1.0 NoSQL

miércoles, 26 de julio de 2017 11:06 PM

Not Only SQL

Tiene una forma no estructurada para consultar la base de datos.
No tiene relaciones.
Son distribuidas, estan replicadas en varias maquinas y tiene mayor rendimiento las consultas.
Son explandibles, particionar datos, parte de datos en una maquina y otros datos en otra maquina.

Por que utilizar NoSQL

El número de personas que utilizan aplicaciones web, necesitan mayor rendimiento, la cantidad de datos y la forma en que se guardan en el disco duro no estaba optimizado, este rendimiento lo podemos lograr con NoSQL.

Esquemas dinamicos, son mas flexibles que permite NoSQL a diferencia de las BD Relacionadas que tiene un esquema fijo.

No existen las transacciones en Base de Datos.

Facilidad de Interacción.

MongoDB

MongoDB esta desarrollado bajo un tipo de dato llamado BSON[Binary Json] el cual es un formato de intermcaio de datos para el almacenamiento y transferencia de datos.

Los objetos BSON consiste en una lista ordenada de elementos. Cada elemento consta de un campo nombre, un tipo y un valor.

Cuando usar NoSQL

Relaciones debiles: Cuando las relaciones que existen entre un tipo de dato y otro no son realmente necesarias para el correcto funcionamiento, por lo tanto se pueden estructurar esos datos de otra manera.

-Usado en Redes Sociales
-Desarrollo web con poca uniformidad en la información
-Desarrollo movil
-BigData

El uso de la base de datos NoSQL implica que el programador debe tener mucho cuidado y queda en manos del programador velar por la integridad de los datos ya que no tiene muchas validaciones que realiza automaticamente las BD SQL Relacionales.

Cuando se tiene mucha información, cuando la aplicación necesita almancenar mucha información de tal manera que supera los limtes de una base de datos relacional es util el uso de NoSQL.

2.0 Introducción MongoDB

miércoles, 26 de julio de 2017 11:58 PM

Base de datos tipo documental, es una base de datos orientada a almacenar, recuperar y gestionar datos en formda de documentos(Datos estructurados)

El concepto de documento hace referencias a encapsular y codificar datos siguinedo un formato estandar.

En este caso mongoBD utliza el estandar BSON, un documento de tipo BSON es:

```
{
        Nombre: "Andres Valencia",
        Direccion: "Loma de los Bernal",
        Profesion:"Ingeniero"
}
```

Tiene una forma similar a la BD relacionar, tiene una base de datos, tenemos un nameSpace al cual asociaremos para guardar los datos y tenemos una Collection, la collection[representa una tabla en SQL] en este caso es la estructura que almacena un conjunto de documentos que no tienen estructura especifica o expresado de otra manera tiene una estructura dinamica.

Estructura

Base de Datos [NameSpace]
Collection [Tabla]
Documentos [Registros]

Documento

```
{
        Id: ObjectID(),  //Identificador del documento
        Campo1: "valor",
        Campor2: 2,
}
```

3.0 Indices

jueves, 27 de julio de 2017 10:27 PM

Permiten tomar ciertos campos de la base de datos, estructurarlos y tenerlos en memoria para acceder en un menor tiempo posible a la información y mejorar el rendimiento.

Btree[Binary tree]: Es el indice principal de MongoDB, es el indice asignado por defecto cuando se crea un indice sin especificar el tipo de indice. Tiene una estructura de arbol y la puede recorrer según la relación de padre e hijo.

Hashed: Es un indice basada en el algoritmo Hash, el cual esta implementado utilizando una llave unica y el valor.

FullText: Permite indexar los documentos utilizando los algoritmos fulltext como lo utilizados por los motores de busqueda.

Geoespaciales: Utilizan la estructura Btree pero optimizadas, y utiliza coordenas guardadas en las collections.

Compund: Son indices compuestos, que permiten tener indices compuestos por mas de un tipo.

Multikey:

REPLICACIÓN

Sistema para instalar MongoDB de forma distribuida y en lso diferentes nodos se tenga la misma información.

Unico nodo (journal): unico nodo
Replica Set: Sistema distribuido de varios nodos.
Sharding: Sistema distribuido de varios nodos,

Crear un incide:
Índices creados por campos
db.mongodb.ensureIndex({ejemplo: 1})

Indice compuesto:
db.motoresDB.ensureIndex({nombre: 1, _id: 1, "lista.0": 1}, {nombre: "IX_COM_MDB"})
Respetar el orden en que son creados los indices, para la consulta, los campos en la consulta tambien deben estar en orden.

