementación en Neo4J. La Sección 4 muestra un escenario de aplicación de la propuesta en el ámbito sanitario. Por último, en la Sección 5 se presentan las conclusiones y trabajo futuro.

2 Trabajos relacionados

El objetivo de la seguridad en ingeniería del software es abordar las vulnerabilidades de seguridad del software teniendo en cuenta las preocupaciones de seguridad y los enfoques de desarrollo en el ciclo de vida del desarrollo de software. Una de las primeras fases del desarrollo es la ingeniería de requisitos donde existen trabajos con propuestas de nuevos métodos y procesos para identificar requisitos de seguridad [5].

Otra importante iniciativa en el desarrollo seguro fue la Seguridad Dirigida por Modelos (MDS) que apoya el desarrollo de aplicaciones críticas para la seguridad como pueden ser UMLsec o SecureUML, por destacar algunas. Además, surgieron propuestas de métodos o procesos de software seguro en la que se incorporan actividades o tareas de seguridad en el ciclo de desarrollo [9], como pueden ser Microsoft SDL o CLASP.

Aunque varios autores han señalado la necesidad de modelar propuestas para el diseño y la gestión de bases de datos NoSQL [19], son pocos los trabajos que han abordado este reto, especialmente desde un punto de vista independiente de la tecnología, y se limitan a encontrar soluciones a problemas concretos en escenarios limitados, orientándolos a documentales [15] o columnares [22]. Otros trabajos se centran en técnicas criptográficas [2], y de modelado a un nivel de abstracción superior, con conceptos de Big Data que no son específicos de ninguna tecnología NoSQL [11]. En cuanto a Big Data, existen varias soluciones de integración interesantes que aplican ontologías [1], pero están limitadas en su aplicación (en este caso, a fuentes implementadas en MongoDB) y no soportan la integración de políticas de seguridad.

Por último, los aspectos de seguridad también deben considerarse en las últimas capas de implementación y control de la solución de Big Data. Sin em-

bargo, se han detectado muchas deficiencias en cuanto a la implementación de elementos de seguridad en las bases de datos NoSQL [2] o en sistemas ampliamente utilizados como Hadoop [23]. Las herramientas actuales proporcionan unos mecanismos de seguridad suficientes para la mayoría de los casos, basados en políticas RBAC y la definición de privilegios a nivel de rol. Sin embargo, aunque avanzan en aspectos como las políticas de seguridad de grano fino, aún queda margen de mejora en la inclusión de características de seguridad para sistemas más exigentes. Un aspecto a mejorar sería la definición de reglas de seguridad que incluyan condiciones con valores que han de ser evaluados en tiempo de ejecución (mostrar pacientes mayores de 18 años), que por el momento no está soportado por las herramientas y debe ser suplido con herramientas que intermedien entre usuarios y base de datos, interceptando sus consultas y añadiendo las cláusulas *where* necesarias para representar las condiciones de la regla. Otro aspecto a mejorar vendría dado por las limitaciones del uso de RBAC frente a un control de acceso más flexible como el basado en atributos (ABAC), que ofrecería la posibilidad de definir reglas basadas en atributos de usuarios u objetos. Por ejemplo, si un usuario de un rol "empleado" dispone de información sobre su centro de trabajo, podríamos limitar sus privilegios a la información relacionada con su centro de trabajo. Es una casuística muy común que actualmente ha de ser resuelta creando un rol por cada centro de trabajo, algo que se complica si pensamos en otros ejemplos en los que no es factible definir una gran cantidad de roles, como que los privilegios dependan de la ubicación del usuario, de su identificación personal, etc.

En conclusión, las propuestas actuales de modelado y desarrollo de bases de datos NoSQL no consideran adecuadamente la seguridad en todas las etapas. Las soluciones centradas en la etapa de implementación proporcionan aspectos o mecanismos de seguridad aislados, mientras que las soluciones centradas en etapas anteriores del desarrollo proporcionan soluciones de modelado e integración, pero consideran únicamente aspectos estructurales.

3 Propuesta para el modelado de políticas de seguridad en bases de datos orientadas a grafos

Nuestra propuesta se centra en la incorporación de políticas de seguridad en bases de datos orientadas a grafos desde etapas tempranas del proceso de desarrollo.

En este sentido, propone un metamodelo que facilita al diseñador la tarea de especificación de políticas de seguridad, debido a que, utilizando la sintaxis de diagramas de clases UML y sin conocer aspectos técnicos de la implementación final, le permite establecer a nivel de diseño las restricciones de seguridad sobre las propias estructuras de la base de datos orientadas a grafos.

