Bases de datos NoSQL. Introducción

Marta Zorrilla – Diego García-Saiz

Enero 2017

Tabla de contenidos

Bibliografía

- Libros
  - Eric Redmond. Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement. 2012
  - Eben Hewitt, Jeff Carpenter. Cassandra: The Definitive Guide, 2nd Edition. 2016.O'Reilly Media, Inc.
  - Kristina Chodorow. MongoDB: The Definitive Guide. 2013. O'Reilly Media, Inc.
  - Ian Robinson, Jim Webber, and Emil Eifrém. Graph Databases 2nd Edition. 2015. O'Reilly Media
  - Harrison et al. Next Generation Databases: NoSQL, NewSQL, and Big Data. 2015. Apress.

La aparición de la tecnología NoSQL

Desde los años 80, la mayoría de los sistemas de información almacenan sus datos en gestores relacionales

- Aplicaciones de banca
- CRM (Costumer Relationship Management)
- ERP (Enterprise Resource Planning)
- Sistemas sanitarios
- Sistemas académicos
- Etc.

Básicamente, estos recogen procesos operacionales que gestionan datos simples y estructurados (p.ej. transacciones bancarias, compras,…)

La aparición de la tecnología NoSQL

Las BD relacionales (R-DBMS) se caracterizan por:

- Utilizar un modelo de datos simple (basado en tablas y relaciones entre tablas)
- Ofrecer herramientas para garantizar la integridad de datos y la consistencia de la información (ACID)
- Utilizar un lenguaje de interrogación estándar, simple y potente
- Proporcionar utilidades para asegurar el acceso, manipulación y la privacidad de los datos
- Ofrecer utilidades para la auditoría y recuperación de datos
- Garantizar la independencia del esquema lógico y físico

Llevan en el mercado más de 40 años!!!

La aparición de la tecnología NoSQL

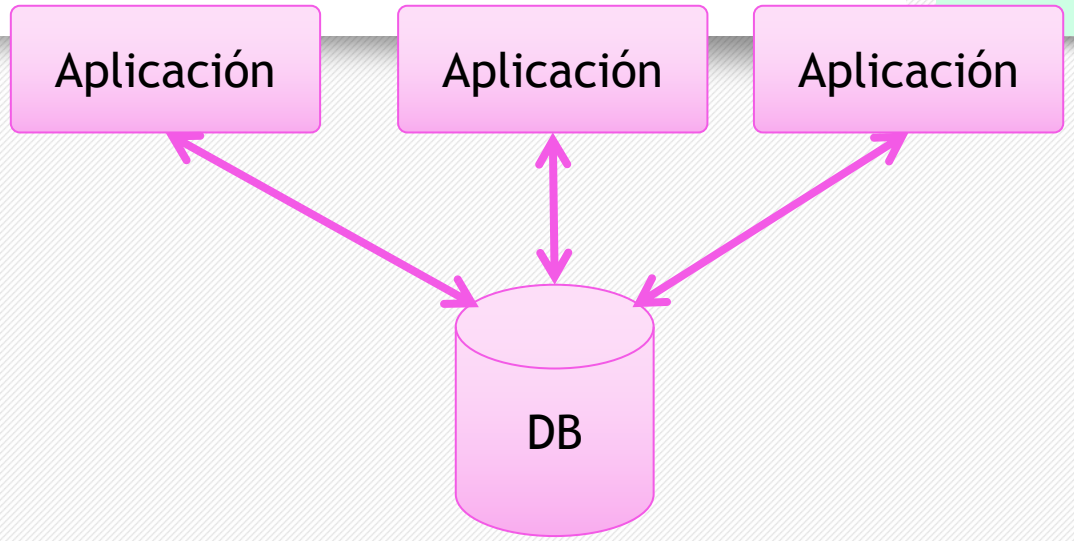
- Pero… los servicios de redes sociales como Facebook, Twitter o Ebay aparecieron; estos necesitaban dar servicio a miles de usuarios concurrentes y responder millones de preguntas diarias y la tecnología relacional no ofrecía ni el nivel de escalabilidad ni el rendimiento adecuado.

- Paralelamente, los avances en virtualización condujeron a la construcción de nodos de computación en la nube (*cloud computing*), que a su vez requería nodos de almacenamiento (*cloud storage*).

