Desde los sistemas de bases de datos SQL hacia NoSQL

Gastón Bruno – 2013 – gaston @ gastonbruno . com . ar

***Abstract*—El modelo de software interactivo tradicional en el cual una persona tiene uso activo de un sistema de información en tiempo real, ha cambiado fundamentalmente a lo largo de los últimos 35 años. Dichos sistemas deben comprender ahora soluciones para poblaciones de usuarios considerablemente mayores, a la vez que tienen la posibilidad de ejecutarse en una infraestructura que ha evolucionado aun más drásticamente en los últimos años.**

Las aplicaciones han cambiado para adaptarse a estas nuevas necesidades y así lo han hecho también las tecnologías de bases de datos. Sin embargo, estos cambios han sido más bien adaptaciones puntuales para cada necesidad de un negocio en particular. Tal es así, que al no haber una solución definida en general para este tipo de situaciones; algunas de aquellas grandes compañías que tuvieron los recursos han optado por desarrollar sus propios sistemas de manejo de información. En este punto surgen las bases de datos NoSQL, para dar un mejor soporte a las aplicaciones modernas. *Este documento explica el origen y desarrollo de esta evolución, dando lugar luego a la profundización de las características de los sistemas de bases de datos NoSQL.*

***Index Terms*— Availability, Distributed databases, Distributed information systems, Internet, Parallel machines, Relational databases**

I. Una nueva necesidad dentro del paradigma tradicional

Aunque desde sus inicios en los años 70s, el modelo de base de datos relacional ha sido el modelo dominante y casi todas las bases de datos han seguido su misma arquitectura básica, con el comienzo del nuevo milenio y el crecimiento de la web surgió la necesidad de proporcionar información procesada a partir de grandes volúmenes de datos.

El término "Base de Datos" se ha convertido en sinónimo de SQL y por mucho tiempo no ha existido ninguna solución alternativa disponible para el almacenamiento de datos.

Las principales compañías de internet, como Google, Amazon, Twitter y Facebook tuvieron que enfrentarse a desafíos con el tratamiento de datos que los RDBMS tradicionales no solucionaban. Estas compañías se dieron cuenta que el rendimiento y sus propiedades de tiempo real eran más importantes que la coherencia, en la que las bases de datos relacionales tradicionales dedicaban una gran cantidad de tiempo de procesamiento.

Como muestra la figura 1 para una aplicación puntual, ha habido diferencias fundamentales en los usuarios, aplicaciones y la infraestructura que yace por debajo; entre los sistemas de software interactivo de los años 1970s y los que son utilizados hoy en día. [1]

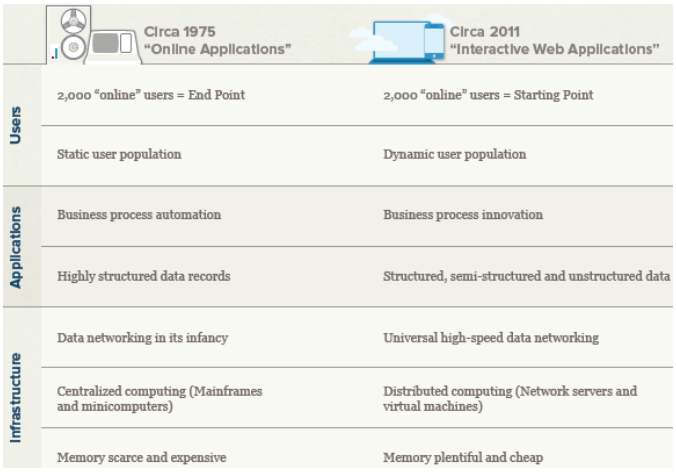

Fig. 1. Relación y límites de la escalabilidad vertical tradicional.

Cuando la mayoría de las bases de datos clásicas fueron diseñadas, el modelo de implementación de hardware predominante consistía en la compra de grandes servidores relacionados a una red de área de almacenamiento (SAN). Las bases de datos eran diseñadas para ejecutarse en un solo equipo, con la responsabilidad de manejar el estado de la consistencia de la información almacenada, administrar los archivos de datos locales y proveer tanto acceso concurrente como pudiera brindarse con las limitaciones del hardware [2].

La figura 2 [1] expone gráficamente esta limitación, mostrando que para soportar una mayor cantidad de usuarios, se necesita un servidor de base de datos más grande; y como resultado de esto, el costo del equipamiento crece exponencialmente, mientras que la cantidad de usuarios crece solamente en forma linear, a la vez que el tiempo de respuesta decrece asintóticamente.