Una vez definido el metamodelo de partida, se analiza el destino. Es decir, las capacidades de implementación de aspectos de seguridad que ofrecen los gestores de bases de datos orientados a grafos (centrándonos en este trabajo en Neo4J). Este análisis nos permite establecer estrategias adecuadas para la

correcta implementación de las restricciones de seguridad definidas a nivel de diseño. Posteriormente, se plantean las transformaciones necesarias para (aplicando un enfoque de desarrollo dirigido por modelos) automatizar el proceso de generación de código a partir del modelo de diseño.

3.1 Modelado de políticas de seguridad de bases de datos orientadas a grafos

A nivel de diseño, se propone un metamodelo que permite a los diseñadores establecer las políticas de seguridad de una forma más clara y sencilla, a la vez que le abstrae de conceptos de más bajo nivel relacionados con tecnologías y sintaxis concretas de los gestores de bases de datos.

La Figura 1 muestra el metamodelo, el cuál está enfocado a bases de datos orientadas a grafos, representando sus aspectos estructurales y permitiendo que las políticas de seguridad del sistema se definan relacionándolas con dichas estructuras.

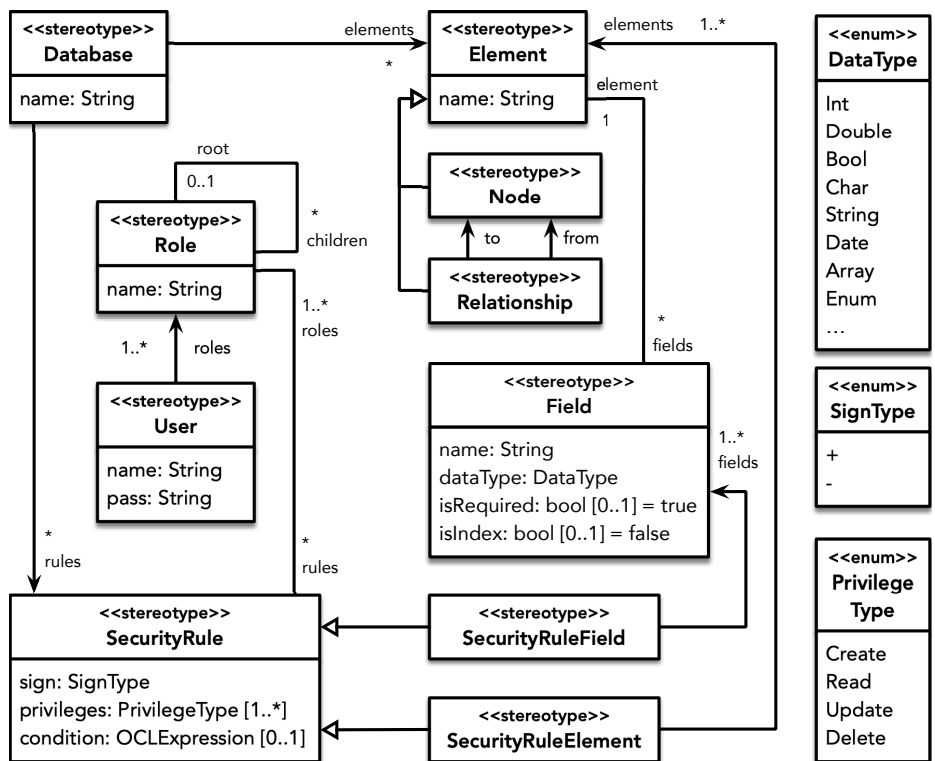


Fig. 1. Metamodelo para bases de datos orientadas a grafos.

Dentro de la parte estructural podemos encontrar los elementos (Element) de la base de datos, que pueden ser nodos (Node) o relaciones entre ellos (Relationship). Tanto nodos como relaciones pueden especificarse en más detalle mediante campos (Field) con tipos de datos básicos (DataType).

La parte relativa a la seguridad considera un sistema de control de acceso basado en roles (RBAC), al ser el sistema utilizado en prácticamente la totalidad de las herramientas gestoras de bases de datos. Esto nos permite clasificar a los usuarios del sistema (User) en base a roles (Role) que a su vez pueden mantener relaciones de herencia creando una jerarquía.

La definición de la política de seguridad se representa mediante reglas de seguridad (Security Rule) que, asociadas a determinados roles, les otorgan o retiran (SignType) privilegios (PrivilegeType) sobre ciertos elementos estructurales (nodos, relaciones o campos). Dependiendo de las estructuras a las que involucran, diferenciamos entre reglas sobre elementos del tipo nodo o relación (SecurityRuleElement) y reglas de grano fino sobre campos de nodos o relaciones (SecurityRuleField). Ejemplos de estas reglas podrían ser la retirada del permiso de lectura para cierto nodo, u otorgar el permiso de modificación sobre determinado campo de una relación.

3.2 Implementación de políticas de seguridad en bases de datos orientadas a grafos

En esta sección se describen los mecanismos para la implementación de medidas de seguridad que ofrecen los gestores de bases de datos orientadas a grafos, centrándonos en la herramienta Neo4J como referente en este tipo de sistemas.

