ReCIBE. Revista electrónica de Computación, Informática, Biomédica y Electrónica
ISSN: 2007-5448
recibe@cucei.udg.mx
Universidad de Guadalajara
México

Chingo Esquivel, Washington; López Sevilla, Galo
Paralelismos entre bases de datos relacionales y no relacionales (un enfoque en seguridad)
ReCIBE. Revista electrónica de Computación, Informática,
Biomédica y Electrónica, vol. 10, núm. 2, 2021, Noviembre-, pp. 1-16
Universidad de Guadalajara
Guadalajara, México

COMPUTATION E INFORMATICA

Recibido 06 jun. 2021 *ReCIBE, Año 10 No. 2, nov. 2021*

Aceptado 14 sep. 2021

Paralelismos entre bases de datos relacionales y no relacionales (un enfoque en seguridad)

Parallelisms between relational and non-relational databases

(a security approach)

Washington Chingo Esquivel[1]
bryanbyr@gmail.com

Galo López Sevilla[1]
glopez@pucesa.edu.ec

[1]Pontificia Universidad Católica del Ecuador - Sede Ambato

Resumen: Las bases de datos nacen como una herramienta para el almacenamiento estandarizado en las aplicaciones, con el avance tecnológico surgen diferentes enfoques a la forma de relacionar los datos que se almacenan en las mismas. Comparar dos tecnologías con una misma finalidad, pero con diferente filosofía puede ayudar a comprender de mejor manera para que fue concebida cada una; y tener una mejor perspectiva sobre cómo se pueden complementar y mejorar, a través de distintas prácticas. A pesar de existir diferentes estudios comparativos sobre las bases de datos relacionales y no relacionales, todos se basan en el rendimiento y no en la seguridad. El estudio tiene como objetivo comparar las seguridades en bases de datos relacionales y no relacionales con la finalidad de encontrar diferencias y similitudes entre bases de datos relacionales y no relacionales .La metodología de comparación, se realiza con servidores en la nube aporta un ambiente apropiado para realizar el experimento entre dos bases de datos muy conocidas y de acceso libre PostgreSQL y MongoDB, siendo la primera un tipo de base relacional y la segunda no relacional (NoSql), para generar como resultado una comparativa que se difunda en la comunidad científica.

Palabras Clave: Seguridad de la información; Bases de Datos; Bases de Datos Relacionales; Bases de Datos No Relacionales; PostgreSQL; MongoDB.

Abstract: Databases were born as a tool for standardized storage in applications, with technological advance it arises different approaches of how to relate the data that is stored on them. Comparing two technologies with the same purpose, but with a different philosophy, can help to understand why each one was conceived; and have a better perspective on how they can be complemented and improved, through different practices. Despite there are different comparative studies on relational and non-relational databases, all based on performance and not in security perspective. The study aims to compare the securities in relational and non-relational databases in order to find differences and similarities between relational and non-relational databases. The comparative methodology, which is carried out with servers in the cloud, provides an appropriate environment to carry out the experiment between two well-known and open-access databases PostgreSQL and MongoDB, the first being a type of relational base and the second is an non-relational (NoSql), to generate as a result a comparison that is disseminated in the community scientific.

Keywords: Information Security; Databases; Relational Databases; Non-Relational Databases; PostgreSQL; MongoDB.

1. Introducción

Las bases de datos son parte fundamental de los sistemas informáticos, Coronel et al. (2010) define una base de datos como una estructura informática compartida e integrada que almacena una colección, mientras que Date (2001) considera a las bases de datos como una especie de armario electrónico para archivar; es decir, es un depósito o contenedor de una colección de archivos de datos computarizados, para administrar una base de datos es necesario un sistema gestor de bases de datos o DBMS (por sus siglas en inglés, *Data Base Management System*) para Sánchez Asenjo (2009) es el software que permite a los usuarios procesar, describir, administrar y recuperar los datos almacenados en una base de datos. En estos Sistemas se proporciona un conjunto coordinado de programas, procedimientos y lenguajes que permiten a los distintos usuarios realizar sus tareas habituales con los datos, además de garantizar la seguridad de los mismos.

Existen diferentes tipos de bases de datos que van desde las bases de datos jerárquicas, en red, relacionales y no relacionales. Codd (1970) diseñó el modelo de bases de datos relacionales en el que propone que: Los usuarios interactúen con un modelo relacional de los datos que consiste en una colección de relaciones que varían en el tiempo y cada usuario no necesita saber más acerca de cualquier relación que su nombre junto con los nombres de sus dominios. Durante muchos años las bases de datos relacionales fueron las predilectas por su facilidad de uso, pero en 2005 Google enfrento un problema con el manejo de grandes cantidades de datos ya que las bases de datos relacionales, eran inadecuadas para hacer frente a los volúmenes y la velocidad de los datos que enfrentaba Google. Para Harrison (2015) los retos que las empresas enfrentan hoy en día con "*big data*" son problemas que Google encontró por primera vez hace casi 20 años. Acorde a Edward y Sabharwal (2015) Carlo Strozzi acuñó el término NoSQL en 1998, al usar este término para identificar su base de datos porque su base de datos no tenía una interfaz SQL por lo que NoSQL es un término general para los almacenes de datos que no siguen los principios de DBMS relacionales. Las bases de datos no relacionales comienzan a tener una gran importancia para el manejo de grandes cantidades de información a partir de los problemas enfrentados por Google.

El DBMS orientado a objetos PostgreSQL (llamado Postgres95) está derivado del paquete Postgres escrito en la universidad de Berkeley. Perkins et al. (2018) detalla las siguientes características: “tiene compatibilidad integrada con Unicode, secuencias, herencia de tablas y sub-selecciones, y es una de las bases de datos relacionales más compatibles con ANSI SQL del mercado”, esto sumado a su longevidad le convierte en uno de los DBMS más probados y estables existentes.

