Diseño de Bases de Datos Multimedia

Conceptos básicos de bases de datos NoSQL (I).

Grado en Ingeniería Multimedia

Universitat d'Alacant
Universidad de Alicante

Departamento de
Lenguajes y Sistemas Informáticos

ÍNDICE

Objetivos

- **Dar al alumno una visión del diseño de las bases de datos más allá del modelo relacional.**
- **Proponer herramientas para el desarrollo rápido de soluciones que necesiten persistencia en la información.**

Introducción a NoSQL

- **NoSQL** – "not only SQL" – es una categoría general de sistemas de gestión de bases de datos que:
  - No tienen esquemas de definición de datos predefinidas.
  - Fijan sus prioridades en la escalabilidad y la disponibilidad en contra de la atomicidad y consistencia de los datos
  - Tanto las bases de datos NoSQL como las relacionales son tipos de **Almacenamiento Estructurado**.

- El término fue acuñado en 1998 por Carlo Strozzi y resucitado en 2009 por Eric Evans
  - Evans sugiere mejor referirse a esta familia de BBDD de nueva generación como **"Big Data"** mientras que Strozzi considera ahora que **NoREL** es un mejor nombre.

Introducción a NoSQL

- La **principal diferencia radica en cómo guardan los datos**:
  - En un SGBDR tendríamos que partir la información en diferentes tablas (normalizar), y luego usar el lenguaje SQL para transformar estos datos en objetos de la vida real.
  - En NoSQL, simplemente guardas los datos:
    - NoSQL está **libre de esquemas:** no necesitas diseñar la estructura por adelantado. Permite añadir campos libremente sin tener que modificar ninguna estructura previamente definida.

- **¡¡¡ NoSQL no es la panacea !!!**
  - Si tus datos son relacionales, y tu sistema necesita control riguroso de transacciones (ACID), elegir un SGBDR sería la opción correcta.
  - Si no eres riguroso en NoSQL , tu sistema de base de datos puede ser caótico e ineficiente.

Ventajas de NoSQL

- **Se ejecutan en máquinas con pocos recursos:** Estos sistemas, a diferencia de los sistemas basados en SQL, no requieren de apenas computación, por lo que se pueden instalar en máquinas de un coste más reducido.

- **Replicación y particionado:** mejorar el rendimiento de estos sistemas simplemente se consigue añadiendo más nodos (replicación) y/o distribuyendo la información entre ellos según criterios pre-establecidos (particionado) , con la única operación de indicar al sistema cuáles son los nodos que están disponibles y definimiento reglas de dónde queremos alojar la información.

- **Pueden manejar gran cantidad de datos:** Esto es debido a que utilizan una estructura distribuida, escalable fácilmente de manera horizontal.

- **No genera cuellos de botella:** El principal problema de los sistemas SQL es que necesitan transcribir cada sentencia para poder ser ejecutada, y cada sentencia compleja requiere además de un nivel de ejecución aún más complejo, lo que constituye un punto de entrada común, que ante muchas peticiones puede ralentizar el sistema.

Tipos de Bases de Datos noSQL

- **Clave-Valor:** Son las bases de datos NoSQL más simples. Cada elemento de la base de datos se almacena con un nombre de atributo (o clave) junto a su valor
- **Documentales:** Es similar al clave valor, pero cada clave se asocia a una estructura compleja que se conoce como documento. Éste puede contener diferentes pares clave-valor, o pares de clave-array o incluso documentos anidados, como en un documento JSON. Los ejemplos más conocidos son MongoDB y CouchDB. (SON LAS QUE ESTUDIAREMOS)
- **Grafos:** se usan para almacenar información sobre redes, como pueden ser conexiones sociales. Son complejas pues almacenan también las relaciones de interés entre la información. (Neo4J, HyperGraphDB, etc.)
- **Columnas:** almacenan los datos como columnas, en vez de como filas, de modo que cada fila puede contener un número diferente de columnas. Optimizados para consultas sobre grandes conjuntos de datos. So las mas complicadas de manejar (Cassandra, Bigtable, etc .)

¿Cuándo usar NoSQL?

- Cuando el volumen de entrada de datos crece muy rápidamente en momentos puntuales, pudiendo llegar a superar el Terabyte de información en cuestión de minutos.
- Cuando la escalabilidad de la solución relacional no es viable tanto a nivel de costes como a nivel técnico.
- Cuando nuestro sistema NO requiera de un control exhaustivo de transaccionalidad.
- Cuando importa más la flexibilidad, la velocidad, y la capacidad de escalado horizontal que otras cuestiones tradicionalmente cruciales como la consistencia o disponer de una estructura perfectamente definida para los datos.
- Cuando el esquema de la base de datos no es homogéneo, es decir, cuando en cada inserción de datos la información que se almacena puede tener campos distintos.

Caso de uso: FIFA ONLINE 3.

https://www.mongodb.com/blog/post/ea-scores-mongodb-based-fifa-online-3
http://fo3.garena.com/home

MongoDB

- **MongoDB** (de la palabra en ingles "hu**mongo**us" que significa enorme) es un sistema de base de datos NoSQL orientado a documentos.