Estas herramientas suelen utilizar un sistema de control de acceso basado en roles (RBAC) y un conjunto de autorizaciones positivas (para otorgar) o negativas (para denegar) a ciertos roles privilegios (de creación, lectura, modificación o eliminación) sobre ciertos elementos estructurales. Los sistemas suelen funcionar como un mundo cerrado en el que los roles en principio no tienen ningún privilegio a no ser que se existan explícitamente autorizaciones positivas que se los otorguen. El uso conjunto de autorizaciones positivas y negativas dan más flexibilidad y facilidad a la hora de establecer las autorizaciones, ya que, por ejemplo, si quisiéramos denegar el acceso a un determinado nodo, podríamos establecer una autorización positiva sobre todos los nodos y una negativa sobre el nodo en cuestión. La otra alternativa sería realizarlo con una autorización positiva en la que incluiríamos uno a uno todos los elementos salvo el nodo en cuestión. Sin embargo, hay que tener en cuenta que el uso de autorizaciones positivas y negativas generan conflictos cuando se establecen sobre los mismos elementos, lo cual los sistemas suelen resolver favoreciendo la estrategia de mundo cerrado, de forma que sólo se concede el permiso al rol si tiene una autorización positiva y no una negativa sobre dicho elemento.

La Tabla 1 muestra la sintaxis de reglas de autorización para Neo4J, en la que puede observarse cómo se incluye información relativa al signo de la autorización, privilegios, elementos y roles a los que afecta. En el caso concreto de Neo4J, debemos comentar que los privilegios clásicos (creación, lectura, modificación y

eliminación) están especializados en más tipos de permisos y a la vez, agrupados en forma jerárquica. Por un lado, los permisos de lectura se refieren al acceso del valor (Read) o a poder alcanzar el elemento (Traverse), teniendo un privilegio que agrupa a los dos anteriores (Match). En cuanto a la escritura, existe un privilegio que agrupa a los demás (Write) que se especializa a su vez en la creación (Create), borrado (Delete), modificación (Set Property) y gestión de etiquetas (set y delete label).

Por otro lado, los elementos estructurales también se categorizan en jerarquías, de modo que podemos referirnos a nodos (Node), relaciones (Relationship) o a ambos, siendo elementos (Element). En este sentido, si deseamos establecer una autorización de grano fino sobre determinadas propiedades, deberíamos indicar tanto el nombre de dichas propiedades como el de los elementos a las que pertenecen.

<regla> ::= <signo><privilegio>[<propiedades>] ON <grafo><entidad>TO <lista>
<signo> ::= GRANT DENY
<privilegio> ::= MATCH TRAVERSE READ WRITE CREATE DELETE SET PROPERTY SET LABEL DELETE LABEL
<propiedades> ::= “{“ * <lista> “}”
<grafo> ::= DEFAULT GRAPH (GRAPH[S] (* <lista>))
<entidad> ::= <tipoEntidad> (* <lista>)
<tipoEntidad> ::= ELEMENT[S] NODE[S] RELATIONSHIP[S]
<lista> ::= <nombre>[“,” <nombre>]*

Table 1. Sintaxis de reglas de autorización

Una vez presentados los mecanismos que aportan las herramientas gestoras de bases de datos orientadas a grafos en cuanto al establecimiento de políticas de seguridad, pasamos a analizar sus limitaciones. Como hemos visto anteriormente, estas herramientas nos permiten establecer un sistema de control de acceso basado en roles y una serie de autorizaciones positivas y negativas sobre nodos y relaciones, e incluso de grano fino sobre sus atributos. Los usuarios del sistema pertenecen a uno o varios roles, de los que heredan su configuración de seguridad, pero no permiten refinarlos más estableciendo privilegios de seguridad para usuarios específicos. Lo que sí permiten herramientas como Neo4J, es la modificación de la configuración de seguridad una vez está el sistema en marcha (sin necesidad de volver a arrancarlo), de forma que si se añade o elimina un privilegio de seguridad a un rol, esta restricción es considerada desde ese momento para todos los usuarios pertenecientes a dicho rol.

Por otro lado, estas autorizaciones actúan como un todo o nada, afectando a todas las instancias del elemento (nodo, relación o propiedad) independientemente de su valor, lo cuál es suficiente para la mayoría de los casos. Por ejemplo, en un sistema médico nos permitiría ocultar a un rol doctor las direcciones de todos los pacientes. Sin embargo, en ocasiones interesa refinar aún más una regla

de seguridad de grano fino para que sólo actúe ante ciertos valores de la instancia, siguiendo con el ejemplo anterior, para ocultarle las direcciones de sólo aquellos pacientes que sean menores de edad. Este tipo de autorizaciones no podrían implementarse directamente en la herramienta, sino que deberíamos disponer de un gestor de consultas que actuara como intermediario entre el usuario y la base de datos, y modificara las consultas para añadirles las condiciones (clausulas where) necesarias. Del mismo modo, no permiten establecer reglas de seguridad a usuarios concretos, sino que han de definirse a nivel de rol afectando por igual a todos los usuarios de dicho rol.