Consultar los Indices de la Collections
db.motoresBD.getIndexKeys(
db.motoresBD.getIndexSpecs(
db.motoresBD.getIndexes(
db.motoresBD.getIndices(

Forzar el uso de un indice
Con el metodo hint() podemos indicar el uso de un Indice.

db.motoresBD.find()**.hint("Compuesto")**

4.0 Esquemas

jueves, 27 de julio de 2017 10:53 PM

Diseño de Esquemas en MongoDB.

EL diseño de esquemas es flexible pero necesario y son operacions atómicas.

Para diseñar un esqueme es importanta saber cuales van hacer las preguntas que le haremos a la base de datos.

Ejemplo de Documentos con campos emebidos:

```
{
        "_id":ObjectID(….),       //-> Identificador unico del documento
        "tittle": "apples",       //-> Atributo o campo del documento
        "relations": [{…}. {…}],  //-> Atributo o campo que contiene una lista de documentos embebidos.
        "description": {"text": "….", "Sensitive": false}   //-> Atributo o campo que contiene otro documento embedido
}
```

Ejemplo de Documentos con campos con referencia:

```
{
        "_id":ObjectID(….),       //-> Identificador unico del documento
        "tittle": "apples",       //-> Atributo o campo del documento
        "relations": [ObjectID(….), ObjectID(…)],  //-> Atributo o campo que contiene una lista de documentos embebidos.
        "description": {"text": "….", "Sensitive": false}   //-> Atributo o campo que contiene otro documento embedido
}
```

Para escoger cual de los tipos de esquema utilizar (embeber o referencias) es analizar la pregunta:
¿Cada vez que yo consulte el documento voy a necesitar tener disponible la información del campo que embebe otros documentos?

Si la respuesta es **NO:** Es recomendable usar referencias debilies.
Si la respuesa es **SI:** Es recomendable embeber los documentos.

5.0 Ejecución

martes, 09 de mayo de 2017 10:57 PM

Para instalar MongoDB se accede a la página de MongoDB https://www.mongodb.com/download-center#previous

MongoDB requiere una carpeta o directorio donde almacenar la información on la estructura **\data\db**
Por lo tanto podemos definir alguna carpeta persona y en su interior compiamos la estructura "**\data\db"**

1. Para iniciar el servidor de MongoDB, ejecutamos en consola el siguiente comando:
   Ejecutar en consola
   "C:\Program Files\MongoDB\Server\3.2\bin\mongod.exe" --dbpath "D:\3.0 DESARROLLOS\1.0 MongoDB\data"

2. Para conectarnos e inicar la BD de MongoDB, ejecutamos el siguiente comando:
   "C:\Program Files\MongoDB\Server\3.2\bin\mongo.exe"
   O desde el directorio doble clic sobre el archivo "mongo.exe"

6.0 Comandos MongoDB

martes, 09 de mayo de 2017 10:59 PM

Listar las bases de datos

- Show dbs: mostrar base de datos

Crear una base de datos

- Use [dataBase] (La base de datos en realidad no se crea hasta que se insertan registros)

Validar la instación de la base de datos

- Db: se valida la instancia de la base de datos en la que estamos.

Lista de comandos permitidos sobre la BD

- db.[tab]

Crear Collection[tabla]

- db.[NombreCollection]: No se crea en realidad la collections por que no se ha ingresado ningun registro.

Listar las Collections

- Show collections: Muestra las collections[tablas] de la base de datos.

Insertar Registros

- db.mongodb.insert({….})
  [InstanciaBaseDatos].[collections].insert
  [BaseDatos].[tabla].insert

```
db.mongodb.insert({tittle: "Curso MongoDB"})
```

Consultar Registros:

- Db.mongodb.find(): Trae todos los registros
- Db.mongodb.findOne(): Trae el primer registro
- Db.mongodb.find({tittle: "texto"}): Filtro para buscar los registros que tengan la propiedad tittle y coincida con el valor t exto .
- Db.mongodb.remove({tittle: "texto"})

Validación de indices creados:

```
>db.getCollectionNames().forEach(function(collection) {
  indexes = db[collection].getIndexes();
  print("Indexes for " + collection + ":");
  printjson(indexes);
});
```

```
db.motoresBD.getIndexKeys(
db.motoresBD.getIndexSpecs(
db.motoresBD.getIndexes(
db.motoresBD.getIndices(
```

Crear un incide:

- Índices creados por campos
  db.mongodb.ensureIndex({ejemplo: 1})

Se debe crear el indice por cada campo, este se especifica dentro del paramtero de ensurIndex en formato BSON,
Y se debe especificar el orden de la creación del indice en memoria :
1: Ascendente
-1: Descendente

Update:

Función para actualizar los registros se compone de la siguiente estructura:

db.collection.**update**({ *filtros* } , {"$set": {*campo: valor*} }, *upsert*, *multi*)

