¿Qué es NoSQL?

Definición

Las NoSQL son una clase de sistemas de gestión de bases de datos pensadas no para rivalizar con las bases de datos relacionales, sino para ofrecer funcionalidades que estas últimas, por definición, no pueden soportar.

El objetivo es mejorar por software el rendimiento de las consultas a costa de sacrificar ciertas restricciones que imponen los sistemas relacionales tradicionales.

Los RDBMS (sistemas de gestión de bases de datos relacionales) ofrecen la flexibilidad para resolver problemas o afrontar nuevos requisitos de manera sencilla, pero no representan la solución óptima a problemas particulares.

Características de las NoSQL (I)

No relacionales.

- No tienen esquema. La información no se organiza de acuerdo al esquema de una base de datos tradicional. Aunque una NoSQL pueda ser libre de esquema, es necesario seguir modelando los datos, pero la rigidez del modelo relacional desaparece.
- Diferentes lenguajes de consulta (no SQL). No se utiliza SQL como lenguaje de consulta en consecuencia. Se utilizan otros paradigmas:
    - GQL (subconjunto SQL para BigTable)
    - POO
    - Closures (JavaScript)

Ideal para almacenamiento masivo de datos en las que priman las operaciones de lectura (por ejemplo: para búsquedas o presentación de documentos).

Alta velocidad de respuesta a peticiones. Ideales para sistemas de apoyo a la decisión en tiempo real.

Escalabilidad horizontal sencilla.

- Particionamiento automático de datos entre servidores.
- Las lecturas y escrituras son distribuidas en fragmentos.

Distribuidas y tolerantes a fallos.

- Servidores replicados en caso fallo en el servidor principal.
- Al ser distribuidas el cuello de botella es menor.

Se ejecutan en clusters de máquinas baratas, puesto que estos sistemas no requieren apenas de computación, en comparación con los sistemas relacionales.

Requieren de menor mantenimiento que las tradicionales. Tanto porque se ejecutan en máquinas de pocos recursos y porque el modelado y despliegue son más sencillos.

Características de las NoSQL (II)

No cumplen con los requisitos ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad):

- La integridad de los datos no suele estar garantizada. No gestionan bloqueos entre transacciones complejas puesto que esto las haría más lentas.
- Transacciones limitadas a operaciones atómicas.

    - Mucho más rápidas que en las relacionales.
    - Actualizaciones en tiempo real.
    - Algunas BD tienen soporte para versiones y resolución de conflictos.
- Existen proyectos como AppScale o CloudTPS que implementan una capa de middleware para conseguir transacciones ACID distribuidas aparentemente sin perdida en el rendimiento.
- En algunas bases de datos puedes realizar transacciones explicitando que la operación que vas a realizar es "isolated", es decir, que durante la operación estás asegurando que vas a ser la única persona que acceda a los datos hasta su actualización.

No preparan las consultas ni las planifican. Un motor de base de datos relacional traduciría una consulta a un árbol de decisiones, comprobaría si las tablas o vistas a las que se accede son existen o son visibles por el usuario, calcularía un plan de ejecución adecuado y por último ejecutaría la consulta. Estos pasos preliminares en las NoSQL no existen, puesto que se accede directamente a los datos con el consiguiente aumento de rendimiento.

No existen las operaciones JOIN.

- A consecuencia de como se almacenan los datos y a la filosofía de las SQL (operación más costosa).
- No existen claves ajenas. A mayor cantidad de datos mayor es la latencia por las búsquedas a través de claves ajenas.
- Se deben desnormalizar los datos.
    - Es posible reducir el uso de joins, por ejemplo, en las bases de datos de tipo documento. La desnormalización de datos no es un problema puesto que puedes embeber tipos y colecciones de datos en el propio documento, optimizando el espacio utilizado, permitiéndote trabajar con un modelo orientado a objetos.
- Suele ser común duplicar la información modificando su estructura para facilitar las búsquedas por criterios distintos.
    - Suele ser común duplicar la información modificando su estructura para facilitar las búsquedas por criterios distintos.

