

## Bases de datos NoSQL

*Ing Hansel Gracia del Busto<sup>1</sup>, Ing Osmel Yanes Enríquez<sup>2</sup>*

<sup>1</sup> DISERTIC. Ingeniero [hansel@tesla.cujae.edu.cu](mailto:hansel@tesla.cujae.edu.cu)

<sup>2</sup> DISERTIC, Ingeniero [yanes@tesla.cujae.edu.cu](mailto:yanes@tesla.cujae.edu.cu)

### RESUMEN

Las bases de datos NoSQL han experimentado un importante incremento en su aplicación en los últimos tiempos. La gran flexibilidad que ofrecen y las posibilidades que brindan desde el punto de vista de la optimización en sus diseños de acuerdo al problema a resolver las convierten en una atractiva variante a tener en cuenta para los desarrolladores de aplicaciones de gestión de información. En el presente artículo se hace un recorrido por la evolución de los tipos de bases de datos hasta llegar a las relacionales, las cuales se analizan con el objetivo de mostrar los aspectos asociados a estas que propiciaron el surgimiento de las NoSQL.

**Palabras claves:** bases de datos relacionales, bases de datos NoSQL, gestión de información digital,

### ABSTRACT

*NoSQL databases have experienced a significant increase in use in recent times. The great flexibility offered and the possibilities provided from the point of view of optimizing their designs according to the problem to solve it an attractive option to consider for developers of information management applications. In this paper, we journey through the evolution of the types of databases to reach the relational, which are analyzed in order to show the aspects associated with these that led to the rise of NoSQL.*

**Key words:** relational databases, NoSQL databases, digital management information.

## **INTRODUCCIÓN**

En la actualidad existe una gran polémica alrededor del tema relacionado con la gestión de la información digital que se necesita almacenar para su posterior recuperación y análisis por parte de los diseñadores, arquitectos y desarrolladores de aplicaciones informáticas de cualquier tipo. Por un lado están los tradicionales sistemas de gestión de bases de datos relacionales (RDBMS, por sus siglas en inglés) y por otro los prometedores sistemas de bases de datos no relacionales y distribuidos conocidos como NoSQL (Not only SQL). Los primeros, dueños de la mayor parte del mercado del almacenamiento de datos, con una robustez innegable y años de explotación en múltiples entornos de gestión de información. Los segundos, emergentes y novedosos, ofrecen sin embargo una nueva forma de pensar en el desarrollo de aplicaciones web orientadas y centradas en el usuario [1]. Se hace necesario entonces hacer un análisis consciente de ambas variantes por parte de los especialistas en la gestión de la información encargados de la selección de uno u otro, con el objetivo de escoger la solución óptima al problema presentado. A esta idea en particular se pretende tributar con el análisis que se presenta en este artículo, realizando un estudio cronológico sobre el surgimiento y evolución hasta la actualidad de los sistemas de gestión de bases de datos para finalmente llegar a conclusiones al respecto.

Con la intención de organizar el análisis y ganar en claridad sobre el objetivo principal del artículo se expondrán las causas que originaron el surgimiento de los RDBMS y los problemas que resolvieron en sus inicios. Luego se mostrarán los problemas presentados con la variante anterior y la necesidad del surgimiento de nuevas alternativas al mismo. Finalmente se introducen los aspectos relacionados con el nacimiento y evolución de los sistemas de bases de datos NoSQL. Al final el escenario quedará listo para que las personas involucradas en el proceso de selección del gestor de bases de datos a utilizar puedan juzgar en su justa medida las conclusiones a las que se arriban.