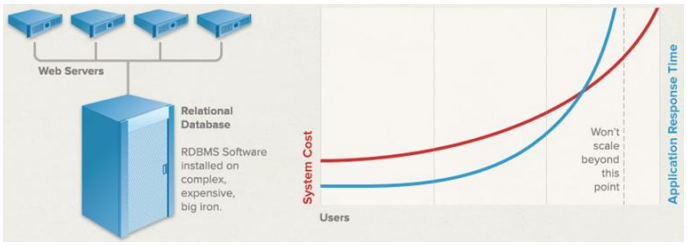

Fig. 2. Relación y límites de la escalabilidad vertical tradicional.

Adicionalmente los RDBMS tradicionales presentan la desventaja de ser rígidos en su esquema, lo que presenta un problema para el soporte a los sistemas de software interactivos actuales. Bajo el esquema tradicional, es necesario definir desde el primer momento y en forma bien clara los tipos de datos que

se van a manejar. Pero las cambiantes necesidades de negocio actuales no pueden prosperar en este esquema, necesitando una opción de manejo y definición de datos más dinámica.
Fueron estos supuestos de arquitectura y diseño de la mayoría de las bases de datos relacionales los que fallaron en alcanzar los requerimientos de escalabilidad, tiempo de respuesta y disponibilidad, en el momento en el cual los mayores sitios de Internet comenzaron a crecer masivamente.

II. Surgimiento de las bases de datos NoSQL como respuesta a la nueva problemática

El concepto de base de datos NoSQL [3] hace referencia a una nueva arquitectura de sistemas de bases de datos que aparece para brindar soluciones a las limitaciones que posee el esquema tradicional, para dar soporte a las necesidades actuales de los modelos de negocio de la internet moderna. Las soluciones NoSQL comenzaron con un conjunto de metas diferentes a las del modelo relacional y evolucionan en un entorno diferente, siendo operacionalmente diferentes, proveyendo mejores soluciones y ajustándose a los requerimientos de almacenamiento de información actual.
Los sistemas de bases de datos NoSQL comenzaron como soluciones in-house a problemas planteados en compañías como Amazon con Dynamo [4], Google con BigTable [5], LinkedIn con Voldemort [6], Twitter con FlockDB [7], Facebook con Cassandra [8] y Yahoo! con PNUTS [9], entre otras. Estas compañías no comenzaron rechazando las tecnologías tradicionales, sino que las utilizaron y encontraron que ellas no satisfacían sus requerimientos. Estas compañías enfrentaron en particular, tres problemas principales: grandes volúmenes de datos que no habían manejado con anterioridad, necesidad de tiempos de respuesta inmediatos para consulta sobre todos esos grandes conjuntos de datos y disponibilidad casi perfecta en un entorno poco fiable.
En un principio, estas compañías intentaron con el enfoque tradicional: agregaron más hardware actualizándose al más veloz disponible. Cuando esta solución no funcionó, se simplificaron los esquemas de las bases de datos, se recurrió a la desnormalización de los modelos, se disminuyó la integridad referencial, se introdujeron varios niveles de consultas preprocesadas y se separaron las bases de datos en réplicas de solo lectura y de escritura dedicada. Estas alternativas extendían la funcionalidad de las bases referenciales, pero no ofrecían una solución definitiva.
Estas compañías modernas requerían una solución que fuera escalable y eficiente. Los ingenieros requerían que las bases de datos escalaran como los servidores web, simplemente adicionando más equipos teniendo como resultado un incremento de performance lineal.
Las bases de datos NoSQL difieren del modelo clásico de bases de datos relacionales en varios aspectos importantes:

- Usan una interface de llamada o protocolo simple en vez de SQL como el principal lenguaje de consultas.
- Los datos almacenados no requieren estructuras fijas como tablas.
- Normalmente no soportan operaciones JOIN.
- No garantizan completamente el esquema ACID.
- Escalan bien horizontalmente.

La figura 3 [1] ilustra el contraste entre el crecimiento no linear del costo total en conjunto con la degradación en forma asintótica de la performance anteriormente vista con las tecnologías de RDBMS y las tecnologías NoSQL que aplanan ambas curvas.

