Bases de datos NoSQL

Bases de datos

Tipos de bases de datos

- Una Base de Datos (BBDD) en un conjunto almacenado de información para ser procesada. Los Sistemas Gestores de Bases de Datos son las aplicaciones de software que permiten almacenar y acceder a la información.

- Los principales tipos de BBDD son:
  - Relacionales: la información que almacena la BBDD está relacionada entre sí. Los datos relacionados (registros o filas) son almacenados en tablas que constan de varios campos (columnas).
  - No relacionales: los datos no tienen porqué estar relacionados entre sí y por lo tanto no tienen que almacenarse en estructuras fijas como las tablas del modelo de base de datos relacional.

BD NoSQL

- Las bases de datos NoSQL son bases de datos (BBDD) no relacionales, pensadas para aplicaciones que necesitan baja latencia y modelos flexibles para gestionar grandes volúmenes de datos.
- Incluyen una amplia variedad de tecnologías de BBDD diseñadas para cumplir con los requisitos de desarrollo de las aplicaciones modernas —que generan constantemente enormes cantidades de datos nuevos—.
- La migración de bases de datos relacionales a no relacionales está muy motivada por el crecimiento constante del número de usuarios de Internet y el crecimiento exponencial del número de dispositivos interconectados, entre otras tendencias.

NoSQL

- Aunque estas bases de datos existen desde los años 60, el término «NoSQL» se acuñó a principios del siglo XXI con el crecimiento exponencial de los datos a raíz de la Web 2.0. Ya que el aumento de la cantidad de datos generados por los usuarios en todo el mundo requería nuevas formas de gestionar los datos en las BBDD.
- NoSQL es la abreviatura de «Not only SQL». Es decir, la mayoría de las bases de datos NoSQL no usan el lenguaje SQL para consultas, o lo usan como un lenguaje secundario.
- Las bases de datos no relacionales ofrecen una arquitectura distribuida que permite almacenar información en casos en los que las bases de datos relacionales no son capaces de ofrecer el rendimiento y la escalabilidad necesarios.
  - Al contrario que las BBDD relacionales, no usan estructuras fijas.
  - En su lugar, usan diversas estructuras de datos flexibles, como los pares de clave-valor o los grafos, para el almacenamiento y la recuperación de datos.

NoSQL

- Mientras que las bases de datos relacionales están optimizadas para almacenar datos, reduciendo la redundancia, las BBDD NoSQL están optimizadas para ofrecer escalabilidad horizontal y desarrollo ágil.
- Las bases de datos NoSQL ayudan a los desarrolladores y profesionales de TI a administrar los nuevos desafíos de un abanico de tipos y modelos de datos en constante expansión y son muy eficaces para controlar datos impredecibles, a menudo con velocidades de consulta ultrarrápidas. También permiten realizar una migración de base de datos suave a la nube para las cargas de trabajo NoSQL existentes.

Características

- Características y ventajas (comparado con las BBDD relacionales) de las bases de datos NoSQL:
  - Estructuras de datos flexibles, en lugar de las relaciones tabulares estándar.
    - Pueden administrar datos muy estructurados, pero no están limitadas exclusivamente a modelos de datos fijos, como las bases de datos relacionales (SQL).
  - Baja latencia.
  - Escalabilidad horizontal.
    - A mayor número de nodos mayor rendimiento
  - Gran número de usuarios simultáneos.
  - Optimizado para grandes volúmenes de datos
    - Estructurados, semiestructurados o sin estructurar.
  - Arquitectura distribuida que permite gestionar grandes cantidades de datos.
    - Se almacena la información en más de una máquina del sistema

Características

- Adaptadas a sprints de desarrollo ágil.
  - Con la capacidad de responder a situaciones imprevistas, las bases de datos NoSQL satisfacen los ciclos frecuentes de lanzamiento de software y son adecuadas para un desarrollo de aplicaciones más rápido y ágil.
- Mayor rendimiento, velocidad y escalabilidad.
  - Más eficientes en el procesamiento de los datos que las BBDD relacionales, por eso son la elección para aplicaciones que hacen un uso intensivo de datos ("streaming", etc.).
  - Es posible escalar horizontalmente o agregar servidores más económicos sin tener que realizar ninguna actualización.
  - Pueden escalarse para administrar más datos o tener una única base de datos de gran tamaño dentro de clústeres de servidores muy distribuibles.
- Utilizan lo que se conoce como consistencia eventual
  - Los cambios realizados en los datos serán replicados a todos los nodos del sistema, lo cual aumenta el rendimiento de estos sistemas en contraposición a las propiedades ACID de las BBDD relacionale
    - ("Atomicity, Consistency, Isolation and Durability" – Atomicidad, Consistencia/Integridad, Aislamiento y Durabilidad).

Ventajas y desventajas

- Los Sistemas de Gestión de Bases de Datos NoSQL no contemplan por definición la atomicidad de las instrucciones
  - Cuando una operación sobre los datos consta de varios pasos, no se tienen que ejecutar todos, cosa que sí sucede en los modelos relacionales (transacciones completas).
  - Hay algunas BBDD NoSQL que contemplan la atomicidad.
- Los gestores NoSQL no contemplan obligatoriamente la consistencia o integridad de la BBDD
  - No se comprueba que la operación a ejecutar sobre los datos se pueda completar desde un estado de la Base de Datos válido a otro válido
  - Por ejemplo, la no violación de ninguna restricción de tipos de datos o reglas.