Además, el hecho de que el sistema de control de acceso esté basado en roles frente a otras opciones como el basado en atributos (ABAC), limita que puedan establecerse reglas de seguridad más ricas que utilicen información asociada a las propias instancias de los usuarios u objetos del sistema. Siguiendo con el ejemplo de un sistema médico, sería interesante que usuarios de la base de datos pudieran tener asociadas propiedades. En ese caso, un usuario de un rol paciente podría tener información sobre su identificador de paciente y establecer en el sistema autorizaciones positivas para que cada paciente pueda acceder a su información pero no al de resto de pacientes. Del mismo modo, un usuario de un rol doctor podría mantener información sobre su especialidad, centro de trabajo, etc. y establecer autorizaciones que limiten sus funciones a dichos entornos.

3.3 Generación de la implementación de políticas de seguridad

Una vez definida una propuesta para el modelado de políticas de seguridad en bases de datos orientadas a grafos y habiendo analizado los mecanismos para la implementación de seguridad que ofrecen los gestores de este tipo de base de datos, este apartado establece las correlaciones necesarias para generar de forma automatizada la implementación de las políticas de seguridad. Para ello se toma a Neo4J como herramienta de destino al ser referente en este tipo de bases de datos.

En primer lugar, generamos la jerarquía de roles y usuarios del sistema. Para ello, tomamos cada rol del modelo y creamos su rol en la implementación (`CREATE ROLE r.name`). Con los usuarios procedemos igual (`CREATE USER u.name SET PASSWORD u.pass`) y finalmente asociamos cada usuario con los roles a los que pertenece (`GRANT ROLE u.roles TO u.name`). Las posibles herencias entre roles representadas en el modelo las trataríamos a la hora de propagar las reglas de seguridad a todos los roles hijos del rol afectado por la regla.

Una vez disponemos de la configuración de roles del sistema, procedemos a generar y asociarles reglas de seguridad que limitan sus privilegios según lo establecido en el modelo. El proceso consiste en ir analizando las reglas de seguridad especificadas en el modelo y por cada una de ellas generar una o varias autorizaciones. Cada una de estas autorizaciones afectan a uno o varios roles, concediéndoles o denegándoles un privilegio sobre uno o varios elementos (siguiendo la sintaxis de implementación presentada en la Tabla 1).

Por cada regla de seguridad que afecta a nodos o relaciones (Security Rule Element) se procesan los siguientes elementos. El signo, para generar autorizaciones positivas (GRANT) o negativas (DENY). Los privilegios que concede o deniega (Create, Read, Update o Delete). Los elementos (nodos, relaciones o propiedades) a los que afecta y los roles a los que se les asigna dicha regla.

Los privilegios de creación y borrado se trasladan como tal a la implementación. Sin embargo, para el permiso de lectura, se opta por implementarlo con un privilegio MATCH, al incluir la lectura de valores (READ) y el poder alcanzar instancias (TRAVERSE). En cuanto al privilegio de modificación, se opta por la implementación como modificación de propiedades (SET PROPERTY *) y no de etiquetas.

Existen varios casos en los que una regla de seguridad del modelo deberá generar varias reglas a nivel de implementación. Esto sucede cuando una misma regla se refiere a la vez a varios tipos de privilegios (creación, lectura, modificación y borrado) o afecta a varios tipos de elementos (nodos y relaciones). En este caso se observa como trabajar con el modelo ofrece al diseñador una reducción de complejidad frente a la implementación.

Por otro lado, las reglas de grano fino definidas sobre propiedades (Security Rule Field) mantienen la misma estructura, pero adicionalmente se ha de extraer las propiedades a las que afecta así como el elemento al que pertenecen. También en este tipo de reglas, se opta por traducir el privilegio de lectura por un READ que afecte a los valores de dichas propiedades, frente al uso de MATCH que considerábamos con las reglas que afectan a nodos o relaciones.

Por último, si cualquier tipo de regla de seguridad tiene indicada una condición (ej. mostrar sólo empleados con más de 5 años de experiencia), ésta no podría implementarse de forma directa en el gestor de bases de datos, ya que las autorizaciones que nos permite definir afectan a todas las instancias de los elementos involucrados (ej. o mostramos u ocultamos todos los empleados a cierto rol). En este caso habría que recurrir a que una capa intermedia entre usuario y base de datos interceptara la consulta, la modificara añadiendo las condiciones de la regla como cláusulas where y lanzara dicha consulta modificada a la base de datos.

