Comparación de Bases de Datos Relacionales y no Relacionales para el Análisis de Datos

Jesús García Viana[1], Griselda Cortés[2], Mercedes Flores[2], Xóchitl R.Wong[2]

Resumen

Prácticamente cualquier tarea relevante en nuestro mundo requiere el uso de computadoras, ya que es a través de ellas, de los sistemas de información y del software, que es posible llevar el control de infinidad de acciones de las cuales pueden depender decisiones importantes o gestiones de cualquier índole. El punto en común de todos esos sistemas, son las bases de datos, las cuales se encargan de guardar información y permitir su consulta cuando sea necesario. Una vez que los datos se encuentran recopilados, deben ser analizados de forma estadística para sacar de ellos el mejor partido posible, no obstante en ocasiones estos datos no son medibles

Acerca de los autores...

[1] Alumno de Maestría en Ingeniería en Sistemas Computacionales, del Tecnológico de Estudios Superiores de Ecatepec

[2] Docente de Ingeniería y de la Maestría en Sistemas Computacionales, del Tecnológico de Estudios Superiores de Ecatepec

prfviana@hotmail.com
griscar7772@yahoo.com.mx
merfloresflores@yahoo.com.mx,
xochitlwong@yahoo.com

de forma clara y es en este punto donde la lógica difusa (LD) con el empleo de conjuntos e inferencia difusa, puede ayudar a medirlos de manera útil.

Palabras Clave: Bases de datos, SQL, NoSQL, Lógica difusa, Análisis de datos.

Introducción

Día tras día, el manejo de la información se hace más complejo; diferentes factores hacen que las personas involucradas en el manejo computacional de datos, busquen tecnologías que les ayuden con este problema; sin embargo, el tinte comercial que se le da al tema dificulta realizar una buena elección. La creciente y enorme cantidad de datos generados por las aplicaciones empresariales actuales, han originado conjuntos masivos de éstos. Los sistemas de gestión de bases de datos (SGBD) NoSQL han surgido como una alternativa a los SGBD relacionales para la gestión de estos conjuntos. En este artículo, se abordan las ventajas de los SGBD basados en SQL y los que se decantaron por alternativas NoSQL para el manejo de datos, así como la interpretación de la información gracias a la lógica difusa.

1. Antecedentes de las bases de datos relacionales

Desde su creación, las bases de datos han sido un soporte para la organización de la información dentro de los diferentes tipos de entidades, debido a que "las bases de datos comenzaron a aparecer a finales de 1950 y comienzos de 1960, impulsadas por dos factores tecnológicos: el incremento de la fiabilidad de los procesadores de ordenador y la expansión de la capacidad de almacenamiento secundario en cintas y unidades de disco" [1].

El modelo relacional fue presentado por Edgar Frank Codd en 1970, convirtiéndose en uno de los modelos matemáticos más importantes para la representación de las bases de datos. Es un modelo basado en registros, como lo es también el modelo de red y el jerárquico, pero supuso un gran avance con respecto a estos dos modelos, fundamentalmente por la forma de conexión entre los registros. Mientras en los modelos de red y jerárquico la conexión entre registros se realiza mediante punteros, en el modelo relacional se utiliza el valor de los registros. Esto último permitió una definición matemática formal del modelo [2] que implicó un cambio radical en el manejo de la información, apoyándose en operaciones de conjuntos que combinan tablas de datos separadas (o relaciones) para producir un conjunto de respuestas. Las consultas se especifican utilizando el lenguaje de consulta estructurado SQL (por las siglas en inglés de Structured Query Language), soportado en el álgebra relacional, y que permite a un usuario expresar su consulta en forma declarativa, sin ningún tipo de instrucciones detalladas de programación [1].

Cuando se implementa el modelo relacional sobre un sistema de gestión de bases de datos, se definen dos lenguajes: el de definición de datos (DDL) y el de manipulación de datos (DML). El DDL permite la creación, modificación y eliminación de elementos de la base de datos. Estos elementos pueden ser: tablas, vistas, índices y otros más sofisticados, en dependencia del SGBD. El DML permite la consulta, modificación e inserción de datos en la base de datos. La función de consulta es la más importante y se basa en el álgebra relacional y/o el cálculo relacional [2].

