Agregación & MapReduce en MongoDB
Taller de repaso
Parte II

Prof. Dr. Marlon Cárdenas Bonett

unir LA UNIVERSIDAD EN INTERNET

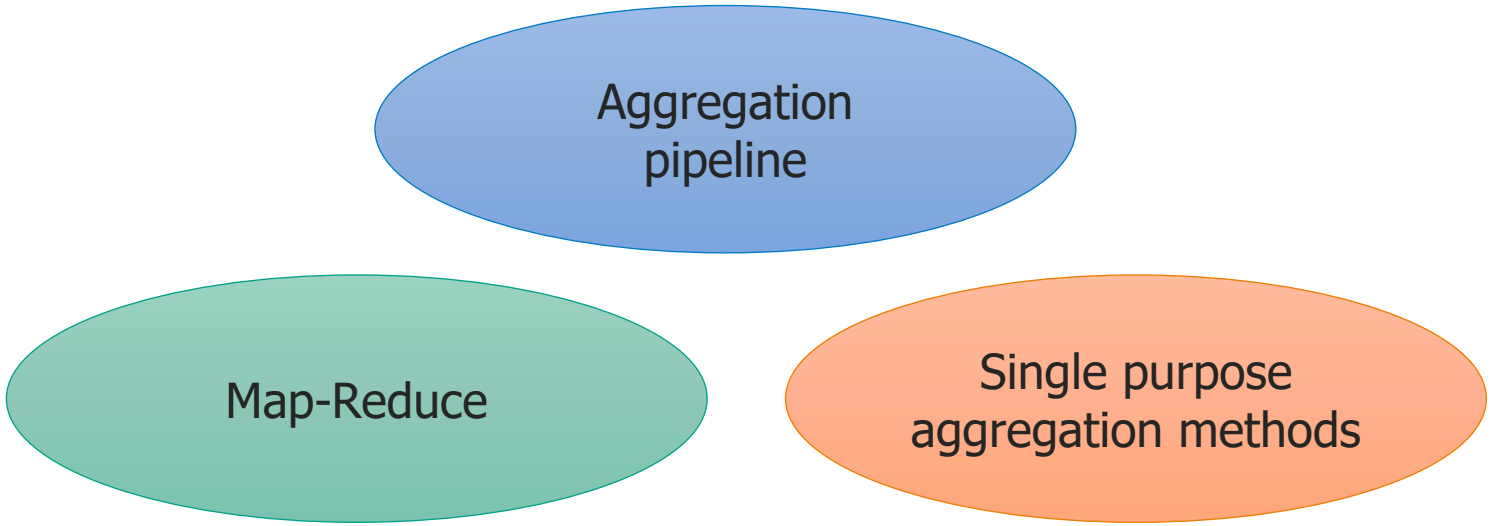
Universidad Internacional de La Rioja

Índice

- Aggregation Framework: primera forma
- Métodos de agregación simple: segunda forma
- Map-Reduce: tercera forma

Agregación de Documentos

- Las operaciones de agregación procesan registros de datos y devuelven resultados calculados.
- Las operaciones de agregación agrupan valores de varios documentos y pueden realizar una variedad de operaciones en los datos agrupados para obtener un único resultado.
- MongoDB proporciona tres formas de realizar la agregación:

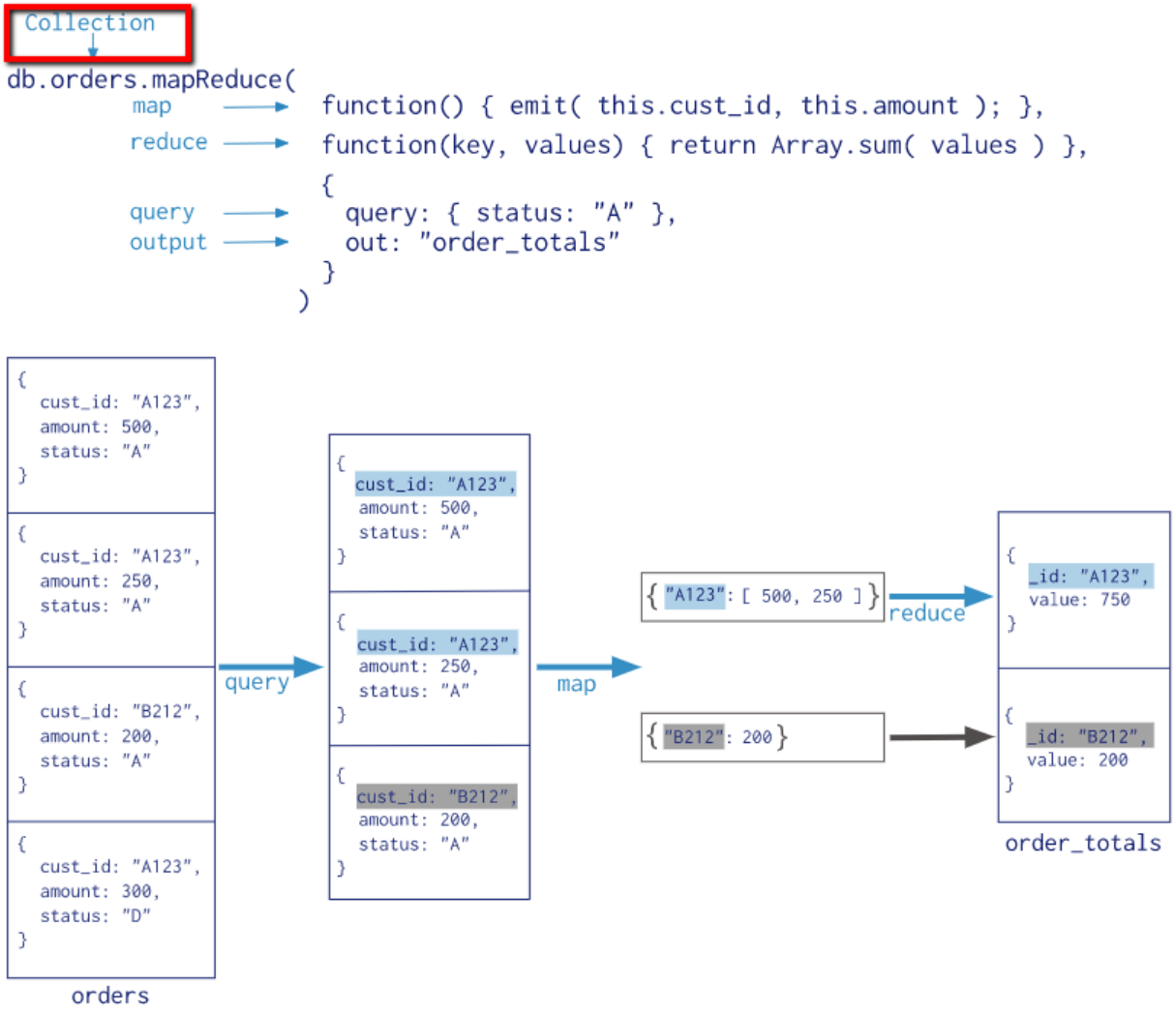


Map-Reduce: tercera forma

Map-Reduce: tercera forma

- MongoDB también proporciona operaciones de map-reduce para realizar la agregación.
- Las operaciones de map-reduce tienen dos fases:
  - Una etapa de map que procesa cada documento y emite uno o más objetos para cada documento de entrada, y
  - Una fase de reduce que combina la salida de la operación del map.
- Opcionalmente, map-reduce puede tener una etapa de finalización para realizar modificaciones finales al resultado.
- Map-reduce puede especificar una condición de consulta para seleccionar los documentos de entrada, así como ordenar y limitar los resultados.
- Map-reduce utiliza funciones de JavaScript personalizadas para realizar el map y el reduce de las operaciones, así como la operación de finalización opcional.
- El uso de JavaScript proporciona una gran flexibilidad en comparación con los pipelines, pero map-reduce es menos eficiente y más complejo.
- Map-reduce puede operar en una colección fragmentada y su resultado puede ser una colección fragmentada.

Map-Reduce: tercera forma



Map-Reduce: tercera forma

- A partir de MongoDB 2.4, ciertas funciones y propiedades de shell mongo son inaccesibles en las operaciones de reducción de mapas.
- MongoDB 2.4 también proporciona soporte para múltiples operaciones de JavaScript para ejecutarse al mismo tiempo.
- Antes de MongoDB 2.4, el código JavaScript se ejecutaba en un solo hilo, lo que generaba problemas de concurrencia para reducir el mapa.

