Sistemas de Gestión de Datos y de la Información
Máster en Ingeniería Informática
Práctica 3

Fecha de entrega: viernes 22 de enero de 2016, 16:55h

Objetivos mínimos
Diseñar el esquema implícito adecuado para una base de datos MongoDB a partir de la descripción de los datos y las consultas. Realizar modificaciones y consultas básicas en MongoDB desde Python utilizando pymongo.

Entrega de la práctica
La práctica se entregará en un único fichero **GrupoXX.zip** mediante el Campus Virtual de la asignatura. Este fichero **ZIP** contendrá:

- A) Un **documento PDF** detallando el esquema implícito, su razonamiento y ejemplos de cada tipo de documento que aparece en el esquema implícito. Además contendrá una carpeta llamada `data` con un fichero **JSON** por cada colección con los contenidos iniciales de cada una.
- B) Fichero `consultas.py` con las funciones Python necesarias para realizar las consultas y modificaciones que se detallan en el enunciado.
- C) Fichero `consultas.js` con las funciones JavaScript necesarias para realizar las consultas avanzadas utilizando el *aggregation framework* y *MapReduce*.

Lenguaje de programación
Python 2.7

Calificación
Los apartados A y C tienen un peso del 30 % cada uno. El apartado B tiene un peso del 40 %. Además de la corrección se valorará también la claridad y la organización del código y los documentos de texto.

Declaración de autoría
Todos los ficheros entregados contendrán una cabecera en la que se indique la asignatura, la práctica, el grupo, los autores y una declaración de integridad expresando que el documento es fruto exclusivamente del trabajo de sus miembros. No se permite ningún tipo de colaboración entre distintos grupos.

En esta práctica consideraremos una web de preguntas y respuestas sobre tecnología similar a `http://stackoverflow.com`. Esta web almacenará toda la información necesaria de los usuarios y de las entradas.

Sobre los **usuarios** queremos almacenar al menos la siguiente información: alias de usuario (único), nombre, apellidos, experiencia (tecnologías en las que es especialista), fecha en la que creó su cuenta de usuario y dirección (al menos su país, cuidad y código postal).

Con respecto a las entradas consideraremos 3 tipos. Las **entradas principales** serán preguntas formuladas por los usuarios. Estas preguntas constarán de un título, una serie de *tags* relativos a la pregunta, una fecha y un texto. Las preguntas pueden recibir **contestaciones** de otros usuarios, que contendrán la fecha en la que se realizó la contestación y un texto. Finalmente las contestaciones (no las preguntas) pueden recibir **comentarios** de otros usuarios con un texto y una fecha. Además se permite que los usuarios **puntúen** las contestaciones (no así las preguntas) como *buena* o *mala* para que otros usuarios puedan detectar rápidamente las contestaciones más interesantes.

Las operaciones que queremos soportar son las siguientes:

1. Añadir un usuario.
2. Actualizar un usuario.
3. Añadir una pregunta.
4. Añadir una respuesta a una pregunta.
5. Comentar una respuesta.
6. Puntuar una respuesta.
7. Modificar una puntuación de *buena* a *mala* o viceversa.
8. Borrar una pregunta junto con todas sus respuestas, comentarios y puntuaciones.
9. Visualizar una determinada pregunta junto con todas sus contestaciones y comentarios. A su vez las contestaciones vendrán acompañadas de su número de puntuaciones *buenas* y *malas*.
10. Buscar preguntas con unos determinados *tags* y mostrar su título, su autor y su número de contestaciones.
11. Ver todas las preguntas o respuestas generadas por un determinado usuario.
12. Ver todas las puntuaciones de un determinado usuario ordenadas por fecha. Este listado debe contener el título de la pregunta original cuya respuesta se puntuó.
13. Ver todos los datos de un usuario.
14. Obtener los alias de los usuarios expertos en un determinado tema.

15. Visualizar las n preguntas más actuales ordenadas por fecha, incluyendo el número de contestaciones recibidas.

16. Ver n preguntas sobre un determinado tema, ordenadas de mayor a menor por número de contestaciones recibidas.

Como características adicionales de esta web que pueden influir en el diseño de la base de datos y sus consultas tenemos que:

- El número de contestaciones a una pregunta no está acotado.
- Deseamos que las contestaciones puedan tener contenido multimedia si es necesario, por lo que su tamaño puede ser grande.
- Los usuarios son muy participativos, por lo que esperamos que cada contestación obtenga un número muy elevado de puntuaciones.
- La web recibirá muchas más peticiones de lectura (consultar preguntas y sus respuestas) que de escritura (añadir entradas o modificar puntuaciones).
- Es tolerable un poco de inconsistencia temporal con respecto al número de contestaciones recibidas por una pregunta o la puntuación concreta de una contestación.

