SISTEMA DE GESTIÓN DE DATOS Y DE INFORMACIÓN

TEMA 2

Bases de datos NoSQL

1º La base de datos MongoDB

- MongoDB es una base de datos orientada al almacenamiento de documentos complejos, los cuales se pueden consultar y manipular desde el propio lenguaje. La gran versatilidad de los documentos documentos definidos en Mongo trae consigo un gran conste computacional al realizar consultas de tipo JOIN o MapReduce.
- Las bases de datos en MongoDB se organizan mediante colecciones, las cuales almacenan un conjunto de documentos definidos en formato JSON, cada uno de los cuales porta una clave única que le identifica.
- Cada uno de los documentos definidos en la base de datos se corresponde con la definición de un objeto JSON, los cuales se encuentran indicados entre llaves y están constituidos por un conjunto de parejas *clave : valor*, donde el valor puede ser:
  - UN número.
  - Una cadena de caracteres.
  - Un booleano (puede expresarse mediante true y false o con los números 1 y 0 respectivamente).
  - Un array de valores (encerrados entre corchetes y separados por comas).
  - Un objeto (definido entre llaves).
  - Null.
- Un documento JSON puede contener otros documentos como valor para alguna de sus claves, de modo que para acceder a alguna de las claves de un subdocumento deberemos utilizar la operación la nomenclatura “subdocumento”.”clave”.
- En el caso de que uno de los campos del documento sea un array, podemos realizar accesos mediante el índice si especificamos el mismo como si fuera un subcampo de dicho array, es decir: “nombreArray”.”indice”.

1.1 Métodos de la base de datos

- **db. list_collection_names():** Devuelve una lista con los nombres de todas las colecciones que conforman la base de datos.

1.2 Métodos de colecciones

- **db.colec.find(query, proyection, options):** Devuelve el conjunto de elementos de la colección que coinciden con la búsqueda realizada. Por defecto se muestran los elementos en bloques de 10 en 10.
  - **query (document)?:** Filtra los documentos a buscar en la colección mediante el uso de operadores de consulta. En el caso de no especificar nada o indicar un documento vacío, se devolverán todos los documentos de la colección.
  - **proyection (document)?:** Especifica los elementos de los documentos filtrados que serán devueltos mediante la consulta. En el caso de no especificar nada indicar un documento vacío, se devolverán todos los campos.
    - La selección se realiza mediante un documento {clave: Boolean, … clave: Boolean}, donde se expresan o todos los campos que se quieren retornar o todos los que no, es decir, sin mezclar booleanos Verdaderos y Falsos (con excepción de “_id”: false).

- **options (document)?:** Especifica opciones adicionales para la realización del filtrado, las cuales pueden modificar el comportamiento o los documentos devueltos.

– **db.colec.countDocuments(query, options):** Retorna el número de documentos de la colección que coinciden con la consulta especificada mediante los parámetros.

- **Query (document):** Consulta que filtrará los documentos de la colección. En el caso de no especificar un documento vacío, se devolverán todos los documentos de la colección.
- **Options (document)?:** Opciones adicionales que afectan al comportamiento del conteo.

– **db.colec.update(query, update, options):** Realiza una actualización a los documentos de la colección que son filtrados por el primer documento indicado como parámetro.

- **query (documet):** Consulta que filtrará los documentos a actualizar.
- **Update (document | pipeline):** Modificaciones a realizar en los elementos filtrados.
  - En el caso de querer remplazar el documento entero, únicamente necesitaremos especificar un nuevo documento en ese campo.
  - En el caso de únicamente querer modificar una parte del documento, necesitaremos especificarlo en este parámetro mediante el uso de algún operador como $set{}.
- **Options(document)?:** Opciones adicionales que afectan al comportamiento de la actualización. Algunas de estas son:
  - **upsert: Boolean:** Flag que le indica a la actualización que en el caso de no encontrar ningún documento que verifique el filtrado, añada un documento nuevo en su lugar. El nuevo documento añadido se corresponderá con el especificado en el capo update.

– **db.colec.remove(query):** Elimina todos aquellos elementos de la colección que son filtrados por la el documento indicado como parámetro.

- **query (documet):** Consulta que filtrará los documentos a eliminar.

– **db.colec.insert(document, ordered):** Inserta un nuevo documento en la colección.