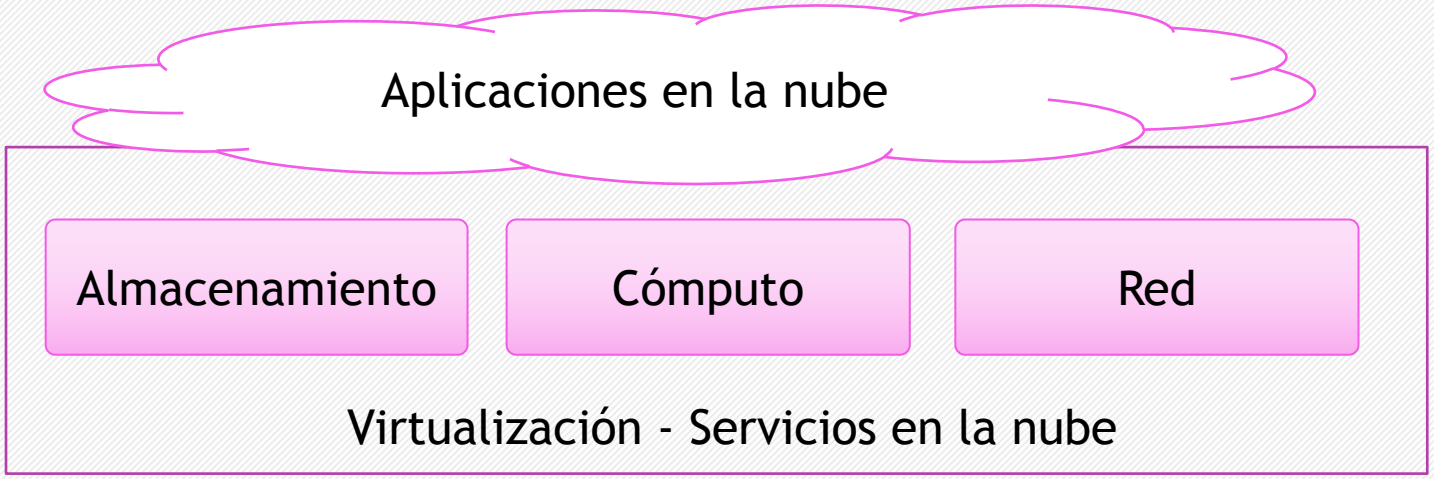
Cambio en la arquitectura de las aplicaciones

