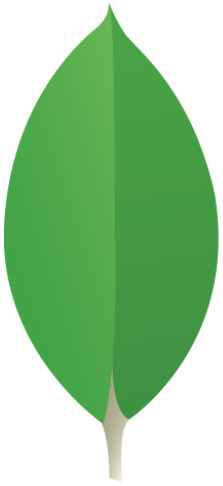

mongoDB®

Víctor Custodio

Introduccion

- **MongoDB** (de la palabra en ingles “hu**mongo**us” que significa enorme) es un sistema de base de datos NoSQL orientado a documentos
  - MongoDB guarda estructuras de datos en documentos tipo BSON (*Binary JSON* (JSON Binario) con un esquema dinámico , haciendo que la integración de los datos en ciertas aplicaciones sea mas fácil y rápida

Introducción

- Lo que MongoDB no puede hacer:
  - No hay tablas de BBDD
  - No hay “joins”
  - No hay transacciones
  - Y MongoDB no usa esquemas de datos

Características Principales

- Modelo de datos basado en documentos
  - Frente al modelo de datos relacional
- Consultas ad hoc
- Índices secundarios
- Replicación
- Velocidad y durabilidad
- Escalabilidad

Características Principales

- **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.

Características Principales

- **Replicación**
  - MongoDB soporta el tipo de **replicación maestro-esclavo.**
    - El maestro puede ejecutar comandos de lectura y escritura.
    - El esclavo puede copiar los datos del maestro y sólo se puede usar para lectura o para copia de seguridad, pero no se pueden realizar escrituras.
    - El esclavo tiene la habilidad de poder elegir un nuevo maestro en caso del que se caiga el servicio con el maestro actual.

Características Principales

- **Balanceo de carga**
  - MongoDB se puede escalar de forma horizontal usando el concepto de “shard”.
  - El desarrollador elige una llave shard, la cual determina cómo serán distribuidos los datos en una colección. Los datos son divididos en rangos (basado en la llave shard) y distribuidos a través de múltiples shard.
  - Un shard es un maestro con uno o más esclavos.
  - MongoDB tiene la capacidad de ejecutarse en múltiple servidores, balanceando la carga y/o duplicando los datos para poder mantener el sistema funcionando en caso que exista un fallo de hardware.
- **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 (que es llamada GridFS) está incluida en los drivers de MongoDB y disponible para los lenguajes de programación que soporta MongoDB.

Casos de Uso

- Almacenamiento y registro de eventos
- Para sistemas de manejo de documentos y contenido
- Comercio Electrónico
- Juegos
- Problemas de alto volumen
- Aplicaciones móviles
- Almacén de datos operacional de una página Web
- Manejo de contenido
- Almacenamiento de comentarios
  - Votaciones
  - Registro de usarios
  - Perfiles de usuarios
  - Sesiones de datos
- Proyectos que utilizan metodologías de desarrollo iterativo o ágiles
- Manejo de estadísticas en tiempo real

Colecciones y Documentos

- 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 son llamados **documentos** y se guardan en **colecciones**
- Una colección puede tener un número indeterminado de documentos
  - Las **colecciones son como tablas** y los **documentos como filas**
  - Cada documento en una colección puede tener diferentes campos.
- La estructura de un documento es simple y compuesta por “key-value pairs” parecido a las matrices asociativas en un lenguaje de programación.
- Como valor se pueden usar números, cadenas o datos binarios como imágenes o cualquier otro “key-value pairs”.

6 Conceptos Clave

- 1.MongoDB tiene el concepto de **“base de datos”** con el que estamos familiarizados (schema en el mundo relacional).
  - Dentro de un servidor MongoDB podemos tener 0 o más BBDD, cada una actuando como un contenedor de todo lo demás.
- 2.Una base de datos puede tener una o más **“colecciones”**, equivalente en el mundo relacional a una “tabla”.
- 3.Las colecciones están hechas de 0 o más **“documentos”**, donde un documento puede considerarse equivalente a una fila de una tabla de un RDBMS.
- 4.Un documento está compuesto de uno o varios “**campos**” que son equivalentes a las columnas de una fila.
- 5.Los “**índices**” en MongoDB funcionan como los de las RDBMS.
- 6.Los “**cursores”** son utilizados para acceder progresivamente a los datos recuperados con una consulta
  - Pueden usarse para contar o moverse hacia delante entre los datos

Comenzando con MongoDB

- 1º Instalación -> https://www.mongodb.com/
- 2º Ejecutamos el servidor de base de datos de MongoDb (carpeta bin de instalación mongod).
- 3º desde una consola, escribimos mongo.
  - Ya estamos en el Shell de mongodb.

