Práctica 3. Bases de datos documentales: MongoDB

Sistemas de información de nueva generación

6 de marzo de 2018

Objetivos

El objetivo de esta práctica es dotar al estudiante con la capacidad básica para trabajar en un entorno NoSQL. Las tareas que se van a realizar son:

- Operar con una base de datos de documentos, entendiendo documento como información semi-estructurada de parejas key/value.
- Crear la base de datos en sistemas sin esquema.
- Operaciones de inserción, actualización y borrado de información.
- Operaciones de consulta básicas y avanzadas.
- Operaciones con las bases de datos.

Índice

1 Introducción

MongoDB is a document database that provides high performance, high availability, and easy scalability.

- Document Database
  - Documents (objects) map nicely to programming language data types.
  - Embedded documents and arrays reduce need for joins.
  - Dynamic schema makes polymorphism easier.
- High Performance
  - Embedding makes reads and writes fast.
  - Indexes can include keys from embedded documents and arrays.
  - Optional streaming writes (no acknowledgments).
- High Availability
  - Replicated servers with automatic master failover.

- Easy Scalability
  - Automatic sharding distributes collection data across machines.
  - Eventually-consistent reads can be distributed over replicated servers.
- Advanced Operations
  - With MongoDB Management Service (MMS) MongoDB supports a complete backup solution and full deployment monitoring.

1.1 MongoDB Data Model

A MongoDB deployment hosts a number of databases. A database holds a set of collections. A collection holds a set of documents. A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

1.2 MongoDB Queries

Queries in MongoDB provides a set of operators to define how the find() method selects documents from a collection based on a query specification document that uses a combination of exact equality matches and conditionals using a query operator.

1.3 Deployment Architectures

Although MongoDB supports a "standalone" or single-instance operation, production MongoDB deployments are distributed by default. Replica sets provide high performance replication with automated failover, while sharded clusters make it possible to partition large data sets over many machines transparently to the users. MongoDB users combine replica sets and sharded clusters to provide high levels redundancy for large data sets transparently for applications.

1.4 MongoDB Design Philosophy

> *MongoDB wasn't designed in a lab. We built MongoDB from our own experiences building large scale, high availability, robust systems. We didn't start from scratch, we really tried to figure out what was broken, and tackle that. So the way I think about MongoDB is that if you take MySql, and change the data model from relational to document based, you get a lot of great features: embedded docs for speed, manageability, agile development with schema-less databases, easier horizontal scalability because joins aren't as important. There are lots of things that work great in relational databases: indexes, dynamic queries and updates to name a few, and we haven't changed much there. For example, the way you design your indexes in MongoDB should be exactly the way you do it in MySql or Oracle, you just have the option of indexing an embedded field.*
>
> —*Eliot Horowitz, MongoDB CTO and Co-founder*

- New database technologies are needed to facilitate horizontal scaling of the data layer, easier development, and the ability to store order(s) of magnitude more data than was used in the past.
- A non-relational approach is the best path to database solutions which scale horizontally to many machines.
- It is unacceptable if these new technologies make writing applications harder. Writing code should be faster, easier, and more agile.
- The document data model (JSON/BSON) is easy to code to, easy to manage(dynamic schema), and yields excellent performance by grouping relevant data together internally.

- It is important to keep deep functionality to keep programming fast and simple. While some things must be left out, keep as much as possible – for example secondaries indexes, unique key constraints, atomic operations, multi-document updates.

- Database technology should run anywhere, being available both for running on your own servers or VMs, and also as a cloud pay-for-what-you-use service.

1.5 Key MongoDB Features

- **Flexibility**

  MongoDB stores data in JSON documents (which we serialize to BSON). JSON provides a rich data model that seamlessly maps to native programming language types, and the dynamic schema makes it easier to evolve your data model than with a system with enforced schemas such as a RDBMS.

- **Power**

  MongoDB provides a lot of the features of a traditional RDBMS such as secondary indexes, dynamic queries, sorting, rich updates, upserts (update if document exists, insert if it doesn't), and easy aggregation. This gives you the breadth of functionality that you are used to from an RDBMS, with the flexibility and scaling capability that the non-relational model allows.

- **Speed/Scaling**

  By keeping related data together in documents, queries can be much faster than in a relational database where related data is separated into multiple tables and then needs to be joined later. MongoDB also makes it easy to scale out your database. Autosharding allows you to scale your cluster linearly by adding more machines. It is possible to increase capacity without any downtime, which is very important on the web when load can increase suddenly and bringing down the website for extended maintenance can cost your business large amounts of revenue.

- **Ease of use**

  MongoDB works hard to be very easy to install, configure, maintain, and use. To this end, MongoDB provides few configuration options, and instead tries to automatically do the "right thing" whenever possible. This means that MongoDB works right out of the box, and you can dive right into developing your application, instead of spending a lot of time fine-tuning obscure database configurations.

2 Objetivo de la práctica

En esta práctica se van a prácticar las funciones básicas de una base de datos orientada a documentos como es MongoDB. Las operaciones que se van a desarrollar son:

- Creación de la base de datos e introducción de los datos.