- MongoDB es de tipo documental, y **guarda las estructuras de datos en documentos tipo BSON** (*Binary JSON, o* JSON Binario) con un esquema dinámico, haciendo que la integración de los datos en ciertos tipos de aplicaciones sea mas fácil y rápida.

http://db-engines.com/en/ranking

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 pueden 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 compuestos.
    - El concepto de índice en MongoDB es similar al encontrado en base de datos relacionales.
- **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
      - El esclavo tiene la habilidad de poder elegir un nuevo maestro en caso de 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".
  - MongoDB tiene la capacidad de ejecutarse en múltiple servidores, balanceando la carga y/o duplicando los datos .
- **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 almacenamiento
  - GridFS es una librería de manejo de LOB que está incluida en los drivers de MongoDB y disponible para los lenguajes de programación que soporta MongoDB.
- **Agregación**
  - La función MapReduce puede ser utilizada para el procesamiento por lotes de datos y operaciones de agregación.
    - Permite que los usuarios puedan obtener el tipo de resultado que se obtiene cuando se utiliza el comando SQL "group-by".
- **Ejecución de JavaScript del lado del servidor**

Manipulación 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 llaman **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 en un Sistema de bases de datos relacionales.**
  - Cada documento en una colección puede tener diferentes campos.
- La estructura de un documento es simple y compuesta por "key-value pairs". Como valor se pueden usar números, cadenas o datos binarios (imágenes, vídeos, etc., con un límite máximo de 16 Megabytes por documento).

Manejo de Datos

- **MongoDB**, a través de JSON, puede utilizar los siguientes tipos de datos:
  - String: guardados en UTF-8. Van siempre entre comillas dobles.
  - Number: números. Al guardarse en BSON pueden ser de tipo byte, int32, int64 o double.
  - Boolean: con valor true o false.
  - Array: van entre corchetes [] y pueden contener de 1 a N elementos, que pueden ser de cualquiera de los otros tipos.
  - Documentos: un documento en formato JSON puede contener otros documentos embebidos que incluyan más documentos o cualquiera de los tipos anteriormente descritos.
  - Null.

Referencias

- MongoDB
  - http://www.mongodb.com/
- MongoDB University
  - https://education.mongodb.com/
- Robert Stam. Introducción a NoSQL y MongoDB Webinar
  - http://www.mongodb.com/presentations/introducci%C3%B3n-nosql-y-mongodb-webinar
- MongoDB University. Curso M102: MongoDB for DBAs
  - https://university.mongodb.com/courses/M102/about
- MongoDB Manual
  - http://docs.mongodb.org/manual
- Tutorial básico de MongoDB
  - http://www.charlascylon.com/mongodb

Guion de la práctica

- Actividad 1: Configuración preliminar.
- Actividad 2: Puesta en marcha del sistema.
  - Ejercicio 1: Arranque del servidor.
  - Ejercicio 2: Monitorizar el servidor mediante un navegador Web.
  - Ejercicio 3: Instalar el shell de MongoDB.
- Actividad 3: Sentencias CRUD en MongoDB.
  - Ejercicio 4: Incorporar la colección (CREATE).
  - Ejercicio 5: Ejecución de consultas sencillas (READ).
  - Ejercicio 6: Ejecución de inserciones (INSERT).
  - Ejercicio 7: Modificando información (UPDATE).
  - Ejercicio 8: Borrando información (DELETE).
- Actividad 4: Relacionando la información.
  - Ejercicio 9: Vincular imágenes a documentos.
  - Ejercicio 10: Consulta de documentos heterogéneos.
- Actividad 5: Extracción de información multimedia.
  - Ejercicio 11: Obtener archivo desde BD hacia el Sistema operativo.

Guion para los ejercicios

- Actividad 3: Carga de información en la BD.
  - Ejercicio 4: Incorporar la colección (CREATE).

  El comando que debéis usar es "mongoimport". Está en la carpeta:

  c:\program files\MongoDB\Server\3.2\bin

  Su sintaxis es:

Mongoimport –-db BASEDATOS –-collection COLECCIÓN –-file FICHERO –-jsonArray

Donde:

BASEDATOS = Nombre de la base de datos destino de la información.
COLECCIÓN = Nombre de la colección en BASEDATOS donde se guardarán los datos.
FICHERO = Nombre del fichero que incluye la información a guardar (entre comillas dobles).
--jsonArray = Indica que el formato de los datos del FICHERO es tipo Json.