- A partir de la versión 4.2, se depreca la opción de map-reduce para crear una nueva colección fragmentada, así como el uso de la opción de fragmentación para map-reduce. Para enviar a una colección fragmentada, cree primero la colección fragmentada.
- MongoDB 4.2 también depreca el reemplazo de una colección fragmentada existente y la especificación explícita de *nonAtomic: false*.

Map-Reduce: tercera forma

Inserta estos datos en tu base de datos

```
db.norders.insertMany([
   { _id: 1, cust_id: "Ant O. Knee", ord_date: new Date("2020-03-01"), price: 25,
items: [ { sku: "oranges", qty: 5, price: 2.5 }, { sku: "apples", qty: 5, price: 2.5
} ], status: "A" },
   { _id: 2, cust_id: "Ant O. Knee", ord_date: new Date("2020-03-08"), price: 70,
items: [ { sku: "oranges", qty: 8, price: 2.5 }, { sku: "chocolates", qty: 5, price:
10 } ], status: "A" },
   { _id: 3, cust_id: "Busby Bee", ord_date: new Date("2020-03-08"), price: 50,
items: [ { sku: "oranges", qty: 10, price: 2.5 }, { sku: "pears", qty: 10, price: 2.5
} ], status: "A" },
   { _id: 4, cust_id: "Busby Bee", ord_date: new Date("2020-03-18"), price: 25,
items: [ { sku: "oranges", qty: 10, price: 2.5 } ], status: "A" },
   { _id: 5, cust_id: "Busby Bee", ord_date: new Date("2020-03-19"), price: 50,
items: [ { sku: "chocolates", qty: 5, price: 10 } ], status: "A"},
   { _id: 6, cust_id: "Cam Elot", ord_date: new Date("2020-03-19"), price: 35, items:
[ { sku: "carrots", qty: 10, price: 1.0 }, { sku: "apples", qty: 10, price: 2.5 } ],
status: "A" },
   { _id: 7, cust_id: "Cam Elot", ord_date: new Date("2020-03-20"), price: 25, items:
[ { sku: "oranges", qty: 10, price: 2.5 } ], status: "A" },
   { _id: 8, cust_id: "Don Quis", ord_date: new Date("2020-03-20"), price: 75, items:
[ { sku: "chocolates", qty: 5, price: 10 }, { sku: "apples", qty: 10, price: 2.5 } ],
status: "A" },
   { _id: 9, cust_id: "Don Quis", ord_date: new Date("2020-03-20"), price: 55, items:
[ { sku: "carrots", qty: 5, price: 1.0 }, { sku: "apples", qty: 10, price: 2.5 }, {
sku: "oranges", qty: 10, price: 2.5 } ], status: "A" },
   { _id: 10, cust_id: "Don Quis", ord_date: new Date("2020-03-23"), price: 25,
items: [ { sku: "oranges", qty: 10, price: 2.5 } ], status: "A" }
])
```

Map-Reduce: tercera forma

Recuerda:

- La función de map emite pares clave-valor. Para aquellas claves que tienen múltiples valores, MongoDB aplica la fase de reduce, que recopila y condensa los datos agregados.
- MongoDB luego almacena los resultados en una colección.
- Opcionalmente, la salida de la función de reduce puede pasar a través de una función de finalización para condensar o procesar aún más los resultados de la agregación.
- Todas las funciones de map-reduce en MongoDB son JavaScript y se ejecutan dentro del proceso mongod.

Map-Reduce: ejercicio 1

```
var mapFunction1 = function() {
   emit(this.cust_id, this.price);
};
```

```
var reduceFunction1 = function(keyCustId, valuesPrices) {
   return Array.sum(valuesPrices);
};
```

```
db.norders.mapReduce(
   mapFunction1,
   reduceFunction1,
   { out:"map_reduce_example"}
)
```

```
db.map_reduce_example.find().sort({_id:1})
{ "_id" : "Ant O. Knee", "value" : 95 }
{ "_id" : "Busby Bee", "value" : 125 }
{ "_id" : "Cam Elot", "value" : 60 }
{ "_id" : "Don Quis", "value" : 155 }
```

```
db.norders.aggregate([
   { $group: { _id: "$cust_id", value: { $sum: "$price" } } },
   { $out: "agg_alternative_1" }
])
```