2. Lenguaje de consulta estructurado SQL

El lenguaje de consulta estructurado SQL brinda soporte para la creación y manipulación de las bases de datos relacionales, así como de la gestión y recuperación de los datos almacenados en ellas. SQL es un lenguaje no procedimental o declarativo, es decir, se le indica a la computadora lo que se desea obtener o se está buscando, pero no se especifica cómo se debe recuperar dicha información. Algunos proveedores de los sistemas gestores de bases de datos relacionales complementaron el lenguaje SQL para brindar capacidades de lenguajes procedimentales, por ejemplo Transact-SQL, Microsoft SQL Server y PL/SQL de Oracle [3].

Los estándares SQL publicados hasta el año 1992 se basaron únicamente en el modelo relacional; a partir de la versión del año 1999, se llegó a la conclusión que un lenguaje estrictamente relacional no satisfacía las demandas del mundo real, por esta razón en la edición de 1999 se incluyeron conceptos característicos de la programación orientada a objetos. El hecho de que SQL no soportara tipos de datos complejos y definidos por el usuario, hizo que en la versión del año 2006 se incorporarán éstas y otras capacidades orientadas a objetos, como métodos y encapsulación, lo que convierte a SQL en un lenguaje de base de datos relacional a objeto [3].

Desde la aparición del Modelo Relacional, muchos fabricantes de sistemas comenzaron a incluir el modelo en sus propuestas y, por ende, SQL [1]. Actualmente la gran mayoría de los sistemas de bases de datos que podemos encontrar en el mercado están fabricados sobre la base de tecnología relacional. Entre los principales vendedores de soportes para el desarrollo de sistemas de bases de datos que lo utilizan se encuentran:

1. *DB2* de IBM. Este es considerado el primer producto que incorporó el lenguaje SQL, según Michael Stonebraker, aunque aún persiste la pelea entre Oracle y DB2 respecto a quién fue primero.

2. *dBase.* Es uno de los primeros sistemas de bases de datos relacionales para microcompuradoras. Tuvo una gran popularidad en la década de los 80.

3. *Informix.* Gestor de base de datos desarrollado por Informix Software. A pesar de tener éxito en los 90 (fue el segundo sistema de bases de datos más popular después de Oracle); su popularidad ha ido desapareciendo progresivamente.

4. *InterBase.* Desarrollado y comercializado por *Borland.* Las características de este sistema que lo distinguen del resto, son pocos recursos de

administración y una arquitectura multigeneracional; se ejecuta en Windows, Linux y Solaris.

5. *Microsoft SQL Server*. Es uno de los principales gestores de bases de datos utilizados en negocios e instituciones.

6. *Oracle*. En la década de los 90 fue el sistema de bases de datos más popular. Actualmente es líder en bases de datos de gran tamaño como las que se utilizan en grandes compañías.

7. *Microsoft Access*. Sistema de bases de datos relacional orientado a las pequeñas empresas.

8. *PostgreSQL*. Sistema de bases de datos con soporte para el modelo relacional distribuido de forma gratuita. En los últimos años ha aumentado significativamente su utilización en diferentes tipos de aplicaciones.

9. *Berkeley DB*. Es una base de datos empotrada, que puede ser utilizada por varios lenguajes de programación.

10. *MySQL*. Es uno de los gestores de bases de datos relacionales más instalados en el mundo. Aunque sus comienzos fueron totalmente como software libre, actualmente está patrocinado y es propiedad de una empresa privada, aunque aún existe una distribución GPL.

11. *Apache Derby*. Es una base de datos relacional *open source* implementada enteramente en Java.

Existen muchos más sistemas gestores de bases de datos, pero esta lista engloba a los más significativos y populares [1] [2] y [3].

2.1 Ventajas de bases de datos relacionales

Los sistemas de bases de datos presentan numerosas ventajas que se pueden dividir en dos grupos: las que se deben a la integración de datos y las que se deben a la interfaz común que proporciona el SGBD.

Ventajas por la integración de datos

Control sobre la redundancia de datos. Los sistemas de ficheros almacenan varias copias de los mismos datos en ficheros distintos. Esto hace que se desperdicie espacio de almacenamiento, además de provocar la falta de consistencia de datos (copias que no coinciden). En los sistemas de bases de datos todos estos ficheros están integrados, por lo que no sealmacenan varias copias de los mismos datos. Sin embargo, en una base de datos no se puede eliminar la redundancia completamente, ya que en ocasiones es necesaria para modelar las relaciones entre los datos, o bien es necesaria para mejorar las prestaciones.