Upsert: Es un parametro boolean, si es <true> indica que si ningun registro coindice con el creiterio de actualización entonces crea un nuevo documento.
Multi: Parametro boolean, si es <true> indica que actualiza todos los documentos que coincidan con el criterio, si es falsa solo actulizará el primer documento.
[Upsert y Multi, no pueden ser true al mismo tiempo]

El siguiente update, actualizará todos los registros y les creara un campo lista vacio.
db.motoresDB.update({} , {"$set": {lista: [] } }, false, true)

Al registro que tiene la palabra MongoDB en el campo nombre, agregue el campo lista con los datos 1,2,3
db.motoresDB.update({nombre: "MongoDb"}, {"$set": {lista: [1,2,3]}}, false, ture)

db.collection.**update**({ *filtros* } , {"$push": {*campo: valor*} })
Agrega un nuevo valor al campo, en caso de ser un Array, agrega un nuevo documento al array

db.mercado.update({producto: "Frutas", tipo: "Manzana"},{"$set": {ventas: {valor: 1500, cantidad: 3}}})

7.0 Explain Plan

martes, 01 de agosto de 2017 11:46 PM

Cursor.explail(verbosity)

El explail Plan es un comando que provee información en el plan de la consulta para la **db.collection.find()**

Explain() tiene los siguientes parametros:
-queryPlanner: Es el metodo por defecto que utiliza el cursor.
-executionStats, allPlansExecution

Cursor utlitizado en la consulta:
Si el queyPlanner no utiliza indices y recorre la DB para obenter los resultado en el "winningPlan" muestra el **COLLSCAN** significa que esta utilizando el **BasicCursor** [No es recomendado utilizar este tipo de cursor ya que no esta haciendo usos de ningun indice es decir que no esta obteniendo los resultados de la memoria si no que esta yendo l disco por la información lo que hace mas lentas las consultas]

Si el queryPlanner utiliza indices en el "winningPlan" muestra **IXSCAN**

Cuando se crea un indice para un solo campo y se consulta todo el registro el queryPLanner debe ir de todas maneras al disco por que los otros campos del registro no tienen indices por lo tanto se debe crear un indice compuesto y en la consulta indicar solo el campo que tiene el indice.

Para forzar el uso de un indice y Seleccionar los campos deseados
db.motoresBD.find({nombre: "MongoDB", _id: {"$lt": ObjectId()}},**{nombre: 1, _id: 1}).hint("Compuesto")**.explain("allPlansExecution")

>**db.collections.find().explain("allPlansExecution")**

```
{
    "queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "platzi.motoresBD",
        "indexFilterSet" : false,
        "parsedQuery" : {
            "$and" : [ ]
        },
        "winningPlan" : {
            "stage" : "COLLSCAN",
            "filter" : {
                "$and" : [ ]
            },
            "direction" : "forward"
        },
        "rejectedPlans" : [ ]
    },
    "executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 4,
        "executionTimeMillis" : 0,
        "totalKeysExamined" : 0,
        "totalDocsExamined" : 4, [Objetos leidos desde el disco-> Objetivo ver en 0]
        "executionStages" : {
            "stage" : "COLLSCAN",
            "filter" : {
                "$and" : [ ]
            },
            "nReturned" : 4,
            "executionTimeMillisEstimate" : 0,
            "works" : 6,
            "advanced" : 4,
            "needTime" : 1,
            "needYield" : 0,
            "saveState" : 0,
            "restoreState" : 0,
            "isEOF" : 1,
            "invalidates" : 0,
            "direction" : "forward",
            "docsExamined" : 4
```

```
            }
        },
        "serverInfo" : {
            "host" : "pb0b0815613",
            "port" : 27017,
            "version" : "3.2.13",
            "gitVersion" : "23899209cad60aaafe114f6aea6cb83025ff51bc"
        },
        "ok" : 1
}
```

8.0 Arrays

jueves, 10 de agosto de 2017 07:05 AM

En mongo DB podemos tener tipos de datos listas, este campo puede guardar en un campo del registro un array []

Para ingresar un array se listan los elementos dentro de un corchete.

Db.motoresDB.insert({lista: []})

En una consulta se puede preguntar por un valor especifico dentro del campo y devuelve todo el campo.

db.motoresDB.find({lista: 1}) -> retorna el registro que dentro del campo lista uno de sus valores sea igual a 1.