MongoDB fue diseñado como una base de datos escalable, el nombre Mongo proviene de " humongous", con el rendimiento y el fácil acceso a los datos como núcleos objetivos de diseño. Para Perkins et al. (2018) MongoDB es una base de datos de documentos, que le permite almacenar objetos anidados a la profundidad que desee, y puede consultar esos datos anidados en un modo *ad hoc*. No aplica ningún esquema (similar a HBase), por lo que los documentos pueden contener campos o tipos que ningún otro documento en la colección contiene.

Las bases de datos en la actualidad están marcadas por estos dos grupos: relacionales y no relacionales (SQL y NoSQL), existen varios estudios que comparan las ventajas y desventajas a nivel de rendimiento entre ambos grupos, sin embargo, no existe la suficiente bibliografía en la que compare las seguridades de las mismas, para esto es necesario tener en cuenta sus filosofías y principios, ACID en el caso de las bases de datos relacionales y BASE en el caso de las bases de datos no relacionales. Para realizar la comparación se ha seleccionado PostgreSQL uno de los DBMS relacionales más extendidos y MongoDB uno de los DBMS no relacionales más usados, para poder hallar los paralelismos de seguridad existentes entre bases de datos relacionales y no relacionales.

Según Coronel et al. (2010) las bases de datos relacionales reflejan los eventos del mundo real a través de transacciones cuyas propiedades son atomicidad, consistencia, aislamiento y durabilidad:

- La atomicidad (*atomicity*) requiere que se completen todas las operaciones (solicitudes SQL) de una transacción; si no, la transacción se cancela.
- La consistencia (*consistence*) indica la permanencia del estado coherente de la base de datos. Una transacción lleva una base de datos de un estado coherente a otro estado coherente; si alguna de las partes de la transacción viola una restricción de integridad, se aborta toda la transacción.

- Aislamiento (*isolation*) significa que los datos utilizados durante la ejecución de una transacción no pueden ser utilizados por una segunda transacción hasta que se complete la primera.
- La durabilidad (*durability*) garantiza que una vez que se realizan (confirman) los cambios de transacción, no se pueden deshacer ni perder, incluso en el caso de una falla del sistema.

Las Bases de Datos no relacionales por su parte tienen los principios BASE que se basan en el Teorema CAP, para Gilbert y Lynch (2012) el teorema presenta tres requerimientos de un sistema distribuido, sólo es posible proporcionar dos al mismo tiempo como se observa en la figura 1. Los tres requerimientos son:

- Consistencia. Todos los nodos ven los mismos datos al mismo tiempo.
- Disponibilidad. El fallo de uno o más nodos no impide a los demás seguir en funcionamiento.
- Tolerancia a las Particiones. El sistema continúa funcionando a pesar de pérdidas en los mensajes.

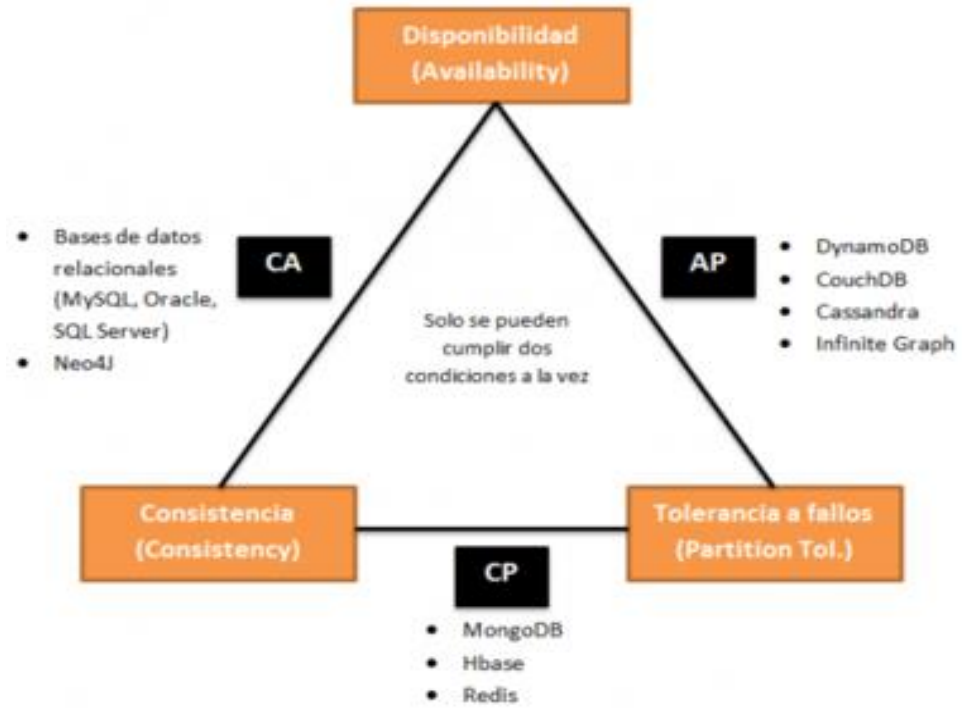

Figura 1. Clasificación de las Bases De Datos según El Teorema De CAP (Rubenfa 2014).

Edward y Sabharwal (2015) establecen las características de BASE que se utiliza en contraposición a ACID:

- *Basically Available* (básicamente disponible) significa que el sistema estará disponible en términos del teorema CAP.
- *Soft State* (estado suave) indica que incluso si no se proporciona ninguna entrada al sistema, el estado cambiará con el tiempo. Esto está de acuerdo con la consistencia eventual.
- *Eventual consistency* (consistencia eventual) significa que el sistema alcanzará consistencia a largo plazo, siempre que no se envíe información al sistema durante ese tiempo.