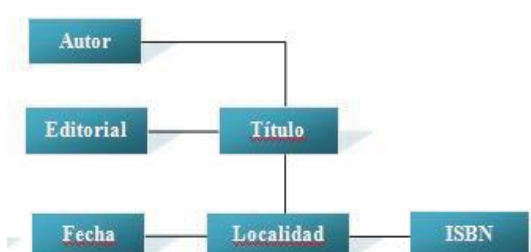
### **1. Bases de datos relacionales**

#### **1.1. Causas de su origen**

La aparición del modelo relacional para las bases de datos estuvo precedida por la existencia de las variantes jerárquicas y de red, las cuales permitían almacenar estructuras de datos tales como listas y árboles. En el caso de la primera variante se trata de un sistema de gestión de bases de datos que almacena la información en una estructura jerárquica que enlaza los registros en forma de estructura de árbol, donde un nodo padre de información puede tener varios nodos hijo. Esta relación jerárquica no es estrictamente obligatoria, de manera que pueden establecerse relaciones entre nodos hermanos. En este caso específico la estructura en forma de árbol se convierte en una estructura en forma de grafo dirigido, convirtiéndose entonces en un sistema del tipo de red antes mencionado. [2] Una estructura de base de datos de red, en resumen, abarca más que la estructura de árbol debido a que un nodo hijo en la estructura puede tener más de un nodo padre. [3] Ver en la figura 1 y 2.



**Figura 1. Representación de una base de datos jerárquica**



**Figura 2. Representación de una base de jerarquía de datos**

Los problemas típicos de las bases de datos jerárquicas, y en menor medida de red, derivan del hecho de que los sistemas gestores de ambas no implementan ningún control sobre los datos, sino que queda en manos de las aplicaciones que garantizan que se cumplan las condiciones invariantes que se requieran para su buen funcionamiento. En resumen estas implementaciones delegaban en los desarrolladores de aplicaciones la responsabilidad de definir el mecanismo de salva y recuperación de la información contenida en los orígenes. En la práctica todas las aplicaciones están sujetas a errores y fallos por lo que es prácticamente imposible que no se produzcan inconsistencias a la hora de almacenar los datos en la base de datos. Por otra parte, dichas condiciones suelen romperse ex profeso por motivos operativos (generalmente ajustes debidos a cambios en la lógica del negocio que se está modelando) o en busca de optimizar la rapidez en el acceso sin evaluar de manera consciente sus consecuencias.

Los problemas comunes que se producen en estas variantes de bases de datos, producto de lo anteriormente expuesto son:

- **Duplicidad de registros:** no se garantiza la inexistencia de registros duplicados. Esto también es cierto para los campos "clave". Es decir, no se garantiza que dos registros cualesquiera tengan diferentes valores en un subconjunto concreto de campos.
- **Integridad referencial:** no existe garantía de que un registro hijo esté relacionado con un registro padre válido. Esto significa que es posible borrar un nodo padre sin eliminar antes los nodos hijos que tiene asociados, de manera que éstos últimos quedan relacionados con un registro inválido o inexistente.
- **Desnormalización:** este no es tanto un problema del modelo jerárquico como del uso que se hace de él. Sin embargo, a diferencia del modelo relacional, las bases de datos jerárquicas no tienen controles que impidan la desnormalización de una base de datos. No existe el concepto de campos clave o campos únicos.

En este contexto se inserta la propuesta de modelo relacional, como respuesta a estos inconvenientes.

### 1.1. Definición

Una base de datos relacional es una base de datos en donde todos los datos visibles al usuario están organizados estrictamente como tablas de valores, y en donde todas las operaciones de la base de datos operan sobre estas tablas. Estas bases de datos son presentadas a los usuarios como una colección de relaciones normalizadas de diversos grados que varían con el tiempo. El modelo relacional representa

un sistema de bases de datos en un nivel de abstracción un tanto alejado de los detalles de la máquina subyacente. El modelo relacional puede considerarse como un lenguaje de programación más bien abstracto, orientado de manera específica hacia las aplicaciones de bases de datos. En términos tradicionales una relación se asemeja a un archivo, una tupla a un registro, y un atributo a un campo.