Comenzando con MongoDB

- Para mostrar todas las base de datos que tenemos en el servidor ejecutamos el comando:

```
show dbs
```

Para crear base de datos seguimos los siguientes pasos:

```
use nombredb
```

- Donde nombredb es el nombre de la base de datos que vamos a crear, al ejecutar este comando aun no se crea la base de datos, ya que necesitamos insertar como mínimo un dato para crear la base de datos.

Comenzando con MongoDB

- Para eliminar la base de datos seleccionada

```
db.dropDataBase()
```

- Para mostrar las colecciones que tenemos en una db:

```
show collections
```

```
db.getCollectionNames()
```

CRUD:Insertar

- Para insertar valores en una colección, no es necesario que la colección este creada, simplemente escribimos:

```
db.micoleccion.insert({"nombre":"Victor","apellido":"Custodio","domicilio":"Gran via 52"})
```

- Los datos tienen un formato tipo JSON, la estructura básica de una inserción es {“clave”:”valor”}, donde las diferentes relaciones clave-valor están separadas por comas “,”.
- Al hacer cada inserción, mongodb automáticamente generara un ID unico, similar al “auto_increment primary key”, pero este ID es del tipo ObjectId de mongodb y es alfanumérico, muy similar a sha1 o md5.

CRUD:Insertar

- Ejemplo de un documento en MongoDB


```
{
"_id": ObjectId("4efa8d2b7d284dad101e4bc7"),
"Last Name": "PELLERIN",
"First Name": "Frank",
"Age": 29,
"Address": {
"Street": "1 chemin des Loges",
"City": "VERSAILLES"
}
}
```

CRUD:Insertar

- Los documentos se almacenan en formato **BSON**, o Binary JSON
- BSON es una versión modificada de JSON que permite búsquedas rápidas de datos.
  - BSON guarda de forma explícita las longitudes de los campos, los índices de los arrays, y demás información útil para el escaneo de datos.
- El mismo documento en BSON ocupa un poco más de espacio de lo que ocuparía de estar almacenado directamente en formato JSON. (almacenamiento es barato)

CRUD: Recuperar

- Para mostrar todos los datos de una colección hacemos lo siguiente:

```
db.micoleccion.find()
```

- Seria lo mismo que ejecutar en SQL “select * from micollecion”, es decir nos muestra todos los datos.
- Para filtrar o hacer algo similar al where:

```
db.micoleccion.find({"apellido":"Ramos"})
```

- en este caso filtrare todos los resultados en los que el campo apellido sea igual a Ramos:

CRUD: Recuperar

- También se puede especificar con mayor flexibilidad que documentos, o campos de los documentos queremos con ayuda de operadores, por ejemplo:

```
db.people.find({age:{$gt:30}},{name:1,age:1})
```

- El operador $gt (greater than) filtrará para los mayores de 30.
- El segundo json indica los campos que queremos recuperar
- Ver más aquí.

CRUD: Editar

- Para actualizar un registro, ejecutamos:

```
db.micoleccion.update(
{"_id":ObjectId("56c9a1ffbb6e73925f958b1a")},
{$set:{"apellido":"Ramos Escalante"}})
```

- Primero indicamos condición de busqueda, después con $set es el operador de asignación. Si el campo no existe en el documento, lo crea, si existe sobrescribe su valor.

CRUD: Eliminar

- Para eliminar algunos registros de una colección hacemos lo siguiente:

```
db.micoleccion.remove({"_id":ObjectId("56c9a1ffbb6e73925f958b1a")})
```

- Se eliminan todos los registros que coincidan.
- Para eliminar todos los registros de una colección:

```
db.micoleccion.remove()
```