- Conceptos básicos de operaciones de actualización y borrado.

- Operaciones de consulta, con especial enfasis en las operaciones de agregación.

- Borrado de la base de datos.

3 Creación de la base de datos

El fichero `municipios.json.zip` contiene varios datos sobre los municipios de España, tales como la Comunidad Autónoma, la Provincia, el nombre de la población, sus coordenadas geográficas (latitud y longitud), la altitud, el número de habitantes varones, el número de mujeres y el número total de habitantes y finalmente, la lista de códigos postales asociados a la población. El formato utilizado es BSON y puede ser importado directamente en una base de datos MongoDB. Un ejemplo concreto de la estructura de la información se muestra a continuación:

```
{
  "comunidad" : "Valencia",
  "provincia" : "Valencia",
  "poblacion" : "Valencia",
  "localizacion" : {
    "latitud" : 39.47024,
    "longitud" : -0.376805
  },
  "altitud" : 23.335,
  "habitantes" : 814208,
  "varones" : 392300,
  "mujeres" : 421908,
  "zip" : [
      46001, 46002, 46003, 46004, 46005, 46006, 46007, 46008, 46009,
      46010, 46011, 46012, 46013, 46014, 46015, 46016, 46017, 46018,
      46019, 46020, 46021, 46022, 46023, 46024, 46025, 46026, 46070,
      46071, 46080, 46116
   ]
}
```

Ejercicio 1

Utilizar el comando de shell `mongoimport` para importar los datos en MongoDB. Utilizar como nombre de la base de datos el nombre de usuario (`sing_uXX`) y almacenar los datos en la colección `municipios`. Usar el comando de la shell `man mongoimport` para mostrar cuáles son las opciones de importación necesarias.

4 Operaciones de inserción, actualización y borrado

Ejercicio 2

Introducir la siguiente población ficticia en la base de datos:

```
{
  "comunidad" : "Valencia",
  "provincia" : "Valencia",
  "poblacion" : "Camelot",
  "localizacion" : {
    "latitud" :  0.000000
    "longitud" : 0.000000
  },
  "altitud" : 0.0,
  "habitantes" : 14,
  "varones" : 13,
  "mujeres" : 1,
  "zip" : [
      46999
   ]
 }
```

Ejercicio 3

El Rey Arturo se acaba de casar con Ginebra, por lo que el número de habitantes en Camelot se ha incrementado. Del mismo modo, Morgade ya no es la única mujer del reino. Actualizar los datos de Camelot: `{ "habitantes": 15, "mujeres": 2 }`.

Ejercicio 4

No todos los documentos contienen el campo `zip`. Actualizar los códigos postales de las siguientes poblaciones:

- **Vinaròs**, con el código postal 12500.
- **Callosa d'en sarrià** tiene el código postal 03510.

Ejercicio 5

Borrar el documento `{ poblacion : "Camelot" }`.

5 Operaciones de consulta

Ejercicio 6

Obtener la lista de todos los municipios de la provincia de `Barcelona` ¿Cuál es el número total de muncipios?

Ejercicio 7

- Obtener el nombre de todas las comunidades sin duplicados.
- Mediante una consulta similar, obtener el nombre de todas las provincias.

Ejercicio 8

Obtener el número total de habitantes de la Comunidad Valenciana. Obtener el número total de varones y total de mujeres de la Comunidad Valenciana.

Ejercicio 9

Obtener el número total de municipos de la provincia de Barcelona.

Ejercicio 10

Obtener todos los datos del municipio más alto de España.

Ejercicio 11

Encontrar el nombre de población de todos los documentos que no tienen código postal.

Ejercicio 12

Responder a las siguientes cuestiones:

- Nombre del municipio más septentrional.
- Provincia en la que se encuentra el municipio más occidental.
- Comunidad con el municipio más meridional.
- Comunidad y provincia que contiene el municipio más oriental.

Ejercicio 13

Obtener el listado de la población total por comunidades.

Ejercicio 14

Obtener la población total de España, la población total de varones y la población total de mujeres.

Ejercicio 15

Obtener el listado de las comunidades que tienen una población total mayor o igual a 4 millones de habitantes.

Ejercicio 16

Obtener la población promedio de los municipios agrupada por provincia.

Ejercicio 17

Obtener un listado con la ciudad más grande y su número de habitantes y la población más pequeña y su número de habitantes agrupado por comunidad.

Ejercicio 18

Repetir el ejercicio anterior, pero agrupado por provincia.

Ejercicio 19

Obtener el nombre y el número de municipios de la provincia con menos municipios y el nombre y número de municipios de la provincia con más municipios.

6 Borrado de la base de datos

Ejercicio 20

Borrar la base de datos.

7 Entrega

El material a entregar consiste en una memoria con los resultados obtenidos:

- las sentencias utilizadas y
- la salida generada por la base de datos.

Fecha de entrega: antes del 20 de Marzo de 2018