Sistemas de Bases de Datos 2

2024

Ing. Luis Alberto Arias Solórzano

Unidad 5

NoSQL

Es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico del sistema de gestión de bases de datos relacionales (RDBMS) en aspectos importantes, el más destacado es que no usan SQL como el principal lenguaje de consultas. Los datos almacenados no requieren estructuras fijas como tablas, normalmente no soportan operaciones JOIN, ni garantizan completamente ACID (atomicidad, consistencia, aislamiento y durabilidad), y habitualmente escalan bien horizontalmente.

Se refieren a este tipo de bases de datos como almacenamiento estructurado, término que abarca también las bases de datos relacionales clásicas. A menudo, las bases de datos NoSQL se clasifican según su forma de almacenar los datos, y comprenden categorías como clave-valor, las implementaciones de BigTable, bases de datos documentales, y Bases de datos orientadas a grafos.

NoSQL

Las bases de datos NoSQL están altamente optimizadas para las operaciones recuperar y agregar, y normalmente no ofrecen mucho más que la funcionalidad de almacenar los registros (p.ej. almacenamiento clave-valor). La pérdida de flexibilidad en tiempo de ejecución, comparado con los sistemas SQL clásicos, se ve compensada por ganancias significativas en escalabilidad y rendimiento cuando se trata con ciertos modelos de datos.

Los sistemas de bases de datos NoSQL crecieron con las principales compañías de Internet, como Google, Amazon, Twitter y Facebook. Estas tenían que enfrentarse a desafíos con el tratamiento de datos que las tradicionales RDBMS no solucionaban. Con el crecimiento de la web en tiempo real existía una necesidad de proporcionar información procesada a partir de grandes volúmenes de datos que tenían unas estructuras horizontales más o menos similares. Estas compañías se dieron cuenta de 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 proceso.

NoSQL

Ventajas

- Estos sistemas responden a las necesidades de escalabilidad horizontal que tienen cada vez más empresas.
- Pueden manejar enormes cantidades de datos.
- No generan cuellos de botella.
- Escalamiento sencillo.
- Diferentes DBs NoSQL para diferentes proyectos.
- Se ejecutan en clusters de máquinas baratas.

NoSQL

Bases de datos orientadas a objetos

- ObjectDB
- Zope Object Database
- db4o
- GemStone S
- Objectivity/DB

Bases de datos documentales

- CouchDB, de Apache CouchDB
- MongoDB, de 10gen
- RavenDB, de Hibernating Rhinos.
- BaseX
- djondb
- eXist
- DocumentDB, de Amazon
- IBM Lotus Domino
- Terrastore

Bases de datos en grafo

- Neo4j
- DEX/Sparksee
- AllegroGraph
- OrientDB
- InfiniteGraph
- Sones GraphDB
- InfoGrid
- HyperGraphDB

NoSQL

Bases de datos clave/valor

- Cassandra, de Apache The Apache Cassandra
- BigTable, de Google
- Dynamo, de Amazon
- MongoDB
- Project Voldemort, de LinkedIn
- Riak
- Redis
- Oracle NoSQL

Bases de datos multivalor

- OpenQM
- Extensible storage engine

Bases de datos tabular

- HBase, de Apache
- BigTable, de Google
- LevelDB, versión abierta de BigTable
- Hypertable

Mongo DB

- MongoDB (que proviene de «humongous») es la base de datos NoSQL líder y permite a las empresas ser más ágiles y escalables. Organizaciones de todos los tamaños están usando MongoDB para crear nuevos tipos de aplicaciones, mejorar la experiencia del cliente, acelerar el tiempo de comercialización y reducir costes.

- Es una base de datos ágil que permite a los esquemas cambiar rápidamente cuando las aplicaciones evolucionan, proporcionando siempre la funcionalidad que los desarrolladores esperan de las bases de datos tradicionales, tales como índices secundarios, un lenguaje completo de búsquedas y consistencia estricta.

Mongo DB - Características

Consultas Ad hoc

MongoDB soporta la búsqueda por campos, consultas de rangos y expresiones regulares. Las consultas pueden devolver un campo específico del documento pero también puede ser una función JavaScript definida por el usuario.

Indexación

Cualquier campo en un documento de MongoDB puede ser indexado, al igual que es posible hacer índices secundarios. El concepto de índices en MongoDB es similar a los encontrados en base de datos relacionales.

Replicación

MongoDB soporta el tipo de replicación primario-secundario. Cada grupo de primario y sus secundarios se denomina replica set. El primario puede ejecutar comandos de lectura y escritura. Los secundarios replican los datos del primario y sólo se pueden usar para lectura o para copia de seguridad, pero no se pueden realizar escrituras. Los secundarios tiene la habilidad de poder elegir un nuevo primario en caso de que el primario actual deje de responder.