M.Zorrilla - D.García

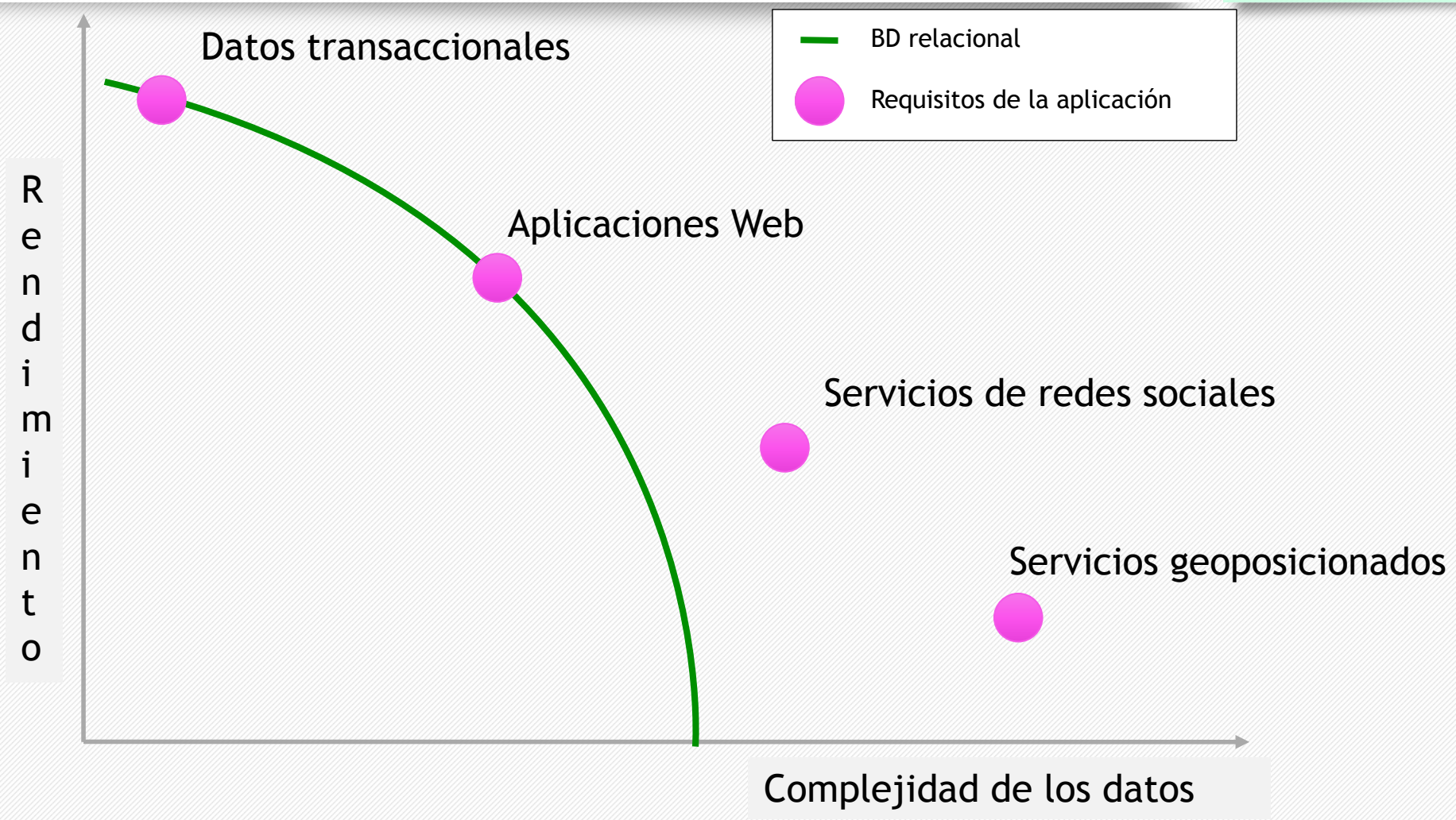
1990's: Integración de bases de datos



2010's: Nube



Cambio en los requisitos de las aplicaciones



El término Big Data

Entendiendo el término BIG DATA: las 3 V's (2013)

- Volume: gran cantidad de datos
- Velocity: necesidad de ser analizados rápidamente
- Variety: datos estructurados y no estructurados, densos y dispersos, conectados y desconectados

Ahora, 7 V's (2016)

- Variability: datos cuyo significado cambia constantemente
- Veracity: veracidad, precisión
- Visualisation: presentar los datos de forma comprensible
- Value: extraer información para la toma de decisiones

¿Por qué los RDBMS no sirven?

- Los RDBMS presentan varios cuellos de botella:
  - Gestión de *Log*: *Redo log*, *Undo log*.
  - Control de concurrencia.
  - Protocolos de transacciones distribuidas.
  - Administración de buffers.
  - Interfaces CLI (JDBC, ODBC, etc.).
- SOLUCIÓN:
  - Implementar modelos alternativos que se ajusten a lo que realmente se necesita.
  - Google, Facebook, Amazon diseñan su solución propia.

El término NoSQL

- NoSQL – es un término utilizado para describir un conjunto de bases de datos que se diferencian de las bases de datos relacionales (*RDBMS*) en los siguientes aspectos:
  - Esquema prescindible, desnormalización, escalan horizontalmente y en sus premisas no está garantizar ACID
- El término fue acuñado por primera vez en 1998 por Carlo Strozzi e instaurado en 2009 por Eric Evans, quien sugiere mejor referirse a esta familia de BDs de nueva generación como "*Big Data*"

NoSQL: principios

- Tanto las bases de datos NoSQL como las relacionales son tipos de **Almacenamiento Estructurado.**
- La principal diferencia radica en cómo guardan los datos (p.ej., el almacenamiento de una factura):
  - En RDBMS separaríamos la información en diferentes tablas (cliente, factura, líneas_factura,…) y luego el aplicativo, ejecutaría el *JOIN* y mapearía esta consulta SQL para mostrárselo al usuario.
  - En NoSQL, simplemente se guarda la factura como una unidad, sin separar los datos.

NoSQL: principios

- ¡¡¡NoSQL **no siempre es la mejor solución**!!!
  - Si tus datos son relacionales, quedarte con tu RDBMS sería la opción correcta.
- NoSQL se basa en 4 principios:
  - El control transaccional ACID no es importante.
  - Los *JOINs* tampoco lo son. En especial los complejos y distribuidos. Se persigue la desnormalización.
  - Algunos elementos relacionales son necesarios y aconsejables: claves (***keys***).
  - Gran capacidad de escalabilidad y de replicación en múltiples servidores.