Consistencia de datos. Eliminando o controlando las redundancias de datos se reduce en gran medida el riesgo de que haya inconsistencias. Si un dato está almacenado una sola vez, cualquier actualización se debe realizar sólo una vez, y está disponible para todos los usuarios inmediatamente. Si un dato está duplicado y el sistema conoce esta redundancia, el propio sistema puede encargarse de garantizar que todas las copias se mantienen consistentes. Desgraciadamente, no todos los SGBD de hoy en día se encargan de mantener automáticamente la consistencia.

Compartición de datos. En los sistemas de ficheros, los ficheros pertenecen a las personas o a los departamentos que los utilizan. Pero en los sistemas de bases de datos, la base de datos pertenece a la empresa y puede ser compartida por todos los usuarios que estén autorizados. Además, las nuevas aplicaciones que se vayan creando pueden utilizar los datos de la base de datos existente.

Mantenimiento de estándares. Gracias a la integración es más fácil respetar los estándares necesarios, tanto los establecidos a nivel de la empresa como los nacionales e internacionales. Estos estándares pueden establecerse sobre el formato de los datos para facilitar su intercambio, pueden ser estándares de documentación, procedimientos de actualización y también reglas de acceso.

Ventajas por la existencia del SGBD

Mejora en la integridad de datos. La integridad de la base de datos se refiere a la validez de los datos almacenados. Normalmente, la integridad se expresa mediante restricciones o reglas que no se pueden violar. Estas restricciones se pueden aplicar tanto a los datos, como a sus relaciones, y es el SGBD quien se debe encargar de mantenerlas.

Mejora en la seguridad. La seguridad de la base de datos es la protección de la base de datos frente a usuarios no autorizados. Sin unas buenas medidas de seguridad, la integración de datos en los sistemas de bases de datos hace que éstos sean más vulnerables que en los sistemas de ficheros. Sin embargo, los SGBD permiten mantener la seguridad mediante el establecimiento de claves para identificar al personal autorizado. Las autorizaciones se pueden

realizar a nivel de operaciones, de modo que un usuario puede estar autorizado a consultar ciertos datos pero no a actualizarlos, por ejemplo.

Mejora en la accesibilidad a los datos. Muchos SGBD proporcionan lenguajes de consultas o generadores de informes que permiten al usuario hacer cualquier tipo de consulta sobre los datos, sin que sea necesario que un programador escriba una aplicación que realice tal tarea.

Mejora en la productividad. El SGBD proporciona muchas de las funciones estándar que el programador necesita escribir en un sistema de ficheros. A nivel básico, el SGBD proporciona todas las rutinas de manejo de ficheros típicas de los programas de aplicación. El hecho de disponer de estas funciones permite al programador centrarse mejor en la función específica requerida por los usuarios, sin tener que preocuparse de los detalles de implementación de bajo nivel. Muchos SGBD también proporcionan un entorno de cuarta generación consistente en un conjunto de herramientas que simplifican, en gran medida, el desarrollo de las aplicaciones que acceden a la base de datos. Gracias a estas herramientas, el programador puede ofrecer una mayor productividad en un tiempo menor.

Mejora en el mantenimiento gracias a la independencia de datos. En los sistemas de ficheros, las descripciones de los datos se encuentran inmersas en los programas de aplicación que los manejan. Esto hace que los programas sean dependientes de los datos, de modo que un cambio en su estructura, o un cambio en el modo en que se almacena en disco, requiere cambios importantes en los programas cuyos datos se ven afectados. Sin embargo, los SGBD separan las descripciones de los datos de las aplicaciones. Esto es lo que se conoce como independencia de datos, gracias a la cual se simplifica el mantenimiento de las aplicaciones que acceden a la base de datos.

Aumento de la concurrencia. En algunos sistemas de ficheros, si hay varios usuarios que pueden acceder simultáneamente a un mismo fichero, es posible que el acceso interfiera entre ellos de modo que se pierda información o, incluso, que se pierda la integridad. La mayoría de los SGBD gestionan el acceso concurrente a la base de datos y garantizan que no ocurran problemas de este tipo [4].