Los sistemas de bases de datos relacionales pueden presentar un inconveniente asociado al proceso de normalización que los caracteriza. Este se evidencia en el siguiente ejemplo relacionado con una empresa comercializadora: supongamos que se cuenta con una tabla de codificadores Producto, una tabla Clientes y una tabla Factura. La primera tabla contiene los productos que oferta la empresa, la segunda los datos relacionados con los clientes de la misma y la última muestra los detalles de las facturas realizadas donde se registra de manera obligatoria el identificador del cliente que efectuó la compra y el identificador del producto que adquirió, además de otros datos asociados a la transacción que no son requeridos en todo momento. El problema de este diseño normalizado se presenta en el momento de actualizar el nombre de un producto presente en múltiples facturas anteriores, a partir de ese momento el producto con el nombre anterior desaparece de todas las facturas y no podrá ser recuperado con su nombre inicial. Esta situación llevada a otra escala afecta a los diseñadores de bases de datos, que se ven obligados a denormalizar los esquemas o buscar variantes de otro tipo para sortear esta limitación. Por otra parte está el conocido problema del Object-Relational Impedance Mismatch, el cual se produce por una incompatibilidad entre el paradigma orientado a objetos y el relacional. El primero está basado en principios de la ingeniería de software y el segundo en principios matemáticos (debido a la existencia del álgebra relacional). Por la diferencia entre las dos tecnologías están son incapaces de alcanzar un acople perfecto. Este problema obliga a los desarrolladores a buscar soluciones alternativas, afectando así la capa de acceso a datos definida en el sistema.

Producto de lo anterior y desde hace unos años ya, los especialistas en gestión de información digital se dieron a la tarea de buscar soluciones particulares al almacenamiento de los datos en sus aplicaciones, donde no estuvieran atados al controvertido modelo relacional. Surgen entonces las variantes no relacionales que no implementan el lenguaje de consultas SQL.

## **2. Bases de datos NoSQL**

La respuesta a la necesidad de gestionar volúmenes masivos de información surge de la base de datos NoSQL, término acuñado a finales de los 90 y que engloba todas las tecnologías de almacenamiento estructurado que no cumplen el esquema relacional.

La cantidad de información manejada por comunidades, redes sociales, buscadores, y muchos otros proyectos en el ámbito de la Web 2.0 es abrumadora, lo que ha hecho que surjan nuevas arquitecturas de almacenamiento de información, que deben ser de alto rendimiento, escalables y distribuidas.

Aunque esta tecnología surgió de unas necesidades muy concretas, su difusión y algunos proyectos para encapsular sus funcionalidades y hacerlas más amigables a desarrolladores acostumbrados a SQL está provocando que también se usen en proyectos de pequeño tamaño, con lo que todo indica que a medio plazo convivirán con las bases de datos tradicionales independientemente del volumen de datos a gestionar.

Dentro de las plataformas NoSQL encontramos varios grupos:

- Basadas en clave/valor. Se almacenan valores asociados a una clave. Son sencillas y las de mayor rendimiento.
- Basadas en documento. Son una particularización de las clave/valor, en las que el valor puede ser un documento. Permiten consultas complejas.
- Basadas en columna. Los valores se almacenan en columnas en lugar de filas. Son útiles cuando se gestionan datos agregados.
- Basadas en grafo. Las relaciones se tratan como un dato más.
- Basadas en objetos. Los datos son objetos y las relaciones punteros entre ellos. Permiten operaciones muy complejas pero suelen tener bajo rendimiento.
- Otras. Cubren necesidades muy específicas y tienen escasa implantación: basadas en tupla, multivaluadas, jerárquicas, etc. [10]

NoSQL (Not Only SQL) realmente es una categoría muy amplia para un grupo de soluciones de persistencia que no siguen el modelo de datos relacional, y que no utilizan SQL como lenguaje de consulta; pero en resumen, las bases de datos NoSQL pueden clasificarse en función de su modelo de datos en las siguientes cuatro categorías:

- Orientadas a clave-valor (Key-Value stores)
- Orientadas a columnas (Wide Column stores)
- Orientadas a documentos (Document stores)
- Orientadas a grafos (Graph databases)

Para los sistemas de clave-valor la unidad más pequeña de modelado es el par clave-valor; mientras que en los sistemas de columnas anchas las tuplas tienen un número variable de atributos. Los sistemas documentales poseen como unidad básica de almacenamiento la definición abstracta de un documento y los sistemas orientados a grafos modelan el conjunto de datos como una gran y densa estructura de nodos en una red. [1]