Ventajas y desventajas

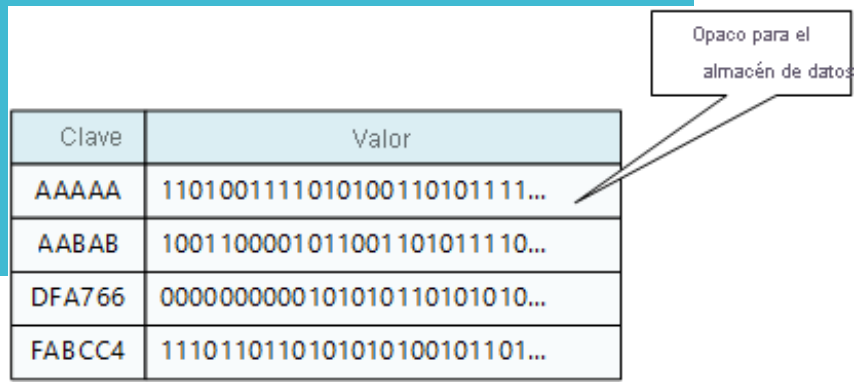
- Al utilizar el mecanismo de consistencia eventual se puede dar el caso de que la misma consulta a diferentes máquinas del sistema produzca resultados diferentes porque las modificaciones de la BBDD aún no han sido replicadas a todos los nodos.
  - Algunas BBDD de este tipo contemplan la propiedad de consistencia.
- Estas BBDD utilizan sus propios lenguajes de consulta de datos y APIs, por lo que no tienen una gran interoperabilidad
  - Dificultad de migraciones de una BBDD a otra, integración con aplicaciones, consultas heredadas en SQL, etc..
- No hay estandarización para este tipo de BBDD, algo que sí es un punto fuerte de las relacionales.
- Las Bases de Datos NoSQL funcionan ampliamente en máquinas Linux, pero no existe en general soporte a otros Sistemas Operativos.
- Las interfaces de gestión de estas BBDD no son intuitivas ni sencillas y en algunos casos carecen de IDEs, gestionándose directamente desde consola de comandos.

¿Cuándo no se recomienda una BD NoSQL?

- Cuando se necesita una BBDD para una aplicación que hace una consulta/lectura intensiva de grandes cantidades de datos.
- Cuando no hay la necesidad de que los datos sean consistentes.
- Si los datos a almacenar no tienen una estructura fija.
- Una misma aplicación puede usar una BBDD relacional y una BBDD NoSQL y guardar cosas diferentes en cada una de ellas.

Tipos

- Bases de datos de clave-valor
  - Es básicamente una tabla hash grande
    - Puede asociar cada valor de datos con una clave única y el almacén clave/valor usará esta clave para almacenar los datos mediante el uso de una función hash adecuada. La función hash se selecciona para proporcionar una distribución uniforme de claves hash en el almacenamiento de datos.
  - Las BBDD de clave-valor destacan por ser muy eficientes tanto para lectura como para escritura.
  - Los datos suelen estar almacenados como objetos binarios (blob)
  - Muy optimizados para aplicaciones que realizan búsquedas simples mediante el valor de la clave, o por un intervalo de claves, pero son menos adecuados para aquellos sistemas que necesitan consultar datos en diferentes tablas de claves/valores, como combinar datos de varias tablas.
  - Ejemplos de bases de datos de clave-valor: Redis, CouchDB, BigTable, Riak, Dynamo, Voldemort, Amazon WS.



Tipos

- Bases de datos en columna
  - Ofrecen una solución híbrida entre los sistemas de bases de datos (DBMS) tabulares y los de clave-valor.
  - Organiza los datos en columnas y filas.
    - En su forma más simple, un almacén de datos de familias de columnas se puede parecer mucho a una base de datos relacional, al menos desde el punto de vista conceptual.
    - La potencia real de una base de datos de la familia de columnas radica en su enfoque desnormalizado para la estructuración de datos dispersos, que viene del enfoque orientado hacia las columnas para el almacenamiento de datos.
  - Las columnas se dividen en grupos conocidos como familias de columnas.
    - Cada familia de columnas contiene un conjunto de columnas que están relacionadas de forma lógica y que normalmente se recuperan o se manipulan como una unidad.
  - Acelera las consultas al almacenar los datos en columnas.
  - Ejemplos de bases de datos tabulares: HBase, BigTable, Cassandra, Hypertable, SimpleDB, Scylla.

CustomerID	Column Family: Identity
001	**First name:** Mu Bae **Last name:** Min
002	**First name:** Francisco **Last name:** Vila Nova **Suffix:** Jr.
003	**First name:** Lena **Last name:** Adamcyz **Title:** Dr.

CustomerID	Column Family: Contact Info
001	**Phone number:** 555-0100 **Email:** someone@example.com
002	**Email:** vilanova@contoso.com
003	**Phone number:** 555-0120

Tipos

- Bases de datos documentales
  - Las BBDD documentales o bases de datos orientadas a documentos son las bases de datos NoSQL más versátiles y se pueden usar en una amplia gama de proyectos.
  - Administra un conjunto de campos de cadena con nombre y valores de datos de objeto en una entidad que se conoce como un documento.
  - Permiten realizar consultas más avanzadas sobre el contenido de un documento, además de consultas de clave-valor.
  - Se apoyan en estructuras simples como JSON, XML, YAML, BSON…
    - Cada valor del campo puede ser un elemento escalar, como un número, o un elemento compuesto

Key	Document
1001	{ "CustomerID": 99, "OrderItems": [{ "ProductID": 2010, "Quantity": 2, "Cost": 520 }, { "ProductID": 4365, "Quantity": 1, "Cost": 18 }], "OrderDate": "04/01/2017" }
1002	{ "CustomerID": 220, "OrderItems": [{ "ProductID": 1285, "Quantity": 1, "Cost": 120 }], "OrderDate": "05/08/2017" }