2.1.1 Desventajas de bases de datos relacionales

Costo. La costosa creación y mantenimiento del sistema. Por lo general, se necesita comprar un software especial si la empresa es grande y necesitas una base de datos más robusta, tendrás que contratar a un programador para crear una base de datos y a un administrador de base de datos para el mantenimiento una vez instalada. Más allá del tamaño de la empresa, si almacena información confidencial o legalmente protegida en la base de datos, como la información sanitaria, números de seguro social o números de

tarjetas de crédito, también se tendrá que proteger los datos contra el acceso no autorizado con el fin de cumplir con las normas reglamentarias.

Abundancia de información. Las imágenes complejas, números, diseños y productos multimedia desafían una fácil categorización liderando el camino para un nuevo tipo de base de datos relacional de objetos llamada sistemas de gestión de bases de datos. Estos sistemas están diseñados para manejar aplicaciones más complejas y tienen la capacidad de ser escalables.

Límites estructurados. Algunas bases de datos relacionales tienen límites en cuanto a la longitud de los campos. Al diseñar la base de datos, debes especificar la cantidad de datos que pueden caber en un campo. Algunos de los nombres o las consultas de búsqueda son más cortos que el actual, y esto puede conducir a la pérdida de datos.

Bases de datos aislados. Los sistemas complejos de bases de datos pueden conducir a que estas bases de datos se conviertan en "islas de información", donde la información no puede ser compartida fácilmente de un sistema a otro. A menudo, las grandes empresas o instituciones, se encuentran con que las bases de datos relacionales en divisiones separadas crecieron de manera diferente. Por ejemplo, tal vez el departamento de facturación del hospital utiliza una base de datos, mientras que el departamento de personal del hospital utiliza una base de datos diferente. Lograr que las bases de datos se "comuniquen" entre sí puede ser largo, costoso y abrumador, sin embargo, en un sistema complejo hospitalario, todas las bases de datos deben estar comunicadas para un buen cuidado del paciente y del personal [3] [4] y [5].

2.2 Bases de datos no relacionales

El término bases de datos no-relacionales se refiere a las BD que no satisfacen el modelo relacional. Esta denominación puede llevar a malinterpretaciones, ya que sugiere que son BD (o para ser más precisos, sistemas de gestión de BD) que carecen de las características disponibles en un SGBD relacional (cuyo lenguaje de gestión de datos es usualmente SQL); por ello se prefiere el nombre SGBD Not Only SQL (SGBD NoSQL) para enfatizar que aunque estos SGBD carecen de algunas características propias de los SGBD relacionales, también suelen incluir características no disponibles en estos últimos[6].

En nuestra época, la recopilación de datos se ha convertido en algo común que se utiliza día a día, un ejemplo claro son las redes sociales [7] y [8] que si bien no tienen el objetivo de recopilar grandes volúmenes de información dado su uso actual pueden ser empleadas para ese fin. Para almacenar la información que se genera de estas redes las bases de datos relacionales no son lo suficientemente capaces dadas sus limitaciones [8] y [9] para esta tarea se crearon las bases de datos no relacionales [9] [10] y [11] que resultan más eficientes a la hora de manejar de forma mucho más veloz grandes volúmenes de datos [7]. Existen varios SGBD para bases de datos no relacionales, de entre ellos destaca MongoDB [12] y [13], que es una base de datos documental, y se caracteriza por ser sumamente rápida, pero también suele presentar algunos problemas en consultas complejas [14] pero con sus actualizaciones y herramientas van solucionando estas deficiencias [16] y [17]. Otro SGBD no relacional es NEO4J, enfocado a bases de datos orientadas a grafos el cual aporta múltiples ventajas entre ellas su organización, fiabilidad y rapidez [18] y [19].

2.2.1 Tipos de SGBD No SQL

En los últimos años ha crecido el interés por los sistemas de gestión de bases de datos NoSQL y continuamente aumenta el número de empresas que las utilizan o que lo harán en el corto plazo Las modernas aplicaciones surgidas, principalmente, con la Web 2.0 tienen requisitos exigentes de escalabilidad y rendimiento en el manejo de grandes volúmenes de datos complejos, las cuales no pueden ser satisfechas por los sistemas relacionales. Para satisfacer estos requisitos han ido surgiendo, a lo largo de esta década, más de doscientos sistemas no relacionales que son agrupados bajo el término No SQL [6].