A continuación se mencionan los diferentes tipos de gestores de datos NoSQL más conocidos y se detallan brevemente los cuatro primeros de estos tipos con algunos elementos de interés. [2]

1. Key Value / Tuple Store
2. Wide Column Store / Column Families
3. Document Store
4. Graph Databases
5. Multimodel Databases

6. Object Databases
7. Grid & Cloud Database Solutions
8. XML Databases
9. Multidimensional Databases
10. Multivalue Database

### 2.1. Key Value / Tuple Store

Los sistemas de clave-valor prometen un rendimiento excelente para volúmenes de datos muy grandes, a cambio de ser muy simples y renunciar a funcionalidades que tenemos en otros sistemas como la verificación intrínseca de la integridad de datos, llaves extranjeras y disparadores. Las validaciones de los datos se delegan completamente en la aplicación cliente, siendo la base de datos, simplemente el lugar donde se guardan los datos. No se verifican integridades, no se comprueban referencias cruzadas, todo esto se ha de implementar a nivel de aplicación, en el código del cliente.

En un sistema relacional existen bases de datos y dentro de cada base de datos tenemos tablas formadas por filas y columnas. En un sistema clave-valor existen contenedores, también se les llama cabinets, en cada contenedor podemos tener tantas parejas de clave-valor como queramos. Hay sistemas que permiten tener claves duplicadas y hay otros que no, o que se puede indicar que no queremos que se dupliquen. En cada contenedor es posible tener datos de la misma naturaleza (por ejemplo productos, pedidos, clientes, etc.) o totalmente diferentes (puede haber un contenedor por cliente), todo depende de los desarrolladores de la aplicación.[3]

En la figura 3 se ilustra la forma de almacenamientos de estos sistemas. A cada clave se le asocia un valor (clave=valor).

<pre>[users.cab] john=myp4ssw0rd sony=4f5h0r8vn0</pre>	<pre>[users_data.cab] john_name=Jonh Clax john_email=jonh@yahoo.com john_country=Canada john_birthdate=04/05/1960 sony_name=Sony Sand sony_email=sony@yahoo.com sony_country=England sony_birthdate=04/05/1948</pre>
--------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Figura 3. Ejemplo donde se utiliza un contenedor para los usuarios y otro para sus datos.**

Esta es una versión simplificada, hay bases de datos clave-valor más sofisticadas, llamadas orientadas a documentos, como MongoDB y CouchDB. Otras tienen estructuras más complejas que permiten almacenar arreglos asociados a una clave; incluso hasta simular tablas con sus columnas.

El almacenamiento de clave-valor proporciona el modelo de datos más sencillo posible. Sin embargo, esto tiene un costo, las consultas de rango no son directas (a menos que el gestor de datos proporcione apoyo explícito), y de modo general resultan muy complejas de ejecutar. [4]

Recientemente las bases de datos orientadas a clave/valor han aumentado en popularidad, en parte gracias a los sistemas basados en la implementación del paradigma Cloud Computing como Amazon SimpleDB o Google BigTable, que proveen a los programadores sistemas de almacenamiento sencillos. Entre las más utilizadas se encuentran las siguientes: DynaDB, Azure Table Storage, Couchbase Server, Riak y Redis.

## 2.2. Wide Column Store / Column Families

Las bases de datos orientadas a columnas son probablemente más conocidas por la aplicación BigTable de Google o por la implementación Cassandra de Apache. A primera vista son muy similares a las bases de datos relacionales, pero en realidad son muy diferentes. Una de las principales diferencias radica en el almacenamiento de datos por filas (sistema relacional) versus el almacenamiento de datos por columnas (sistema orientado a columnas) y otra la optimización de consultas para mejorar los tiempos de respuesta en comparación con los sistemas relacionales.