- **Document (document | [document]):** documentos a insertar en la colección.
- **Ordered (Boolean)?:** Si es True, la inserción se realizará ordenada y en caso de haber un error se devolverán todos los documentos a partir de aquel donde se produjo. En el caso de ser False se realizará de forma desordenada y únicamente se devolverán los documento donde se produjeron errores.

– **db.colec.doc():** Elimina todos los documentos que conforman la colección.

– **db.colec.createIndex(key, options, commitQuorum):** Genera un índice asociado a un determinado parámetro de los documentos de una colección.

- **Key (document): In**dica los campos sobre los que se creará el índice. El valor asociado a cada uno de los campos indica la ordenación de los mismos:
  - **1:** Orden ascendente.
  - **-1:** Orden descendente.
- **Options(document)?:** Opciones adicionales que afectan al comportamiento del índice.
- **CommitQuorum (alfanumérico)?:** Número mínimo de documentos del índice que deben contener datos.

– **db.colec.aggregate(pipeline, options):** Agrega los resultados de un conjunto de operaciones realizadas sobre una misma colección, las cuales son indicadas como primer parámetro.

- **Pipeline ([documents]):** Secuencia de operaciones o etapas de agregación de datos. Dichas operaciones se realizan mediante el uso de operadores de agregación, cada uno de los cuales conforma un documento.
- **Options (document)?:** Opciones adicionales que afectan al comportamiento del la agregación.

1.3 Métodos de cursores

- **Cursor.prety():** Formatea la salida del cursor para que sea más fácil de leer.
- **Cursor.limit(number):** Limita el número de documentos devueltos por el cursor a tantos como los indicados como parámetro.
- **Cursor.skip(number):** Realiza un salto en los documentos devueltos por el cursor de la longitud indicada como parámetro.
- **Cursor.sort(document):** Ordena los documentos devueltos por el cursor utilizando como referencia el documento indicado como parámetro, en el cual se le indicará la clave por la cual de deberá realizar la ordenación. La ordenación puede ser {clave:1} para un orden ascendente o {clave:-1} para un orden descendente.
- **Cursor.count():** Retorna el número de documentos que han sido devueltos por el cursor.

1.4 Operadores

– Operadores aritméticos:

- **{ $eq:“value” }:** Coincide con valores que son iguales al valor especificado.
- **{ $ne:“value” }:** Coincide con todos los valores diferentes al valor especificado.
- **{ $in:“[value] }”:** Coincide con cualquiera de los valores especificados en una matriz.
- **{ $nin:“[value] }”:** No coincide con ningún valor especificado en una matriz.
- **{ $gt“number” }:** Coincide con valores que son mayores que el valor especificado.
- **{ $gte:“number” }:** Coincide con valores que son mayores o iguales que el valor especificado.
- **{ $lt:“number” }:** Coincide con valores que son menores que el valor especificado.
- **{ $lte:“number” }:** Coincide con valores que son menores o iguales a el valor especificado.

– Operadores lógicos:

  - **{ $and: [document] }:** Aplica la operación AND al resultado de todos los documentos indicados en el array, los cuales pueden ser documentos de operadores.
  - **{ $or: [document] }:** Aplica la operación OR al resultado de todos los documentos indicados en el array, los cuales pueden ser documentos de operadores.

– Operadores de evaluación:

  - **{ $expr: <consult> }:** Permite el uso de expresiones de agregación dentro de la consulta. Las expresiones de agregación nos permite acceder a los campos que componen los documentos mediante al especificación “$nombreCampo”

– Operadores de elementos:

  - **{ $exists: Boolean }:** Cuando es true, coincide con aquellos documentos para los cuales existe el campo sobre el que se aplica el operador, aunque este sea null. En el caso de ser false, coincidiría con aquellos para los cuales no existiese dicho campo.

– Operadores de arrays:

  - **{ $all: [value] }:** Coincide con aquel array que contenga al menos todos los elementos especificados como parámetros.

– Operadores de actualización:

  - **{ $set: { [field: value] } }:** Reemplaza los campos especificados como parámetros por los respectivos valores indicados.
  - **{ $inc: { [field: number } }:** Incrementa el valor numérico de los campos especificados tantas veces como el valor indicado para cada uno de dichos campo.