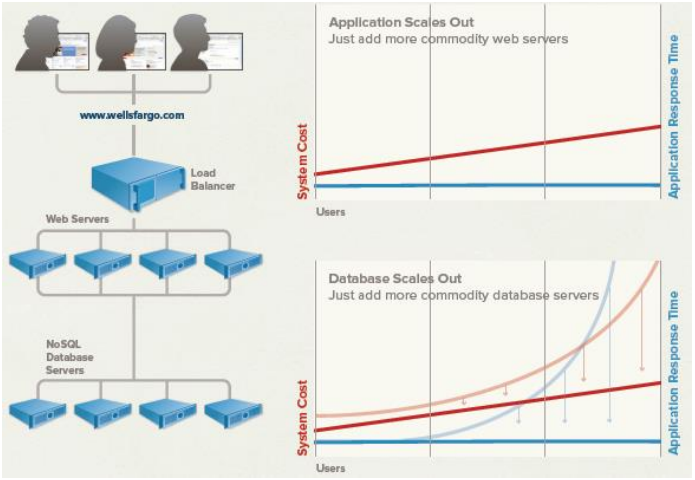

Fig. 3. Escalamiento horizontal de las bases de datos NoSQL.

III. Definición de NoSQL

Muchos de los nuevos sistemas son referidos como NoSQL. La realidad es que en cuanto a la definición de NoSQL, se dice mucho más de lo que no es; que lo que efectivamente es. La definición de NoSQL, que proviene de del inglés “Not Only SQL” o “Not Relational”, no es acordada enteramente todavía [10].
En lo único que coinciden los proveedores de soluciones NoSQL es que el término NoSQL no es perfecto, sino que es más bien pegadizo. La justificación de “Not Only” no se basa en rechazar SQL; sino que al contrario, compensar las limitaciones técnicas que comparten la mayoría de las implementaciones de los RDBMS convencionales.
Lo que sí es cierto acerca de la gran mayoría de los sistemas de bases de datos NoSQL, es que no respetan el modelo relacional estándar de Boyce-Codd [11] en el cual los datos son normalizados a la tercera forma lógica. Estas estructuras de datos suelen requerir operaciones JOIN que demandan recursos intensivos para satisfacer las consultas de los usuarios. Los datos en una base de datos NoSQL son en cambio, desnormalizados y residen en estructuras organizadas en una gran variedad de formatos (columnar, documento, llave/valor, grafo). En aquellos casos en los que los datos son imposibles de almacenar correctamente en un RDBMS o el éste ofrece poca performance siendo accedido de una manera relacional, los sistemas de bases de datos NoSQL se definen por cuán bien manejan esos datos y la velocidad a la que lo hacen.

IV. Características de las bases de datos NoSQL

Mientras que el tipo de implementación puede variar, los sistemas de bases de datos NoSQL comparten algunas características que se desarrollan a continuación.

A. *No necesitan la definición de un esquema*

Los datos pueden ser insertados sin haber definido un esquema previamente. En este sentido, el formato de los datos puede variar en cualquier momento sin interrupción para la aplicación. Esto provee muchísima flexibilidad, que a la vez redunda flexibilidad para el negocio.

B. *Auto-Sharding o elasticidad*

Una base de datos NoSQL distribuye datos automáticamente entre servidores, sin requerir que participen en esto las aplicaciones. Los servidores se pueden agregar o inclusive quitar de la capa de datos sin incurrir en downtime de la aplicación, con los datos y el I/O automáticamente distribuido en los servidores.

Muchas de las bases de datos NoSQL soportan replicación de datos, almacenamiento de múltiples copias dentro de un clúster y hasta inclusive entre centros de datos, asegurando de esta manera alta disponibilidad y soporte para recuperación ante desastres. Un sistema de bases de datos NoSQL bien administrado no debería quedar *offline* por ningún motivo, soportando una operación continua 24x7x365 de las aplicaciones.

C. *Soporte para consultas distribuidas*

La elasticidad anteriormente mencionada, puede en ciertos casos reducir o eliminar la capacidad de realizar consultas complejas de datos. Los sistemas de bases de datos NoSQL mantienen el poder completo de la expresión de una consulta incluso en situaciones de datos distribuidos en cientos o miles de servidores.

D. *Cacheo integrado*

De manera de reducir la latencia e incrementar el *throughput* de datos sostenido, las tecnologías avanzadas de los sistemas de bases de datos NoSQL *cachean* los datos en memoria de manera transparente. Este comportamiento es transparente tanto para los equipos de desarrolladores de las aplicaciones como para los equipos de operaciones; a diferencia de las tecnologías RDBMS donde el cacheo usualmente reside en una capa de infraestructura por separado que debe ser desarrollada, distribuida en distintos servidores y administrada explícitamente por el equipo operativo.