4 Caso de estudio

Esta sección presenta la aplicación de nuestra propuesta al dominio de la salud, más concretamente a la gestión de diagnósticos, involucrando pacientes, doctores y tratamientos asociados. En primer lugar el diseñador modela a alto nivel el sistema incluyendo aspectos de seguridad, para posteriormente generar de forma automatizada la correspondiente implementación considerando Neo4J como plataforma de destino. Finalmente, comprobamos cómo la implementación generada satisface los requisitos de seguridad establecidos.

4.1 Modelo de diseño

Haciendo uso del metamodelo definido en nuestra propuesta, el diseñador modela el sistema a alto nivel. Por motivos descriptivos hemos dividido el modelo en dos partes, una con los elementos estructurales (Figura 2) y otra con las políticas de seguridad (Figura 3).

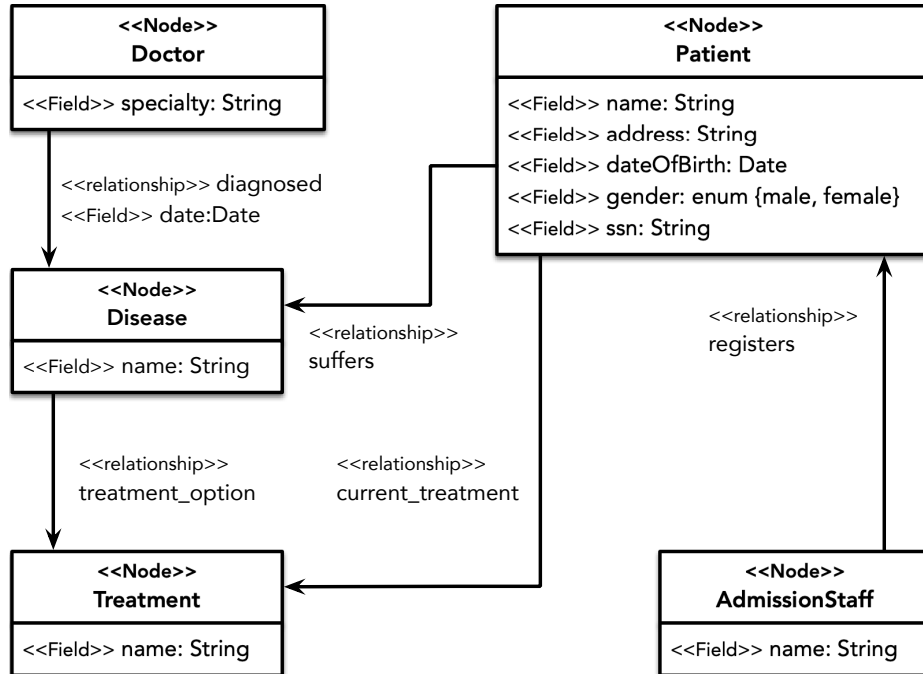


Fig. 2. Modelo del caso de estudio (estructura).

Comenzando con la parte estructural (Figura 2) se definen los siguientes nodos, relaciones y propiedades. El personal de administración (nodo AdmissionStaff) es el encargado de registrar a los pacientes (nodo Patient) y mantener su información asociada (nombre, dirección, número de la seguridad social, etc.). Los doctores (nodo Doctor) cuentan con una especialidad asociada y se encargan de realizar diagnósticos a los pacientes, de forma que los pacientes tienen asociadas enfermedades (nodo Disease) diagnosticadas en cierta fecha por cierto doctor. Además, cada enfermedad tiene asociados una serie de posibles tratamientos (nodo Treatment), siendo el doctor el que selecciona uno de ellos como el tratamiento actual que está siguiendo un determinado paciente que sufre dicha enfermedad.

En cuanto a la parte de seguridad (Figura 3), en primer lugar se decide definir un rol por cada tipo de usuario que podrá interactuar con el sistema:

personal de administración (RoleAdmisionStaff), pacientes (RolePatient) y doctores (RoleDoctor). A continuación, para cada uno de estos roles se establecen un conjunto de autorizaciones que limitan sus privilegios según la política de seguridad buscada. Entrando en más detalle, se definen las siguientes reglas de seguridad:

- El rol de paciente tiene definida una autorización positiva para la consulta de doctores.
- El rol de personal de admisión tiene asociadas varias autorizaciones para otorgar privilegios sobre nodos y relaciones. Por un lado, los privilegios de lectura, creación y modificación (no de eliminación) para pacientes y para la relación que indica que cierto personal ha registrado a determinado paciente. Y por otro lado, para otorgar el privilegio de lectura sobre el personal de admisión.
- El rol doctor presenta dos autorizaciones parecidas a las vistas anteriormente, que otorgan varios privilegios sobre nodos y relaciones. Pero además, define dos autorizaciones de grano fino sobre propiedades. La primera de ellas establece una autorización negativa que retira el permiso de lectura sobre el número de seguridad social de los pacientes (sobre los que tenía acceso de lectura completo). La segunda regla, también refina el acceso a la información de pacientes, retirando esta vez el privilegio de lectura de sus direcciones pero únicamente para aquellos pacientes menores de edad.

