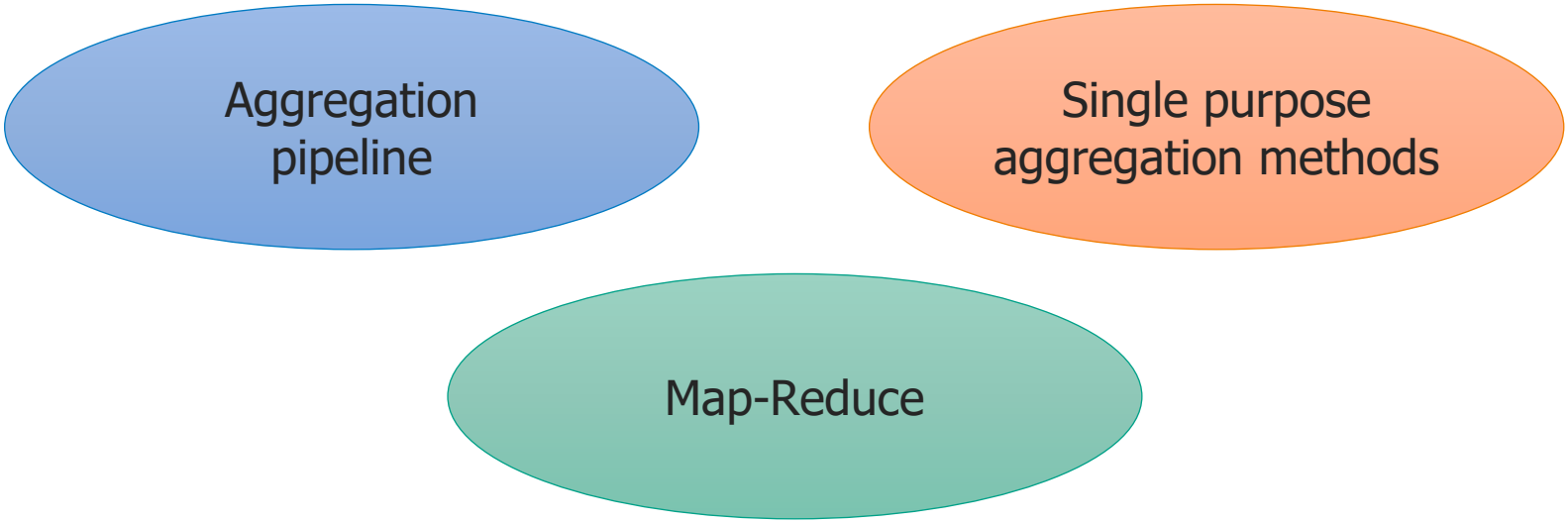
Agregación & MapReduce en MongoDB
Taller de repaso
Parte I

Prof. Dr. Marlon Cárdenas Bonett

Índice

Aggregation 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:



Aggregation Framework

Aggregation Pipeline: primera forma

- El framework de agregación de MongoDB se basa en el concepto de canalización (pipelines) de procesamiento de datos.
- Los documentos entran en una tubería de etapas múltiples que transforma los documentos en un resultado agregado.

```
db.orders.aggregate([
        { $match: { status: "A" } },
        { $group: { _id: "$cust_id", total: { $sum: "$amount" } } }
])
```

- **Primera etapa**: la etapa $match filtra los documentos por el campo indicado y pasa a la siguiente etapa aquellos documentos que cumplan dicha condición (en este caso status = "A“).
- **Segunda etapa**: la etapa $group agrupa los documentos por el campo indicado (cust_id) para calcular la suma de la cantidad para cada cust_id único.
- Las etapas de canalización más básicas proporcionan filtros que funcionan como consultas y transformaciones de documentos que modifican la forma del documento de salida.

Aggregation Pipeline: primera forma

- Otras operaciones de pipeline proporcionan herramientas para agrupar y clasificar documentos por campo o campos específicos, así como herramientas para agregar el contenido de los arrays, incluidas los arrays de documentos.
- Los pipelines pueden usar **operadores** para tareas como calcular el promedio o concatenar una cadena.
- Los pipelines proporcionan una agregación de datos eficiente utilizando operaciones nativas dentro de MongoDB, y es el método preferido para la agregación de datos en esta base de datos.