Las bases de datos orientadas a columnas son en realidad lo que se podría suponer, tablas de datos donde las columnas de valores de datos representan el almacenamiento estructural. Los datos son almacenados como secciones de las columnas de datos en lugar de filas de datos, como en la mayoría de los gestores relacionales. Esto tiene ventajas para los almacenes de datos, sistemas de gestión de relaciones con clientes, catálogos de bibliotecas de tarjetas y otros sistemas ad-hoc de consulta donde los agregados se calculan a través de un gran número de elementos de datos similares. (Figura 4).

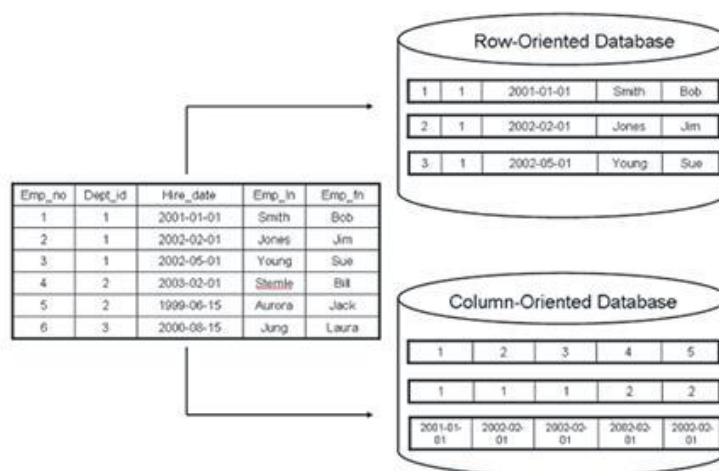


Figura 4. Ilustración de las dos variantes de almacenamiento del tipo de base de datos Wide Column Store.

Los siguientes conceptos son fundamentales para entender cómo funcionan las bases de datos orientadas a columna:

- Familia de Columnas: Las familias de columnas puede contener súper columnas o columnas.
- Súper Columna: Una súper columna es un diccionario, se trata de una columna que contiene otras columnas pero no otras súper columnas.
- Columna: Una columna es una tupla de nombre y el valor.

Las columnas y las súper columnas no son imprescindibles, lo que significa que ocupan exactamente 0 bytes si no tienen un valor almacenado en ellas. Las familias columnas son muy parecidas a una tabla de datos de un sistema relacional, pero a diferencia de esta, solamente es necesario definir el nombre y la forma de ordenamiento de la llave pues no existe un esquema. [5]

Algunos de los beneficios que ofrecen las bases de datos orientadas a columnas son:

- Alto rendimiento en las consultas de agregación (como COUNT, SUM, AVG, MIN, MAX)
- Alta eficiencia en la compresión y distribución de los datos.
- Verdadera escalabilidad y carga rápida de datos para grandes volúmenes de datos.
- Gran accesibilidad por muchas herramientas analíticas de BI de terceros.

Debido a sus capacidades de agregación que calculan un gran número de elementos de datos similares, las bases de datos orientadas a columnas ofrecen ventajas clave para ciertos tipos de sistemas:[6]

- Almacenes de datos e inteligencia de negocios.
- Sistemas de gestión de relaciones de clientes.
- Catálogos de bibliotecas de tarjetas.
- Sistemas de consulta ad-hoc.

Entre los sistemas orientados a columnas más utilizados se encuentran:

- Hadoop / HBase
- Cassandra
- Hypertable
- Accumulo
- Amazon SimpleDB

### **2.3. Document Store**

Una base de datos orientada a documentos está diseñada para gestionar información orientada a documentos o datos semi-estructurados. Este tipo de bases de datos constituye una de las principales categorías de las llamadas bases de datos NoSQL. La popularidad del término "base de datos orientada a documentos" o "almacén de documentos" ha crecido a la par con el uso del término NoSQL en sí. A



diferencia de las conocidas bases de datos relacionales con su definición de "tabla", los sistemas documentales están diseñados entorno a la definición abstracta de un "documento".