– Operadores de agregación se utilizan dentro del método de colección *agregate*. Algunos son:

  - **{ $group: { [_id: expresion, value: expresion, ...] } }:** Separa los documentos en grupos de acuerdo con la clave _id como parámetro. La clave puede ser un campo, grupo de campos o el resultado de una expresión. Podemos crear más campos cuyos valores sean el resultado de operar con todos los documentos agrupados mediante el uso de operadores.
  - **{ $sum: <expresion> }:** Calcula y retorna la suma colectiva de los valores numéricos filtrados según la expresión indicada como parámetro, ignorando los no numéricos.
  - **{ $avg: <expresion> }:** Calcula y retorna el valor medio colectivo de los valores numéricos filtrados según la expresión indicada como parámetro, ignorando los no numéricos.
  - **{ $addToSet: <expresion> }:** Devuelve un array con todos los valores únicos resultantes de aplicar la expresión indicada como parámetro al grupo de documentos. No se determina el orden de dichos documentos.
  - **{ $push: <expresion> }:** Devuelve un array con todos los valores resultantes de aplicar la expresión al grupo de documentos.

- **{ $unwind: <expresion> }:** Desglosa todos los valores existente para el campo indicado como parámetro (el cual suele ser un array) y crea un documento por cada uno de dichos valores. Los campos restantes de los documentos creados son completados con los respectivos valores que el documento original contiene.
- **{ $max: <expresion> }:** Retorna el valor máximo entre todos los valores resultantes de aplicar la expresión indicada como parámetro.
- **{ $max: <expresion> }:** Retorna el valor mínimo entre todos los valores resultantes de aplicar la expresión indicada como parámetro.
- **{ $project: <expresion> }:** Genera documentos compuestos por los campos indicados dentro de la expresión pasada como parámetro.
- **{ $macht: <expresion> }:** Filtra los documentos dejando pasar únicamente aquellos especificados en la expresión indicada como parámetro.
- **{ $first: <expresion> }:** Retorna el valor resultante de aplicar la expresión indicada como parámetro al primero documento del grupo. Solo tiene sentido utilizarlo cuando los documentos se encuentran en un orden definido.
- **{ $last: <expresion> }:** Retorna el valor resultante de aplicar la expresión indicada como parámetro al último documento del grupo. Solo tiene sentido utilizarlo cuando los documentos se encuentran en un orden definido.
- **{ $sort: <expresion> }:** Ordena los documentos de entrada conforme a la expresión indicada como parámetro. Esta expresión es de la forma: {field_1: value, field_2: value}, donde se ordenará de forma jerárquica respecto a los distintos campos indicados y donde el valor puede ser:
  - **1:** Ordenación ascendente.
  - **-1:** Ordenación descendente.
- **{ $limit: number }:** Limita el número de documentos devueltos por la agregación a tantos como los indicados como parámetro. Siempre se usa en combinación con el operador $short.
- **{ $skip: number }:** Realiza un salto en los documentos devueltos por la agregación de la longitud indicada como parámetro. Siempre se usa en combinación con el operador $short.
- **{ $out: value }:** Toma todos los elementos devueltos por la agregación y los retorna juntos en una colección, donde el valor pasado por parámetro será el nombre de la misma.
- **{ $bucket: {**

  groupBy: expression,

  boundaries: [lowerbound1, lowerbound2, ...],

  default: literal,

  output: { [output1: {<$accumulator expression>}] }

  } }: Agrupa los documentos en intervalos según la clave indicada como parámetro:
  - **groupBy:** Especifica la clave por la que queremos agrupar la agrupación.
  - **Boundaries:** Array que indica los límites por los que agrupar.
  - **default?:** Nombre del grupo en el que se incluirán los valores que no encajen.
  - **output:** Claves a incluir en la salida, si no se especifica ninguna se incluirá por defecto una suma de los respectivos elementos agrupados en cada colección.
- **{ $bucketAuto: {} }:** Igual que $bucket pero sin definir el límite de los intervalos que queremos crear. Expresamos cuantos intervalos queremos mediante buckets:”number” y mongo lo organiza.
- **{ $facet: { [outField:[<expresion>]] }:** Permite agrupar múltiples etapas de ejecución en una misma etapa.