Map-Reduce: ejercicio 2

```
var mapFunction2 = function() {
    for (var idx = 0; idx < this.items.length; idx++) {
       var key = this.items[idx].sku;
       var value = { count: 1, qty: this.items[idx].qty };

       emit(key, value);
    }
};
```

```
var reduceFunction2 = function(keySKU, countObjVals) {
   reducedVal = { count: 0, qty: 0 };
   for (var idx = 0; idx < countObjVals.length; idx++) {
       reducedVal.count += countObjVals[idx].count;
       reducedVal.qty += countObjVals[idx].qty;
   }
   return reducedVal;
};
```

Esta es una nueva función

```
var finalizeFunction2 = function (key, reducedVal) {
  reducedVal.avg = reducedVal.qty/reducedVal.count;
  return reducedVal;
};
```

Map-Reduce: ejercicio 2

```
db.norders.mapReduce(
   mapFunction2,
   reduceFunction2,
   {
     out: { merge: "map_reduce_example2" },
     query: { ord_date: { $gte: new Date("2020-03-01") } },
     finalize: finalizeFunction2
   }
 );

db.map_reduce_example2.find().sort( { _id: 1 } )
```

Mirad la nueva función

Map-Reduce: ejercicio 2

```
db.norders.aggregate( [
   { $match: { ord_date: { $gte: new Date("2020-03-01") } } },
   { $unwind: "$items" },
   { $group: { _id: "$items.sku", qty: { $sum: "$items.qty" }, orders_ids: {
$addToSet: "$_id" } }  },
   { $project: { value: { count: { $size: "$orders_ids" }, qty: "$qty", avg: {
$divide: [ "$qty", { $size: "$orders_ids" } ] } } } },
   { $merge: { into: "agg_alternative_3", on: "_id", whenMatched: "replace",
whenNotMatched: "insert" } }
] )
```

```
db.agg_alternative_3.find().sort( { _id: 1 } )
```

Comparad los dos resultados:
map_reduce_example2 y agg_alternative_3

Map-Reduce: ejercicio 2 (explicación)

```
$match: ord_date >= new Date("2020-03-01")
$unwinds:
{ "_id" : 1, "cust_id" : "Ant O. Knee", "ord_date" : ISODate("2020-03-
01T00:00:00Z"), "price" : 25, "items" : { "sku" : "oranges", "qty" : 5,
"price" : 2.5 }, "status" : "A" }
{ "_id" : 1, "cust_id" : "Ant O. Knee", "ord_date" : ISODate("2020-03-
01T00:00:00Z"), "price" : 25, "items" : { "sku" : "apples", "qty" : 5, "price"
: 2.5 }, "status" : "A" }
$group:
{ "_id" : "chocolates", "qty" : 15, "orders_ids" : [ 2, 5, 8 ] }
{ "_id" : "oranges", "qty" : 63, "orders_ids" : [ 4, 7, 3, 2, 9, 1, 10 ] }
{ "_id" : "carrots", "qty" : 15, "orders_ids" : [ 6, 9 ] }
{ "_id" : "apples", "qty" : 35, "orders_ids" : [ 9, 8, 1, 6 ] }
{ "_id" : "pears", "qty" : 10, "orders_ids" : [ 3 ] }
$project:
{ "_id" : "apples", "value" : { "count" : 4, "qty" : 35, "avg" : 8.75 } }
{ "_id" : "pears", "value" : { "count" : 1, "qty" : 10, "avg" : 10 } }
{ "_id" : "chocolates", "value" : { "count" : 3, "qty" : 15, "avg" : 5 } }
{ "_id" : "oranges", "value" : { "count" : 7, "qty" : 63, "avg" : 9 } }
{ "_id" : "carrots", "value" : { "count" : 2, "qty" : 15, "avg" : 7.5 } }
```

Map-Reduce: ejercicios 3 y 4

3

```
db.usersessions.insertMany([
   { userid: "a", start: ISODate('2020-03-03 14:17:00'), length: 95 },
   { userid: "b", start: ISODate('2020-03-03 14:23:00'), length: 110 },
   { userid: "c", start: ISODate('2020-03-03 15:02:00'), length: 120 },
   { userid: "d", start: ISODate('2020-03-03 16:45:00'), length: 45 },
   { userid: "a", start: ISODate('2020-03-04 11:05:00'), length: 105 },
   { userid: "b", start: ISODate('2020-03-04 13:14:00'), length: 120 },
   { userid: "c", start: ISODate('2020-03-04 17:00:00'), length: 130 },
   { userid: "d", start: ISODate('2020-03-04 15:37:00'), length: 65 }
])
```