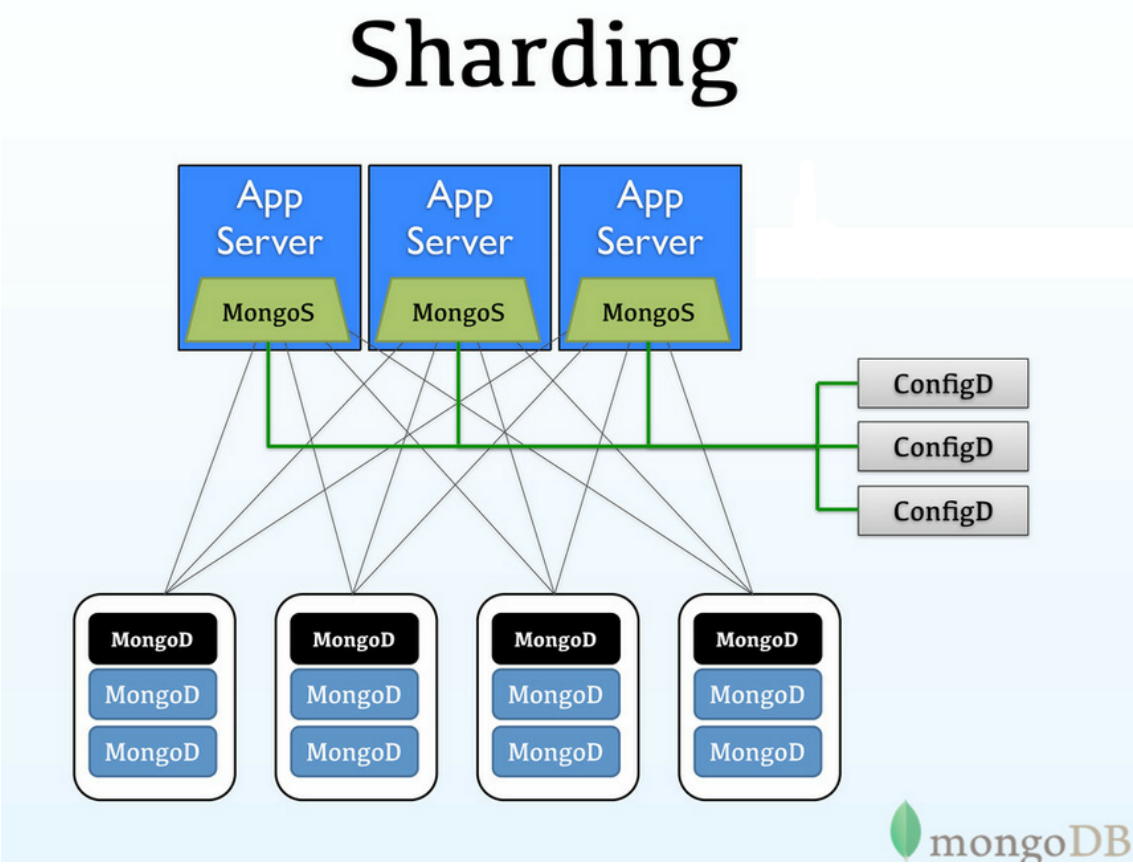
Tampoco existen las operaciones GROUP BY. En su lugar implementar el llamado patrón MapReduce:

- Concepto introducido por Google para dar soporte a la computación paralela sobre grandes colecciones de datos y dar soporte a operaciones de agregación (para por ejemplo, hacer estadísticas).
- Suele ser común duplicar la información modificando su estructura para facilitar las búsquedas por criterios distintos.
- Operación Map
    - Operación que define el conjunto que de datos de entrada que recibirá la operación Reduce.
    - Deben definirse dos parámetros:
        - Clave
            - Debe ser un número o una cadena.
            - Debe actuar de "clave primaria" de nuestra colección.
            - Aquel atributo de cada instancia de nuestra colección sobre el cuál aplicaremos la operación de agregación (Reduce).

        - Valor
            - Conjunto de datos relacionados con nuestra clave que se incluirán en nuestra agregación.
- Operación Reduce
    - Operación a realizar sobre cada par de datos clave-valor.
    - Se suele devolver un valor escalar.
- Operación Filter
    - Restricción que aplicaremos sobre nuestro elemento Clave, para procesar solo aquellos que nos interesen.

Sharding o fragmentación

La fragmentación en las NoSQL se realiza de forma transparente para el administrador. Basta con añadir un nuevo servidor en caliente para seguir generando fragmentos en nuestro cluster de datos. Realizar esto en las bases de datos relacionales resulta más complejo.