Con la compresión de los requerimientos que busca cada tipo de base de datos, se puede identificar las seguridades de cada sistema y el porqué del enfoque de cada una de estas. De acuerdo al teorema CAP, se puede señalar que los dos tipos bases de datos convergen en el requerimiento de consistencia.

Seguridad Informática y Seguridad de la Información

La triada de la seguridad informática tiene como objetivos la confidencialidad, integridad y disponibilidad; la seguridad de la información agrega a estos tres objetivos la autenticación. La confidencialidad se refiere a una característica que asegura que solo los usuarios autorizados

puedan leer la información a la cual tengan acceso, la integridad asegura que la información no ha sido modifica por terceros, la disponibilidad asegura que la información pueda accederse bajo cualquier circunstancia y la autenticación asegura que el usuario sea quien dice ser.

2. Metodología

La comparación de los DBMS se realiza a través del método comparativo, que para Tonon (2011) objetivo la búsqueda de similitudes y disimilitudes en la modalidad descriptiva que no busca evaluar las características a estudiar, para el proceso de análisis se toma como referencia PostgreSQL ya que al iniciar el proceso de conexión al DBMS se encuentra la primera característica que es la autenticación, para después pasar a su contraparte en MongoDB, como siguiente paso se analiza la confidencialidad que asegura la conexión a los DBMS, por último la integridad y disponibilidad se analizan por medio de la concurrencia, por su parte la comparación de vulnerabilidades se realiza a través de una variante de la metodología Open Web Application Security Project (OWASP) en los ítems que se considere necesario como el caso de inyección de consulta (*QUERY Injection*). Se ha seleccionado las últimas versiones disponibles al momento de realizar las pruebas que son PostgreSQL 13.0 y MongoDB 4.4.1.

Autenticación (Control de acceso)

PostgreSQL al ser un DBMS relacional ocupa dos formas de control de acceso, Date (2001) conceptualiza el control discrecional “un usuario específico tendrá generalmente diferentes derechos de acceso (también conocidos como privilegios) sobre diferentes objetos”, también el mismo autor define control obligatorio “ cada objeto de datos está etiquetado con un nivel de clasificación determinado y a cada usuario se le da un nivel de acreditación”, en cuanto a MongoDB el control de acceso de acuerdo a Edward y Sabharwal (2015) se realiza en dos niveles “La autenticación verifica la identidad del usuario y la autorización determina el nivel de acciones que el usuario puede realizar en la base de datos autenticada”.

Las vistas que permiten comprobar estas características en PostgreSQL son pg_user y pg_roles, para esto existen comandos \du y \dg que equivalen a consultas SQL, en la figura 2 se realiza la consulta acerca de la información del usuario.

```
postgres=# \du+
********* CONSULTA **********
SELECT usename AS role_name,
       CASE
              WHEN usesuper AND usecreatedb THEN
              CAST('superuser, create database' AS pg_catalog.text)
              WHEN usesuper THEN
              CAST('superuser' AS pg_catalog.text)
              WHEN usecreatedb THEN
              CAST('create database' AS pg_catalog.text)
              ELSE
              CAST('' AS pg_catalog.text)
       END role_attributes
FROM pg_catalog.pg_user ORDER BY role_name desc;
```

Figura 2. Consulta De Información Acerca Del Usuario.

Tanto pg_user como pg_roles son parte del esquema pg_catalog que contiene los metadatos e información sobre el funcionamiento interno de PostgreSQL por lo que tanto usuarios y roles pueden ser consultados por medio de este esquema, en la figura 3 se indica la consulta para los roles del usuario.

```
postgres=# \dg+

********* CONSULTA **********

SELECT r.rolname, r.rolsuper, r.rolinherit, r.rolcreaterole, r.rolcreatedb,
r.rolcanlogin, r.rolconnlimit, r.rolvaliduntil, ARRAY(SELECT b.rolname FROM
pg_catalog.pg_auth_members m JOIN pg_catalog.pg_roles b ON (m.roleid = b.oid)
WHERE m.member = r.oid) as memberof , pg_catalog.shobj_description(r.oid,
'pg_authid') AS description , r.rolreplication FROM pg_catalog.pg_roles r
ORDER BY 1;
```

Figura 3. Consulta de Roles del Usuario.

Las dos consultas dan como resultado una salida similar a la presentada en la figura 4.

```
                                List of roles
Role name    |      Attributes    | Member of | Description
-------------+--------------------+-----------+-----------------
Someone      |                    | {}        | THIS IS A ROLE COMMENT
```

Figura 4. Resultado de las consultas.

Como se resultado se obtiene información acerca del nombre de usuario, sus atributos, grupos y roles. MongoDB por su parte ocupa privilegios de acuerdo a cada base de datos, para esto dispone de métodos de manejo de usuarios y roles, para obtener información de los usuarios se ocupa el comando getUser() y para la información del rol del usuario getRole(), estos métodos se ejecutan a través de db que hace referencia a la base de datos actual y se puede comprobar con el siguiente comando observado en la figura 5.

```
use Nombredb
db.getRole("readWrite", { showPrivileges: true })
"role" : "dbAdmin",
"db" : "Nombre_db",
"isBuiltin" : true,
"roles" : [ ],
"inheritedRoles" : [ ],
"privileges" : [ {
        "resource" : {
                "db" : "Nombre_db",
                "collection" : ""
        },
        "actions" : [
                "collMod",
                "collStats",
                "compact",
                "convertToCapped",
                "createCollection",
                "createIndex",
                "dbStats",
                "dropCollection",
                "dropDatabase",
                "dropIndex",
                "enableProfiler",
                "indexStats",
                "planCacheIndexFilter",
                "planCacheRead",
                "planCacheWrite",
                "reIndex",
                "renameCollectionSameDB",
                "repairDatabase",
                "storageDetails",
                "validate"
        ]
} ]
```

Figura 5. Comando GETROLE().