Entre las características más demandadas por las aplicaciones actuales y que han incidido en la creación e incremento en el uso de bases de datos NoSQL se pueden enumerar:

- Gran cantidad de peticiones de lectura y escritura de forma concurrente. La propia complejidad de la lógica detrás del funcionamiento de las bases de datos relacionales, tiende a perder eficiencia en relación al crecimiento de los datos. Esto hace difícil responder con poca latencia en el caso de aplicaciones que atienden un gran número de pedidos a la misma vez.
- Necesidad de almacenar y gestionar eficientemente cantidades masivas de datos. Aplicaciones multimedia, redes sociales o buscadores requieren almacenar y gestionar cantidades de datos continuamente crecientes sin que esto implique una pérdida de eficiencia. Estas son aplicaciones para los cuales los RDBMS en general no están a diseñados.
- Brindar escalabilidad y disponibilidad. Para la gestión de altos volúmenes de datos y para garantizar su disponibilidad, se hacen necesarios sistemas redundantes y fáciles de escalar. Los sistemas NoSQL están diseñados para lograr fácilmente un escalamiento horizontal, a diferencia de los sistemas relacionales tradicionales que escalan en forma vertical.

V. Diferencias en cuanto a la consideración de las transacciones

Tanto los sistemas de bases de datos tradicionales como los NoSQL, manejan un conjunto de operaciones que conforman una unidad de procesamiento para obtener un resultado en particular. Los RDBMS contienen mecanismos para garantizar consistencia en este sentido. Sin embargo, las tecnologías NoSQL han optado por resignar alguna de estas características a cambio de ganar performance y disponibilidad.

A. *El esquema ACID tradicional*

En bases de datos se denomina ACID a un conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción. En este sentido se dice entonces que si un sistema de gestión de bases de datos es ACID *compliant*, quiere decir que éste cuenta con las funcionalidades necesarias para que sus transacciones tengan las características ACID [12].

En concreto, ACID es un acrónimo de las palabras en inglés *Atomicity, Consistency, Isolation* y *Durability*. Dichas palabras en el mismo orden al español: Atomicidad, Consistencia, Aislamiento y Durabilidad.

- Atomicidad es la propiedad que asegura que la operación se ha realizado o no; y por lo tanto ante un fallo del sistema ésta no puede quedar a medias.
- Consistencia se refiere a la integridad. Es la propiedad que asegura que sólo se empieza aquello que se puede finalizar. Por lo tanto se ejecutan aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos.
- Aislamiento es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información sean independientes y no generen ningún tipo de error.
- Durabilidad es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

A medida que se fueron desarrollando las soluciones NoSQL se hizo claro que para alcanzar la escalabilidad deseada, se hacía necesario redefinir algunas de las cualidades ACID; en particular la consistencia y la durabilidad [13].

Obtener consistencia completa en un entorno distribuido requiere de un sistema de comunicación, el cual involucra bloqueos de datos; lo que fuerza a los sistemas a esperar tiempos mayores antes compartir la información que esta siendo actualizada.

B. *El esquema BASE emergente*

La mayoría de las soluciones NoSQL elijen disminuir la consistencia como tal, rebajándola a algo llamado Consistencia Eventual [14]. Esto permite a cada sistema realizar actualizaciones de información y aprender de otras actualizaciones realizadas por los otros sistemas en un período de tiempo corto, sin ser totalmente consistentes en todo momento.

Por otro lado, lograr la durabilidad de los datos siempre ha sido el cuello de botella de los sistemas de base de datos. El motivo de esto es que la escritura a disco reduce la velocidad de la base de datos, siendo que el tiempo de acceso a disco es ampliamente mayor que la escritura en memoria. La mayoría de las aplicaciones permiten balancear la durabilidad en contraposición con el tiempo de espera de las consultas, para satisfacer los requerimientos de la aplicación. Esto puede generar pequeñas ventanas de tiempo en las que es posible perder la información que no ha sido impactada de forma permanente mediante la instrucción COMMIT.

Los sistemas de bases de datos NoSQL no proveen, en general, propiedades transaccionales bajo el modelo ACID. Las actualizaciones son propagadas eventualmente, habiendo garantías limitadas en cuanto a la consistencia de las lecturas. El acrónimo BASE se contrapone al de ACID conformándose de la siguiente manera*: Basically Available, Soft state, Eventually consistent*. La idea es que resignando las limitaciones de ACID, se puede lograr mucha más performance y escalabilidad [15].