Como puede observarse, el modelo presentado facilita la labor del diseñador al abstraerle de detalles técnicos de la plataforma destino a la vez que obtiene una representación más sencilla, ya que varias autorizaciones del sistema pueden representarse a nivel de modelo con una misma regla de seguridad (que involucre a la vez a varios roles, privilegios o elementos).

4.2 Generación de la implementación de políticas de seguridad

En este apartado mostramos cómo las transformaciones propuestas trabajan para que los elementos de seguridad del modelo de diseño del caso de estudio (Figuras 2 y 3) sean transformados en su implementación para un gestor de bases de datos orientadas a grafos (Neo4J) y cómo realmente dicha implementación respeta las medidas de seguridad especificadas.

En primer lugar, los tres roles definidos en el modelo (RoleAdmisionStaff, RolePatient y RoleDoctor) son creados a nivel de implementación, para posteriormente ir estableciendo sobre ellos políticas de seguridad concretas. Para ello, se analizan las reglas de seguridad asociadas a los mismos (Figura 3), generando cada una de ellas una o varias políticas de seguridad a nivel de implementación. En este sentido, a modo de ejemplo, analizaremos cómo se transforman algunas de estas reglas de seguridad.

En cuanto al rol destinado a personal de administración (RoleAdmisionStaff) encontramos varias reglas de seguridad asociadas. Como ejemplo, nos centramos en la regla que autoriza los privilegios de lectura, creación y modificación de

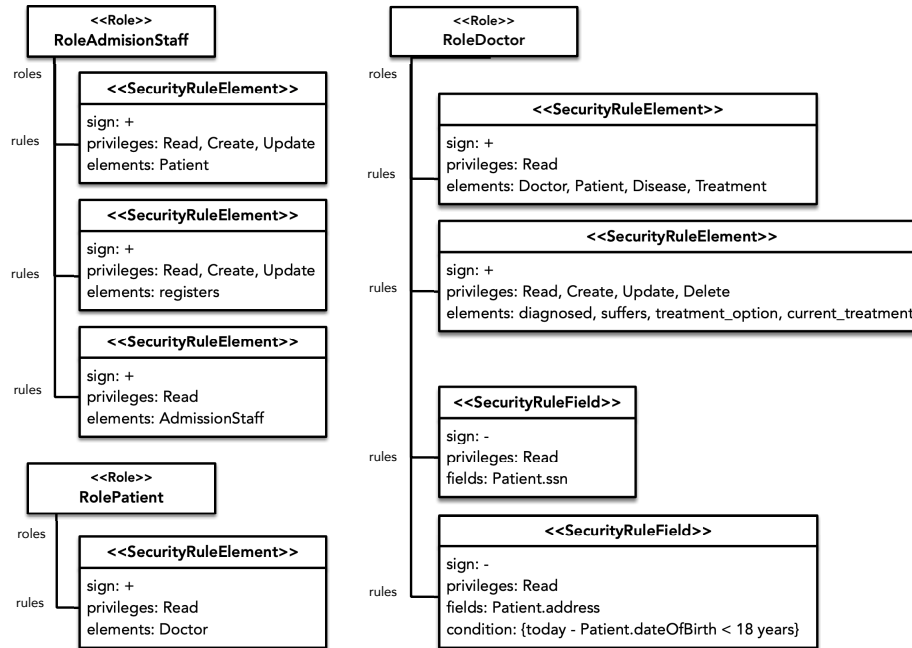


Fig. 3. Modelo del caso de estudio (seguridad).

pacientes. Dicha regla ha de ser transformada en tres autorizaciones positivas a nivel de implementación (una por cada privilegio), tal y como se muestra a continuación.

```

GRANT MATCH{*} ON GRAPH Health NODE Patient
  TO RoleAdmisionStaff;
GRANT CREATE ON GRAPH Health NODE Patient
  TO RoleAdmisionStaff;
GRANT SET PROPERTY{*} ON GRAPH Health NODE Patient
  TO RoleAdmisionStaff;
  
```

Para comprobar que la implementación cumple las restricciones de seguridad especificadas, creamos un usuario de cada rol, con el que realizaremos varias pruebas. Cada usuario hereda los privilegios de su rol, no permitiendo la herramienta refinar más estos permisos a nivel concreto de cada usuario.

A la hora de ingresar al sistema con un usuario del rol RoleAdmisionStaff e interactuar con la base de datos, observamos cómo las operaciones están limitadas al nodo de pacientes. La Figura 4 muestra cómo, tal y como establece la política de seguridad modelada, está limitada la lectura a los pacientes y todos sus campos (se han omitido algunos campos en la figura).

