MongoDB Enterprise 4.0.10

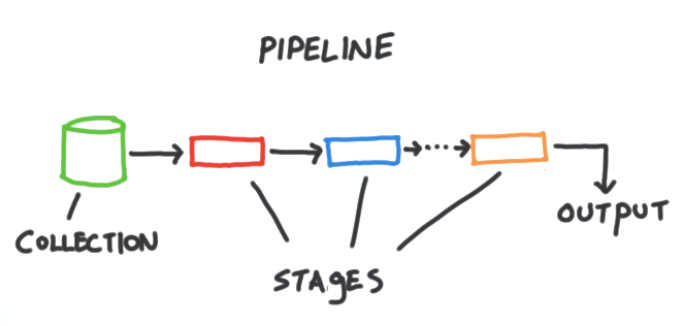
Instructor: Carlos Carreño
Email: ccarrenovi@Gmail.com

Agregación de datos en MongoDB

MongoDB Enterprise 4.0.10

Aggregation Framework

- Es habitual tener que realizar consultas para agrupar datos y calcular valores a partir de ellos.
- En las bases de datos relacionales para agrupar usamos operadores como GROUP BY y para cálculos usamos AVG, SUM, COUNT, En MongoDB tenemos dos opciones: MapReduce y Aggregation Framework



Partes de una consulta de agregación

- Una consulta de agregación con Aggregation Framework tiene el siguiente formato

db.<collection>.aggregate([<pipeline>])

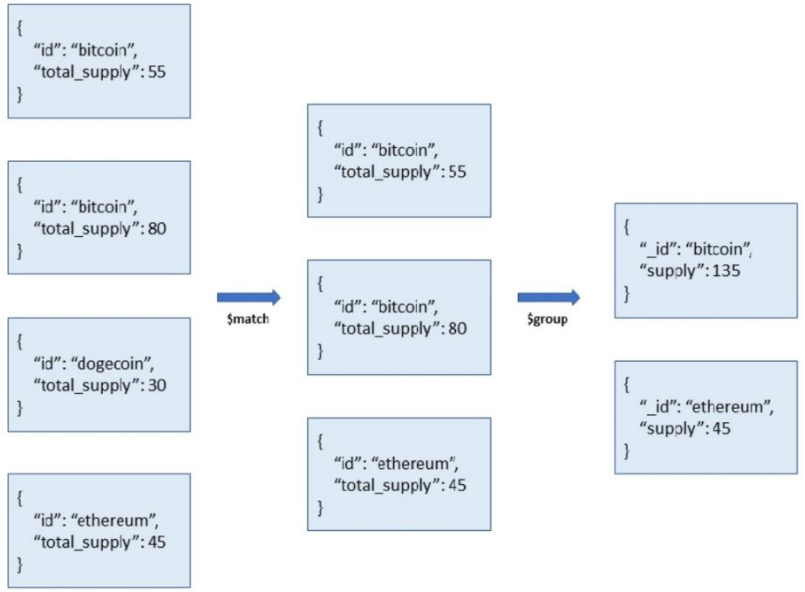
- Los ***pipelines*** o ***tuberías***, son similares a las que se utilizan en la línea de comandos de los sistemas Unix, pasando los resultados de un comando a otro para producir resultados de forma conjunta.

Pipelines de agregación

- ***$project*** : se utiliza para modificar el conjunto de datos de entrada, añadiendo, eliminando o recalculando campos para que la salida sea diferente.
- ***$match***: filtra la entrada para reducir el número de documentos, dejando solo los que cumplan las condiciones establecidas.
- ***$limit***: restringe el número de resultados al número indicado.
- ***$skip***: ignora un número determinado de registros, devolviendo los siguientes.
- ***$unwind***: convierte un array para devolverlo separado en documentos.
- ***$group***: agrupa documentos según una determinada condición.
- ***$sort***: ordena un conjunto de documentos según el campo especificado.
- ***$geoNear***: utilizado con datos geoespaciales, devuelve los documentos ordenados por proximidad según un punto geoespacial.

Pipelines

- Los pipelines pueden trabajar en conjunto




```
db.tickers.aggregate([
        {$match: {total_supply: {$gt: 30} } },
        {$group: {_id: "$id", supply: {$sum: "$total_supply"}}}
])
```

Operadores básicos de agregación

- Para realizar cálculos sobre los datos producidos por los *pipelines*, utilizamos las ***expresiones***. Las *expresiones* son funciones que realizan una determinada operación sobre un grupo de documentos, un array o un campo en concreto.
- ***$first***: Devuelve el primer valor de un campo en un grupo. Si el grupo no está ordenado, el valor mostrado será impredecible.
- ***$last***. Devuelve el último valor de un campo en un grupo. Si el grupo no está ordenado, el valor mostrado será impredecible.
- ***$max***. Devuelve el valor más alto de un determinado campo dentro un grupo.
- ***$min***. Devuelve el valor más pequeño de un determinado campo dentro de un grupo.
- ***$avg***. Calcula la media aritmética de los valores dentro del campo especificado y los devuelve.
- ***$sum***. Suma todos los valores de un campo y los devuelve.