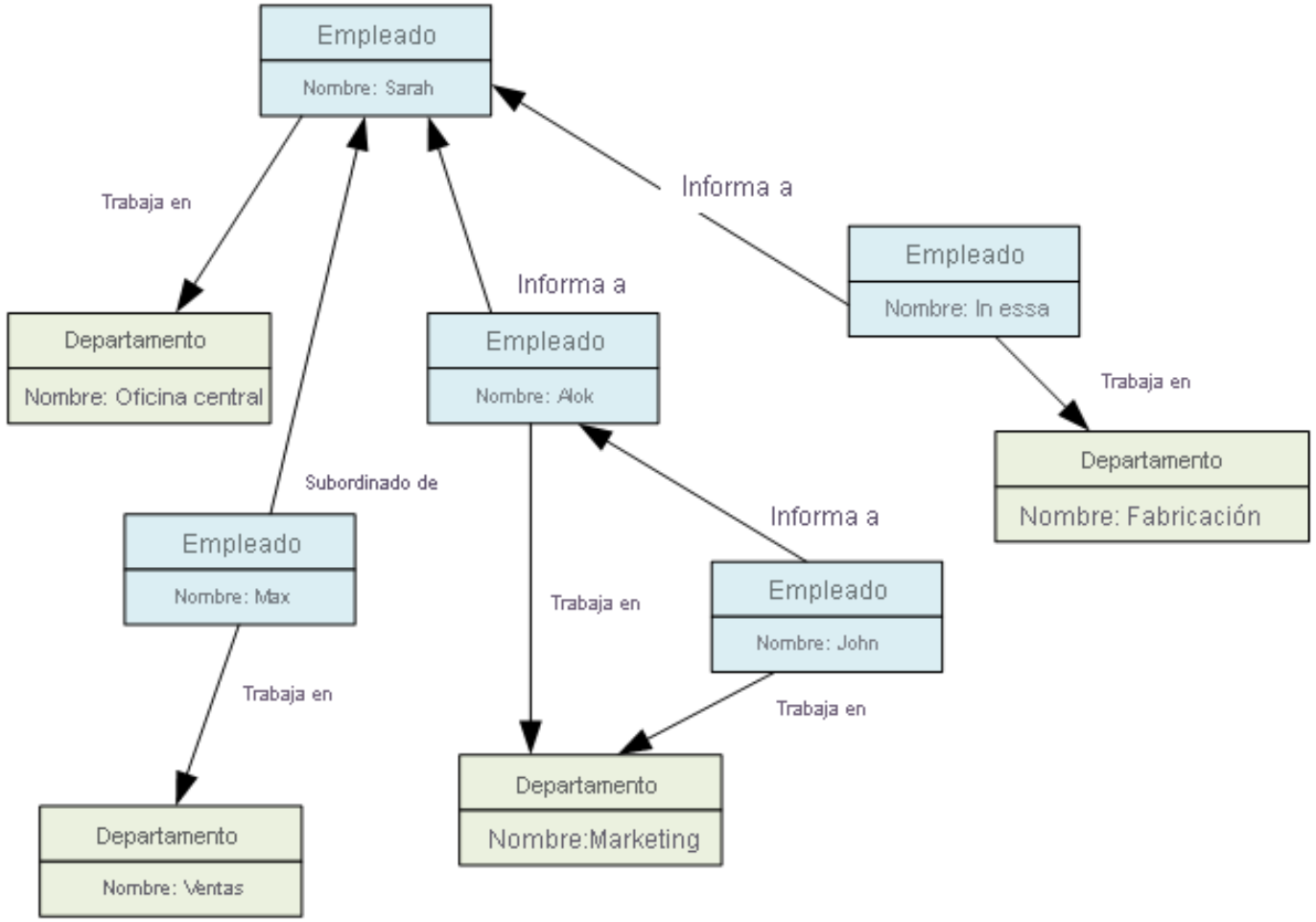
- Bases de datos documentales
  - Normalmente, un documento contiene todos los datos de una entidad.
  - Los elementos que constituyen una entidad son específicos de la aplicación.
  - Un solo documento puede contener información que se puede distribuir a través de varias tablas relacionales de un sistema de administración de bases de datos relacionales (RDBMS).
  - Un almacén de documentos no requiere que todos los documentos tengan la misma estructura.
    - Este enfoque de forma libre proporciona gran flexibilidad.
  - La aplicación puede recuperar documentos mediante el uso de la clave del documento.
    - La clave es un identificador único del documento, al que a menudo se aplica un algoritmo hash para ayudar a distribuir los datos uniformemente.
    - Algunas bases de datos de documentos crean automáticamente la clave del documento. Otras le permiten especificar un atributo del documento para usarlo como clave.
  - La aplicación también puede consultar documentos según el valor de uno o más campos.
    - Algunas bases de datos de documentos admiten la indexación para facilitar la búsqueda rápida de documentos basándose en uno o más campos indexados.
  - Ejemplos de bases de datos documentales: Elastichsearch, MongoDB y CouchDB.

Tipos

- Bases de datos orientadas a grafos
  - Las BBDD orientadas a grafos o bases de datos en grafo ofrecen una experiencia de búsqueda más eficiente entre relaciones, respecto a las bases de datos relacionales.
  - Este modelo usa estructuras de grafos para consultas semánticas y representa los datos como nodos, bordes y propiedades.
    - Los nodos representan entidades y los bordes especifican las relaciones entre estas entidades.
    - Los nodos y los bordes pueden tener propiedades que proporcionan información acerca de ese nodo o borde de forma parecida a las columnas de una tabla.
    - Los bordes también pueden tener una dirección que indica la naturaleza de la relación.
  - Proporcionan un lenguaje de consulta que puede usar para recorrer una red de relaciones de forma eficaz.
  - Ejemplos de bases de datos orientadas a grafos: FlockDB, InfiniteGraph, Neo4J, OrientDB y OpenLink Virtuoso.

Tipos

- Bases de datos orientadas a grafos



Tipos

- Almacenes de datos de series temporales
  - Conjunto de valores organizados por tiempo, y un almacén de datos de está optimizado para este tipo de datos.
  - Soportan un número muy elevado de operaciones de escritura
    - Suelen recopilar grandes cantidades de datos en tiempo real de un gran número de orígenes.
  - Están optimizados para almacenar datos de telemetría. Los escenarios incluyen sensores de IoT o contadores de sistemas y aplicaciones.
  - Las actualizaciones son poco frecuentes y las eliminaciones se realizan a menudo como operaciones masivas.
  - Aunque los registros que se escriben en una base de datos de series temporales suelen ser pequeños, a menudo hay un gran número de registros y el tamaño total de los datos puede crecer rápidamente.
  - Ejemplos: OpenTSDB, Influx, TimeScale, Prometheus

timestamp	deviceid	value
2017-01-05T08:00:00.123	1	90.0
2017-01-05T08:00:01.225	2	75.0
2017-01-05T08:01:01.525	2	78.0

Tipos

- Bases de datos orientadas a objetos
  - Las BBDD orientadas a objetos combinan las capacidades de las bases de datos con las de los lenguajes de programación orientados a objetos. Los datos se representan en forma de objetos.
  - Optimizados para almacenar y recuperar objetos binarios grandes o blobs como imágenes, archivos, transmisiones de vídeo y audio, objetos de datos de aplicación de gran tamaño, documentos e imágenes de disco de máquina virtual.
    - Un objeto se compone de los datos almacenados, algunos metadatos y un identificador único para acceder a él.
    - Los almacenes de objetos están diseñados para admitir archivos que son individualmente muy grandes, así como para proporcionar grandes cantidad de almacenamiento total para administrar todos los archivos.
  - Ejemplos de bases de datos orientadas a objetos: ObjectDB, Db4o, GemStone/S, Versant, Object Storm y Realm.

