Instituto Tecnológico de Costa Rica

Base de Datos II - IC4302

Clase Virtual - Viernes 6 de octubre 2023

Conceptos vistos en clase.

Metadatos: Datos acerca de los datos.

Duplicates: Evita que hayan duplicados en tablas SQL.

Big Large Object(BLOB): Es información binaria que tiene asociado metadatos. Ejemplo tener la información de o datos de un video, un usuario podrá obtener los datos ya sea de una base de datos SQL como NoSQL, sin tener que ingresar al BLOB. Los BLOB no se almacenan en bases de datos.

BSON - Binary JSON: Representación binaria de un documento.

Tired Scaling: Es cuando se tienen diferentes clases de hardware dentro del cluster, conforme los datos menos se usan, los datos se van degradando, hasta llegar al Object Storage.

Federated Queries: Búsquedas desde el Object Storage.

Mongo

Una base de datos como Mongo se conoce como una base de datos NoSQL documental. Esto con lleva al Document Model, se conoce como un documento atómico, la información se representa como un JSON, ayuda a representar cualquier tipo de datos.

Organización de Mongo: Al más alto nivel se encuentra Mongo, se tienen una serie de colecciones(equivalente a tablas), que sería un contenedor de documentos, esto permite tomar los datos y particionarlos y distribuirlos en diferentes servidores.

El problema de manejar con JSON puro, es que, al momento de manejar muchos documentos de este tipo, genera un desperdicio de almacenamiento e información es muy grande. Con la utilización de BSON se evita el desperdicio de información.

Caracteristicas de Mongo

- **Intuitivo:** Al guardar un documento con su estructura original, es muy sencillo que una persona se pueda adaptar con lo que está trabajando.
- **Esquema Dinamico(self describing):** En Mongo es posible tener dos esquemas diferentes, esto ayuda a evitar colocar nulos, evitar temas de espacio o realizar validaciones, permitiendo dar un enfoque a los datos que está recolectando y como la va a guardar en la base de datos.
- **Multi schema:** Es la posibilidad de tener datos que se pueden representar de diferentes formas.
- **Schema Validation:** Son las reglas que definen los usuarios, para realizar validaciones en los espacios que se están guardando, como también verificar los tipos de datos que se están almacenando.

- **JSON:** Hace mucho tiempo en los servidores de aplicación, se hablaba en un protocolo HTTP, que es un protocolo que permite realizar intercambio de datos textuales, esto ayuda a desplegar información en un browser. Los HTML son ASCII visible que pasa por puerto 80, el browser interpreta HTML, el servidor realiza la solicitud vía puerto 80, mediante un **GETHTML**, para luego mostrarlo al usuario. Luego al querer hacer cambios en el envió de datos en HTML, se implementó el uso de un servidor de aplicaciones, que hablaba un meta lenguaje que se encontraba dentro del HTML, como jsp(Java), PHP,asp, entre otros. Esto permitió que las paginas no fueran estáticas y pasaron a ser dinámicas, con la opción enviar cualquier tipo de dato. Seguidamente se quiso realizar un cambio al modelo mencionado y aparecieron los "javascript", y su función era ejecutar en el browser del usuario el código, en lugar de realizarlo en el servidor que le brindaba la información, a esto se le llama **Client Execution**, esto permite eliminar trabajo del servidor de aplicaciones. Con la aparición de la API, su función era implementar mediante HTTP, traer información, información que venía representada en JSON.

Working with Mongo

- **Query API:** Se accede mediante un driver, la API permite acceder a información, agrugar, trasformar, analizar y agrupar información.
- **Agregation Pipeline:** Transforma y analiza los datos que se están resolviendo.
- **Multi Stage:** Es un pipeline que puede encadenar múltiples transformaciones, esto permite acceder a la información y en el mismo pipeline, pasan la información a una transformación, que su salida pasa a otra transformación.
- **ACID Transaction:** Se puede manejar a nivel de documento o múltiples documentos, se puede omitir la consistencia, se puede tener consistencia eventual o consistencia fuerte.
- **Change Stream:** Implementa un mecanismo de push notification, por ejemplo Mongo Atlas notifica a Atlas Search, que obtenga los nuevos datos y los procese, se podría decir que es como una suscripción.