El resultado del comando es mucho más amplio y se ha recortado a la parte que es de interés para el estudio, se puede observar el nombre del usuario, sobre qué base de datos tiene privilegios, sus privilegios sobre la base de datos y las acciones que puede realizar.

Por lo tanto, en cuanto a autenticación y autorización los dos tipos de DBMS manejan iguales formas de control de acceso de datos, observando que según de The MongoDB Manual estas opciones vienen desactivadas por defecto, y se debe activar mediante la característica "*authorization: enabled*" en la configuración del DBMS.

Confidencialidad (Cifrado de datos)

La forma más aceptada para asegurar la confidencialidad es a través de la encriptación o cifrado de datos, aquí se encuentra un símil entre los dos DMS. PostgreSQL ocupa diferentes niveles de encriptación comenzado con encriptación de clave a través de la configuración de password_encryption, encriptación de columnas a través del módulo pgpcrypto, encriptación de partición de datos, encriptación de datos a través de la red por medio de hostssl usando *Secure Socket Layer* (SSL), además de encriptación del lado del cliente. MongoDB por su parte permite encriptación de datos en movimiento a través de *Transport Layer Security* (TLS) y SSL, encriptación de datos en reposo donde de forma nativa se puede encriptar un archivo (desde la versión 3.2), además de encriptación del lado del cliente. Acorde a esto a pesar de ciertas diferencias la confidencialidad es un elemento que los sistemas gestores de bases de datos buscan asegurar.

A continuación, se muestra en la tabla 1 la comparativa de algoritmos de encriptación e intercambio de claves de los sistemas gestores de bases de datos.

Tabla 1. Comparativa entre los algoritmos de encriptación PostgreSQL y MongoDB

PostgreSQL	MongoDB
• MD5 • SHA1 • SHA224/256/384/512 • Blowfish • AES • DES/3DES/CAST5 • Raw encryption • PGP • Symmetric encryption • PGP Public-Key encryption	• AES256-GCM • SCRAM-SHA-1* • SCRAM-SHA-256* *Utilizadas para el intercambio de claves por medio de Public Key Infrastructure-PKI

Fuente: elaboración propia, basado en Lockart (1996) y The MongoDB Manual.

El módulo pgpcyrpto se usaría de una manera similar a la de la figura 6.

```
psql CREATE EXTENSION pgcrypto; SELECT crypt('mypass', gen_salt('bf', 4));

crypt --------------------------------------------------------------
$2a$04$1bfMQDOR6aLyD4q3KVb8/ujG7ZAkyie4d/s3ABwuZNbzkFFgXtC76
```

Figura 6. Ejemplo Criptografía PostgreSQL.

En MongoDB se pude comprobar la encriptación por medio del comando mostrado en la figura 7.

```
db.system.users.findOne({user:dbAdmin})
{
	"_id":"admin.dbAdmin",
	"userId":UUID("f325412e-1943-4a47-b637-ecbcafe58cf5"),
	"user":"dbAdmin",
	"db":"admin",
	"credentials": {
		"SCRAM-SHA-1" : {
			"iterationCount":10000,
			"salt":"jGWNJKQzx8UWIddH8GTyRw==",
			"storedKey":"5uuLgYrOzCBcZ0/rgFy0XhWDFQY=",
			"serverKey":"LfQdMqNa68JWESYebyg86G9tueo"
		},
		"SCRAM-SHA-256" : {
			"iterationCount":15000,
			"salt":"oOyPfEfITiIRvOfvFeuTuXQAe0cFNP60vba14g==",
			"storedKey":"6nwbtJfub5nR4/7+Wgo0kNvWWD86uIUDsHe03/tmgY=",
			"serverKey":"Htw2GA/VQQwZbFgh5soqAVyOIDy1v7YPDgj0rmfDO/g="
		}
	},
	"roles": [
		{
			"role": "root";
			"db": "admin"
		}
	]
}
```

Figura 7. Ejemplo Criptografía MongoDB.

Concurrencia

La concurrencia en PostgreSQL se basa en Control de la Concurrencia MultiVersión (por sus siglas en ingles MMVC - *Multi-Version Concurrency Control*), Bernstein y Goodman (1983) establecen el control de concurrencia como la actividad de sincronizar operaciones emitidas al ejecutar concurrentemente programas en una base de datos compartida. La figura 8 muestra como MMVC, evita la sobrecarga de bloqueo, ya que evita que el bloque 2 cree un bloqueo que habría impedido que la sesión 2 lea los datos hasta que se complete la transacción de la sesión 1.