2º La biblioteca Pymongo

- **insert_one(document, ...):** Inserta un documento nuevo en la colección. Devuelve el _id del documento insertado.
- **insert_many(documents, ordered=True, ...):** Inserta una lista de documentos. Devuelve una lista con todos los _id de los documentos insertados.
- **replace_one(filter, replacement, upsert=False, ...):** Reemplaza el primer documento que coincide con el filtro especificado por parámetro. El flag upsert indica que si el dato a modificar no se encuentra, se insertará uno nuevo con las especificaciones de la modificación.
- **replace_many(filter, replacement, upsert=False, ...):** Reemplaza uno o más documentos que coinciden con el filtro especificado por parámetro. El flag upsert indica que si el dato a modificar no se encuentra, se insertará uno nuevo con las especificaciones de la modificación.
- **update_one(filter, update, upsert=False, …):** Actualiza el primer documento que coincide con el filtro especificado por parámetro.
- **update_many(filter, update, upsert=False, ...):** Actualiza uno o más documentos que coinciden con el filtro especificado por parámetro.
- **delete_one(filter, ...):** Elimina el primer documento que coincide con el filtro especificado por parámetro.
- **delete_many(filter, ...):** Elimina uno o más documentos que coinciden con el filtro especificado por parámetro.
- **bulk_write(requests, ordered=True, ...):** Envía un lote de operaciones de escritura al servidor.
- **create_index([(field, pymongo.”option”)], unique=False):** Crea un nuevo índice asociado al campo indicado de la colección.

3º Neo4j y Cypher

3.1 Definición de entidades mediante Cyper

(“VARIABLE”: [“Etiqueta] { [“Propiedad”: “valor”] })

Los nodos representan un conjunto de valores equivalente a una tupla de bases de datos relacionales, los cuales se definen mediante propiedades. Estos nodos están identificador mediante una variable y ser o no de un determinado tipo o tipos. Estructura de un nodo:

- **Variable:** Valor con el que identificaremos al nodo.
- **Etiqueta:** Asigna un determinado tipo al nodo que estamos representando. Un mismo nodo puede tener asignadas varias etiquetas (estas no están vinculadas a ningún valor).
- **Propiedad:** Valores asociados al nodo, equivalentes a las columnas de una tupla en bases de datos relacionales. Un nodo puede tener asociados una cantidad indeterminada de propiedades.

(Origen) - [“VARIABLE”: [“Etiqueta] { [“Propiedad”: “valor”] } -> (Destino)

Las relaciones son entidades que definen una relación lineal y unidireccional entre dos nodos concretos, las cuales, al igual que los nodos, están identificadas por una variable, ser de uno o varios tipos y tener definidas una serie de propiedades. Estructura de un relación:

- **Origen:** Nodo desde el cual se establece la relación. La definición de este puede contener todas las características de la definición de un nodo de forma individual.
- **Destino:** Nodo hasta el cual se establece la relación. La definición de este puede contener todas las características de la definición de un nodo de forma individual.
- **Variable:** Valor con el que identificaremos la relación.
- **Etiqueta:** Asigna un determinado tipo a la relación que estamos representando. Una misma relación puede tener asignadas varias etiquetas (estas no están vinculadas a ningún valor).
- **Propiedad:** Valores asociados a la relación. Una relación nodo puede tener asociados una cantidad indeterminada de propiedades.

P = (Origen) - [“VARIABLE”: [“Etiqueta] { [“Propiedad”: “valor”] } -> (Destino)

Los caminos son un conjunto de relaciones que unen un determinado nodo origen con un destino concreto, los cuales son especificados en la definición del mismo. La definición de los caminos es muy similar a la de las relaciones, de modo que podemos expresar los mismos campos que cuando definimos una de estas.

- **P:** Se refiere al nombre que daremos al camino. Este campo es la diferencia entre definir una relación y un camino.
- **Origen:** Nodo desde el cual se establece la relación. La definición de este puede contener todas las características de la definición de un nodo de forma individual.
- **Destino:** Nodo hasta el cual se establece la relación. La definición de este puede contener todas las características de la definición de un nodo de forma individual.

3.2 Operaciones sobre la base de datos mediante Cypher

– **CREATE: $Nodo_1$, … , $Nodo_N$, $Relacion_1$, … $Relacion_N$** Operación que introduce en la base de datos aquellos nodos y relaciones expresadas como parámetro.