- El pipeline puede operar en una colección fragmentada.

- El pipeline puede usar índices para mejorar su rendimiento durante algunas de sus etapas.
- Además, estos tienen una fase de optimización interna.

Aggregation Pipeline: operadores

Inserta estos datos en tu base de datos para que pruebes los ejemplos siguientes.

```
{ _id: 1, name: "juan", budget: 5, spent: 8, date: ISODate("2014-03-01T08:00:00Z") }
{ _id: 2, name: “ana“, budget : 4, spent : 4, date: ISODate("2014-03-05T09:00:00Z") }
{ _id: 3, name: "jorge“, budget : 9, spent : 7, date: ISODate("2014-04-01T09:00:00Z") }
{ _id: 4, name: “manuel“, budget : 6, spent : 7, date: ISODate("2014-05-01T09:30:00Z") }
```

- $abs: valor absoluto

```
{$abs: <number>}
> db.budgets.aggregate([
    {
      $project: { total: { $abs: { $subtract: [ "$budget", "$spent" ] } } }
    }
])
```

- $add: sumar o fechas

```
> db.budgets.aggregate([
    { $project: { name: 1, total: { $add: [ "$budget", "$spent" ] } } }
])
```

```
> db.budgets.aggregate(
  [{ $project:{name: 1, billing_date: {$add: ["$date",3*24*60*60000 ]}}}
])
```

Aggregation Pipeline: operadores

Inserta estos datos en tu base de datos para que pruebes los ejemplos siguientes.

```
db.sales.insertMany([
{ "_id" : 1, "item" : "abc", "price" : 10, "quantity" : 2, "date" : ISODate("2014-01-01T08:00:00Z") },
{ "_id" : 2, "item" : "jkl", "price" : 20, "quantity" : 1, "date" : ISODate("2014-02-03T09:00:00Z") },
{ "_id" : 3, "item" : "xyz", "price" : 5, "quantity" : 5, "date" : ISODate("2014-02-03T09:05:00Z") },
{ "_id" : 4, "item" : "abc", "price" : 10, "quantity" : 10, "date" : ISODate("2014-02-15T08:00:00Z") },
{ "_id" : 5, "item" : "xyz", "price" : 5, "quantity" : 10, "date" : ISODate("2014-02-15T09:12:00Z") }])
```

- $addToSet: devuelve un array de todos los valores únicos que resultan de aplicar una expresión a cada documento en un grupo de documentos que comparten el mismo grupo por clave. El orden de los elementos en la matriz de salida no está especificado.

```
> db.sales.aggregate(
   [{
       $group:
         {
           _id: { day: { $dayOfYear: "$date"}, year: { $year: "$date" } },
           itemsSold: { $addToSet: "$item" }
         }
   }]
)
```

```
{ "_id" : { "day" : 46, "year" : 2014 }, "itemsSold" : [ "xyz", "abc" ] }
{ "_id" : { "day" : 34, "year" : 2014 }, "itemsSold" : [ "xyz", "jkl" ] }
{ "_id" : { "day" : 1,  "year" : 2014 }, "itemsSold" : [ "abc" ] }
```

Aggregation Pipeline: operadores

Inserta estos datos en tu base de datos para que pruebes los ejemplos siguientes.

```
db.new_students.insertMany([
{ "_id": 1, "quizzes": [ 10, 6, 7 ], "labs": [ 5, 8 ], "final": 80, "midterm": 75 },
{ "_id": 2, "quizzes": [ 9, 10 ], "labs": [ 8, 8 ], "final": 95, "midterm": 80 },
{ "_id": 3, "quizzes": [ 4, 5, 5 ], "labs": [ 6, 5 ], "final": 78, "midterm": 70 }])
```

- $and: evalúa una o más expresiones y devuelve verdadero si todas las expresiones son verdaderas o si se evocan sin expresiones de argumento. De lo contrario devuelve falso.

```
> db.sales.aggregate(
   [{
       $project:
          {
            item: 1,
            qty: 1,
            result:{ $and:[ { $gt:[ "$price", 10 ] }, {$lt:[ "$price", 20]}]}
          }
   }]
)
```