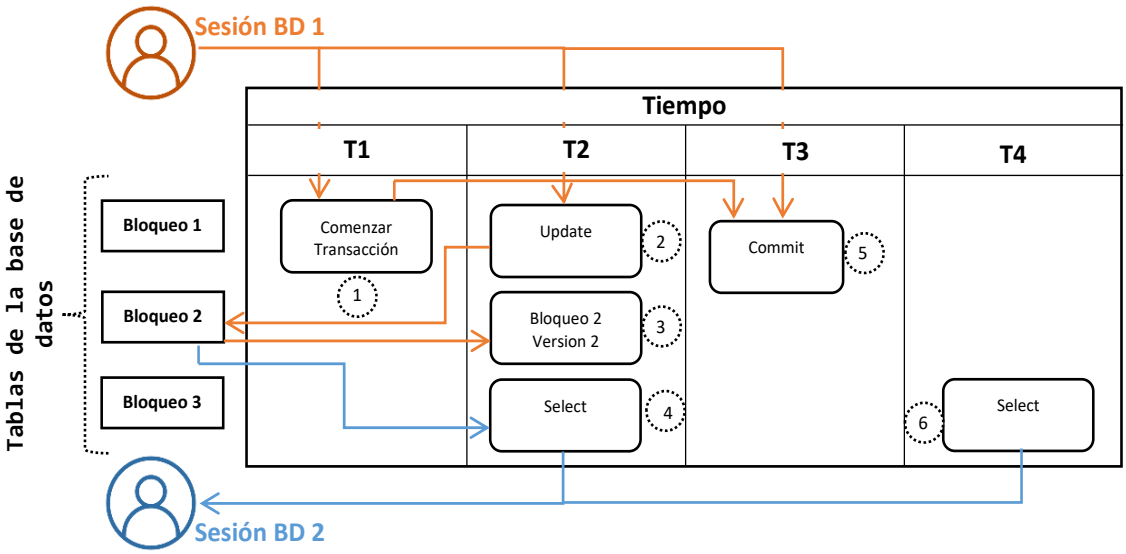

Figura 8. Funcionamiento De MMVC basado en Harrison (2015)

Transacciones Serializables

Una ejecución serial de las transacciones significa que éstas se ejecutan una después de otra, sin cualquier intercalación de operaciones (Ricardo, 2009). Si X y Y son dos transacciones, sólo hay dos posibilidades de ejecución seriales: se termina la transacción X y después se ejecuta Y, o se termina la transacción Y para luego ejecutar X. PostgreSQL maneja la serialización de transacciones a través del aislamiento transaccional por medio de tres acciones no deseadas conocidas como lectura sucia, lectura no repetible y lectura fantasma y cuatro niveles de aislamiento, como se observa en la tabla 2.

Tabla 2. Niveles de aislamiento de PostgreSQL

	Lectura "sucia"	Lectura no repetible	Lectura "fantasma"
Lectura no cursada	Posible	Posible	Posible
Lectura cursada	No Posible	Posible	Posible
Lectura repetible	No Posible	No Posible	Posible
Serializable	No Posible	No Posible	No Posible

Fuente: Elaboración propia basado en Lockhart (1996)

De acuerdo con Lockhart (1996) en el manual de usuario de PostgreSQL las acciones no deseadas se definen como:

- Lecturas sucias. Una transacción lee datos escritos por una transacción no esperada, no cursada.
- Lecturas no repetibles. Una transacción vuelve a leer datos que previamente había leído y encuentra que han sido modificados por una transacción cursada.
- Lectura fantasma. Una transacción vuelve a ejecutar una consulta, devolviendo un conjunto de filas que satisfacen una condición de búsqueda y encuentra que otras filas que satisfacen la condición han sido insertadas por otra transacción cursada.

En cuanto a niveles de aislamiento Lockhart (1996) enumera:

- Nivel de lectura cursada: Nivel de aislamiento por defecto, una consulta a este nivel sólo ve datos cursados antes de que la consulta comenzara. Si una fila devuelta por una consulta mientras se ejecuta una declaración UPDATE (o DELETE, o SELECT FOR UPDATE) está siendo actualizada por una transacción concurrente no cursada, entonces la segunda transacción que intente actualizar esta fila esperará a que la otra transacción se curse o pare. Los resultados de la ejecución de SELECT o INSERT (con una consulta) no se verán afectados por transacciones concurrentes.
- Nivel de aislamiento serializable: Nivel más alto de aislamiento transaccional, una consulta en este nivel sólo ve los datos cursados antes de que la transacción comience y nunca ve ni datos sucios ni los cambios de transacciones concurrentes cursados durante la ejecución de la transacción. La ejecución de una declaración UPDATE (o DELETE, o SELECT FOR UPDATE) está siendo actualizada por una transacción concurrente no cursada, la segunda transacción que trata de actualizar esta fila esperará a que la otra transacción se curse o pare. Los resultados de la ejecución de SELECT o INSERT (con una consulta) no se verán afectados por transacciones concurrentes.

Las transacciones y sus diferentes estados se pueden comprobar a través de la vista pg_stat_activity, como se muestra en la figura 9.

Figura 9. Consulta De Transacciones Serializables.

Como se observa en la figura la consulta regresa un registro de las transacciones que se realizan desde el inicio del DBMS y tienen los siguientes pid que es identificador de la transacción, usesysid identificador del usuario, usename con el nombre del usuario, backend_start que indica cuando realmente se realizó la transacción en caso de haber una serie y tiene las variaciones de milisegundos entre transacciones, además se observa el tipo de evento en espera, el evento, la consulta.

Bloqueos

Los candados (también llamados bloqueos) y las estampas de tiempo son las dos técnicas que generalmente se usan para garantizar la seriabilidad de transacciones concurrentes (Ricardo, 2009). PostgreSQL ofrece varios modos de bloqueo para controlar el acceso concurrente a los datos en tablas. Algunos de estos modos de bloqueo los adquiere PostgreSQL automáticamente antes de la ejecución de una declaración, mientras que otros son proporcionados para ser usados por las aplicaciones.