Las bases de datos de documentales son consideradas por muchos como un escalón superior ante los simples gestores de llave-valor, puesto que permiten encapsular pares de llave-valor en estructuras más complejas denominadas documentos. Por otra parte no existe un esquema estricto a seguir para definir estos documentos, lo cual simplifica sustancialmente su uso.[7]

Almacenar y recuperar todos los datos relacionados como una sola unidad puede entregar ventajas enormes en el rendimiento y la escalabilidad. De este modo, los gestores de datos no tienen que hacer operaciones complejas como las uniones para encontrar los datos que normalmente están relacionados, ya que todo se encuentra en un mismo lugar. Ver ejemplo en la figura 5.

```
{
  FirstName:"Jonathan",
  Address:"15 Wanamassa Point Road",
  Children:[
    {Name:"Michael",Age:10},
    {Name:"Jennifer", Age:8},
    {Name:"Samantha", Age:5},
    {Name:"Elena", Age:2}
  ]
}
```

**Figura 5. Ejemplo de una definición de un documento que pretende almacenar algunos datos de una persona.**

A excepción de algunas, estas bases de datos generalmente proporcionan sus datos a través de HTTP, almacenan los datos como documentos con la notación de objetos de JavaScript (JSON) y ofrecen diferentes API para varios lenguajes. Los intereses generales son la sencillez, velocidad y escalabilidad. Entre las más utilizadas se encuentran: [8]

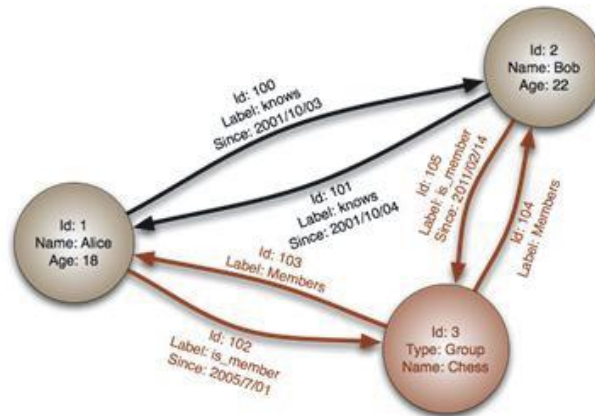
- MongoDB ([mongodb.org](http://mongodb.org))
- CouchDB ([couchdb.apache.org](http://couchdb.apache.org))
- RavenDB ([ravendb.net](http://ravendb.net))

## **2.4. Graph Databases**

Las bases de datos orientadas a grafos representan la información como nodos de un grafo y sus relaciones con las aristas del mismo, de manera que se pueda usar teoría de grafos para recorrer la base de datos ya que esta puede describir atributos de los nodos (entidades) y las aristas (relaciones).

Una base de datos orientada a grafos debe estar absolutamente normalizada, esto quiere decir que cada tabla tendría una sola columna y cada relación tan solo dos, con esto se consigue que cualquier cambio en la estructura de la información tenga un efecto tan solo local.

Hay ocasiones en que se requiere almacenar no solamente datos "sueltos" sino que una parte importante de dichos datos son las relaciones entre ellos como se aprecia en la figura 6:



**Figura 6. Representación del modo de operación de una base de datos de grafo.**

Este tipo de base de datos está diseñada para los datos cuyas relaciones son bien representadas en forma de grafo, o sea, los datos son elementos interconectados con un número no determinado de relaciones entre ellos. La información a gestionar por este tipo de almacenamiento pudiera ser las relaciones sociales, el transporte público, mapas de carreteras o topologías de red, entre otros ejemplos.

Por definición, una base de datos orientada a grafos es cualquier sistema de almacenamiento que permite la adyacencia libre de índice. Esto quiere decir que cada elemento contiene un puntero directo a sus elementos adyacentes por lo cual no es necesario realizar consultas por índices.

A pesar que las estructuras de datos en forma de grafos son normalizables en teoría, incluso en sistemas relacionales, esto tendría serias implicaciones en el rendimiento de las consultas debido a las características de implementación de bases de datos relacionales. En un sistema relacional cada operación sobre una relación resultaría en una operación de unión para el gestor de datos, lo cual es un proceso lento y no escalable ante un creciente número de tupas en estas tablas.