– **MERGE: $Nodo_1$, … , $Nodo_N$, $Relacion_1$, … $Relacion_N$:** Introduce en la base de datos los nodos y relaciones indicados como parámetros pero sin producir duplicados de aquellos ya existentes. Se pueden agregar los siguientes parámetros:

- **ON CREATE SET <expresion>?:** Ejecuta la expresión indicada como parámetro en el caso de que el elemento se introduzca con éxito en la base de datos (no exista previamente).
- **ON MATCH SET <expresion>?:** Ejecuta la expresión indicada como parámetro en el caso de que el elemento ya se encontrara previamente registrado en la base de datos, por lo cual no se ha podido introducir de nuevo.

– **RETURN <value>:** Retorna el valor introducido como parámetro. Se puede utilizar junto a varios atributos opcionales para modificar su comportamiento.

- **SKIP <number>?:** Realiza un salto en los datos mostrados de tantos resultados como el valor especificado por parámetro.
- **LIMIT <number>?:** Limita el número de resultados mostrados al valor indicado por parámetro.
- **ORDER BY <ASC | DESC>?:** Ordena los parámetros devueltos por orden ascendente o descendente según el valor indicado como parámetro.

– **MATCH <selecction>, RETURN <value>:** Realiza una consulta a base de datos en búsqueda de los elementos que cumplen la selección indicada como parámetro y devuelve los datos indicados en la clausula Return. Se pueden utilizar los siguientes atributos opcionales indicados entre las clausulas MATCH y RETURN:

- **WHERE <expresion>?:** Incluye una expresión que filtra los documentos seleccionados por la clausula MATCH. La expresión acepta una gran cantidad de operadores.
- **OPTIONAL MATCH <expresion>?:** Permite realizar una segunda operación MATCH dentro de la primera, la cual puede tener todos los parámetros opcionales requeridos por el usuario. Es equivalente a realizar la operación *outer join* en SQL.
- **WITH <ORDER BY | SKIP | LIMIT>?:** Permite realizar consultas MATCH agregadas, las cuales recibirán como entrada el tratamiento que realiza la operación WITH a la salida del MATCH que le precede.
- **SET?:**Clausula que permite modificar una propiedad de los datos devueltos por la consulta MATCH. En el caso de querer eliminar ducha propiedad, debemos insertar en ella el valor NULL.
- **[EXPLAIN]:** Muestra gráficamente los pasos seguidos para desarrollar la consulta MATCH.
- **[PROFILE]:** Muestra gráficamente los pasos seguidos para desarrollar la consulta MATCH pero con un mayor número de detalles respecto a las acciones realizadas en los mismos.

– **DELETE <selection>:** Eliminamos el elemento seleccionado de la base de datos, el cual puede ser tanto nodos como relaciones. No podremos eliminar nodos que formen parte de alguna relación, por lo que deberemos eliminar estas antes.

  - **[DETACH]:** Palabra clave indicada antes de la instrucción que elimina aquellas relaciones que formen parte del nodo indicado a eliminar.

– **CREATE INDEX ON <selection>:** Genera un índice sobre la selección de datos indicados como parámetro. Los índices mejoran el acceso de las consultas pero retrasan la inserción y modificación de los datos.

– **DROP INDEX ON <selection>:** Elimina el índice existente sobre la selección de datos indicados como parámetro.

3.3 Operaciones de agregación

- **COUNT:** Retorna el número de elementos devuelto por la consulta MATCH.
- **MAX:** Devuelve el valor máximo la propiedad indicada como parámetro entre todo el conjunto de elementos con los que opera.
- **MIN:** Devuelve el valor mínimo la propiedad indicada como parámetro entre todo el conjunto de elementos con los que opera.
- **SUM:** Devuelve el valor resultado de sumar la propiedad indicada para todos los elementos con los que opera.
- **AVG:** Devuelve el valor resultado de calcular la media aritmética de la propiedad indicada para todos los elementos con los que opera.
- **COLLECT:** Recopila las salidas de las consultas indicadas en una lista.

3.4 Funciones

- **exists():** Devuelve True si el patrón o propiedad indicada como parámetro existe en el grafo.
- **Relationship():** Devuelve las relaciones que conforman el camino indicado como parámetro.
- **Nodes ():** Devuelve los nodos que conforman el camino indicado como parámetro.
- **Type ():** Devuelve el tipo de la relación indicada como parámetro.
- **Labels ():** Devuelve las etiquetas del nodo indicado como parámetro.