- $avg

```
db.new_students.aggregate([
   { $project:{ quizAvg: { $avg:"$quizzes"}, labAvg:{$avg: "$labs" }, examAvg: {
     $avg: [ "$final", "$midterm" ] } } }
])
```

Aggregation Pipeline: operadores

Inserta estos datos en tu base de datos para que pruebes los ejemplos siguientes.

```
db.new_items.insertMany([
{ _id: 1, value: 9.25 }, { _id: 2, value: 8.73 },
{ _id: 3, value: 4.32 }, { _id: 4, value: -5.34 }])
```

- $ceil: devuelve el entero más pequeño mayor o igual al número especificado.

```
> db.new_items.aggregate(
   [ { $project: { value: 1, ceilingValue: { $ceil: "$value" } } } ]
)
```

```
{ "_id" : 1, "value" : 9.25, "ceilingValue" : 10 }
```

- $cmp: compara dos valores y retorna: -1 si el primero es menor que el segundo, 0 si son iguales y 1 si el primero es mayor que el segundo.

```
> db.sales.aggregate(
   [{ $project:
         { item: 1,
           price: 1,
           cmpTo10: { $cmp: [ "$quantity", 10 ] },
           _id: 0 }
     }])
```

Aggregation Pipeline: operadores

- $concat: devuelve el entero más pequeño mayor o igual al número especificado.

```
> db.sales.aggregate(
   [{ $project: { salesDetail: { $concat: [ "$item", " - ", "$price" ] } }
} ]
)
```

- $concatArrays

```
> db.new_students.aggregate(
   [{ $project: { calificaciones: { $concatArrays: [ "$quizzes", "$labs" ] }
} } ]
)
```

- $divide

```
> db.budgets.aggregate(
   [{ $project: { name: 1, discount: { $divide: [ "$spent", 2 ] } } } ]
)
```

Aggregation Pipeline: operadores

Inserta estos datos en tu base de datos para que pruebes los ejemplos siguientes.

```
db.warehouse.insertMany([
{ "_id" : 1, instock: [ "chocolate" ], ordered: [ "butter", "apples" ] },
{ "_id" : 2, instock: [ "apples", "pudding", "pie" ] },
{ "_id" : 3, instock: [ "pears", "pecans"], ordered: [ "cherries" ] },
{ "_id" : 4, instock: [ "ice cream" ], ordered: [ ] }])
```

- $cond: dermite el uso de condiciones para realizar la agregación.

```
> db.warehouses.aggregate([
   { $project:
      { items:
          { $cond:
            {
              if: { $and: [ { $isArray: "$instock" }, { $isArray: "$ordered"
} ] },
              then: { $concatArrays: [ "$instock", "$ordered" ] },
              else: "One or more fields is not an array."
            }
          }
      }
   }
])
```

- $isArray: con base al ejemplo, ¿qué hace este operador?

Aggregation Pipeline: operadores

- $push: devuelve una matriz de todos los valores que resultan de aplicar una expresión a cada documento en un grupo de documentos que comparten el mismo grupo por clave.

```
> db.sales.aggregate(
   [{
       $group:
         {
           _id: { day: { $dayOfYear: "$date"}, year: { $year: "$date" } },
           itemsSold: { $push:  { item: "$item", quantity: "$quantity" } }
         }
   }]
)
```

```
> db.products.update( { "category_ids": 12 }, { "$push": { "category_ids": 12
} } )
```

```
> db.products.update(
    { "category_ids": 12 }, { "$pull": { "category_ids": { "$gt": 20 } } }
)
```

```
  $push (+end) $pop (-end), $pull (-n), $pullAll (-*)
```

Aggregation Pipeline: operadores

- $pop: -1/1: remueve el primero o último elemento de un array.

```
> db.norders.update({ _id: 1}, {$pop : { items: -1  } })
> db.norders.update({ _id: 1}, {$pop : { items: 1  } })
```