PostgreSQL presenta bloqueos a nivel de tabla y fila, de acuerdo con Lockhart (1996) los bloqueos a nivel de tabla son:

- AccessShareLock modo de bloqueo adquirido automáticamente sobre tablas que están siendo consultadas.
- RowShareLock adquirido por SELECT FOR UPDATE y LOCK TABLE para declaraciones IN ROW SHARE MODE.
- RowExclusiveLock adquirido por UPDATE, DELETE, INSERT y LOCK TABLE para declaraciones IN ROW EXCLUSIVE MODE.
- ShareLock adquirido por CREATE INDEX y LOCK TABLE para declaraciones IN SHARE MODE.
- ShareRowExclusiveLock adquirido por LOCK TABLE para declaraciones IN SHARE ROW EXCLUSIVE MODE.
- ExclusiveLock adquirido por LOCK TABLE para declaraciones IN EXCLUSIVE MODE.
- AccessExclusiveLock adquirido por ALTER TABLE, DROP TABLE, VACUUM y LOCK TABLE.
- Bloqueos a nivel de fila: se producen cuando campos internos de una fila son actualizados (o borrados o marcados para ser actualizados). SELECT FOR UPDATE modificará las filas seleccionadas marcándolas, de tal modo que se escribirán en el disco. Los bloqueos a nivel de fila no afectan a los datos consultados. Estos son usados para bloquear escrituras a la misma fila únicamente.

  Los bloqueos se pueden comprobar por medio de la consulta mostrada en la figura 10.

```
SELECT * from pg_locks pl LEFT JOIN pg_stat_activity psa ON pl.pid=psa.pid;
locktype|database|relation|page|tuple|virtualxid|transactionid|
virtualtransaction|classid|objid|objsubid|pid|mode|granted|
fastpath|datid|datname|pid|leader_pid|usesysid|usename|
application_name|client_addr|client_hostname|client_port|backend_start|xact_start|
query_start|state_change|wait_event_type|wait_event|state|backend_xid|backend_xmin|
query|backend_type

relation|13395|12250|--|--|--|--|
--|--|--|3/20|66|AccessShareLock|t|
t|1395|postgres|66|--|10|postgres|
psql|127.0.0.1|--|38346|2020-11-25 16:33:11.130921+00|--|
--|--|--|--|active|--|486|
SELECT * from pg_locks pl LEFT JOIN pg_stat_activity psa ON pl.pid=psa.pid|clientbackend
```

Figura 10. Consulta De Bloqueos En PostgreSQL.

La consulta retorna el tipo de bloqueo, id de la base de datos, la relación en este caso AcessSharelock que se adquiere automáticamente al ser la tabla consultada y como se observa lo que se hizo fue una consulta a pg_locks, además se tienen los mismos campos que en la consulta de transacciones.

Bloqueos en MongoDB

MongoDB no implementa un sistema MVCC y, por lo tanto, la lectura no se puede realizar en un documento que se está actualizando. MongoDB de forma predeterminada proporciona un documento estricto único de consistencia, en una implementación de un solo servidor en una base de datos. Cuando se modifica un documento de MongoDB, se bloquea tanto para lecturas como para escrituras para otras sesiones. En implementaciones de conjuntos de réplicas, es posible configurar algo más cercano a consistencia eventual al permitir que las lecturas se completen en servidores secundarios que pueden contener datos desactualizados.

Acorde a Harrison (2015) la coherencia para documentos individuales se logra en MongoDB mediante el uso de bloqueos. Se utilizan cerraduras para asegurar que dos escrituras no intentan modificar un documento simultáneamente, y también que un lector no verá una Vista inconsistente de los datos.

Para Edward y Sabharwal (2015) en MongoDB, aunque las lecturas se pueden enrutar a nodos secundarios, las escrituras siempre se enrutan al nodo primario, erradicando el escenario en el que dos nodos intentan simultáneamente actualizar el mismo conjunto de datos. El conjunto de datos del nodo principal siempre es coherente. Si las solicitudes de lectura se enrutan al nodo principal, siempre verá los cambios actualizados, que significa que las operaciones de lectura son siempre coherentes con las últimas operaciones de escritura, así se logra un bloqueo a nivel granular.

Sin embargo, si la aplicación ha cambiado la preferencia de lectura para leer desde secundarios, puede haber una probabilidad de que el usuario no vea los últimos cambios o los estados anteriores. Esto se debe a que las escrituras son replicadas asincrónicamente en los secundarios.

El bloqueo granular que tiene cuatro niveles, acorde a Giamas (2019) estos niveles son:

- Global (Instancia de MongoDB): Todas las bases de datos en la instancia son afectadas.
- Base de Datos: Sólo la base de datos donde se aplica el bloqueo es afectada.
- Colección: Sólo la colección donde se aplica el bloqueo es afectada.
- Documento: Sólo el documento donde se aplica el bloqueo es afectado

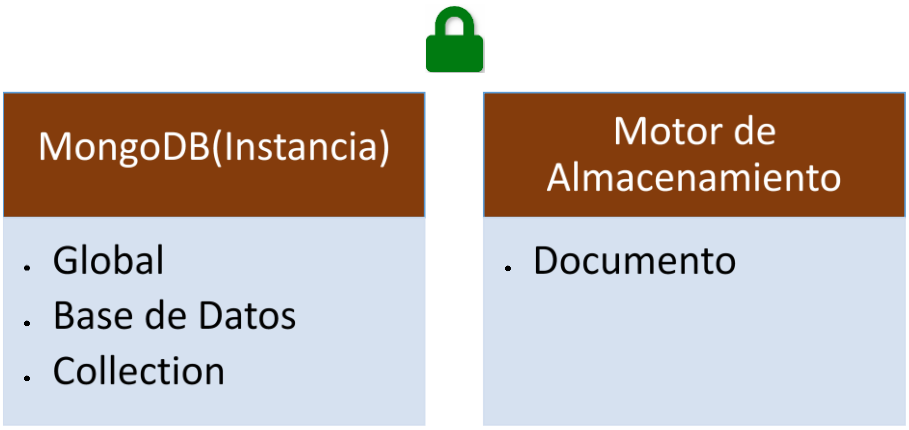

Figura 11. Bloqueos En MongoDB elaboración propia Basado en MongoDB Locks - Shared, Exclusive And Intent Modes (2018).