Esquema implícito [Objetivo mínimo, 30 %]

Diseñar el **esquema implícito** de una base de datos MongoDB para almacenar toda la información necesaria para la web de preguntas y respuestas. El esquema implícito debe contener:

1. Descripción de las colecciones que existirán.
2. Por cada colección, descripción detallada de los distintos tipos de documentos que almacenará. Por cada documento distinto hay que detallar el nombre de sus atributos y su tipo.
3. Por cada tipo de documento diferente, incluir un ejemplo con valores concretos.

Todo el esquema implícito debe entregarse como un único **documento PDF**. Además de la descripción de las colecciones y documentos es **imprescindible** incluir el razonamiento acerca de la adecuación del esquema escogido al problema concreto. Este razonamiento debe explicar las decisiones tomadas, incluyendo por qué se ha decidido anidar o referenciar, por qué se evita o se permite cierto grado de replicación, etc. También se deben incluir y resaltar otras consideraciones que no aparezcan expresamente en el enunciado y que hayáis tenido en cuenta porque tienen impacto en el esquema implícito. Además del PDF debéis incluir un directorio `data` que contenga un fichero **json** con el contenido inicial de cada colección considerada. Podéis exportar colecciones completas a ficheros **json** utilizando el binario `mongoexport` que está incluido en el directorio `bin` de MongoDB.

MongoDB desde Python [Objetivo mínimo, 40 %]

Implementar en Python las operaciones de consulta y modificación presentadas anteriormente. Por cada consulta se creará un función Python que acepta como parámetros los valores necesarios para realizar la consulta/modificación y se comunica con MongoDB utilizando la librería `pymongo`. Como pretendemos utilizar estas funciones en una API REST, todas las funciones deben devolver una **cadena de texto** representando los resultados como **documentos JSON**. *Para la actualización de un usuario suponed que recibís todos los atributos del usuario y los actualizáis a la vez, es decir, no es necesario tener varias funciones para actualizar el nombre, el apellido, añadir una tecnología, etc.*

Por razones de manejabilidad todas las funciones deberán aparecer en el mismo fichero `consultas.py` cuyo esqueleto os podéis descargar del Campus Virtual. Además, todas las consultas/modificaciones deberán realizarse en la base de datos **sgdi_grupoXX** donde **XX** es el número de grupo.

Aggregation Pipeline y MapReduce [30 %]

Además de las consultas usuales, MongoDB admite dos mecanismos para realizar consultas complejas. El primero de ellos es el *aggregation pipeline* (`https://docs.mongodb.org/manual/core/aggregation-pipeline/`), que permite aplicar distintos procesamientos de manera ordenada sobre una colección. Por otro lado MongoDB soporta *MapReduce* (`https://docs.mongodb.org/manual/core/map-reduce/`), con un comportamiento similar al que hemos explicado durante el curso (*¡pero revisad bien la documentación que hay sorpresas!*).

En este apartado implementaremos consultas *avanzadas* utilizando estos mecanismos más complejos. Todas las consultas se realizarán sobre una colección de usuarios básica de nombre **agg** dentro de la colección **sgdi**. Esta colección la podéis importar usando el binario `mongoimport` a partir del fichero `agg.json` que podéis descargar del Campus Virtual. Por simplicidad vamos a centrarnos únicamente en cómo expresar las consultas, por lo que vamos a dejar Python aparcado y realizaremos las consultas directamente en el *shell* de MongoDB (léase JavaScript). Por tanto en este apartado se entregará un único fichero `consultas.js` cuyo esqueleto se puede descargar del Campus Virtual. Este fichero contiene una función por cada consulta, aunque podéis declarar las funciones auxiliares que consideréis necesarias.

Consultas con *Aggregation Pipeline*

1. Listado de *país-número de usuarios* ordenado de mayor a menor por número de usuarios.

2. Listado de *país-número total de posts* de los 3 países con mayor número total de posts, ordenado de mayor a menor por número de *posts*.

3. Listado de *afición-número de usuarios* ordenado de mayor a menor número de usuarios.

4. Listado de *afición-número de usuarios* restringido a usuarios españoles y ordenado de mayor a menor número de usuarios.

Consultas con MapReduce

1. Listado de *afición-número de usuarios* restringido a usuarios españoles.
2. Listado de *número de aficiones-número de usuarios*, es decir, cuántos usuarios tienen 0 aficiones, cuántos una afición, cuántos dos aficiones, etc.
3. Listado de *país-número de usuarios que tienen más posts que contestaciones*.
4. Listado de *país-media de posts por usuario*.