C. *Control optimizado de concurrencia*

Otro enfoque es el del control optimizado de concurrencia, que emplea técnicas como las de control de concurrencia multi versión. (Del acrónimo en inglés, MVC). Estas técnicas permiten la lectura consistente de datos en una transacción con concurrencia escribiendo en otra transacción, pero no contemplan los conflictos de lectura y pueden generar más reintentos de transacciones cuando ellas se solapan o tardan mucho tiempo. [16]

Tanto la Consistencia Eventual como el Control de Concurrencia Multi Versión requieren que los programadores piensen en forma diferente en cuanto a la información que manejan en la capa de aplicación. Ambos introducen el potencial que implica que los datos leídos en una transacción quizás no estén completamente actualizados a pesar de que sean consistentes.

Las restricciones de las bases de datos tradicionales generalmente no se encuentran disponibles o son muy básicas en NoSQL por un motivo muy simple: todo lo que puede reducir el tiempo de respuesta del sistema se contrapone a la disponibilidad. Esto implica que son las aplicaciones que utilizan soluciones de almacenamiento de datos de tipo NoSQL, las que deben tener la lógica de control por encima de la capa de base de datos para satisfacer cualquier requerimiento de consistencia adicional.

VI. Teorema CAP / Brewer

En el año 2000, el profesor Eric Brewer de la Universidad de California presento el teorema de CAP, también conocido como teorema de Brewer [17]. Este teorema se refiere a las propiedades de coherencia, disponibilidad y particionamiento en un sistema distribuido. En resumen, afirma que un sistema distribuido no puede garantizar de manera constante estas tres propiedades, sino que en cualquier red de trabajo de datos compartidos, solo se puede satisfacer al mismo tiempo dos de estas tres propiedades.

A partir del teorema CAP es posible establecer un criterio de clasificación tomando en cuenta cuáles son los dos aspectos que satisfacen los sistemas respecto a las tres propiedades deseadas.

A. *DC: Disponibilidad - Consistencia*

Clasifican a la mayoría de los sistemas de bases de datos tradicionales, donde la consistencia y disponibilidad se logran mediante la replicación de los datos sin brindar una buena tolerancia a su particionamiento.

B. *CT: Consistencia – Tolerancia al particionamiento*

En esta variante se logra garantizar la consistencia y que los datos sean almacenados en nodos distribuidos en la red; es decir son tolerantes al particionamiento de datos, pero no son lo suficientemente buenos para garantizar una buena disponibilidad de éstos.

C. *TD: Tolerancia al particionamiento - Disponibilidad*

Este caso se refiere a aquellos sistemas que priorizan garantizar la disponibilidad de los datos y su tolerancia al particionamiento por sobre la consistencia de los datos.

VII. Modelos de datos y acceso NoSQL

Según Greg Burd el modelo de datos relacional con sus tablas, vistas, filas y columnas ha sido muy exitoso y puede ser usado para modelar casi todos los problemas [2]. Empleando restricciones, *triggers*, control de acceso granular y otras funcionalidades, los desarrolladores pueden crear sistemas que fuerzan la estructura e integridad referencial y que aseguren los datos. Para Burd, éstas son todas buenas características, pero tienen un precio. Primero, no existe un solapamiento entre la representación de datos en bases de datos SQL y en los lenguajes de programación; cada acceso requiere una traducción desde y hacia la base de datos. Las soluciones de correlación de Objeto a Relacional [18] (del acrónimo en inglés ORM), permiten en forma transparente; transformar, almacenar y obtener objetos en bases de datos relacionales. Pero por más que funcionen bien, agregan una carga adicional al proceso, haciéndolo más lento. Esto es un impedimento que presenta sobrecarga donde menos se necesita.

En segundo lugar, administrar restricciones globales en un ambiente distribuido es difícil e involucra crear bloqueos para coordinar cambios, de manera tal que estas restricciones se cumplan. Esto introduce sobrecarga en la red y en algunos casos pueden detener el progreso del sistema.

Los sistemas de bases de datos NoSQL han tomado diferentes enfoques. De hecho, las soluciones NoSQL divergen un poco entre sí de la misma manera que lo hacen con respecto a un RDBMS estándar.

Dentro de los sistemas de bases de datos de tipo NoSQL existen diferentes modelos. Algunos de estos modelos de datos más comunes son los que se describen a continuación [19].