Calculo en una agrupación

- Examinemos la siguiente muestra de datos:

```
{idcliente:"102",mount:456.0, dateorder:"2019-01-25", status:"delivered"},
{idcliente:"103",mount:234.0, dateorder:"2019-01-27", status:"complete"},
{idcliente:"101",mount:13456.0, dateorder:"2019-01-19", status:"pending"},
{idcliente:"103",mount:10456.0, dateorder:"2019-02-18", status:"delivered"},
{idcliente:"104",mount:21366.0, dateorder:"2019-03-21", status:"delivered"},
{idcliente:"105",mount:3457.0, dateorder:"2019-03-25", status:"complete"},
{idcliente:"101",mount:2016.0, dateorder:"2019-04-22", status:"pending"}
```

- Si fuera relacional y necesitáramos calcular la suma de los montos de las ordenes por cliente la consulta seria:

```
SELECT SUM(MOUNT) total  FROM ORDERS GROUP BY IDCLIENTE
```

- En MongoDB con agregaciones es:

```
db.orders.aggregate(
 [{
   $group:{_id:"$idcliente", total:{ $sum:"$mount" } }
}]);
```

Usando operadores en agregaciones

```
db.people.aggregate({
   $sort: {   name: 1    }
  }, {   $group: {
       _id: {
          age: "$age",
          gender: "$gender"
       },
       count: {
          $sum: 1
       },
       avgweight: {
          $avg: "$weight"
       }
   } });
```

Pipeline $project

- Este *pipeline* tiene una funcionalidad parecida a la que tiene el clásico *SELECT* en una consulta SQL. Con él, podremos cambiar los campos originales que tiene un documento añadiendo otros nuevos, eliminando, cambiando el nombre o añadiendo datos calculados.
- Ejemplo:

```
db.people.aggregate({
    $project: {
        isActive: 1,
        company: 1
    }
});
```

Pipeline $match

- El *pipeline $match* se utiliza para filtrar los documentos que se pasarán al siguiente pipeline de la consulta con **Aggregation Framework**

```
db.people.aggregate({
    $match: {
        isActive: true
    }
});
```

Pipeline $group

- Permite agrupar los documentos. Es importante destacar que *$group* siempre tiene que tener un campo *_id*. Es el campo por el que vamos a agrupar los resultados.

```
db.people.aggregate({
   $group: {
      _id: {
         gender: "$gender"
      },
      averageAge: {
         $avg: "$age"
      },
      count: {
         $sum: 1
      }
   }
});
```

Pipeline $sort

- En cualquier consulta de agregación que se precie, es probable que nos veamos en la necesidad de ordenar los resultados. Y es aquí dónde entra el pipeline *$sort*.

```
db.people.aggregate({
    $sort: {
        age: 1
    }
});
```

Prioridad en $sort

- Cuando ordenamos por campos de distintos tipos $sort sigue la siguiente prioridad:

1. Null
2. Valores numéricos (int, log, double)
3. Cadenas de caracteres
4. Objetos
5. Arrays
6. Datos binarios
7. ObjectID (como el campo _id que MongoDB genera para cada documento)
8. Boolean
9. Fechas
10. Expresiones regulares

Pipelines $limit y $skip

- Estos dos pipelines son muy sencillos de utilizar. Con *$limit*, reduciremos el número de documentos devueltos por la consulta hasta el número que indiquemos. Con *$skip* lo que haremos será ignorar un número de elementos determinado que no serán devueltos en la consulta.

```
db.people.aggregate({
    $limit: 3
});

db.people.aggregate({
    $skip: 2
});
```

$unwind

- Utilizando *$unwind* conseguiremos separar los elementos de un array, creando como resultado tantos documentos iguales como elementos tenga el array, pero incluyendo sólo el valor del array.

```
db.people.aggregate({
   $match: {
      name: "Yenny"
    } },
    {   $project: {
           name: 1,
           address: 1
     } },{
         $unwind: "$address"
    });
```

Operadores de agregación y comparación

- Este tipo de operadores de expresión se utilizan para comparar valores y devolver un resultado. Los operadores disponibles son los siguientes:
- $cmp: compara dos valores y devuelve un número entero como resultado. Devuelve -1 si el primer valor es menor que el segundo, 0 si son iguales y 1 si el primer valor es mayor que el segundo.
- $eq: compara dos valores y devuelve true si son equivalentes.
- $gt: compara dos valores y devuelve true si el primero es más grande que el segundo.
- $gte: compara dos valores y devuelve true si el primero es igual o más grande que el segundo.
- $lt: compara dos valores y devuelve true si el primero es menor que el segundo.
- $lte: compara dos valores y devuelve true si el primero es igual o menor que el segundo.
- $ne: compara dos valores y devuelve true si los valores no son equivalentes.