Mongo DB - Características

Balanceo de carga

MongoDB se puede escalar de forma horizontal usando el concepto de "shard". El desarrollador elige una clave de sharding, la cual determina cómo serán distribuidos los datos de una colección. Los datos son divididos en rangos (basado en la clave de sharding) y distribuidos a través de múltiples shard. Cada shard puede ser una replica set. MongoDB tiene la capacidad de ejecutarse en múltiple servidores, balanceando la carga y/o replicando los datos para poder mantener el sistema funcionando en caso que exista un fallo de hardware. La configuración automática es fácil de implementar bajo MongoDB y se pueden agregar nuevas servidores a MongoDB con el sistema de base de datos funcionando.

Almacenamiento de archivos

MongoDB puede ser utilizado como un sistema de archivos, tomando la ventaja de la capacidad que tiene MongoDB para el balanceo de carga y la replicación de datos utilizando múltiples servidores para el almacenamiento de archivos. Esta función se llama GridFS y es mas bien una implementación en los drivers, no en el servidor , por lo que está incluida en los drivers oficiales que la compañía de MongoDB desarrolla. Estos drivers exponen funciones y métodos para la manipulación de archivos y contenido a los desarrolladores. En un sistema con múltiple servidores, los archivos pueden ser distribuidos y replicados entre los mismos y de una forma transparente, de esta forma se crea un sistema eficiente que maneja fallos y balanceo de carga.

Mongo DB - Características

Agregación

MongoDB proporciona un framework de agregación que permite realizar operaciones similares a las que se obtienen con el comando SQL "GROUP BY". El framework de agregación está construido como un pipeline en el que los datos van pasando a través de diferentes etapas en los cuales estos datos son modificados, agregados, filtrados y formateados hasta obtener el resultado deseado. Todo este procesado es capaz de utilizar índices si existieran y se produce en memoria. Asimismo, MongoDB proporciona una función MapReduce que puede ser utilizada para el procesamiento por lotes de datos y operaciones de agregación.

Ejecución de JavaScript del lado del servidor

MongoDB tiene la capacidad de realizar consultas utilizando JavaScript, haciendo que estas sean enviadas directamente a la base de datos para ser ejecutadas.

Mongo DB - Desventajas

No implementa las propiedades ACID

El no implementar las propiedades ACID genera que la base de datos no asegure la durabilidad, la integridad, la consistencia y el aislamiento requeridos obligatoriamente en las transacciones. Es posible que en futuras versiones esto se solucione.

Sobre la base de este punto se detallan los cuatro siguientes:

Problemas de consistencia

Las lecturas estrictamente consistentes ven versiones obsoletas de documentos, también pueden devolver datos incorrectos de lecturas que nunca deberían haber ocurrido.

Bloqueo a nivel de documento

MongoDB bloquea la base de datos a nivel de documento ante cada operación de escritura. Sólo se podrán hacer operaciones de escritura concurrentes entre distintos documentos.

Mongo DB

Las escrituras no son durables ni verificables

MongoDB retorna cuando todavía no se ha escrito la información en el espacio de almacenamiento permanente, lo que puede ocasionar pérdidas de información. En MongoDB 2.2 se cambia el valor por defecto para escribir en al menos una réplica, pero esto sigue sin satisfacer la durabilidad ni la verificabilidad.

Problemas de escalabilidad

Tiene problemas de rendimiento (v2.2)cuando el volumen de datos supera los 100GB.

Mongo DB – Manejo de Datos

MongoDB guarda la estructura de los datos en documentos tipo JSON con un esquema dinámico llamado BSON, lo que implica que no existe un esquema predefinido.

Los elementos de los datos se denominan documentos y se guardan en colecciones. Una colección puede tener un número indeterminado de documentos. Comparando con una base de datos relacional, se puede decir que las colecciones son como tablas y los documentos son registros en la tabla. La diferencia es que en una base de datos relacional cada registro en una tabla tiene la misma cantidad de campos, mientras que en MongoDB cada documento en una colección puede tener diferentes campos.

En un documento, se pueden agregar, eliminar, modificar o renombrar nuevos campos en cualquier momento, ya que no hay un esquema predefinido. La estructura de un documento es simple y compuesta por "key-value pairs" parecido a las matrices asociativas en un lenguaje de programación, esto es debido a que MongoDB sigue el formato de JSON. En MongoDB la clave es el nombre del campo y el valor es su contenido, los cuales se separan mediante el uso de ":", tal y como se puede ver en el siguiente ejemplo. Como valor se pueden usar números, cadenas o datos binarios como imágenes o cualquier otro "key-value pairs".