A. *Modelos llave-valor*

Este modelo de datos es muy sencillo, a cada llave le corresponde un valor. Aunque su estructura es muy simple, permite velocidades de consulta mayores que las bases de datos relacionales, lo que lo hace muy útil para ser utilizado en bases de datos masivas y que requieran alta concurrencia, etc. Este modelo soporta bien las operaciones de consulta y modificación o basadas en la llave primaria.
Algunos ejemplos de productos que implementan el modelo llave-valor son:

- Cassandra, de Apache [8]
- BigTable, de Google [5]
- Dynamo, de Amazon [4]
- Project Voldemort, de LinkedIn [6]
- Riak [20]
- Redis [21]

B. *Modelos orientados a columnas*

Estos modelos se basan en la utilización del concepto de tabla pero sin las relaciones de asociación. Los datos son separados y almacenados por columnas, cada columna es un índice, todos los datos en una columna tienen el mismo tipo y por lo general para el acceso concurrente, se dividen las consultas de manera que las consultas para una misma tabla sean realizadas por un mismo proceso. En general, las ventajas de este modelo lo hacen más adecuado para aplicaciones que explotan la agregación y para crear almacenes de datos.
Algunos ejemplos de productos que implementan el modelo orientado a columnas son:

- HBase, de Apache [22]
- BigTable, de Google [5]
- Hypertable [23]

C. *Modelos de documentos*

El modelo orientado a documentos es muy parecido en estructura al modelo llave-valor, pero la principal diferencia reside en el tipo de datos que no son simples, sino semánticos; expresados por lo general en formato JSON [24] o a XML [25], [26]. Otra diferencia es que los modelos orientados a documentos suelen utilizar índices secundarios y en el modelo llave-valor esto no es posible.
Algunos ejemplos de productos que implementan el modelo de documentos son:

- CouchDB, de Apache [27]
- MongoDB, de 10gen [28]
- RavenDz, de Hibernating Rhinos [29]
- BaseX [30]
- eXist [31]
- SimpleDB [32]
- IBM Lotus Domino [33]
- Terrastore [34]

D. *Modelos basados en grafos*

En el modelo basado en grafos, la información se almacena partiéndola en trozos más básicos y estableciendo relaciones entre ellas. Este tipo de base de datos es capaz de obtener rendimientos altísimos en consultas sobre información relacionada, como los contactos de un usuario en una red social. En un modelo relacional, estas relaciones implicarían una enorme cantidad de operaciones JOIN; que dependiendo del volumen y la infraestructura, simplemente son muy ineficientes por el enorme consumo de tiempo.
Algunos ejemplos de productos que implementan el modelo basado en grafos son:

- Neo4j [35]
- DEX [36]
- AllegroGraph [37]
- OrientDB [38]
- InfiniteGraph [39]
- Sones GraphDB [40]
- InfoGrid [41]
- HyperGraphDB [42]

VIII. Teorema CAP / Brewer dentro de los diferentes tipos de sistemas NoSQL

La relación existente entre los diferentes modelos de datos de soluciones NoSQL y su clasificación según el teorema CAP se puede visualizar a continuación en la figura 4 [43].

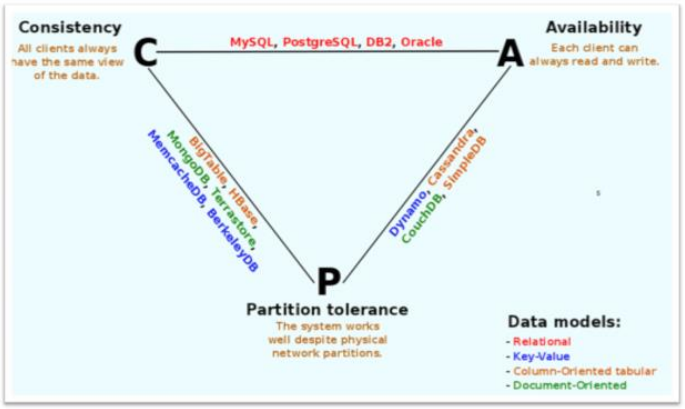

Fig. 4. Guía visual para los sistemas NoSQL

IX. Conclusiones

Es fundamental aclarar antes que nada que los sistemas de bases de datos NoSQL no surgieron para remplazar a los RDBMS tradicionales.
Si bien se han desarrollado algunas funcionalidades de los sistemas de bases de datos NoSQL que presentan ventajas respecto a los RDBMS, los primeros no son adecuados para todos los escenarios. A pesar de que hayan habido grandes compañías que dedicaron recursos e inversión para desarrollar este tipo de sistemas NoSQL, lo han hecho para buscar una solución a una problemática emergente, pero sin poder desligarse de los RDBMS tradicionales. Ambos sistemas de bases de datos conviven en un esquema en el cual se complementan las fortalezas y debilidades de cada una para dar soporte a todo el modelo de negocio.