Operadores de agregación booleanos

Un operador booleano recibe parámetros booleanos y devuelve otro booleano como resultado. Los operadores booleanos que podemos utilizar son:

- *$and*: devuelve true si todos los parámetros de entrada son true.
- *$or*: devuelve true si alguno de los parámetros de entrada es true.
- *$not*: devuelve el valor contrario al parámetro de entrada. Si el parámetro es true, devuelve false. Si el parámetro es false, devuelve true.

Operadores de agregación condicionales

Los operadores condicionales en **MongoDB**se utilizan para verificar una expresión y según el valor de esta expresión, devolver un resultado u otro. En **Aggregation Framework**existen dos operadores condicionales:

- *$cond*: operador ternario que recibe tres expresiones. Si la primera es verdadera, devuelve la segunda. Si es falsa, devuelve la tercera. Muy similar, por ejemplo, al operador ternario de JavaScript
- *$ifNull*: operador que recibe dos parámetros y en el caso de que el primero sea null, devuelve el segundo. Muy similar al *IsNull* de SQL Server o al *NVL* de Oracle.

Operadores de agregación de cadenas

Los operadores de cadena son aquellos que se utilizan para realizar operaciones con strings. Son los siguientes:

- *$concat*: concatena dos o más strings.
- *$strcasecmp:* compara dos strings y devuelve un entero con el resultado. Devolverá un número positivo si el primer string es mayor, un número negativo si el segundo es mayor o un 0 en el caso de que sean iguales.
- *$substr:* crea un substring con una cantidad determinada de caracteres.
- *$toLower:* convierte el string a minúsculas.
- *$toUpper:* convierte el string a mayúsculas.

Operadores de agregación y fechas

Los operadores de fecha se utilizan para realizar operaciones con campos de tipo fecha. Tenemos disponibles los siguientes operadores:

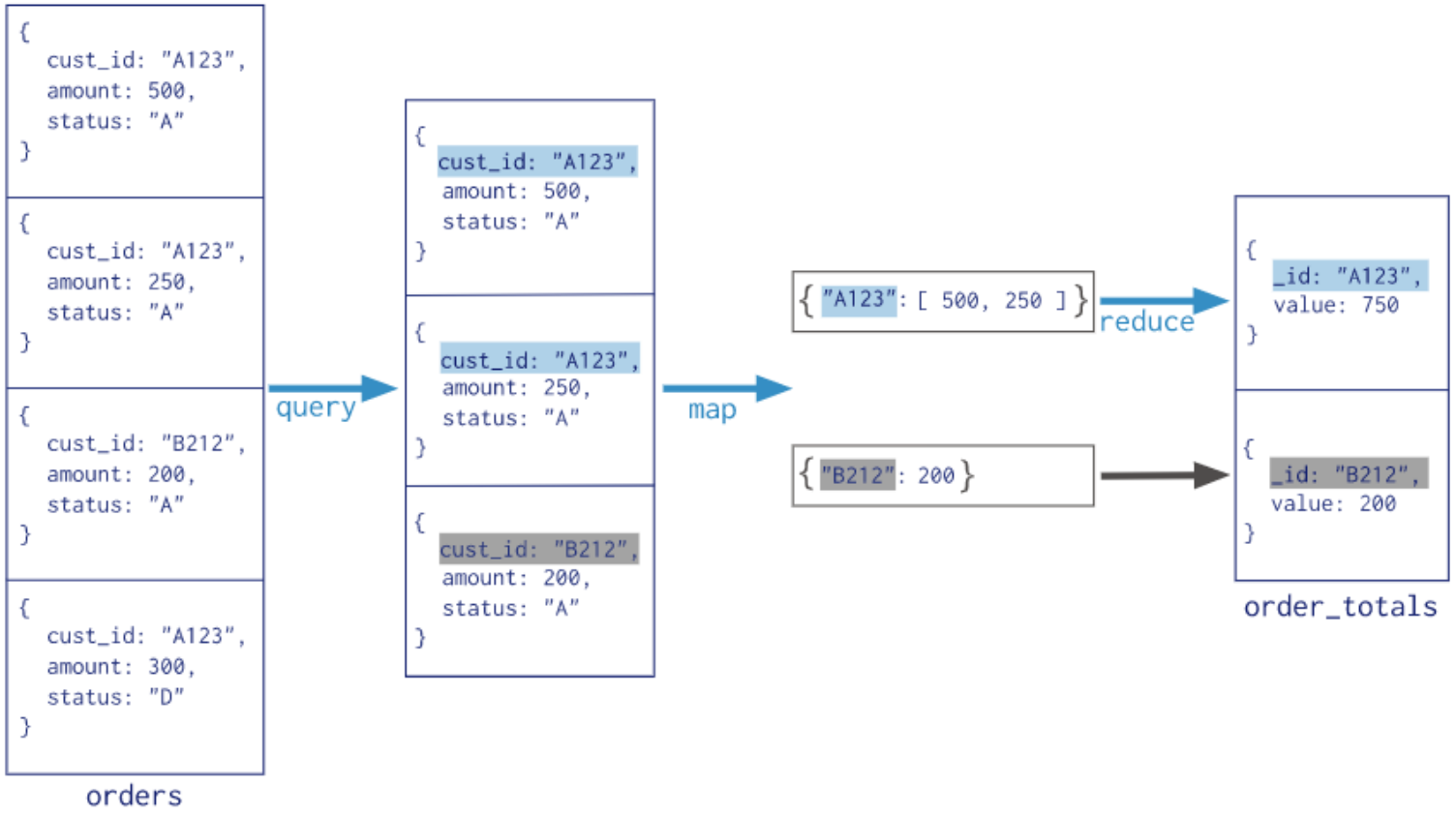
- *$dayOfYear*: convierte una fecha a un número entre 1 y 366.
- *$dayOfMonth*: convierte una fecha a un número entre 1 y 31.
- *$dayOfWeek*: convierte una fecha a un número entre 1 y 7.
- *$year:* convierte una fecha a un número que representa el año (2000,1998 etc.)
- *$month*: convierte una fecha a un número entre el 1 y el 12.
- *$week*: convierte una fecha a un número entre 0 y 53.
- *$hour:* convierte una fecha a un número entre 0 y 23.
- *$minute*: convierte una fecha a un número entre 0 y 59
- *$second*: convierte una fecha a un número entre 0 y 59. Aunque puede ser 60 para contar intervalos.
- *$millisecond:* devuelve los milisegundos de la fecha, con un número entre 0 and 999.