- $pull

```
db.stores.update(
    { },
    { $pull: { fruits: { $in: [ "apples", "oranges" ] }, vegetables:
"carrots" } },
    { multi: true }
)
```

- $push

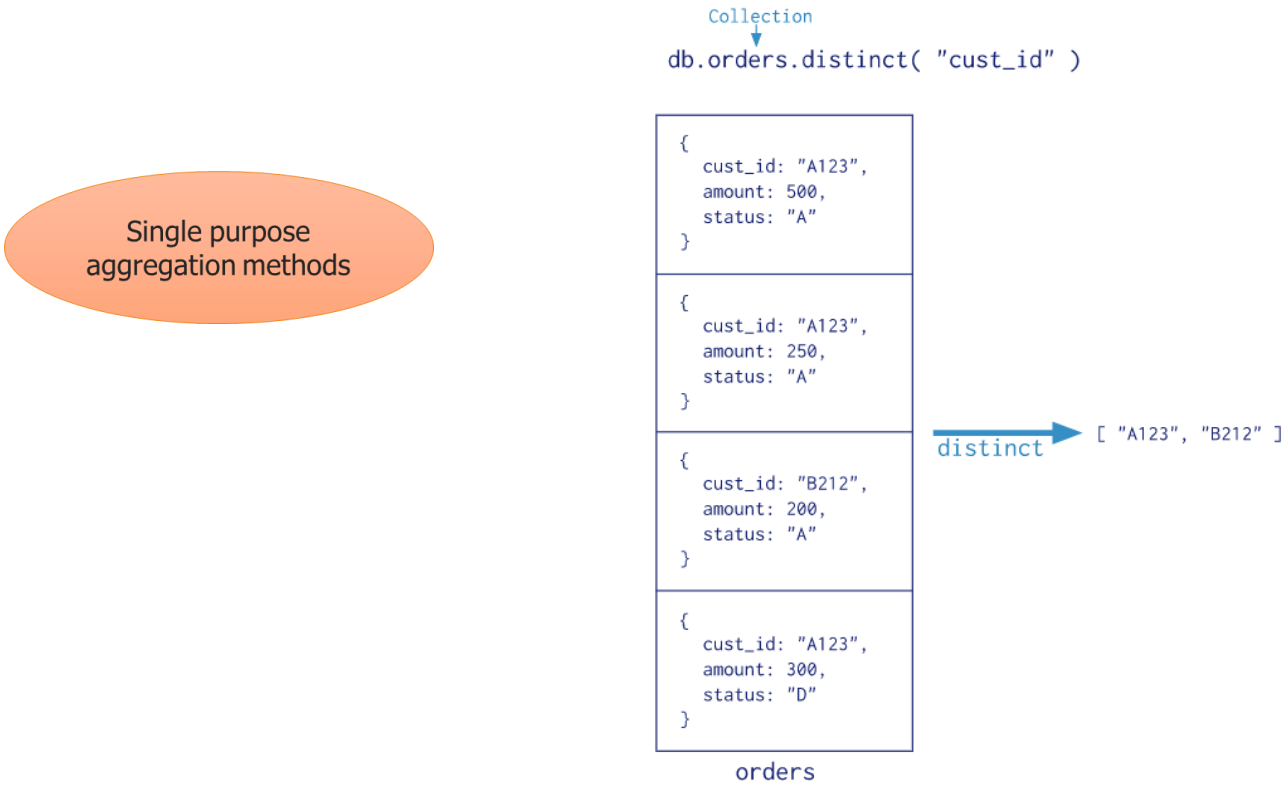
```
> db.students.update(
   { _id: 1 },
   { $push:{scores:89}}
)
```

$each

```
> db.students.update(
   { name: "joe" },
   { $push:{scores:{$each:[90,92,85]} }}
)
```

Single Aggregation Methods

Métodos de agregación simple: segunda forma



- MongoDB proporciona db.collection.estimatedDocumentCount(), db.collection.count() y db.collection.distinct().
- Todas estas operaciones agregan documentos de una sola colección. Si bien estas operaciones proporcionan acceso simple a los procesos de agregación comunes, carecen de la flexibilidad y las capacidades de los pipelines o map-reduce.

www.unir.net