La mayoría de las bases de datos NoSQL ofrecen acceso vía RESTful JSON API, lo cual en bases de datos relacionales es poco común. Existen conectores de bases de datos (JDBC, ODBC, etc...) para cualquier lenguaje de programación.

NoSQL como modelos de datos

En el modelo relacional, el modo de consultar los datos va guiada por el modo en el que nuestros datos están estructurados, es decir, por el esquema de nuestra base de datos.

En el modelo NoSQL, no existe esquema en la base de datos. El modo de consultar los datos va guiado por el tipo de operaciones de consulta disponible en nuestra aplicación NoSQL.

Puesto que existen diferentes tipos de SGBD NoSQL, deberemos estructurar nuestros datos

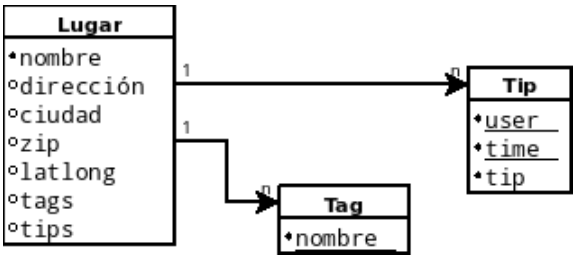
para que se adecuen de la mejor manera al modo que tiene de acceder a ellos nuestro SGBD y/o a los requerimientos solicitados.

Por lo tanto, antes de realizar el “diseño” de nuestro modelo de datos, deberemos estudiar primero el tipo de SGBD NoSQL que vamos a utilizar y asegurarnos de que dicho SGBD cumple con nuestras expectativas.

Cuando hablamos del modelado de datos en el mundo NoSQL las cosas pueden variar un poco. Si se va utilizar una base de datos documental será necesario modelar los datos como documentos, si se va a usar un herramienta clave-valor el diseño será completamente diferente, y lo mismo ocurre con soluciones tabulares y basadas en grafo. Un consejo: cuanto menos recursos tengas para modelar, más difícil será el proceso (por ejemplo, modelar usando clave-valor es algo más complicado que modelar usando documentos).