path	blob	metadata
/delays/2017/06/01/flights.cscv	0XAABBCCDDEEF...	{created: 2017-06-02}
/delays/2017/06/02/flights.cscv	0XAADDCCDDEEF...	{created: 2017-06-03}
/delays/2017/06/03/flights.cscv	0XAEBBDEDDEEF...	{created: 2017-06-03}

Tipos

- Almacenes de datos de índice externo
  - Proporcionan la capacidad de buscar información que se encuentra en otros almacenes de datos y servicios.
  - Un índice externo actúa como un índice secundario de cualquier almacén de datos y puede utilizarse para indexar grandes volúmenes de datos y proporcionar acceso casi en tiempo real acceso a ellos.
  - Los almacenes de datos de índice externo a menudo se utilizan para admitir la búsqueda de texto completo y basada en web.

- Ejemplos de bases de datos orientadas a objetos: Azure Search.

id	search-document
233358	{"name": "Pacific Crest National Scenic Trail", "county": "San Diego", "elevation":1294, "location": {"type": "Point", "coordinates": [–120.802102,49.00021]}}
801970	{"name": "Lewis and Clark National Historic Trail", "county": "Richland", "elevation":584, "location": {"type": "Point", "coordinates": [–104.8546903,48.1264084]}}
1144102	{"name": "Intake Trail", "county": "Umatilla", "elevation":1076, "location": {"type": "Point", "coordinates": [–118.0468873,45.9981939]}}

Beneficios BD relacionales

- Persistencia de datos. Una de las misiones más importantes de las bases de datos es mantener cantidades enormes de datos persistentes.
- Concurrencia. En general, en las aplicaciones normales suele haber muchas personas que están trabajando sobre los mismos dados, y puede que estén modificando los mismos datos.
- Integración. Con frecuencia, las aplicaciones requieren interaccionar con otras aplicaciones de forma que compartan los mismos datos, y unas y otras utilizan los datos que han sido modificados por el resto de las aplicaciones.
- Modelo estándar. Las bases de datos relacionales han tenido éxito debido a que proporcionan los beneficios clave de constituir un sistema estándar.

Problemas BD relacionales

- Impedancia: Diferencia entre el modelo relacional y las estructuras de datos en memoria.
  - El modelo relacional organiza los datos en una estructura de tablas y filas (también denominado relaciones y tuplas).
- En el modelo relacional, una tupla es un conjunto de pares nombre-valor y una relación es un conjunto de tuplas. Todas las operaciones en SQL consumen y retornan relaciones de acuerdo con el algebra relacional.
  - Los valores en una tupla relacional tienen que ser simples (no pueden tener estructura tal como un registro anidado o una lista).
- Esta limitación no se da en las estructuras de datos en memoria, donde existe un conjunto de tipos de estructuras mucho más ricas que las relaciones.
  - Como consecuencia, si se quiere usar una estructura de datos más compleja que la relación, hay que traducirla a una representación relacional para poderla almacenar en disco.
- Este problema llevó a pensar que las bases de datos relacionales serían sustituidas por bases de datos que replicaran las estructuras de datos en memoria.

Problemas BD relacionales

- Integración en un clúster. Un clúster es un conjunto de máquinas que permite implementar soluciones para escalar una aplicación (escalado horizontal).
- Las BD relacionales no están diseñadas para ejecutarse sobre un clúster.
  - Las bases de datos relacionales podrían ejecutarse en servidores separados para diferentes conjuntos de datos, implementando lo que se denomina «una solución de **sharding** de la base de datos».
  - Sin embargo, esta solución plantea el problema de que todo el sharding tiene que ser controlado por la aplicación, por lo que deberá mantener el control sobre qué servidor debe ser preguntado para cada tipo de datos.
  - Normalmente las bases de datos relacionales están pensadas para ejecutarse en un único servidor, de manera que si se tienen que ejecutarse en un clúster requieren de varias instancias de la base de datos

Ventajas BD NoSQL

- Productividad del desarrollo de aplicaciones. En los desarrollos demuchas aplicaciones se invierte un gran esfuerzo en realizar mapeos entre las estructuras de datos que se usan en memoria y el modelo relacional.
- Datos a gran escala. Las organizaciones han encontrado muy valiosa la posibilidad de tener muchos datos y procesarlos rápidamente.

Características BD NoSQL

- No usan SQL como lenguaje de consultas, sin embargo, algunas

de ellas utilizan lenguajes de consultas similares a SQL, tales como CQL en Cassandra.

- En general se trata de proyectos de código abierto.
- Muchas de las bases de datos NoSQL nacieron por la necesidad de ejecutarse sobre clúster, lo que ha influido sobre su modelo de datos como su aproximación acerca de la consistencia.
- Las bases de datos NoSQL no tienen un esquema fijo.
  - Muy útil cuando se trata con campos de datos personalizados y no uniformes.
  - Sin embargo, aunque la base de datos no tenga esquemas, existe un esquema implícito, que es el conjunto de suposiciones acerca de la estructura de los datos en el código que manipula estos datos.