Mongo DB - Compatibilidad

Platform	3.2	3.0	2.6	2.4	2.2
Amazon Linux	✓	✓	✓	✓	✓
Debian 8	✓				
Debian 7	✓	✓	✓	✓	✓
Fedora 8+			✓	✓	✓
RHEL/CentOS 6.2+	✓	✓	✓	✓	✓
RHEL/CentOS 7.0+	✓	✓	✓		
SLES 11	✓	✓	✓	✓	✓
SLES 12	✓				
Solaris 64-bit	✓	✓	✓	✓	✓
Ubuntu 12.04	✓	✓	✓	✓	✓
Ubuntu 14.04	✓	✓	✓		
Ubuntu 16.04	✓				
Microsoft Azure	✓	✓	✓	✓	✓
Windows Vista/Server 2008R2/2012+	✓	✓	✓	✓	✓
OS X 10.7+	✓	✓	✓	✓	

Mongo DB - Sintaxis

```
db.users.insertOne(
   {
      name: "sue",
      age: 19,
      status: "P"
   }
)
```

```
db.users.insertMany(
   [
     { name: "bob", age: 42, status: "A", },
     { name: "ahn", age: 22, status: "A", },
     { name: "xi", age: 34, status: "D", }
   ]
)
```

```
db.users.updateOne(
   { "favorites.artist": "Picasso" },
   {
     $set: { "favorites.food": "pie", type: 3 },
     $currentDate: { lastModified: true }
   }
)
```

```
db.users.updateMany(
   { "favorites.artist": "Picasso" },
   {
     $set: { "favorites.artist": "Pisanello", type: 3 },
     $currentDate: { lastModified: true }
   }
)
```

Mongo DB - Sintaxis

SQL INSERT Statements	MongoDB insert() Statements

`INSERT INTO users(user_id,`
`age,`
`status)`
`VALUES ("bcd001",`
`45,`
`"A")`
	`db.users.insert(`
	`{ user_id: "bcd001", age: 45, status: "A" }`
	`)`

Mongo DB - Sintaxis

SQL SELECT Statements	MongoDB find() Statements
`SELECT * FROM users`	`db.users.find()`
`SELECT id, user_id, status FROM users`	`db.users.find( { }, { user_id: 1, status: 1 } )`
`SELECT user_id, status FROM users`	`db.users.find( { }, { user_id: 1, status: 1, _id: 0 } )`
`SELECT * FROM users WHERE status = "A"`	`db.users.find( { status: "A" } )`
`SELECT user_id, status FROM users WHERE status = "A"`	`db.users.find( { status: "A" }, { user_id: 1, status: 1, _id: 0 } )`
`SELECT * FROM users WHERE status != "A"`	`db.users.find( { status: { $ne: "A" } } )`
`SELECT COUNT(user_id) FROM users`	`db.users.count( { user_id: { $exists: true } } )` *or* `db.users.find( { user_id: { $exists: true } } ).count()`
`SELECT COUNT(*) FROM users WHERE age > 30`	`db.users.count( { age: { $gt: 30 } } )` *or* `db.users.find( { age: { $gt: 30 } } ).count()`
`SELECT DISTINCT(status) FROM users`	`db.users.distinct( "status" )`
`SELECT * FROM users LIMIT 1`	`db.users.findOne()` *or* `db.users.find().limit(1)`
`SELECT * FROM users LIMIT 5 SKIP 10`	`db.users.find().limit(5).skip(10)`
`EXPLAIN SELECT * FROM users WHERE status = "A"`	`db.users.find( { status: "A" } ).explain()`

Mongo DB - Sintaxis

SQL Update Statements	MongoDB update() Statements

`UPDATE users`
`SET status = "C"`
`WHERE age > 25`
	`db.users.update(`
	`{ age: { $gt: 25 } },`
	`{ $set: { status: "C" } },`
	`{ multi: true }`
	`)`

`UPDATE users`
`SET age = age + 3`
`WHERE status = "A"`
	`db.users.update(`
	`{ status: "A" } ,`
	`{ $inc: { age: 3 } },`
	`{ multi: true }`
	`)`

Mongo DB - Sintaxis

SQL Delete Statements	MongoDB remove() Statements
`DELETE FROM users` `WHERE status = "D"`	`db.users.remove( { status: "D" } )`
`DELETE FROM users`	`db.users.remove({})`

Gracias