No existe un consenso general sobre la terminología existente en el área de grafos pues hay muchos tipos diferentes de modelos de grafos. Sin embargo, se están realizando algunos esfuerzos para crear el Modelo de Grafo de Propiedad, que unifica la mayoría de las diferentes implementaciones de grafos. De acuerdo con este Modelo, la información en un grafo de propiedad se modela utilizando tres elementos básicos:

- El nodo (vértice)
- La relación (arista) con dirección y tipo (etiquetado y dirigido)
- La propiedad (atributo) en los nodos y en las relaciones

Más específicamente, el modelo propone un multi-grafo etiquetado, dirigido y atribuido. Un grafo etiquetado tiene una etiqueta para cada nodo, que se utiliza para identificar el tipo de ese nodo. Un grafo dirigido permite nodos con una navegación determinada, cada relación define la dirección y el sentido de navegación entre dos nodos. Un grafo atribuido permite una lista variable de atributos para cada nodo y para cada relación, donde un atributo es un valor asociado a un nombre, simplificándose así la estructura del grafo. Un multi-grafo permite múltiples aristas entre dos vértices. Esto significa que dos nodos se pueden conectar varias veces por diferentes aristas, incluso si dos aristas tienen la misma cola, cabeza, y etiqueta.

La teoría de grafos ha sido testigo de una gran utilidad y pertinencia de muchos problemas en disímiles dominios. Los algoritmos de grafos más teóricos aplicados, incluyen varios tipos de cálculos como el camino más corto, rutas geodésicas, medidas de centralidad como PageRank, centralidad del vector propio, cercanía, intermediación, HITS, y muchos otros más. Sin embargo, la aplicación de estos algoritmos, en muchos casos se ha limitado a la investigación, ya que en la práctica no ha habido ningún producto listo para escenarios de alto rendimiento en cuanto a implementaciones de bases de datos orientadas a grafos. Afortunadamente, en los últimos años, esto ha cambiado. Hay varios proyectos que se han desarrollado teniendo en cuenta los grandes escenarios de producción como: [1]

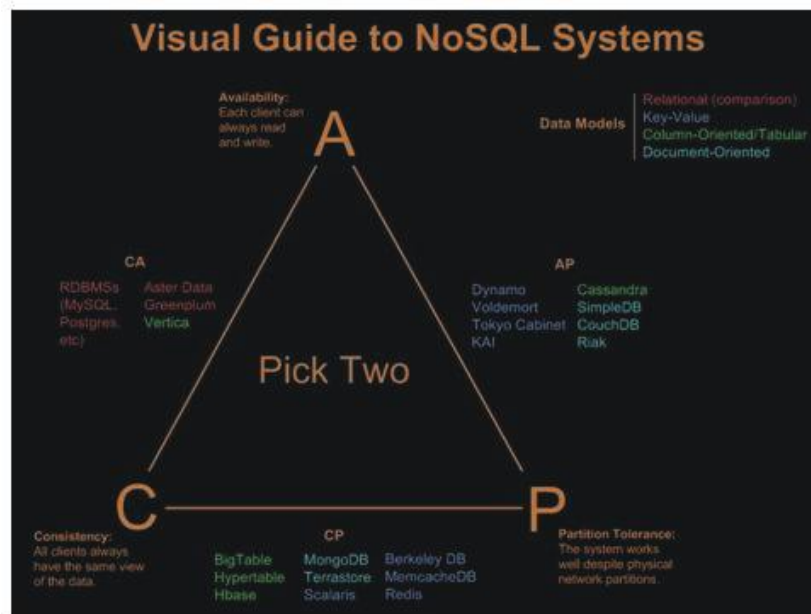
- Neo4j
- Infinite Graph
- InfoGrid
- HyperGraphDB
- DEX
- GraphBase
- Trinity

## CONCLUSIONES

Las bases de datos relacionales están ampliamente extendidas y cuentan con potentes herramientas que facilitan la interacción con las mismas. Existe además, una documentación en línea y una gran comunidad de usuarios entorno a ellas. Numerosas aplicaciones de disimiles propósitos salen al mercado preparadas para integrarse fácilmente con este tipo de bases de datos, como CMS, ITS, entre otras. A diferencia, las bases de datos NoSQL son relativamente incipientes y disponen de pocas herramientas de gestión que no ofrecen muchas de las prestaciones que son encontradas en las herramientas para la gestión de las bases de datos relacionales.