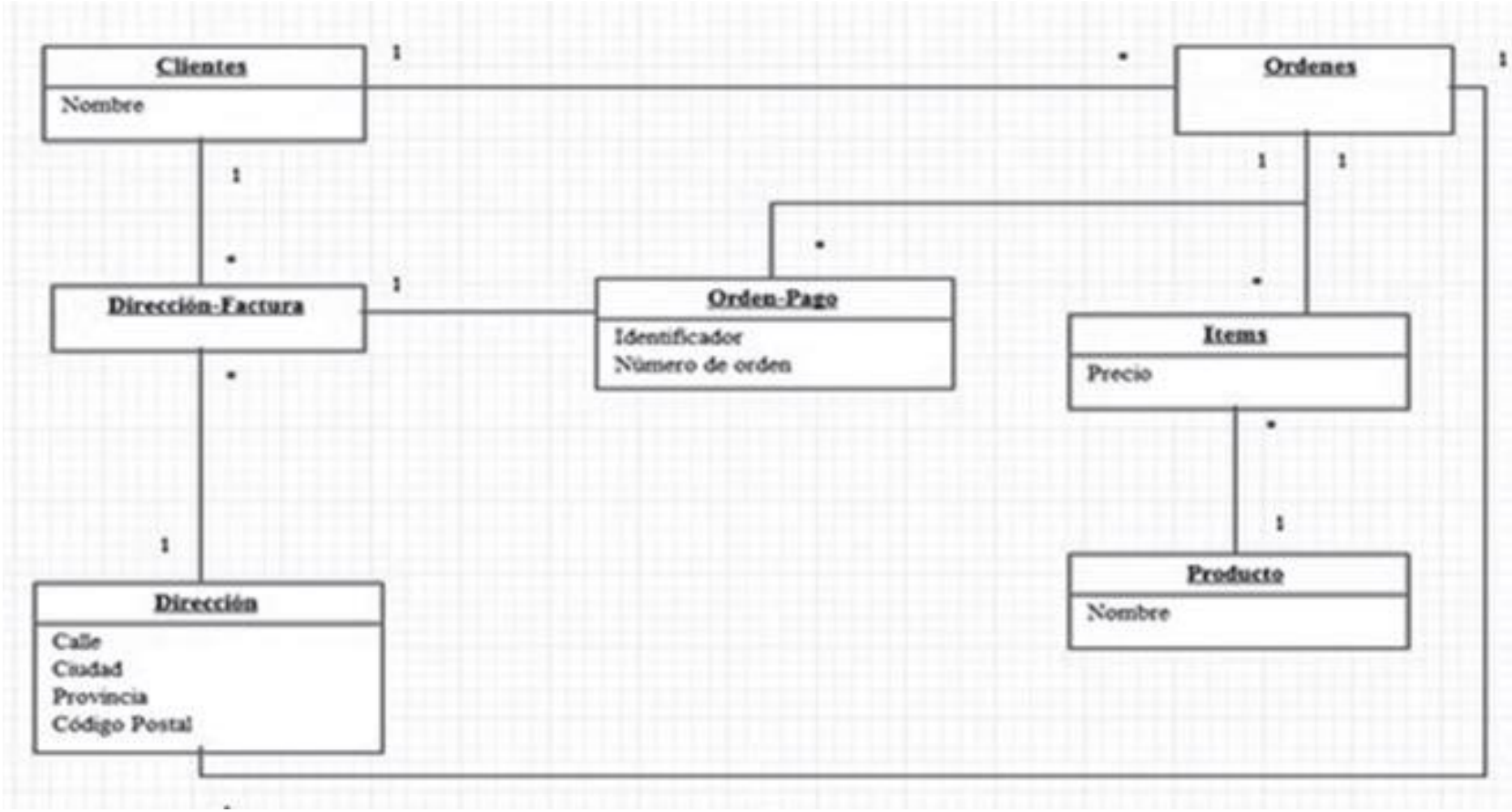
Agregados

- Las familias de bases de datos NoSQL clave-valor, orientadas a documentos y basadas en columnas comparten una característica común en sus modelos de datos: modelos orientados hacia **agregados**.
- En el modelo relacional se toma la información que se quiere almacenar y se divide en tuplas (filas).
  - Una tupla es una estructura de datos limitada, dado que captura un conjunto de valores de manera que no se puede anidar una tupla dentro de otra para conseguir registros anidados, ni se puede poner una lista de valores o tuplas dentro de otra.
- En la orientación agregada hay un cambio de aproximación, y se parte del hecho de que a veces interesa operar con estructuras más complejas que un conjunto de tuplas tales como un registro complejo que puede contener listas y otras estructuras de registro que están anidadas dentro del mismo.