NoSQL: arquitectura

- En general, proven consistencia débil (*weak consistency*), como p.ej. consistencia eventual, o transacciones restringidas a elementos de datos simples.
- Emplean una arquitectura distribuida (distributed architecture), donde los datos están almacenados de forma redundante en varios servidores. A menudo utilizan tablas *hash* distribuidas.
- Generalmente ofrecen estructuras de datos simples (simple data structures) como *arrays* asociativos o estructuras clave-valor.
- Las consultas se realizan exclusivamente por *key* o *índice*.
- Las consultas complejas se realizan mediante una infraestructura de procesamiento externo tal como *MapReduce*.

Modelos de consistencia: BASE

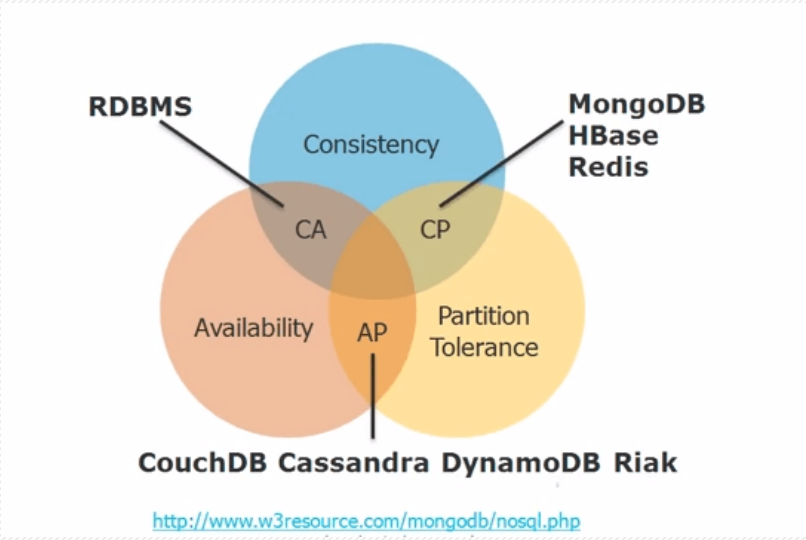
- Las RDBMS nos permiten definir la estructura de un esquema que demanda reglas rígidas y garantizan ACID:
  - ***Atomicity*** - todo o nada
  - ***Consistency*** - coherencia de los datos
  - ***Isolation*** - serialización de transacciones
  - ***Durability*** - los cambios son permanentes
- Pero estas transacciones ACID son más estrictas que lo que el dominio de la aplicación require en muchos casos.
- Por ello NoSQL debilita los requistios de: consistencia inmediata, datos actuales y precisión de respuesta con objeto de ganar escalabilidad → Teorema CAP.
- En vez de usar ACID, se utiliza el término BASE para describir una estrategia de consistencia más optimista.

El teorema CAP

- Teorema de Brewer: “es imposible para un sistema computacional distribuido ofrecer simultáneamente las siguientes tres garantías”:
  - **Consistencia** (*Consistency*)- todos los nodos vean los mismos datos al mismo tiempo
  - **Disponibilidad** *(Availability)* - garantizar que los clientes encuentra una copia de los datos solicitados aunque haya nodos caídos en el cluster.
  - **Tolerancia a la partición** *(Partition)* - el sistema continua funcionando a pesar de la pérdida de mensajes o fallos parciales del sistema.

- Equivalente a:

  “You can have it good, you can have it fast, you can have it cheap: pick two.”



NoSQL: Modelos de consistencia

- ***B****asically* ***A****vailable*, está operativo la mayoría del tiempo
- ***S****oft state*, los datos en las diferentes réplicas no tienen que ser mutuamente consistentes en todo momento.
- ***E****ventually Consistent*, se asegura la consistencia solo después de que pase cierto tiempo.
- En resumen:
  - Consistencia débil – datos obsoletos *OK*
  - Prima la disponibilidad
  - Respuestas aproximadas *OK*
  - Agresivamente optimista, disponibilidad aunque fallen nodos
- ACID vs BASE: http://queue.acm.org/detail.cfm?id=1394128