Podemos utilizar el operador de posición para las listas (.) para ubicarnos en una posición especifica del array

db.motoresDB.find({ "lista.0": 1 }) -> Retorna la lista que tenga el valor 1 en la primera posición de array del campo lista

--

Indices para las posiciones del Array

Se pueden crear indices para las posiciones del array

db.motoresDB.ensureIndex({"lista .0": 1})

9.0 Sort

martes, 15 de agosto de 2017 10:27 PM

Comando para ordenar los registros de un collections.

Sort({}): Ordena los registros según el orden en el que se insertadon.

```
db.motoresDB.find().sort({nombre: 1}) // 1: Orden Ascendente
db.motoresDB.find().sort({nombre: -1}) //2: Orden Descendente
```

10.0 Agregation

jueves, 17 de agosto de 2017 07:02 AM

El framework de agregation cumple una función parecida al Group By de SQL.

Sintaxis:
.aggregate([{<stage>},])

Se le pasa una lista de operaciones que en mongodb se conoce como stage(estaciones), el framework de agregation recorrera en orden hasta que el proceseo termine .

Proceso secuencial: Ejecuta en orden las listas pasadas en el parametro.

Filtrar lo antes posible: Durante el proceso de agregación, lo ideal es realizar el procesamiento sobre la menor cantidad de datos posibles.

Limites en el uso de la memoria: Para el uso de la agregation, mongoDb tiene por defecto 100MB asigandos, si se requierre mas es necesario activar el uso del disco, en caso de que no se tenga activado esta funcionalidad mongoDB cancelara el proceso de agregation.

LISTA DE FUNCIONES DE AGREGATION

$match{<query>}: Operación de filtro para reducir el número de documentos sobre los cuales ejecutará la agregación.
Recomendado usarlo al inicio del pipeline(La secuencia de funciones del agregation) ya que podra utilizar los indices.

```
>db.mercado.aggregate([{$match: {producto: "Frutas"}}])
```

{$sort: {<llalve>:<dir>}}: Si se usa al incio del pipeline podra tambien utitilizar los indices.
Recomendado usar $Sort + $limit se haran mas rapidas las consultas, ya que empezará a ordenar y va contando cuando llegue al limite la operación de sort termina y continua con las siguientes funciones.

{$unwind: <llave>}: Se encarga de descomponer el array, es recomendado cuando necesitamos hacer operaciones de grupo, realizar agrupaciones es recomendado hacer el unwind, por lo tanto crea un documento en un campo por cada elemento del array.

```
>db.mercado.aggregate([{$unwind: "$ventas"}])
>db.mercado.aggregate([{$match: {producto: "Frutas"}},{$unwind: "$ventas"}]).toArray()
```

{$group: {..}}: Operador que permite realizar funciones de agrupación sobre las collections.
Se puede combinar con los acumuladores: ($sum, $avg, $max, $min)

```
> db.mercado.aggregate([{$unwind: "$ventas"},{$group: {_id: "$_id", result:{"$sum":
"$ventas.cantidad"}}}])
```

{$project: <llave>}: Es una operación que nos permite filtrar los campos que queremos tener en nuestro documento final, podemos agegar o podemos quitar campos y ahorrar espacio de momoria utilizado.

{$out: <llave>}: Luego que termina de ejecutar un proceso de agragación, se insertan los resultados en una nueva collection.

11. Autenticación

sábado, 19 de agosto de 2017 10:27 AM

MongoDB no tiene habilitado la autenticación por defecto, por lo tanto al momento de iniciar a trabajar no es necesario tener ususario y constraseña.

Si se requiere habilitar la autenticación:

El usuario se crea a nivel de base de datos, se debe elegir una base de datos y se crea el usuario para la base de datos. La información de los usuarios quedara guardad en la base de datos Admin o local que se crea por defecto con la instancio de MOngoDB

- db.createUser({user: "guest", pwd: "123456", roles: [{role: "readWrite", db: "platzi"}]})
- db.createUser({user: "admin", pwd: "pass", roles: [{role: "userAdminAnyDatabase", db: "admin"}, {role: "readWriteAnyDatabase", db: "admin"}]})

Luego de crear los usuarios, se debe reniciar el servidor de MongoDB para que tome los cambios, al momento de iniciar el servidor de MongoDB se debe habiltar el uso de autenticación colocando la palaba --auth en el comando

"C:\Program Files\MongoDB\Server\3.2\bin\mongod.exe" **--auth** --dbpath

Existen dos formas de autenticarse

1- En la linea de comandos cuando se va a inicar el cliente de mongoDB
mongo.exe -u <user> -p <pass> --authenticationDatabase <database>

```
mongo -u "myUserAdmin" -p "abc123" --authenticationDatabase "admin"
```

2- Después de ejecuar el cliente de mongo.exe y se encutras dentro de las bases de datos, se utliza la función db.auth(<user>,<passs>)
db.auth("guest","123456")