Además, según *The MongoDB Manual (2020)* existen cuatro modos de bloqueo que son:

- S – *Shared* – Este modo permite que los lectores concurrentes compartan el recurso y se utiliza en operaciones READ.
- X – *Exclusive* – Este modo evita que los lectores concurrentes compartan el recurso y se utiliza en operaciones WRITE.
- IS – *Intent Shared* – Este modo indica que el contenedor del bloqueo podrá leer el recurso a un nivel granular. Si se aplica IS a la base de datos, significa que el contenedor aplicará bloqueo en modo S a nivel de colección o documento.

- IX – *Intent Exclusive* – Este modo indica que el contenedor del bloqueo podrá modificar el recurso a nivel granular. Si se aplica IX a la base de datos, significa que el contenedor aplicará un bloqueo en modo X a nivel de colección o documento.

Estos modos de bloqueo se pueden aplicar sobre otro ya existente, con ciertas restricciones entre cada modo como se muestra en la tabla 3.

tabla 3. Modos de bloqueo MongoDB.

Nuevo modo de bloqueo \ Modo de bloqueo ya existente en un recurso	**S**	**IS**	**IX**	**X**
S		SI	NO	NO
IS	SI		SI	NO
IX	NO	SI		NO
X	NO	NO	NO	

Fuente: elaboración propia, basado en Giamas (2019)

Una forma de comprobar los bloqueos en MongoDB es a través del siguiente comando de la figura 12.

```
db.currentOp();
{
      "locks": {"^Nombre_db": "R"},
      "ns": "Nombre_db.bar",
      "op": "query",
      "opid": 1349152,
      "query": {"test": 1},
      "secs_running": 15,
      "waitingForLock": true
}
```

Figura 12. Comando para verificar Bloqueos MongoDB.

Como se observa la base de datos se encuentra esperando el bloqueo esto es más frecuente cuando se trabaja con multiusuarios. PostgreSQL basa la concurrencia en transacciones serializables y los bloqueos basados en MMVC, mientras MongoDB basa la concurrencia en bloqueos.

Así como es posibles hallar paralelismos en las seguridades de los dos tipos de DBMS, también es posible encontrar vulnerabilidades en las que los dos DBMS coinciden independientemente de su finalidad.

Vulnerabilidades

OWASP TOP 10 presenta las vulnerabilidades que más afectan a aplicaciones web, pero estas se pueden tomar como marcadores para otras aplicaciones como el primer ítem de control A1:2017 Inyección, luego A6:2017 Configuración de Seguridad Incorrecta, a pesar de no ser vulnerabilidades directamente relacionadas a los DBMS, sino que se presentan con el uso de los mismos, por lo cual es necesario detallar su funcionamiento.

Inyección

De acuerdo a The OWASP Foundation (2017) las fallas de inyección, como SQL, NoSQL, OS o LDAP ocurren cuando se envían datos no confiables a un intérprete, como parte de un comando o consulta. Los datos dañinos del atacante pueden engañar al intérprete para que ejecute comandos involuntarios o acceda a los datos sin la debida autorización.

Para Nwankwo (2020) las bases de datos NoSQL sufren ataques de inyección de consultas al igual que las bases de datos SQL, a pesar de las modificaciones a los protocolos a través de una serie de técnicas de bajo nivel, los riesgos de inyección, la gestión del control de acceso inadecuado y la exposición de la red insegura siguen siendo relativamente altos.

Por lo general la Inyección se presenta a través del cliente por falta de revisión de parámetros o de validación de consultas, como se observa en la figura 13.

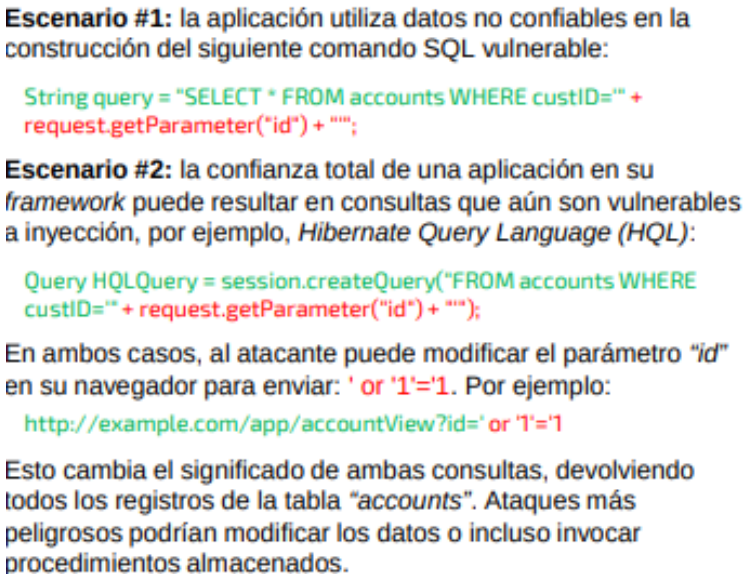

Figura 13. Ejemplos de Inyección (The OWASP Foundation 2017).

Configuración de Seguridad Incorrecta

Para The OWASP Foundation (2017) la configuración de seguridad incorrecta es un problema muy común y se debe en parte a establecer la configuración de forma manual, *ad hoc* o por omisión (o directamente por la falta de configuración). Esta vulnerabilidad está directamente relacionada a los administradores de los DBMS y como se menciona anteriormente se debe principalmente al desconocimiento por parte de los mismos acerca de prestaciones que estas tecnologías ponen a su disposición.