En realidad estos sistemas son un reflejo de diferentes paradigmas de modelado de datos, entre los que destacan:

Clave/valor: el almacenamiento se hace mediante estructuras con dos componentes: una clave y un valor. Ejemplo: {"nombre": "Juan"}, donde la clave es "nombre" y su valor es "Juan". Entre los SGBD NoSQL de este tipo están Redis, Dynamite! y Project Voldemort.

Columna: el almacenamiento se hace por columnas. Es decir, una fila de datos corresponde a un grupo de valores para un mismo atributo, e.g., como se muestra en la Tabla 1. Por otro lado, en una BD relacional una fila de datos corresponde a un grupo de valores, uno para cada atributo del esquema, e.g., como se muestra en la Tabla 2. Entre los SGBD No SQL de este tipo están Apache Hbase, Cassandra y Hypertable.

Grafo: el almacenamiento se hace mediante grafos. En los nodos se almacenan entidades, las cuales se relacionan por medio de aristas. Ejemplo: ver Figura 1. Entre los SGBD No SQL de este tipo están Neo4j, OrientDB e InfiniteGraph [6].

Tabla 1
Ejemplo de filas de empleados en una BD NoSQL orientada a columnas

1	2	3	→ Una fila de ids.
Pedro	Ana	María	→ Una fila de Nombres.
18	20	31	→ Una fila de Edades.
8	10	6	→ Una fila de Grados
10	10	20	→ Una fila de Departamento.

Tabla 2
Ejemplo de filas de empleados en una BD relacional orientada a filas

ID	Nombre	Edad	Grado	Departamento	
1	Pedro	18	8	10	→ Un empleado.
2	Ana	20	10	10	→ Un empleado.
3	María	31	6	20	→ Un empleado.

Documento: en una base de datos documental; en lugar de tablas de registros hay colecciones de documentos que no son más que objetos JSON análogos a las tablas, la diferencia más grande es que el sistema de claves ajenas debe ser simulado mediante métodos de estructuración *programados ya que el SGBD no lo incorpora como es en el caso de las BD relacionales [15].*

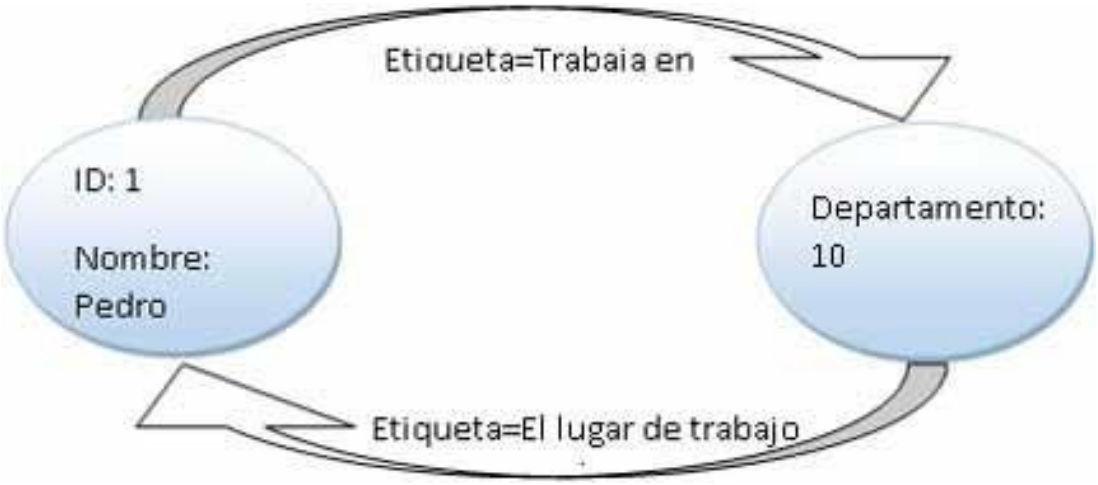

Figura 1

Representación de datos en una BD NoSQL de tipo grafo

2.2.2 Diferencias y ventajas de NoSQL

Algunas de las diferencias más destacables que nos podemos encontrar entre los sistemas NoSQL y los sistemas SQL están:

No utilizan SQL como lenguaje de consultas. La mayoría de las bases de datos NoSQL evitan emplear este tipo de lenguaje o lo utilizan como un lenguaje de apoyo. Por poner algunos ejemplos, Cassandra usa el lenguaje CQL, MongoDB aplica JSON y BigTable hace uso de GQL.