Data Protection: El usuario tiene la posibilidad de escoger como va a ser el data protection. Posee múltiples nodos, que permite la replicación de un lugar a otro. Mongo conoce en todo momento en que ubicación geográfica se encuentra el servidor, esto es importante por temas de **leyes de protección de datos**.

Write Concern: Define el nivel de consistencia que se desea tener. El write concern tiene tres variables que se deben tomar en cuenta y son las siguientes:

{w:<value>, i: <boolean>,wtimeout: <number>}

- Variable **w(Majority):** w especifica el número de servidores a los cuales se les debe hacer la escritura o envió de la información para asumir que la escritura fue completamente satisfactoria, esto ayuda a que si el servidor muere, la cantidad de servidores que se les realizó escritura, nos garantiza que hay copias de los datos en los otros servidores.
- Variable **j**: Se tiene por ejemplo tres replicas, el servidor 1 recibe un insert, y se encarga de enviar el mismo insert a las otras replicas, con **j** se instruye a Mongo, que en el momento que lleguen los datos al servidor, se da por completa la operación, esto podría generar perdida de información, si el booleano de **j** se coloca en true, cuando se hace la inserción se instruye a Mongo DB que espere a que el otro nodo confirme que se escribió en disco.
- Variable **wtimeout:** La función de esta variable es decir cuánto tiempo hay que esperar antes de que se dé la confirmación de que el write concern terminó. Si se pide a j que verifique que la escritura de disco fue satisfactoria y el **wtimeout** en 10 segundos y una de las bases de datos duró más tiempo que la otra, el servidor avisa que la operación no funcionó.

Arquitectura distribuida

- ReplicaSet, capacidad de tener de uno a cincuenta servidores.
- Nodos separados (Ubicados en data centers).
- Puede estar repartida en diferentes ubicaciones geográficas.

Caracterisiticas

- **Scale-out:** Es posible agregar más servidores según sean necesarios.
- **Always-ON HA:** No tiene downtime, implica que se debe hacer un gasto grande de dinero.
- **Distributted Data Low Latency:** Las conexiones entre servidores para hacer actualizadores son muy rápidas.
- **Geo-User Data Access(Low latency):** Al tener localidad de datos y detectar donde se encuentran las personas usuarios, se puede localizar hacia el nodo más cercano de la petición de una persona, y garantiza perdida en latencia.

Como se mencionó anteriormente Mongo puede realizar **Scale out/ Scale up**, ya sea vertical o horizontal, y también **Rolling Update**, lo que quiere decir que actualiza de servidor en servidor.

Sharded Cluster: Son servidores que toman la información y la dividen en partes pequeñas, esto implica a algo que se llama "Config servers", lo que realiza es conocer en todo momento en donde se encuentra la información que se está buscando.

Las particiones de los datos se realizan mediante un método llamada **Shard Key**, una mala escogencia de un shard key puede generar bottleneck.

Mediante **distributes reads/writes**, al tener muchos servidores es posible enviar muchas escrituras y permite el aumento de capacidad de escritura y lectura del sistema. El shard cluster posee alta disponibilidad. Los **Chunks** es la porción del shard.

También presenta un **balancer**, su función es mover dentro de la misma colección de servidores o replicaSet los shards, para lograr un balance de carga.

Sharding Strategy disponibles en Mongo:

- Hashed Sharding, genera un hash de los datos, y se asigna con alguna función de hash, a diferentes shards y esto genera valores continuos, que generan problemas en los queries.
- Ranged Sharding, muy util si y solo si, se escoge un ranged sharding correcto.
- Zoned Sharding, ubica por ubicaciones a los usuarios en un shard.

Datos importantes de Mongo

- Refinar o cambiar los shard key, para analizar los patrones de búsqueda mediante observabilidad.
- Es posible agregar o eliminar shards.
- Es posible aumentar el número de réplicas mediante scale out.
- Rebalance data, los shards se mueven de servidor para obtener el máximo de rendimiento.