Por último, presentamos el ejemplo de transformación de una regla de seguridad de grano fino (SecurityRuleField). El rol doctor (RoleDoctor), dispone de reglas que autorizan la lectura de varios nodos y relaciones, entre ellos el nodo de

```

+-----+
| n |
+-----+
(:Patient {name: "Agnes S Williams", dateOfBirth: "11/21/1982",
(:Patient {name: "John O Just", dateOfBirth: "14/05/1976", addre
(:Patient {name: "Lucius C Robinson", dateOfBirth: "15/06/1978",
(:Patient {name: "Virginia R Garcia", dateOfBirth: "21/10/1983",
(:AdmissionStaff {name: "Tyler Floyd"})
(:AdmissionStaff {name: "Ulric Barnett"})
+-----+

```

Fig. 4. Ejecución de un usuario del rol RoleAdmisionStaff.

pacientes. Y además, presenta una regla de grano fino que deniega explícitamente la lectura de la propiedad que representa el número de seguridad social (ssn) del paciente. Para transformar esta última regla, se crea a nivel de implementación una autorización negativa sobre dicho campo.

```
DENY READ {ssn} ON GRAPH Health NODE Patient TO RoleDoctor;
```

En este caso, la Figura 5 muestra cómo un usuario doctor consulta el nombre y número de seguridad social de los pacientes y cómo el sistema, de acuerdo con la regla de grano fino establecida, le oculta la información sensible.

```

doctorMurphy@neo4j> match (p:Patient) return p.name, p.ssn;
+-----+
| p.name | | p.ssn |
+-----+
| "Agnes S Williams" | | NULL |
| "John O Just" | | NULL |
| "Lucius C Robinson" | | NULL |
| "Virginia R Garcia" | | NULL |
+-----+

```

Fig. 5. Ejecución de un usuario del rol RoleDoctor.

5 Conclusiones

Las bases de datos NoSQL están adquiriendo una importancia considerable y son muy demandadas debido principalmente al incremento de las tecnologías Big Data. Una base de datos NoSQL es escalable, distribuida y altamente fiable. Sin embargo, actualmente existe un importante obstáculo para su adopción generalizada: la seguridad de los datos.

Por esa razón, hemos construido un metamodelo con el que diseñar bases de datos orientadas a grafos junto con sus políticas de seguridad, incluso a un nivel de grano fino. Este modelo de diseño permite al diseñador abstraerse de aspectos

más técnicos de la implementación final, pero a la vez, nuestra propuesta define un conjunto de reglas de transformación que permiten obtener la implementación final en sistemas gestores concretos, en este caso Neo4J. Como trabajo futuro, pretendemos definir otros conjuntos de reglas de transformación para diferentes gestores de bases de datos orientadas a grafos, teniendo en cuenta las particularidades de la tecnología de salida, como OrientDB y ArangoDB. También planteamos la inclusión de una capa más de seguridad a la que aportan las herramientas actuales, mediante la definición de una herramienta intermedia (entre el usuario y la base de datos) que permita añadir restricciones de seguridad en base a información propia de cada usuario y objeto del sistema (como su identificación, localización, propósito, etc.).

Agradecimientos

Este trabajo ha sido desarrollado en el marco de los proyectos AETHER-UCLM (PID2020-112540RB-C42) y PRECON-I4 (TIN2017-86520-C3-3-R) financiados por el Ministerio de Ciencia e Innovación, y GENESIS (SBPLY/17 /180501/000202) financiado por la Consejería de Educación, Cultura y Deportes de la Dirección General de Universidades, Investigación e Innovación de la JCCM.

References

1. Abbes, H., Gargouri, F.: Big Data Integration: A MongoDB Database and Modular Ontologies based Approach. *Procedia Computer Science* **96**, 446–455 (jan 2016). <https://doi.org/10.1016/j.procs.2016.08.099>
2. Alenezi, M., Usama, M., Almustafa, K., Iqbal, W., Raza, M.A., Khan, T.: An efficient, secure, and queryable encryption for NoSQL-based databases hosted on untrusted cloud environments. *International Journal of Information Security and Privacy* **13**(2), 14–31 (apr 2019). <https://doi.org/10.4018/IJISP.2019040102>
3. Bao, H., He, H., Liu, Z., Liu, Z.: Research on Network Privacy Information Security Management Method Based on NoSQL Database. In: 2019 International Conference on Virtual Reality and Intelligent Systems (ICVRIS). pp. 415–419. IEEE (sep 2019). <https://doi.org/10.1109/ICVRIS.2019.00107>
4. Blanco, C., Fernández-Medina, E., Trujillo, J.: Modernizing Secure OLAP Applications with a Model-Driven Approach. *The Computer Journal* p. bxu070 (2014)
5. Bulusu, S.T., Laborde, R., Wazan, A.S., Barrère, F., Benzekri, A.: A Requirements Engineering-Based Approach for Evaluating Security Requirements Engineering Methodologies. In: *Advances in Intelligent Systems and Computing*, vol. 738, pp. 517–525. Springer Verlag (2018). https://doi.org/10.1007/978-3-319-77028-4_67
6. Fernandes, D., Bernardino, J.: Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB. In: *Proceedings of the 7th International Conference on Data Science, Technology and Applications*. pp. 373–380. SCITEPRESS - Science and Technology Publications (2018). <https://doi.org/10.5220/0006910203730380>
7. Gupta, A., Verma, A., Kalra, P., Kumar, L.: Big Data: A security compliance model. In: *Proceedings of the 2014 Conference on IT in Business, Industry and Government: An International Conference by CSI on Big Data, CSIBIG 2014* (2014). <https://doi.org/10.1109/CSIBIG.2014.7056963>