El trabajo con bases de datos de NoSQL requiere, en la mayoría de los casos, conocer bien el negocio que se desea modelar para definir adecuadamente la estructura en la que se van a almacenar los datos. No obstante, este aspecto marca la diferencia en el rendimiento de las consultas, a favor de las NoSQL en comparación con las relacionales. Un esquema de datos bien ajustado a un negocio muy específico permite optimizar los resultados de las consultas desde la etapa de diseño.

Las bases de datos NoSQL ofrecen una alternativa para las bases de datos relacionales, no un reemplazo. Cada una tiene su lugar, y simplemente entregan más alternativas de las cuales podemos elegir una. ¿Pero cómo elegir? Un indicador importante es el teorema de Brewer (o CAP) sobre la coherencia (Consistency), disponibilidad (Availability) y tolerancia a la partición (Partition Tolerance). Plantea que en los sistemas distribuidos solo podemos tener dos de las tres garantías (la C, la A o la P), y por lo tanto es preciso elegir la más importante. Es preciso tener en cuenta que si lo que más importa es la coherencia, entonces se debe optar por una base de datos relacional. El siguiente esquema explica gráficamente el teorema de Brewer:



## Referencias

1. SÁNCHEZ PÉREZ, CARLOS: “Bases de Datos: RDBMS vs No-SQL, una R-Evolución”, disponible en <http://carlossanchezperez.wordpress.com/2011/02/14/bases-de-datos-rdbms-vs-no-sql-una-r-evolucion/>.
2. ANÓNIMO: “Base de datos jerárquica”, disponible en: [http://es.wikipedia.org/wiki/Base\\_de\\_datos\\_jer%C3%A1rquica](http://es.wikipedia.org/wiki/Base_de_datos_jer%C3%A1rquica).
3. ANÓNIMO: “Base de datos de red”, disponible en: [http://es.wikipedia.org/wiki/Base\\_de\\_datos\\_de\\_red](http://es.wikipedia.org/wiki/Base_de_datos_de_red).
4. NEUBAUER, P: “Graph Databases, NOSQL and Neo4j”, disponible en <http://www.infoq.com/articles/graph-nosql-neo4j>.
5. ANÓNIMO: “NoSQL Archive”, disponible en <http://nosql-database.org/>.
6. TALENS, JAH: “Bases de datos clave-valor”, disponible en <http://softinspain.com/desarrollo/bases-de-datos-clave-valor/>.
7. ANÓNIMO: “NoSql Databases Landscape”, disponible en <http://www.vineetgupta.com/2010/01/nosql-databases-part-1-landscape/>.
8. RAHIEN, A: “That No SQL Thing: Column (Family) Databases”, disponible en <http://ayende.com/blog/4500/that-no-sql-thing-column-family-databases>.
9. ANDERSON, D: “Column Oriented Database Technologies”, disponible en <http://dbbest.com/blog/column-oriented-database-technologies/>.
10. STRAUCH, C: “NoSQL Databases”, disponible en <http://www.christof-strauch.de/nosql dbs.pdf>.
11. LERMAN, J: “¿Qué son las bases de datos documentales?”, disponible en <http://msdn.microsoft.com/es-es/magazine/hh547103.aspx>.
12. ZAMUDIO, E: “Neo4J: Base de datos orientada a grafos”, disponible en [http://www.javamexico.org/blogs/ezamudio/neo4j\\_base\\_de\\_datos\\_orientada\\_grafos](http://www.javamexico.org/blogs/ezamudio/neo4j_base_de_datos_orientada_grafos).
13. GARCÍA, JC: “Bases de datos NoSQL y escalabilidad horizontal”, disponible en <http://blog.eltallerdigital.com/2011/03/bases-de-datos-nosql-y-escalabilidad-horizontal/>.