Por otro lado, no sería adecuado establecer predicciones en este punto, siendo que todavía los sistemas de bases de datos NoSQL se encuentran desarrollándose aún desde su definición. Algunos RDBMS tradicionales incorporaron mucho tiempo antes del surgimiento de los sistemas de bases de datos NoSQL algunas de sus características, como el caso de ORACLE RAC; que permite escalar horizontalmente. Sin embargo, siempre fue dentro del marco de los conceptos de bases de datos relacionales SQL. En este sentido entonces, los sistemas de bases de datos NoSQL presentan una nueva alternativa que requiere una forma de pensar desde las bases a la hora de analizar el negocio, de manera tal de encontrar la correlación en la tecnología que dará soporte éste.

X. Trabajo futuro

El objetivo de este documento ha sido tratar los dos sistemas de bases de datos tradicionales y los NoSQL, sin embargo existen otras alternativas contemporáneas pensadas para soportar las nuevas necesidades que demandan los modelos de negocio involucrados dentro del concepto de "Big Data" [44]. Los sistemas de bases de datos NoSQL son una herramienta tecnológica más que se unen al abanico de opciones de sistemas de bases de datos.

En contraste con lo sucedido con aquellas compañías que se han aventurado a desarrollar su propio sistema de bases de datos en vista de no contar con algo existente que responda a su necesidad; como en toda nueva implementación, es importante también tener en consideración que esto es bastante riesgoso en el sentido que en la mayoría de esos casos se ha optado por dejar todo el valor del negocio en un desarrollo nuevo que no brinda garantías de estabilidad ni soporte, al tratarse de un desarrollo en curso. Esto refleja la necesidad de contar con metodologías, mejores prácticas, marcos de trabajo, guías de implementación, etc., que establezcan un marco conceptual unificado. De esta manera sería posible contar con un conjunto de términos y formas de trabajar estandarizado que ayude a la integración de distintos sistemas y elimine la complejidad que surge al tener distintas fuentes de interpretación.

XI. Referencias

[1] COUCHBASE, «NoSQL Database Technology,» 2012.

[2] G. Burd, «NoSQL,» 2011.

[3] DataStax Corporation, «NoSQL in the Enterprise,» 2011.

[4] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall y W. Vogels, «Dynamo: Amazon's Highly Available Key-value Store,» 2007.

[5] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes y R. E. Gruber, «Bigtable: A Distributed Storage System for Structured Data,» OSDI'06: Seventh Symposium on Operating System Design and Implementation, Seattle, 2006.

[6] Voldemort, «Project Voldemort,» [En línea]. Available: http://www.project-voldemort.com/voldemort/. [Último acceso: 17 7 2013].

[7] R. Pointer, «Introducing FlockDB,» [En línea]. Available: https://blog.twitter.com/2010/introducing-flockdb. [Último acceso: 17 7 2013].

[8] DataStax Corporation, «Introduction to Apache Cassandra,» 2012.

[9] B. F. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H.-A. Jacobsen, N. Puz, D. Weaver y R. Yerneni, «PNUTS: Yahoo!'s Hosted Data Serving Platform».

[10] ORACLE, «Oracle NoSQL Database,» 2011.

[11] C. J. Date y R. Fagin, «Simple Conditions for Guaranteeing Higher Normal Forms in Relational Databases,» 2001.

[12] T. Haerder y A. Reuter, «Principles of transaction-oriented database recovery,» 1983.

[13] N. Leavitt, «Will NoSQL Databases Live Up to Their Promise?,» 2010.

[14] D. Bermbach y S. Tai, «Eventual Consistency: How soon is eventual?,» 2011.

[15] R. Cattell, «Scalable SQL and NoSQL Data Stores,» 2011.

[16] J. Scholz, «Coping with Dynamic, Unstructured Data Sets – NoSQL: a buzzword or a savior?,» 2011.

[17] S. Gilbert y N. Lynch, «Brewer's conjecture and the feasibility of consistent, available, and partition-tolerant web services,» 2002.

[18] hibernate.org, «What is Object/Relational Mapping?,» [En línea]. Available: http://www.hibernate.org/about/orm. [Último acceso: 17 7 2013].

[19] R. Hecht y S. Jablonski, «NoSQL Evaluation,» 2011.

[20] Basho, «What is Riak,» [En línea]. Available: http://docs.basho.com/riak/latest/tutorials/fast-track/What-is-Riak/. [Último acceso: 7 17 2013].