8. Kantarcioglu, M., Ferrari, E.: Research Challenges at the Intersection of Big Data, Security and Privacy. *Frontiers in Big Data* **2**, 1 (feb 2019). <https://doi.org/10.3389/fdata.2019.00001>
9. Khan, M.U.A., Zulkernine, M.: A Survey on Requirements and Design Methods for Secure Software Development. *International Journal of Computers & Technology* **16**(2009–562), 7047–7064 (2009), <http://techreports.cs.queensu.ca/files/2009-562.pdf>
10. Kurpanik, J., Pańkowska, M.: NoSQL problem literature review. *Studia Ekonomiczne* **234**, 80–100 (2015)
11. Liu, L.: Security and privacy requirements engineering revisited in the big data era. In: *Proceedings - 2016 IEEE 24th International Requirements Engineering Conference Workshops, REW 2016*. p. 55. Institute of Electrical and Electronics Engineers Inc. (jan 2017). <https://doi.org/10.1109/REW.2016.7>
12. Mendelson, A.: Security and Privacy in the Age of Big Data and Machine Learning. *Computer* **52**(12), 65–70 (dec 2019). <https://doi.org/10.1109/MC.2019.2943137>, <https://ieeexplore.ieee.org/document/8909933/>
13. Moreno, J., Fernandez, E.B., Serrano, M.A., Fernández-Medina, E.: Secure Development of Big Data Ecosystems. *IEEE Access* **7**, 96604–96619 (2019)
14. Moreno, J., Serrano, M.A., Fernandez-Medina, E., Fernandez, E.B.: Towards a security reference architecture for big data. In: *CEUR Workshop Proceedings*. vol. 2062 (2018)
15. Pasqualin, D., Souza, G., Buratti, E.L., de Almeida, E.C., Del Fabro, M.D., Weingaertner, D.: A Case Study of the Aggregation Query Model in Read-Mostly NoSQL Document Stores. In: *Proceedings of the 20th International Database Engineering & Applications Symposium on - IDEAS '16*. pp. 224–229. ACM Press, New York, New York, USA (2016). <https://doi.org/10.1145/2938503.2938546>
16. Ramzan, Bajwa, Kazmi, Amna: Challenges in NoSQL-Based Distributed Data Storage: A Systematic Literature Review. *Electronics* **8**(5), 488 (apr 2019). <https://doi.org/10.3390/electronics8050488>
17. Rawat, D.B., Doku, R., Garuba, M.: Cybersecurity in Big Data Era: From Securing Big Data to Data-Driven Security. *IEEE Transactions on Services Computing* pp. 1–1 (mar 2019). <https://doi.org/10.1109/TSC.2019.2907247>
18. Robinson, I., Webber, J., Eifrem, E.: Graph Databases. New opportunities for connected data (2015)
19. Roy-Hubara, N., Sturm, A.: Design methods for the new database era: a systematic literature review. *Software & Systems Modeling* (jun 2019). <https://doi.org/10.1007/s10270-019-00739-8>
20. Samaraweera, G.D., Chang, M.J.: Security and Privacy Implications on Database Systems in Big Data Era: A Survey. *IEEE Transactions on Knowledge and Data Engineering* pp. 1–1 (jul 2019). <https://doi.org/10.1109/TKDE.2019.2929794>
21. Singh, N., Lai, K.H., Vejvar, M., Cheng, T.C.E.: Big Data Technology: Challenges, Prospects, and Realities. *IEEE Engineering Management Review* **47**(1), 58–66 (mar 2019). <https://doi.org/10.1109/EMR.2019.2900208>
22. Weintraub, G., Gudes, E.: Data integrity verification in column-oriented NoSQL databases. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. vol. 10980 LNCS, pp. 165–181. Springer Verlag (2018). https://doi.org/10.1007/978-3-319-95729-6_11
23. Yadav, D., Maheshwari, D.H., Chandra, D.U.: Big Data Hadoop: Security and Privacy. *SSRN Electronic Journal* (apr 2019). <https://doi.org/10.2139/ssrn.3350308>