NoSQL: aplicaciones

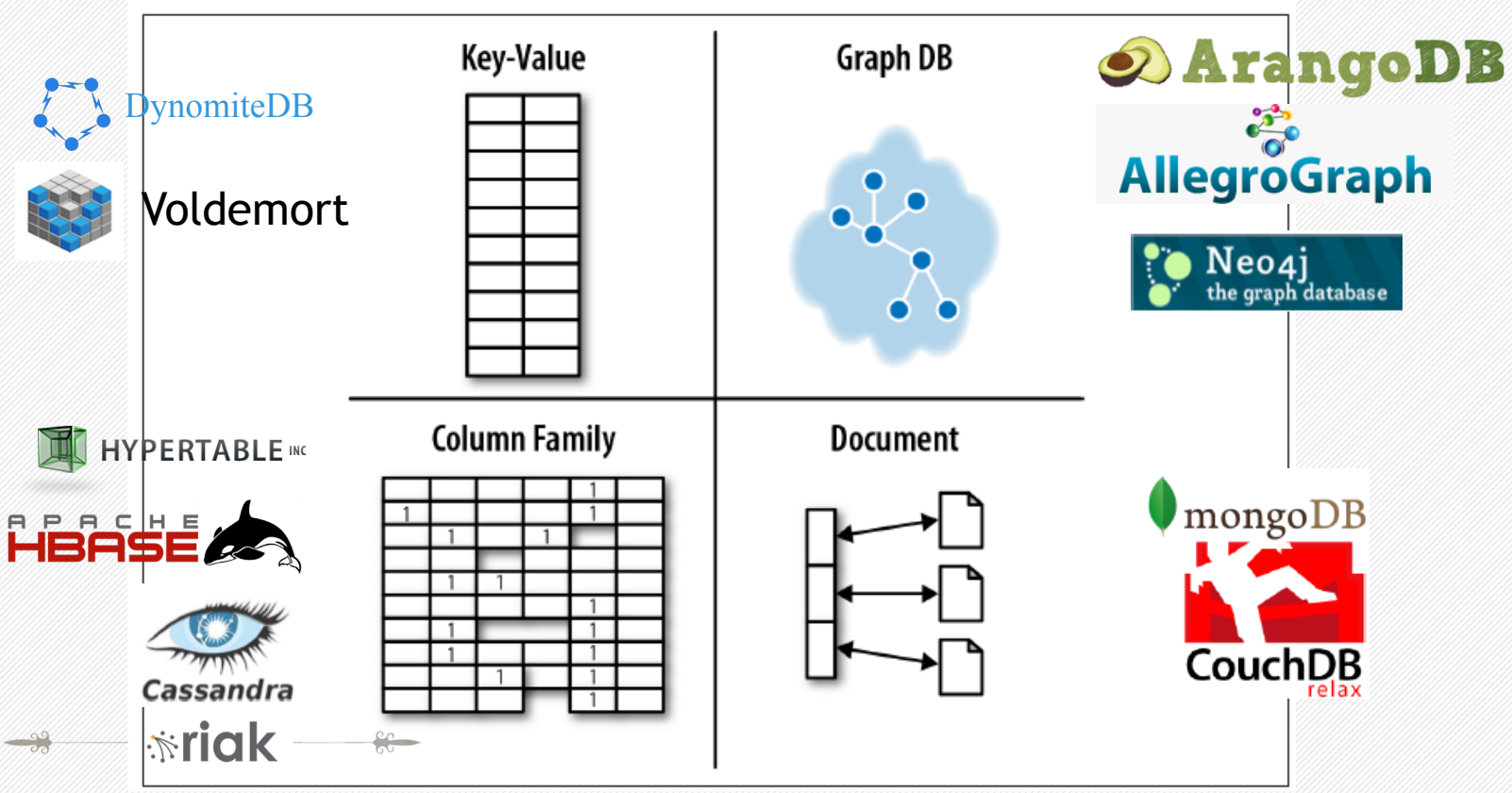
- Aplicaciones más complejas, los desarrolladores deben ser conscientes del comportamiento de las transacciones BASE que soporta la tecnología elegida.
- Deben elegir para cada caso, si aceptan datos inconsistentes o si se configurá el repositorio de datos con consistencia en tiempo de lectura aunque esto suponga un retraso en la respuesta (esto es, verificar que en todas las réplicas se tiene el mismo dato o repararlo si no es así).