No utilizan estructuras fijas como tablas para el almacenamiento de los datos.

Permiten hacer uso de otros tipos de modelos de almacenamiento de información como sistemas de clave valor, objetos o grafos.

No suelen permitir operaciones JOIN.

Al disponer de un volumen de datos tan extremadamente grande, suele resultar deseable evitar los JOIN. Esto se debe a que, cuando la operación no es la búsqueda de una clave, la sobrecarga puede llegar a ser muy costosa. Las soluciones más directas consisten en desnormalizar los datos, o bien realizar el JOIN mediante software, en la capa de aplicación.

Arquitectura distribuida

Las bases de datos relacionales suelen estar centralizadas en una sola máquina o bien en una estructura maestro-esclavo; sin embargo, en los casos NoSQL la información puede estar compartida en varias máquinas mediante mecanismos de tablas Hash distribuidas [7] [11] y [15].

Finalmente, la mayor ventaja de NoSQL sobre SQL es la velocidad y la capacidad de manipular cantidades exorbitantes de datos de forma ágil [2] [3] [6] y [15].

3. Lógica difusa para el análisis de datos

Numerosos estudios se han realizado centralizando a la lógica difusa como principal herramienta en su desarrollo [24] [25] [26] [27] y [28], ya que el dominio de los conjuntos difusos es de gran utilidad para tareas o datos que no pueden ser clasificados de forma tajante.

La LD, cuya creación se atribuye al azerbaiyano Lotfi Zadeh, en la década de los sesenta del siglo pasado, ha sido desarrollada básicamente en distintas prácticas disciplinares, especialmente en las relacionadas con control de procesos industriales, el sector de la computación y numerosas aplicaciones en la economía. [23]

Los conceptos empleados en Lógica Difusa y Probabilidad están relacionados en cierto modo, pero son totalmente diferentes. De forma resumida, la probabilidad representa información sobre frecuencia de ocurrencias relativas de un evento bien definido sobre el total de eventos posibles. Por su parte, el grado de pertenencia difuso representa las similitudes de un evento con respecto a otro, donde las propiedades de esos eventos no están definidos en forma precisa. [29].

El propósito de la agrupación de datos (clustering), es la de segmentar la información de acuerdo con criterios definidos de similitud y de cumplimiento de características o patrones; así, se generan los conjuntos denominados cluster [25].

El análisis de grandes volúmenes de datos para descubrir información oculta dentro de los mismos datos, como pueden ser patrones de conducta o preferencias a ciertas acciones u objetos, recibe el nombre de "Mineria de datos", actualmente es una gran área de investigación y desarrollo [22], la cual se apoya de muchas herramientas para llevar a cabo esta tarea, una de ellas es la "Lógica difusa". Dicha teoria se fundamenta en los conjuntos de números difusos o borrosos, que denotan en esencia grupos de elementos que pertenecen con intensidades o grados diversos a una cierta categoria [23] lo que permite utilizar esta herramienta para diversos análisis de datos de diferentes campos, como la medicina, la selección de personal, el análisis de riesgo, la toma de decisiones, la educación, las reglas lingüísticas, entre otras [24] [25] [26] [27] y [28].

Conclusiones

Este trabajo tiene como objetivo el brindar un panorama general que permita discernir entre la selección de un SGBD relacional o uno no relacional, tomando en cuenta las caracteristicas de los mismos, sus diferencias, sus ventajas y sus desventajas. Al mismo tiempo y dado que se trata de una tendencia relativamente nueva, se exponen diversos tipos de arquitecturas basadas en NoSQL haciendo énfasis en aquellas características que lo han colocado como una opción más que viable en la creación de sistemas modernos con gran afluencia de datos.

De acuerdo con el sitio DBEngines (db-engines.com/en/ranking) el Top 10 dentro de las bases de datos más utilizadas en la actualidad, se encuentra ocupado por Oracle, MySQL, Microsoft SQLServer, PostgreSQL, Mongo DB, DB2, Microsoft Access, Cassandra, Redis y ElasticSearch, seis de estas posiciones se encuentran ocupadas por SGBD dirigidas al modelo relacional, lo cual deja en claro que son un instrumento muy utilizado en la actualidad,

ya que incluso poseen los cuatro primeros lugares; no obstante, desde la aparición de NoSQL hasta la fecha, le ha valido el hacerse con posiciones dentro de este ranking y si la tendencia continúa poco a poco irán ganando terreno, dadas sus capacidades.