[21] Redis, «FAQ,» [En línea]. Available: http://redis.io/topics/faq. [Último acceso: 17 7 2013].

[22] Apache, «Welcome to Apache HBase,» [En línea]. Available: http://hbase.apache.org/. [Último acceso: 17 7 2013].

[23] Hypertable, «Documentation,» [En línea]. Available: http://hypertable.com/documentation/. [Último acceso: 17 7 2013].

[24] http://www.json.org/, «Introducing JSON,» [En línea]. Available: http://www.json.org/. [Último acceso: 7 17 2013].

[25] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau y J. Cowan, «Extensible markup language (XML) 1.1 (Second Edition),» 2006.

[26] N. Nurseitov, M. Paulson, R. Reynolds y C. Izurieta, «Comparison of JSON and XML Data Interchange Formats: A Case Study,» 2009.

[27] Apache, «Technical Overview,» [En línea]. Available: http://wiki.apache.org/couchdb/Technical%20Overview. [Último acceso: 17 7 2013].

[28] mongodb.org, «MongoDB Fundamentals,» [En línea]. Available: http://docs.mongodb.org/manual/faq/fundamentals/. [Último acceso: 17 7 2013].

[29] Hibernating Rhinos, «RavenDB in a nutshell,» [En línea]. Available: http://ravendb.net/docs/2.0/intro/ravendb-in-a-nutshell. [Último acceso: 17 7 2013].

[30] basex.org, «Getting Started,» [En línea]. Available: http://docs.basex.org/wiki/Getting_Started. [Último acceso: 17 7 2013].

[31] exist-db.org, «Documentation,» [En línea]. Available: http://exist-db.org/exist/apps/doc/. [Último acceso: 17 7 2013].

[32] Amazon, «Documentación de Amazon SimpleDB,» [En línea]. Available: http://aws.amazon.com/es/documentation/simpledb/. [Último acceso: 17 7 2013].

[33] IBM, «IBM Lotus Notes and IBM Lotus Domino 8.5 software,» 2012.

[34] terrastore, «Terrastore Documentation,» [En línea]. Available: https://code.google.com/p/terrastore/wiki/Documentation?tm=6. [Último acceso: 17 7 2013].

[35] Neo4j, «What is Neo4j?,» [En línea]. Available: http://www.neo4j.org/learn/neo4j. [Último acceso: 17 7 2013].

[36] Sparsity Technologies, «DEX Overview,» [En línea]. Available: http://www.sparsity-technologies.com/dex. [Último acceso: 17 7 2013].

[37] Franz Inc., «AllegroGraph 4.11 Introduction,» [En línea]. Available: http://www.franz.com/agraph/support/documentation/current/agraph-introduction.html. [Último acceso: 17 7 2013].

[38] orientdb, «Documentation,» [En línea]. Available: https://code.google.com/p/orient/wiki/Presentations. [Último acceso: 17 7 2013].

[39] Objectivity, Inc., «InfiniteGraph 3.1 Technical Specifications,» [En línea]. Available: http://www.objectivity.com/products/infinitegraph/technical-specifications/. [Último acceso: 17 7 2013].

[40] sones GmbH, «sones Whitepaper GraphDB,» 2010.

[41] InfoGrid, «InfoGrid Documentation Overview,» [En línea]. Available: http://infogrid.org/trac/wiki/Docs/Overview. [Último acceso: 17 7 2013].

[42] B. Iordanov, «HyperGraphDB: A Generalized Graph Database».

[43] M. Stelmach, «Introduction to NOSQL databases,» 7 5 2012. [En línea]. Available: http://stelmach.biz/blog/2012/post2.html. [Último acceso: 17 7 2013].

[44] COUCHBASE, «Why NoSQL?,» 2013.

XII. BIOGRAFÍA

Gastón Bruno nació en Buenos Aires, Argentina, el 01 de enero de 1984. A sus 15 años decidió estudiar y trabajar en SIEMENS, donde obtuvo sus títulos de Bachiller en Bienes y Servicios y de Técnico Electrónico en Telecomunicaciones. Se graduó luego de Licenciado en Informática y actualmente está finalizando una Maestría e IT en la Universidad de Palermo. Su experiencia docente inicia en 2005 en la Facultad de Ingeniería de esta misma Universidad. Se ha especializado en el ámbito de infraestructura IT, para desempeñarse profesionalmente en la actualidad como arquitecto regional de soluciones IT en Hewlett-Packard y como Chief Information Officer en General Plastic Corp. S. A.