Tablas vs. Documento

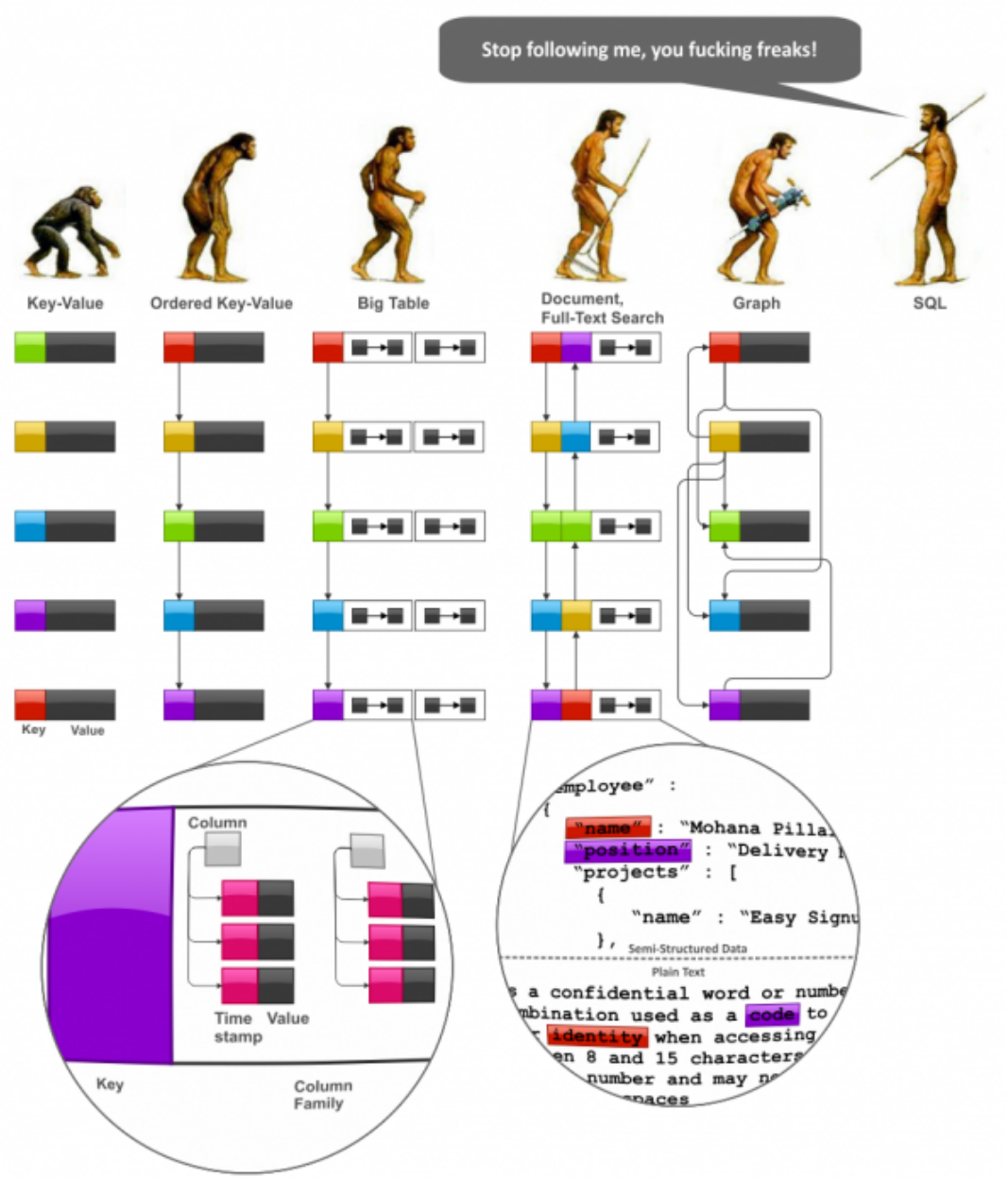
Relacional



NoSQL

```
{
   name: "10gen HQ",
  address: "17 West 18th Street 8th Floor",
   city: "New York",
   zip: "10011",
   latlong: [40.0, 72.0],
   tags: ["business", "cool place"]
   tips: [
      {
         user: "nosh",
         time: 6/26/2010,
            tip: "stop by for office hours on
Wednesdays from 4-6pm"
      },
      {...}
   ]
}
```

Tipos de bases de datos NoSQL



Key-Value: almacenamiento muy simple. Muy potente. Muchas de las técnicas descritas a continuación se basan en esta. No del todo aplicable para casos en los que se requiere procesar rangos de datos. Para ello están las Ordered Key-Values que supera esta limitación, mejorando las capacidades de agregación. Ejemplos: Oracle Coherence, Berkeley DB, Redis, Kyoto Cabinet, LevelDB, Memcached.

BigTable implementa los valores asociados a las claves como colecciones map-of-maps-of-maps, también llamadas columnas con marcas de tiempo. Ejemplos: Apache HBase, Apache Cassandra, Amazon DynamoDB.

Las bases de datos basadas en documento proveen dos significativas mejoras respecto a las BigTable.

Primero, los valores de las claves no están implementados como simples colecciones de matrices dispersas, sino como un conjunto complejo de datos llamado schema.

La segunda es que soporta generación de índices, no sólo para las claves si no también para campos embebidos en el documento. Normalmente los índices son implementados como arboles-B. Ejemplos de tipo documento: MongoDB, CouchDB. Ejemplos de tipo full-text-search: Apache Lucene, Apache Solr.

Las bases de datos basadas en grafo son una evolución de las Ordered Key-Values. Permiten implementar jerarquías de entidades de negocio de forma transparente. Ejemplos: neo4j, FlockDB.

Casos de uso

Aptos en:

- Actualizaciones en tiempo real, pequeñas transacciones pero con un gran tráfico de datos, por ejemplo: redes sociales.
- Búsquedas, gracias a que permiten indexado de gran cantidad de documentos. Por ejemplo: Google.
- Presentación de páginas en sitios que tienen gran tráfico datos o streaming audiovisual. Por ejemplo: Digg.
- Útiles cuando los requisitos de datos no están del todo claros o son cambiantes en el tiempo. Añadir nuevos objetos a una base de datos documental puede ser menos problemático que añadir una nueva restricción en el diseño de una base de datos relacional.

No aptos en:

- Sistemas de alta transaccionalidad, como procesos bancarios o financieros.
- En entornos de data-warehousing que trabajan con datos históricos (en los que la actualización en tiempo real no es un requerimiento), debido a la gran carga SQL que ejecutan los informes de los clientes.
- Cuanto tengamos requerimientos de seguridad, gestión de usuarios, auditoría...