Es obvia la tendencia que está sucediendo actualmente en el mercado, ya que incluso grades empresas como Oracle, Microsoft y Google están apostando por bases de datos no relacionales tales como Microsoft Azure u Oracle NoSQL, Coherence y BerkerleyDB, por mencionar solo algunas de ellas. Además nuevas compañías se perfilan con nuevos modelos de SGBD siendo líderes dentro de las áreas de cada gestor, por ejemplo MongoDB en bases de datos documentales o Neo4J, líder en bases de datos orientadas a grafos. De continuar esta tendencia, en pocos años NoSQL será algo tan común como lo son ahora las bases de datos relacionales.

Por otra parte, los datos almacenados pueden revelar con un debido y correcto estudio, tendencias a un determinado comportamiento por parte de los usuarios; este conocimiento puede ser utilizado por las empresas para mejorar sus ventas o subsanar errores que estén cometido sin siquiera saberlo. Dicho estudio es únicamente gracias a técnicas de clasificación de datos, haciendo especial énfasis en la lógica difusa, que ha probado, a lo largo del tiempo, ser una herramienta más que idónea para la clasificación de datos con cierto grado de incertidumbre y establecer grados de pertenencia.

Al exponer cada punto mencionado anteriormente, se espera sea más sencilla la selección para la integración de tecnologías con respecto al tipo de sistema que se desee implementar y que de acuerdo con sus necesidades, no se opte de manera inmediata por los SGBD tradicionales, sino que se evalué si es la mejor opción dado el mercado actual y la tendencia futura.

Trabajos futuros

Se pretende diseñar un sistema que pueda gestionar de forma íntegra toda el área de tutoría del Tecnológico de Estudios Superiores de Chimalhuacán. Para esta tarea se ha optado por una base de datos no relacional orientada a grafos llama Neo4J, que aunque está en la posición 21 del ranking (consultar db-engines.com/en/ranking) es la número uno de su campo, esto por diversas razones:

Escalabilidad: Es posible dado la naturaleza del proyecto que no solo sea aplicable en esa institución y siempre cabe la posibilidad de que sea implementada en otros centros similares, para ello se requiere una base de datos capaz de manejar grandes volúmenes de datos de forma eficiente.

Carencia de información: Es una base de datos de la cual existe poca documentación y de hecho libera constantes actualizaciones de forma periódica, señal clara de su constante fluctuación y mejora.

Integración: De acuerdo con el sitio oficial de Neo4J (https://neo4j.com) actualmente posee soporte para la integración con diversos lenguajes de programación, entre ellos destacan los principales para programación web o dispositivos móviles, como o son C#, Java, PHP, Ruby y Python.

Estos puntos son clave para aventurarse y brindar la confianza en este nuevo SGBD.

Bibliografía

Cuevas Rodríguez, L. (2009). *Modelo difuso de base de datos objeto-relacional: Propuesta de implementación en software libre*, Granada, España.

Fernández Garciamo, J.D.J. y Segura Londoño, C. (2015). *Utilidad de las bases de datos NoSQL en relación con las técnicas de big data*, Pereira.

Castro Romero, A., González Sanabria, J.S. y Callejas Cuervo, M. "Utilidad y funcionamiento de las bases de datos NoSQL". *Facultad de Ingeniería, UPTC*, Vol.21, No. 33, pp. 21-32, 2012.

Marqués, M. (2009). *Bases de Datos*, España: Universitat Jaume I de Castelló.

Anni, M. "Techlandia", Leaf Group, 18 Junio 2015. [En línea]. Available: https://techlandia.com/desventajas-base-datos-relacional-lista_183865/. [Último acceso: 5 Enero 2018].

Moreno Arboleda, F.J., Quintero Rendón, J.E. y Rueda Vásquez, R. "Una comparación de rendimiento entre Oracle y MongoDB", *Ciencia e Ingeniería Neogranadina*, Vol. 26, No. 1, pp. 109-129, 2016.

Hernández Chillón, A., Feliciano Morales, S., García Molina, J. y Sevilla Ruiz, D. (2017). *Visualización de Esquemas en Bases de Datos NoSQL basadas en documentos*. España: Facultad de Informática, Universidad de Murcia.