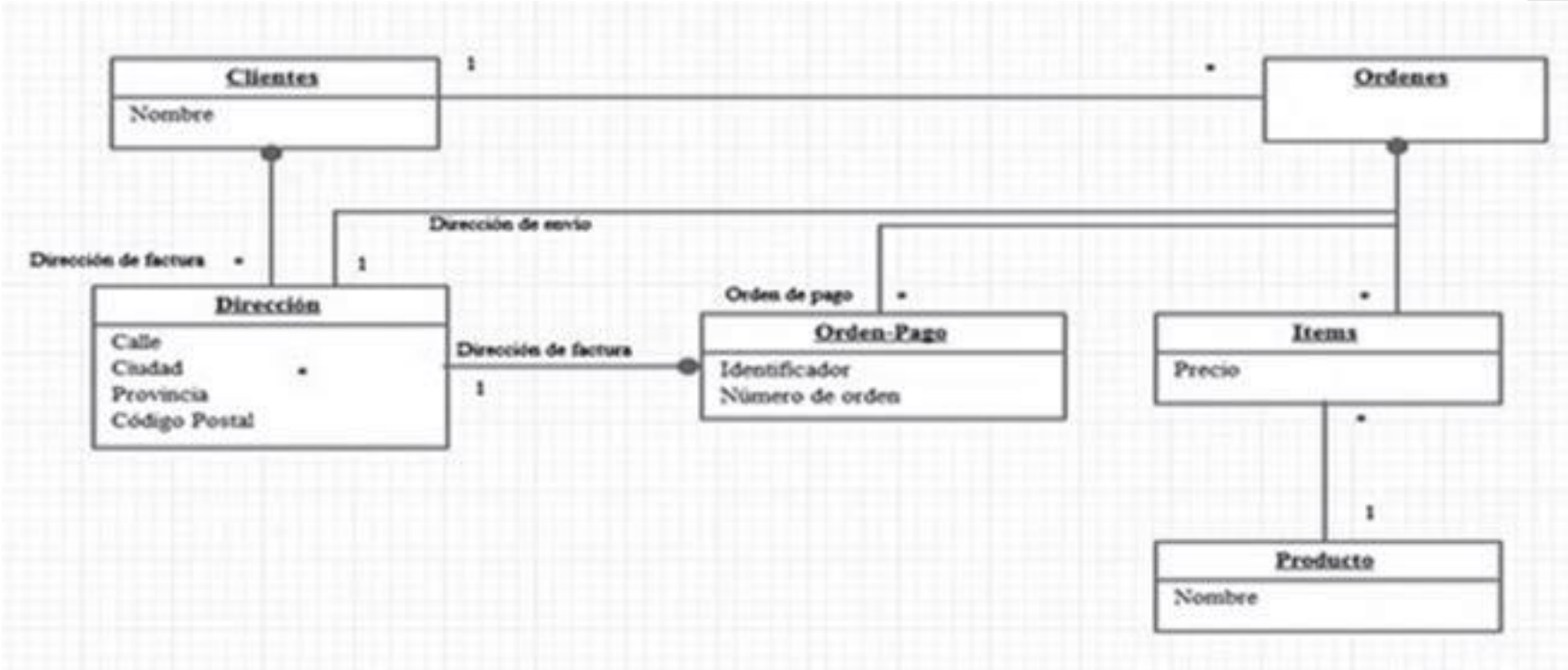
Agregados

- Los agregados son registros complejos que representan un conjunto de datos relacionados que son tratados como una unidad.
- Son una unidad para la manipulación de datos y para la gestión de la consistencia.
- Se actualizarán agregados con operaciones atómicas y se interaccionarán con la base de datos en términos de agregados.
- También el uso de agregados facilita que las bases de datos puedan operar sobre clústeres
  - El concepto de agregado se convierte en una unidad natural para la replicación y la distribución.
- Desde el punto de vista del programador, es más fácil operar con agregados, pues habitualmente operan con datos a través de estructuras agregadas.

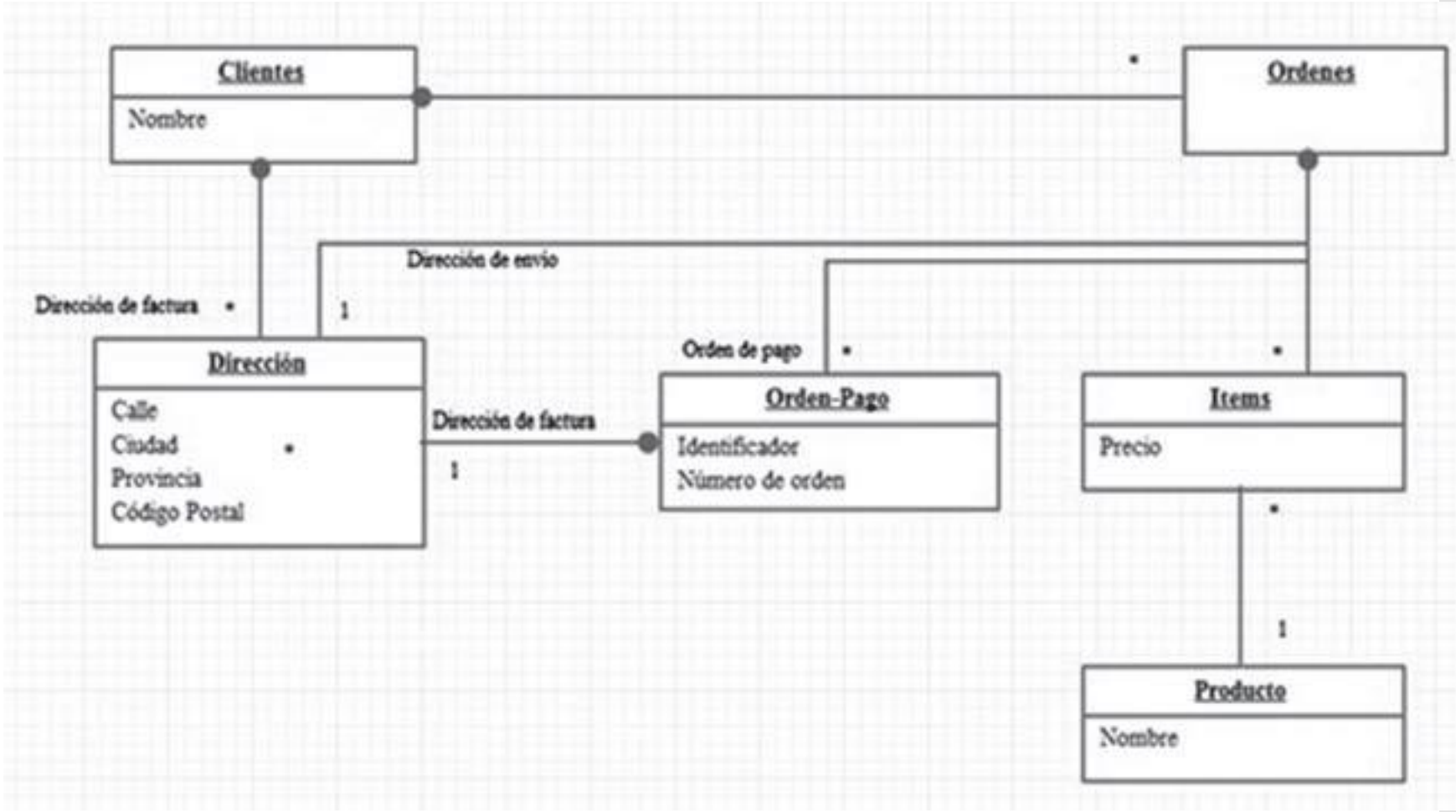
Modelo relacional Tienda virtual



Modelo agregado Tienda virtual



Modelo agregado Tienda virtual



Modelo de distribución de datos NoSQL

- Las bases de datos NoSQL son interesantes por su capacidad para ejecutarse sobre clústeres de grandes dimensiones.
- Sin embargo, la distribución conlleva un coste en términos de complejidad.
- Hay dos modelos de distribución de datos: replicación y sharding.
  - La replicación toma los mismos datos y los copia en múltiples servidores
  - El sharding distribuye diferentes datos a través de múltiples servidores de manera que cada servidor actúa como una única fuente para un subconjunto de datos.
  - Ambos modelos puede usarse en solitario uno de ellos o bien ambos.

Problemas de consistencia de datos en BD NoSQL

- Consistencia en las escrituras. Cuando dos usuarios tratan de actualizar el mismo dato al mismo tiempo se produce un conflicto de escritura-escritura.
- Consistencia en las lecturas. Una inconsistencia de lectura o conflicto de lectura-escritura se produce cuando un usuario realiza una lectura en la mitad de la escritura de otro sobre los mismos datos que se están leyendo.
- Consistencia de sesión. Las ventanas de inconsistencia puedenser problemáticas cuando se producen inconsistencias con uno mismo.
- Aunque es posible diseñar sistemas que eviten las inconsistencias, a menudo no es posible hacerlo sin sacrificar otras características del sistema, de manera que existe una compensación de la consistencia del sistema con otras características.