3. Resultados

Una vez detalladas las características de seguridad comunes entre los dos DBMS, se conceptualiza cada uno de los mecanismos por los cuales aseguran estas características, además de ciertas vulnerabilidades comunes, se obtiene como resultado la tabla 4 que presenta los paralelismos de seguridad entre bases de datos relacionales y no relacionales que al igual que el desarrollo del estudio toma como estructura la autenticación, confidencialidad, concurrencia y por último la vulnerabilidades que afectan a los dos DBMS.

TABLA 4. Paralelismos entre Bases de Datos Relacionales y No Relaciones

DBMS Característica	POSTGRESQL	MONGODB
Autenticación	• Control de Acceso Obligatorio • Control de Acceso Discrecional	• Autenticación • Autorización * Desactivadas por defecto
Confidencialidad	• Encriptación de clave • Encriptación de columnas • Encriptación de partición de datos • Encriptación de datos a través de la red • Encriptación del lado del cliente	• Encriptación de datos en movimiento • Encriptación de datos en reposo • Encriptación del lado del cliente * De forma nativa a nivel de archivos desde la versión 3.2.
Concurrencia	• *Multi-Version Concurrency Control* Transacciones Serializables Bloqueos	• Bloqueos
Vulnerabilidades	Errores de configuración Inyección de consultas	

4. Conclusiones

A pesar de la diferencia de filosofías y de objetivos de los dos sistemas gestores de bases de datos, se halla que los dos DBMS aseguran la autenticación, confidencialidad y concurrencia que incluye integridad y disponibilidad, características propias de la seguridad de sistemas informáticos.

La autenticación se logra a través de dos mecanismos en los dos DBMS y a pesar de que los nombres varían son bastante similares, sin embargo, la autenticación esta deshabilitada por defecto en MongoDB ya que no es parte de sus objetivos. La confidencialidad se asegura por medio de la encriptación y a pesar de estar mucho más desarrollada en el DBMS relacional, el DBMS no relacional la ha puesto especial interés en los últimos años. Por su parte la concurrencia que asegura la disponibilidad e integridad evidencia que parte de MMVC está presente en los dos DBMS, que es el punto en cual los dos tipos de DBMS convergen en el teorema de CAP, así se puede concluir que los paralelismos de seguridad entre bases de datos relacionales y no relacionales existen y ayudan comprender de mejor manera el funcionamiento de cada tipo de DBMS siempre teniendo en cuenta sus filosofías.

Por ultimo las vulnerabilidades que afectan a los dos tipos de DBMS son más de índole administrativa y por parte del cliente, lo que deja en claro que es necesario sin importar el tipo de DBMS leer sus manuales, entender su finalidad e investigar las prestaciones y características de seguridad que cada uno de estos ofrece.

Referencias Bibliográficas

Bernstein, P. A., & Goodman, N. (1983). Multiversion concurrency control—Theory and algorithms.

ACM Transactions on Database Systems, *8*(4), 465-483.

https://doi.org/10.1145/319996.319998

Codd, E. F. (1970). *A Relational Model of Data for Large Shared Data Banks*. *13*(6), 11.

Coronel, C., Morris, S., & Rob, P. (2010). *Bases de datos [recurso electrónico]: Datos, implementación y administración.* (9.ª ed.). Cengage Learning Editores, S.A. de C.V.

Date, C. J. (2001). *Introducción a los sistemas de bases de datos*. Pearson Educación.

Edward, S. G., & Sabharwal, N. (2015). *Practical MongoDB*. Apress. https://doi.org/10.1007/978-1-4842-0647-8

Giamas, A. (2019). *Mastering MongoDB 4.x: Expert techniques to run high-volume and fault-tolerant database solutions using MongoDB 4.x, 2nd Edition*. Packt Publishing Ltd.

Gilbert, S., & Lynch, N. (2012). Perspectives on the CAP Theorem. *Computer*, *45*(2), 30-36. https://doi.org/10.1109/MC.2011.389

Harrison, G. (2015). *Next Generation Databases*. Apress. https://doi.org/10.1007/978-1-4842-1329-2

Lockhart, T. (Ed.). (1996). *Manual del usuario de PostgreSQL*.

MongoDB Locks—Shared, Exclusive and Intent Modes. (2018). *TutorialKart*. https://www.tutorialkart.com/mongodb/mongodb-locks/

Nwankwo, W. (2020). A Review of Critical Security Challenges in SQL-based and NoSQL Systems from 2010 to 2019. *International Journal of Advanced Trends in Computer Science and Engineering*, *9*(2), 2029-2035. https://doi.org/10.30534/ijatcse/2020/174922020

Perkins, L., Redmond, E., & Wilson, J. R. (2018). *Seven Databases in Seven Weeks*. Pragmatic Programmers.

Ricardo, C. M. (2009). *Bases de datos*. The McGraw-Hil.

rubenfa. (2014, enero 28). *NoSQL: Clasificación de las bases de datos según el teorema CAP*. Genbeta. https://www.genbeta.com/desarrollo/nosql-clasificacion-de-las-bases-de-datos-segun-el-teorema-cap

Sánchez Asenjo, J. (2009). *Sistemas Gestores de Bases de Datos* (Autoedicion).

The MongoDB 4.4 Manual—MongoDB Manual. (s. f.). Https://Github.Com/Mongodb/Docs/Blob/Master/Source/Index.Txt. Recuperado 11 de diciembre de 2020, de https://docs.mongodb.com/manual/

The OWASP Foundation. (2017). *OWASP Top 10—2017 Los diez riesgos más críticos en Aplicaciones Web*. The OWASP Foundation. https://wiki.owasp.org/images/5/5e/OWASP-Top-10-2017-es.pdf

Tonon, G. (2011). *La Utilización Del Método Comparativo En Estudios Cualitativos En Ciencia Política Y Ciencias Sociales: Diseño y desarrollo de una tesis doctoral*. https://dialnet.unirioja.es/descarga/articulo/3702607.pdf