Comandos MongoDB

- Actividad 3: Comandos "CRUD".
  - Ejercicio 5: Ejecutar consultas (READ).
    - Seleccionar una base de datos:

      > **use BASEDATOS**

    - Leer información de la base de datos (similar a SELECT de SQL):

      > **db.COLECCIÓN.find({FILTRO},{MOSTRAR}).pretty()**

    Donde:

    FILTRO = COLUMNA: VALOR
    MOSTRAR = COLUMNA1:1, COUMNA2:1, COLUMNA3: 1 …

Comandos MongoDB

- Actividad 3: Comandos "CRUD".
  - Ejercicio 6: Ejecutar inserciones (INSERT).
  - Insertar documentos en una COLECCIÓN (similar a INSERT de SQL):

    > **db.COLECCIÓN.insert({DATOS})**

    Donde:

    DATOS = COLUMNA1: VALOR1, COLUMNA2:VALOR2 …

Comandos MongoDB

- Actividad 3: Comandos "CRUD".
  - Ejercicio 7: Ejecutar modificaciones (UPDATE).
  - Modificar documentos en una COLECCIÓN (similar a UPDATE de SQL):

    > **db.COLECCIÓN.update({CONDICIONES},{DATOS})**

    Donde:

    CONDICIONES = COLUMNA1: VALOR1, COLUMNA2:VALOR2 …
    DATOS = COLUMNA1: VALOR1, COLUMNA2:VALOR2 …

Comandos MongoDB

- Actividad 3: Comandos "CRUD".
  - Ejercicio 8: Borrar documentos (DELETE).
  - Borrar documentos de una COLECCIÓN (similar a DELETE de SQL):

    > **db.COLECCIÓN.remove({CONDICIONES})**

    Donde:

    CONDICIONES = COLUMNA1: VALOR1, COLUMNA2:VALOR2 …

MongoDB. Guía rápida

- Manejar bases de datos desde Shell
  - muestra todas BDs:
    - `> show dbs`
  - muestra bd activa:
    - `> db`
  - muestra colecciones en la bd activa
    - `> show collections`
  - eliminar bd activa
    - `> db.dropDatabase()`

MongoDB. Guía rápida

- Manejar colecciones desde Shell
  - Insertar (crea la colección si no existe):
    - Insertar nuevo objeto (documento):
      - `db.micoleccion.insert({`*`objeto JSON…`*`})`
    - Insertar o actualizar si existe (según _ID):
      - `db.micoleccion.save({`*`objeto JSON…`*`})`
  - Consultar:
    - Contar todos:
      - `db.micoleccion.count()`
    - Mostrar todos:
      - `db.micoleccion.find()`

MongoDB. Guía rápida

- Manejar colecciones desde Shell
  - Seleccionar db.collection.find(**<criteria>, <projection>**)
    - Seleccionar:
      - `db.micoleccion.find({query})`
    - Seleccionar 1:
      - `db.micoleccion.findOne({query})`
    - Seleccionar n (ej.9):
      - `db.micoleccion.find({query}).limit(9)`

MongoDB. Guía rápida.

- Manejar colecciones desde Shell
  - Ejemplos query
    - Documento a buscar
      - `{“codigo”:”A33”, “precio”:{“valor”:10, “moneda”:”euro”}}`
    - Query
      - `{“codigo”:”A33”}` (SQL where codigo=“A33”)
      - `{“precio.valor”:10}` (SQL join)
      - `{“precio.valor”:10, “precio.moneda”:”euro”}` (AND)
      - `{“precio.valor”:{$gte:10}}` (precio.valor>=10)

MongoDB. Guía rápida

- Manejar colecciones desde Shell
  - Operadores $ (ver más en MongoDB manual)
    - `$gte` (>= greater than|equal) `{"precio.valor":{$gte:10}}`
    - `$gt` (> greater than) `{"precio.valor":{$gt:10}}`
    - `$lt` (< lower than) `{"precio.valor":{$lt:10}}`
    - `$lte` (<= lower than|equal) `{"precio.valor":{$lte:10}}`
    - `$or` (OR) `{$or [{"precio.valor":10},{precio.moneda:"euros"}]}`
    - `$not` (NOT)
    - `$exists` (EXISTS) `{"precio.valor":{$exists:true}}`
    - Combinaciones: `{"precio.valor":{$gte:10, $lte:100}}`

MongoDB. Guía rápida

- Manejar colecciones desde Shell
  - Consultar:
    - Contar:
      - db.micoleccion.find({*query*}).count()
    - Seleccionar y proyectar:
      - db.micoleccion.find({*query*},{*proyeccion*})
      - Proyeccion se define como query pero asignando valor:1 para incluir ese campo, o 0 para excluirlo.
      - > db.micoleccion.find({"codigo":"A33"}{"precio.valor":1, "precio.moneda":1} (Muestra valor y moneda)
    - Ordenar
      - db.micoleccion.find({*query*},{*proyeccion*}).sort({*campo*:1})

MongoDB. Guía rápida.

- Manejar colecciones desde Shell
  - Consultar:
    - Agregar (SQL group by):

```
db.micoleccion.aggregate(
    {$match:{query}},
    {$group:{
        _id: "$groupbyfield"},
        resultado: {aggregatefunction}) } } )
```

    - Ejemplo: Count de artículos agrupando por cliente

```
db.articulos.aggregate({$match:{}},{$group:{_id:"$cliente",cuenta:{$sum:1}}});
```