Teorema de CAP

- El teorema establece que, si se consideran las propiedades consistencia, disponibilidad y tolerancia a la partición en un sistema, solo es posible conseguir dos de ellas a la vez.
  - Consistencia
  - Disponibilidad: significa que si se tiene acceso a un nodo en un clúster, se puede leer y escribir datos.
  - Tolerancia al particionamiento: significa que el clúster puede sobrevivir a roturas de la comunicación en el clúster que lo separen en múltiples particiones que no puedan comunicarse entre sí (conocido como “split brain”).

Teorema de CAP

- Aunque el teorema establece que solo se pueden conseguir dos de las tres propiedades, en la práctica lo que está estableciendo es que un sistema puede sufrir particiones y hay que compensar consistencia para conseguir disponibilidad.
- El sistema no será completamente consistente ni disponible, pero tendrá una combinación que será razonable para las necesidades particulares.
- Hay casos donde se puede tratar adecuadamente con respuestas inconsistentes a las peticiones o lecturas inconsistentes.

Modelo de datos documental

- Está fuertemente orientado a agregados, dado que una base de datos consiste en un conjunto de agregados denominados «documentos».
- Las bases de datos documentales se caracterizan por que definen un conjunto de estructuras y tipos permitidos que pueden ser almacenados
  - Es posible acceder a la estructura del agregado, teniendo como ventaja que se consigue más flexibilidad en el acceso.
- Se pueden realizar consultas a la base de datos según los campos del agregado, se pueden recuperar partes del agregado en vez del agregado completo, y además se pueden crear índices basados en el contenido del agregado.
- Dado que cada agregado tiene asociado un identificador, es posible realizar búsquedas del estilo clave-valor.

MongoDB

- MongoDB es una base de datos NoSQL orientada a documentos.
- Creada por la compañía 10gen en el año 2007.
- Se caracteriza por que almacena los datos en documentos de tipo JSON con un esquema dinámico denominado “BSON”.

Documentos

- Son la unidad básica de organización de la información en MongoDB
  - Desempeñan un papel equivalente a una fila en las bases de datos relacionales.
- Un documento es un conjunto ordenado de claves que tienen asociados valores, y que se corresponden con algunas estructuras de datos típicas de los lenguajes de programación tales como tablas hash o diccionarios.
- En general, los documentos contendrán múltiples pares clavevalor
  - {"Nombre":"Juan","País":"España"}

Características de los documentos

- Las claves son cadenas, por lo que se permite cualquier carácter con un par de excepciones:
  - La clave no pueden contener el carácter nulo «\0».
  - El punto «.» y el «$» deben evitarse, pues tienen propiedades especiales.
- MongoDB es sensible a mayúsculas y minúsculas como a los tipos de datos.
  - Los siguientes documentos se consideran distintos:
  - {"Edad":3}, {"Edad": "3"}, {"edad":3}, {"edad":"3"}.
- Los documentos no pueden tener claves duplicadas.
  - El siguiente documento es incorrecto:
  - {"edad":3,"edad":56}.
- Los pares clave-valor están ordenados en los documentos.
  - El documento {"x":3,"y":5} no es igual que {"y":5,"x":3}.
  - Es importante no definir las aplicaciones pensando en el orden de los campos, pues MongoDB puede reordenarlos automáticamente en determinadas situaciones.
- Los valores de un documento pueden ser de diferentes tipos.

Tipos de datos

- Nulo: representa el valor nulo o bien un campo que no existe.
  - Por ejemplo, {"x":null}.
- Booleanos: representa el tipo booleano, que puede tomar los valores de true o false.
  - Por ejemplo, {"x":true}.
- Números: distingue entre números reales
  - Por ejemplo, {"x":3.14}
  - y números enteros, como, por ejemplo, {"x":45}.
- Cadenas: cualquier cadena de caracteres
  - Por ejemplo, {"x":"Ejemplo"}.

Tipos de datos

- Fechas: almacena la fecha en milisegundos, pero no la zona horaria.
  - Por ejemplo, {"x":new Date()}.
  - Para crear un objeto de tipo fecha se usa el comando new Date (). Sin embargo, si se llama sin new (solo Date ()), se retorna una cadena que representa la fecha. Y por tanto se trata de diferentes tipos de datos.
  - Las fechas en el Shell son mostradas usando la configuración local de la zona horaria, sin embargo, la base de datos las almacena como un valor en milisegundos sin referencia a la zona horaria (aunque podría almacenarse este valor definiendo una clave para el mismo).
- Expresiones regulares: se pueden usar expresiones regulares para realizar consultas.

Tipos de datos

- Arrays: se representa como un conjunto o lista de valores.
  - Por ejemplo, {"x":["a","b","c"]}.
  - Pueden ser usados tanto en operaciones en las que el orden es importante, tales como listas, pilas o colas, como en operaciones en las que el orden no es importante, tales como conjuntos.
  - Los arrays pueden contener diferentes tipos de valores, como, por ejemplo, {"Cosas": ["edad",45]} (de hecho, soporta cualquiera de los tipos de valores soportados para los documentos, por lo que se pueden crear arrays anidados).
  - Una propiedad importante en MongoDB es que reconoce la estructura de los arrays y permite navegar por el interior de los arrays para realizar operaciones sobre sus contenidos, como consultas, o crear índices sobre sus contenidos.
  - MongoDB permite realizar actualizaciones que modifican los contenidos de los arrays, tales como cambiar un valor del array por otro.