4

```
db.usersessions.insertMany([
   { userid: "a", ts: ISODate('2020-03-05 14:17:00'), length: 130 },
   { userid: "b", ts: ISODate('2020-03-05 14:23:00'), length: 40 },
   { userid: "c", ts: ISODate('2020-03-05 15:02:00'), length: 110 },
   { userid: "d", ts: ISODate('2020-03-05 16:45:00'), length: 100 }
])
```

Map-Reduce: ejercicio 3

```
var mapFunction = function() {
    var key = this.userid;
    var value = { total_time: this.length, count: 1, avg_time: 0 };
    emit( key, value );
};
```

```
var reduceFunction = function(key, values) {
   var reducedObject = { total_time: 0, count:0, avg_time:0 };
   values.forEach(function(value) {
      reducedObject.total_time += value.total_time;
      reducedObject.count += value.count;
   });
   return reducedObject;
};
```

```
var finalizeFunction = function(key, reducedValue) {
   if (reducedValue.count > 0)
      reducedValue.avg_time = reducedValue.total_time /
reducedValue.count;
   return reducedValue;
};
```

Map-Reduce: ejercicio 3

3

```
db.usersessions.mapReduce(
   mapFunction,
   reduceFunction,
   {
     out: "session_stats",
     finalize: finalizeFunction
   }
)
```

```
db.session_stats.find().sort( { _id: 1 } )
```

```
{ "_id" : "a", "value" : { "total_time" : 200, "count" : 2, "avg_time" : 100 } }
{ "_id" : "b", "value" : { "total_time" : 230, "count" : 2, "avg_time" : 115 } }
{ "_id" : "c", "value" : { "total_time" : 250, "count" : 2, "avg_time" : 125 } }
{ "_id" : "d", "value" : { "total_time" : 110, "count" : 2, "avg_time" : 55  } }
```

Map-Reduce: ejercicio 4

4

```
db.usersessions.mapReduce(
   mapFunction,
   reduceFunction,
   {
     query: { ts: { $gte: ISODate('2020-03-05 00:00:00') } },
     out: { reduce: "session_stats" },
     finalize: finalizeFunction
   }
)
db.session_stats.find().sort( { _id: 1 } )
```

```
{ "_id" : "a", "value" : { "total_time" : 330, "count" : 3, "avg_time" : 110 } }
{ "_id" : "b", "value" : { "total_time" : 270, "count" : 3, "avg_time" : 90  } }
{ "_id" : "c", "value" : { "total_time" : 360, "count" : 3, "avg_time" : 120 } }
{ "_id" : "d", "value" : { "total_time" : 210, "count" : 3, "avg_time" : 70  } }
```

Map-Reduce: ejercicio 5

¡Cuidado!

5

```
db.usersessions.drop();

db.usersessions.insertMany([
   { userid: "a", start: ISODate('2020-03-03 14:17:00'), length: 95 },
   { userid: "b", start: ISODate('2020-03-03 14:23:00'), length: 110 },
   { userid: "c", start: ISODate('2020-03-03 15:02:00'), length: 120 },
   { userid: "d", start: ISODate('2020-03-03 16:45:00'), length: 45 },
   { userid: "a", start: ISODate('2020-03-04 11:05:00'), length: 105 },
   { userid: "b", start: ISODate('2020-03-04 13:14:00'), length: 120 },
   { userid: "c", start: ISODate('2020-03-04 17:00:00'), length: 130 },
   { userid: "d", start: ISODate('2020-03-04 15:37:00'), length: 65 }
])
```

Map-Reduce: ejercicio 5

```
db.usersessions.aggregate([
   { $group: {_id: "$userid", total_time: {$sum: "$length" }, count: {$sum: 1 }, avg_time:
{$avg: "$length" }}},
   { $project: { value: { total_time: "$total_time", count: "$count", avg_time:
"$avg_time" } } },
   { $merge: {
      into: "session_stats_agg",
      whenMatched: [ { $set: {
         "value.total_time": { $add: [ "$value.total_time", "$$new.value.total_time" ] },
         "value.count": { $add: [ "$value.count", "$$new.value.count" ] },
         "value.avg": {$divide: [ {$add: ["$value.total_time", "$$new.value.total_time" ]
},
               {$add: [ "$value.count", "$$new.value.count" ] } ] }
      } } ],
      whenNotMatched: "insert"
   }}
])
```

```
db.session_stats_agg.find().sort( { _id: 1 } )
```

www.unir.net