Blanco Rojas, T., Archila Córdoba, D.M. y Ballesteros-Ricaurte, J.A. "Gestión de datos obtenidos desde redes sociales aplicando Business Intelligence Engineering Process", *Revista Virtual Universidad Católica del Norte*, pp. 73-91, 2016.

Franklin Leung, B.Z. (2016). "Performance evaluation of Twitter datasets", *Web Intelligence*. IOS Press, pp. 275-286.

Rodríguez Pérez, "Selección de Base de Datos No SQL para almacenamiento de Históricos en Sistemas de Supervisión", *Revista Cubana de Ciencias Informáticas*, pp. 85-96, 2016.

Castro Romero, A., González Sanabria J.S., y Callejas Cuervo, M., *Utilidad y funcionamiento de las bases de datos NoSQL*. Facultad de Ingeniería, pp. 21-32, 2012.

Vázquez Ortiz, Y., Mier Pierre L., y Sotolongo León, A. "Características no relacionales de PostgreSQL: incremento del rendimiento en el uso de datos JSON", *Revista Cubana de Ciencias Informáticas*, pp. 70-81, 2016.

Sánchez de Madariaga, R. "Examining database persistence of ISO/EN 13606 standardized electronic health record extracts: relational vs. NoSQL approaches", *BMC Medical Informatics and Decision Making*, pp. 15, 2017.

Sulistyo Heripracoyo, R.K. (2016). *Big Data Analysis with MongoDB for Decision Support System*. TELKOMNIKA, pp. 1083-1089.

Quintero, I. L. (2016). *Node.js*, España: Publicaciones Altaira.

Pejic, S. "Stonebraker on NoSQL and enterprise", *Communications of the ACM*, pp. 10-11, 2011.

Guo, R. (2017). MongoDB's JavaScript Fuzzer, ACMQUEVE, pp. 43-47.

Alomari, E., Barnawi, A. y Sakr, S., (2015). «CDPort: A Portability Framework for NoSQL Datastores,» Cross Mark, p. 2531-2553.

A. Perçukua, D. Minkovskab y L. Stoy, «Modeling and Processing Big Data of Power Transmission Grid Substation Using Neo4j,» Procedia Computer Science , Nº 113, pp. 10-16, 2017.

A. Hernández Chillón, S. Feliciano Morales, J. García Molina y D. Sevilla Ruiz, *Visualización de Esquemas en Bases de Datos NoSQL basadas en documentos*, España: Facultad de Informática, Universidad de Murcia, 2017.

J. Sanabria Garzón, «Herramienta software para implementar minería de datos: clusterización utilizando lógica difusa,» *Orinoquia*, Vol. 8, Nº 1, pp. 15-23, 2004.

B. Asdrúbal Arroyo y N. T. Antolínez, «La lógica difusa como herramienta de evaluación en el sector universitario,» *Alteridad*. Revista de Educación, Vol.10, Nº 2, pp. 132-145, 2015.

A. Echeverría, «Lógica difusa aplicada a la toma de decisiones» *Ingeniería Industrial*, vol. XXXI, Nº 1, pp. 1-5, 2010.

L. González Palacio, J. A. Echeverri y G. Urrego Giraldo, «Herramienta de lógica difusa para definir rasgos de la personalidad de un individuo» *Revista Ingenierías. Universidad de Medellín*, Vol. 10, Nº 19, pp. 137-148, 2011.

C. A. Díaz Contreras, A. Aguilera-Rojas y N. Guillén-Barrientos, «Lógica difusa vs. modelo de regresión múltiple para la selección de personal,» Ingeniare. *Revista chilena de ingeniería*, Vol. 22, Nº 4, pp. 547-559, 2014.

F. J. Ruvalcaba Coyaso y A. Vermonden, «Lógica difusa para la toma de decisiones y la selección de personal,» *Universidad & Empresa*, Vol. 17, Nº 29, pp. 239-256, 2015.

J. I. R. Patiño, A. A. A. Vivas y C. A. T. Isaza, «Definición de un modelo de medición de análisis de riesgos de la seguridad de la información aplicando lógica difusa y sistemas basados en el conocimiento,» *Entre Ciencia e Ingeniería*, Vol. 9, Nº 17, pp. 71-80, 2015.

C. G. Morcillo, «Lógica Difusa Una introducción práctica,» Técnicas de Softcomputing, 2017.