Tipos de datos

- Documentos embebidos: los documentos pueden contener documentos embebidos como valores de un documento padre.
  - Por ejemplo, {"x":{"y":45}}.
  - Los documentos pueden ser usados como valores de una clave, y en este caso se denominan «documentos embebidos».
  - Se suelen usar para organizar los datos de una manera lo más natural posible.
  - Por ejemplo, si se tiene un documento que representa a una persona y se quiere almacenar su dirección, podría crearse anidando un documento «dirección» al documento asociado a una persona, como, por ejemplo:
  - {
  - "nombre": "Juan",
  - "dirección": {
  - "calle":"Mayor 3",
  - "ciudad": "Madrid",
  - "País": "España"
  - }}

Tipos de datos: Documentos embebidos

- MongoDB es capaz de navegar por la estructura de los documentos embebidos y realizar operaciones con sus valores, como, por ejemplo, crear índices, consultas o actualizaciones.
- La principal desventaja de los documentos embebidos se debe a la repetición de datos.
- Por ejemplo, supóngase que las direcciones se encuentran en una tabla independiente y que hay que rectificar un error tipográfico, entonces, al rectificar la dirección en la tabla direcciones, se rectifica para todos los usuarios que comparten la misma dirección. Sin embargo, en MongoDB habría que rectificar las direcciones una a una en cada documento, aunque se compartan direcciones.

Tipos de datos

- Identificadores de objetos: es un identificador de 12 bytes para un documento.
  - Por ejemplo, {"x": OjectId()}.
  - Cada documento tiene que tener un clave denominada «_id»:
  - El valor de esta clave puede ser de cualquier tipo pero por defecto será de tipo ObjectId.
  - En una colección, cada documento debe tener un valor único y no repetido para la clave «_id», lo que asegura que cada documento en la colección pueda ser identificado de manera única. El tipo ObjectID es el tipo por defecto para los valores asociados a la clave «_id».
  - Es un tipo de datos diseñado para ser usado en ambientes distribuidos de manera que permita disponer de valores que sean únicos globalmente.

Timestamp				Machine			PID		Increment		
0	1	2	3	4	5	6	7	8	9	10	11

  - Cuando un documento se va a insertar, si no tiene un valor para la clave «_id», es generado automáticamente por MongoDB.

Tipos de datos

- Datos binarios: es una cadena de bytes arbitraria que no puede ser manipulada directamente desde el Shell y que sirve para representar cadenas de caracteres no UTF8.
- Código Javascript: los documentos y las consultas pueden contener código JavaScript. Por ejemplo, {"x: function () {...}}.

Colecciones

- Una colección es un grupo de documentos
  - Desempeña el papel análogo a las tablas en las bases de datos relacionales.
- Las colecciones tienen esquemas dinámicos, lo que significa que dentro de una colección puede haber cualquier número de documentos con diferentes estructuras.
  - Por ejemplo, en una misma colección podrían estar los siguientes documentos diferentes:
  - {“edad”:34}, {“x”:”casa”} que tienen diferentes claves y diferentes tipos de valores.

Razones para separar documentos en Colecciones

- Es razonable crear un esquema y agrupar los tipos relacionados de documentos juntos aunque MongoDB no lo imponga como obligatorio:
  - Si se mantuvieran diferentes tipos de documentos en la misma colección, produciría problemas para asegurar que cada consulta solo recupera documentos de un cierto tipo o que en el código de la aplicación implementa consultas para cada tipo de documento.
  - Es más rápido obtener una lista de colecciones que extraer una lista de los tipos de documentos en una colección.
  - La agrupación de documentos del mismo tipo juntos en la misma colección permite la localidad de los datos.
  - Cuando se crean índices se impone cierta estructura a los documentos (especialmente en los índices únicos).
  - Estos índices están definidos por colección
    - Poniendo documentos de un solo tipo en la misma colección se podrán indexar las colecciones de una forma más eficiente.

Restricciones del nombre de las Colecciones

- La cadena vacía no es un nombre válido para una colección.
- Los nombres de las colecciones no pueden contener el carácter nulo "\o", pues este símbolo se usa para indicar el fin del nombre de una colección.
- No se debe crear ninguna colección que empiece con "system", dado que es un prefijo reservado para las colecciones internas.
  - Por ejemplo, la colección system.users contiene los usuarios de la base de datos, la colección system.namespaces contiene información acerca de todas las colecciones de la base de datos.
- Las colecciones creadas por los usuarios no deben contener el carácter reservado "$" en su nombre.

Subcolecciones

- Una convención para organizar las colecciones consiste en definir subcolecciones usando espacios de nombres separados por el carácter «.».
- Aunque las subcolecciones no tienen propiedades especiales, son útiles por algunas razones:
  - Existe un protocolo en MongoDB para almacenar archivos muy extensos denominado «GridFS» que usa las subcolecciones para almacenar los archivos de metadatos separados de los datos.
  - Algunas librerías proporcionan funciones para acceder de una forma simple a las subcolecciones de una colección.

Bases de datos

- Las colecciones se agrupan en bases de datos, de manera que una única instancia de MongoDB puede gestionar varias bases de datos cada una agrupando cero o más colecciones.
- Cada base de datos tiene sus propios permisos y se almacena en ficheros del disco separados.
- Una buena regla general consiste en almacenar todos los datos de una aplicación en la misma base de datos.
- Las bases de datos separadas son útiles cuando se almacenan datos para aplicaciones o usuarios diferentes que usan el mismo servidor de MongoDB.

Nombres de Bases de datos

- La cadena vacía no es un nombre válido para una base de datos.
- El nombre de una base de datos no puede contener ninguno de los siguientes caracteres: \,/,.,",*,<,>,:,|,?,$,espacio o \o(valor nulo).
- Los nombres de las bases de datos son sensitivos a mayúsculasy minúsculas incluso sobre sistemas de archivos que no lo sean. Una regla práctica es usar siempre nombres en minúscula.
- Los nombres están limitados a un máximo de 64 bytes.
- Existen nombres que no pueden usarse para las bases de datos por estar reservados:
  - admin. Es el nombre de la base de datos «root» en términos de autenticación.
  - local. Esta base de datos nunca será replicada y sirve para almacenar cualquier colección que debería ser local a un servidor.
  - configura. Cuando en MongoDB se usa una configuración con sharding, se usa esta base de datos para almacenar información acerca de los fragmentos o shards que se crean.

Espacio de nombres

- Mediante la concatenación del nombre de una base de datos con una colección de la base de datos se consigue una cualificación entera del nombre de la colección denominado «espacio de nombres».
  - Por ejemplo, si se usa la colección blog.posts en la base de datos cms, el espacio de nombres de esa colección sería
    - **cms.blog.posts**
- Los espacios de nombres están limitados a 121 bytes de longitud, aunque en la práctica es mejor que sean menores de 100 bytes.