NoSQL: aplicaciones

- Su uso es adecuado para aquéllas que:
  - Manejan volúmenes ingentes de datos
  - Tienen una frecuencia alta de accesos de lectura y escritura
  - Con cambios frecuentes en los esquemas de datos
  - Y que no requieren consistencia ACID

- Casos de aplicación:
  - Servicios Web2.0 (redes sociales, blogs, etc.)
  - Aplicaciones IoT
  - Almacenamiento de perfiles sociales
  - Juegos sociales
  - Gestión de contenidos

NoSQL: taxonomía



BD clave-valor: características

- Los datos se almacenan basados en una única clave (*key*), en un *hash-map*
- El valor asociado con la *key* es opaco, un *blob*
- Alto rendimiento para operaciones CRUD básicas pero bajo para atender consultas complejas o manejar datos interconectados
- Operaciones atómicas a nivel de *key*
- Escalabilidad horizontal a través de *sharding*

BD clave-valor: casos de uso

- Adecuada para:
  - Información de sesión en aplicaciones web
  - Perfiles y preferencias de usuario
  - Carritos de la compra

- No aptas para aplicaciones que requieren
  - Datos interconectados (servicios de redes sociales, recomendadores,…)
  - Consistencia para un conjunto de *keys*
  - Búsquedas por la información almacenada en valor

BD documental: características

- Datos almacenados como documentos
- Documentos contienen su propio metadatos
- Similares a las clave-valor pero lo almacenado en valor es visible y se puede consultar
- Esquema flexible
- Operaciones atómicas a nivel de documento
- Escalabilidad horizontal a través de *sharding*

BD documental: casos de uso

- Adecuada para:
  - Blogs
  - *Event logging*
  - Datos operacionales y meta datos para aplicaciones web y móviles
- No aptas para aplicaciones que requieren
  - Consistencia para un conjunto de documentos
  - O si sus datos son densos, entonces encajan mejor en un modelo relacional

Familia de columnas: características

- Los datos se almacenan en familias de columnas; cada fila tiene una *key* y una o varias columnas. Las columnas no tienen que ser las mismas en cada fila.
- Estructura: Mapa multidimensional
- Operaciones atómicas a nivel de *fila (row)*

Familia de columnas: casos de uso

- Adecuada para:
  - Blogs
  - *Event logging*
  - Contadores
  - Tiempo de validez por columna (*TTL, Time-To-Live*)

- No aptas para aplicaciones que requieren
  - Transacciones ACID para operaciones de lectura/escritura

BD grafos: características

- Los datos se almacenan como nodos (vértices) y enlaces (*links - relationships*)
- Están orientadas a analizar las relaciones (caminos mínimos, adyacencias, medidas de centralidad,…) y no a computar agregaciones
- No se recomienda *sharding* ya que atravesar grafos en distintas máquinas es difícil y costoso.
- Escalabilidad vertical
- ACID, similar a las BD relacionales

BD grafos: casos de uso

- Adecuada para:
  - Almacenar datos conectados
  - Aplicaciones espaciales o de rutas
  - Motores de recomendación

- No aptas para aplicaciones que requieren
  - Actualizaciones frecuentes de todos o un subconjunto de propiedades de nodos y enlaces
  - Requisitos de escalabilidad horizontal

NoSQL: resumiendo

- En NoSQL, los datos se recuperan, en general, de forma más rápida que en RDBMS, sin embargo las consultas que se puede realizar son más limitadas. La complejidad se traslada a la aplicación.
- Si tu aplicación require soporte transaccional, debes usar un RDBMS.
- NoSQL no es adecuado para aplicaciones que generen informes con consultas complejas (necesidad de *JOINs*), aunque tienen buena conexión con entornos como MapReduce que permite paralelizar operaciones complejas como agregaciones, filtros, etc..
- La tendencia actual es hacia la combinación de sistemas SQL y NoSQL

NOSQL: ventajas e inconvenientes

- Ventajas
  - Masivamente escalables
  - Alta disponibilidad
  - Bajo coste comparado con soluciones similares de la misma escala
  - (generalmente) elasticidad predecible
  - Esquema flexible, adecuado para datos dispersos y semiestructurados
- Inconvenientes
  - Capacidad de interrogación limitada
  - Consistencia eventual → su programación no es intuitiva
  - No estandarizado → no portabilidad
  - El desarrollo de aplicaciones clientes más complejo
  - Carecen de herramientas de control de acceso (seguridad)