MapReduce

- ***MapReduce*** es un framework creado por Google, y pensado para realizar operaciones de forma paralela sobre grandes colecciones de datos.
- ***MapReduce*** framework está compuesto de dos funciones principales: la función **Map** y la función **Reduce**.
- La función **Map** se encarga, de forma paralela, de mapear los datos de origen. Para cada dato de origen, se genera una tupla clave-valor, las cuales son unidas en una lista que se pasa a la función **Reduce**.
- Después, la función **Reduce**, trata cada elemento de la lista de pares y realiza operaciones sobre ella para devolver un dato concreto.

Funcionamiento de MapReduce

- Mapear y reducir



Utilizando MapReduce en mongoDB

- Para ejecutar una operación **MapReduce** en **MongoDB**, necesitaremos ejecutar un comando como este:

db.nombreColección.mapReduce(*mapFunction*,*reduceFunction*, *options*);

- Como parámetros al método MapReduce de la colección, debemos pasar una función Javascript que se encargue del Map, otra función que se encargue del Reduce y un JSON con opciones adicionales. Este JSON deberá incluir al menos el campo output, con un string que represente el nombre de la colección que se creará para almacenar los datos de la consulta.

Función Map

- Ejemplo: Función Map aplicada a la colección people

```
var map = function () {
    emit(this.gender, { age: this.age, count: 1 });
};
```

Función Reduce

- Ejemplo: Función reduce aplicada a la colección people

```
var reduce = function (keys,values) {
    var reduced = {
        totalPeople:0,
        totalAge:0
    }
    for (var i=0; i < values.length;i++)  {
      reduced.totalPeople+=1;
      reduced.totalAge+=values[i].age;
    }
    return reduced;
};
```

Usando MapReduce

- Ahora que tenemos las funciones podemos ejecutar MapReduce

- db.people.mapReduce(map, reduce, {out: 'map_reduce_result' });
- db.map_reduce_result.find();

```
MongoDB Enterprise > db.map_reduce_result.find();
{ "_id" : "female", "value" : { "totalPeople" : 2, "totalAge" : 54 } }
{ "_id" : "male", "value" : { "totalPeople" : 6, "totalAge" : 200 } }
MongoDB Enterprise >
```

Practica

- Practica 6 Utilizando Aggregation Framework

Referencias

- https://platzi.com/contributions/introduccion-al-pipeline-de-agregacion-de-mongodb/
- https://www.slideshare.net/mongodb/webinar-exploring-the-aggregation-framework