Bases de Datos No Relacionales

- Framework de agregación

  (continuación)

Operador

- **$unwind**

Sintaxis: db.coleccion.aggregate ([{ $unwind: "$atributoArray" }])

Se especifica un atributo de los documentos de entrada que sea un array y se creará un documento de salida por cada elemento del array, se mantendrán iguales todos los atributos, excepto el campo del array que tendrá sólo un elemento del array.

Ejemplo

Documento de entrada

{ Atributo1: “A”,
Atributo2: “B”,
item: [
{prod: 1, cant: 10},
{prod: 2, cant: 50},
{prod: 3, cant: 70}
] }

Documentos de salida

{Atributo1: “A”,Atributo2: “B”, **item**: {num: 1, cant: 10} } ,
{Atributo1: “A”,Atributo2: “B”, **item**: {num: 2, cant: 50} } ,
{Atributo1: “A”,Atributo2: “B”, **item**: {num: 3, cant: 70} }

Operador

- **$lookup** **Sintaxis:**

```
db.coleccionOrigen.aggregate ([
{ $lookup:
    { from: <coleccionDestino>,
    localField: <atributo de la coleccionOrigen>,
    foreignField: <atributo de la coleccionDestino>,
    as : <atributo de la coleccion de salida (array)>
    }
} ])
```

Los documentos de salida serán como los documentos de entrada, pero se agregará el atributo "as", que será un array con todos los documentos de la colección "from" que tengan el mismo "foreignField" que el "localField" de los documentos de entrada.

Operador

- **$lookup**

Ejemplo

ColeccionOrigen

{ _id: 1,
Atributo1: “A”,
_idDestino: 1_1}
{ _id: 2,
Atributo1: “B”,
_idDestino: 1_1}
{ _id: 1,
Atributo1: “C”,
_idDestino: 1_2}

ColeccionDestino

{ _id: 1_1,
Atributo2: “B”} ,
{ _id: 1_2,
Atributo2: “C”}

Documentos de salida

{Atributo1: “A”,Destino: [{_id: 1_1,Atributo2: “B”}]},
{Atributo1: “B”,Destino: [{_id: 1_1,Atributo2: “B”}]},
{Atributo1: “C”,Destino: [{_id: 1_2,Atributo2: “C”}]}

…
from: coleccionDestino,
LocalField: _idDestino,
ForeingField: _id,
as: "Destino"
…

Operador

- **$lookup** Si localField es un array, se puede hacer coincidir cada elemento con un foreignField escalar sin usar $unwind.

Ejemplo

ColeccionOrigen

{ _id: 1,
Atributo1: “A”,
_idDestino: **[1_1,1_2]**}
{ _id: 1,
Atributo1: “C”,
_idDestino: **[1_2]**}

ColeccionDestino

{ _id: 1_1,
Atributo2: “B”} ,
{ _id: 1_2,
Atributo2: “C”}

Documentos de salida

{Atributo1: “A”,Destino: [{_id: 1_1,Atributo2: “B”},
{_id: 1_2,Atributo2: “C”}]
},
{Atributo1: “C”,Destino: [{_id: 1_2,Atributo2: “C”}] }

Operador

$lookup Si se quiere presentar la salida como **documentos** conformados por los atributos de ambas colecciones, utilizamos **$mergeObjects** y modificamos mediante **$replaceRoot** la estructura del documento actual **($$ROOT).**

```
db.coleccion.aggregate( [
  {
    $lookup: {
      from: "ColeccionDestino",
      localField: "_id",
      foreignField: "_idDestino",
      as: "ArrayDestino"
    }
  },
  {
    $replaceRoot: { newRoot: { $mergeObjects: [ { $arrayElemAt: [ "$ArrayDestino", 0 ] }, "$$ROOT" ] } }
  },
  { $project: { ArrayDestino: 0 } }
                    ] )
```

Toma el elemento del array en la pos. indicada

Hace referencia al documento actual

Operador

Ejemplo de $lookup y $mergeObjects

ColeccionOrigen

{ _id: 1,
Atributo1: “A”,
_idDestino: 1_1}
{ _id: 2,
Atributo1: “B”,
_idDestino: 1_1}
{ _id: 1,
Atributo1: “C”,
_idDestino: 1_2}

ColeccionDestino

{ _id: 1_1,
Atributo2: “B”} ,
{ _id: 1_2,
Atributo2: “C”}

Documentos de salida

{Atributo1: “A”, Atributo2: “B”} ,
{Atributo1: “B”, Atributo2: “B”} ,
{Atributo1: “C”, Atributo2: “C”}

Operador

- **$out** Permite crear una colección luego de ejecutarse el pipeline. Siempre va al final del mismo. Si ya existiera la colección la sobreescribe. *Es muy útil para migraciones.*

  Lo podemos asociar con las sentencias SQL :

  `INSERT INTO T2 SELECT * FROM T1` ó `SELECT * INTO T2 FROM T1`

  ### Sintaxis

  { $out: "<coleccion>" } // Coleccion se creará en la misma base

  { $out: { db: "<Base>", coll: "<coleccion>" } } //Especifico coleccion y base distinta

Operador

- **$bucketAuto** Clasifica los documentos entrantes en un número específico de grupos, denominados cubos o cubetas (bucket), en función de una expresión especificada. Los límites de los cubos se determinan automáticamente en un intento de distribuir uniformemente los documentos en el número especificado de cubos .
  Cada bucket se representa como un documento en la salida. El documento de cada cubo contiene:
- Objeto **_id** que especifica los límites del cubo.
  El campo **_id.min** especifica el límite inferior inclusivo para el bucket.
  El campo **_id.max** especifica el límite superior del cubo. Este límite es exclusivo para todos los buckets, excepto para el último de la serie, en el que es inclusivo.
- Un campo de recuento que contiene el número de documentos del bucket. El campo de recuento se incluye de forma predeterminada cuando no se especifica el documento de salida.

Operador

- **$bucketAuto**

Sintaxis

```
{
$bucketAuto: {
    groupBy: <expression>,
    buckets: <number>,
    output: {
    <output1>: { <$accumulator expression> },
    ...
}
```

Operador

- **$bucketAuto**

Ejemplo Trabajar con la colección art.

Esta operación agrupará en 3 cubos los documentos por año

```
db.art.aggregate( [
{
$bucketAuto: {
groupBy: "$year",
buckets: 3
}}
])
```

Resultado

```
{ " id" : { "min" : null, "max" : 1913 }, "count" : 3}
{ " id" : { "min" : 1913, "max" : 1926 }, "count" : 3}
{ "_id" : { "min" : 1926, "max" : 1931 }, "count" : 2}
```

Operador

- **$bucketAuto** Se puede especificar la salida mediante **output** y allí además utilizar otros opeardores.

```
db.art.aggregate( [
 {
        $bucketAuto: {
          groupBy: "$year",
          buckets: 3,
          output: {
            "count": { $sum: 1 },
            "years": { $push: "$year" }
          }
        }
      }
] )
```

Resultado

{"_id":{"min":null,"max":1913},"count":3,"years":[1902]},
{"_id":{"min":1913,"max":1926},"count":3,"years":[1913,1918,1925]},
{"_id":{"min":1926,"max":1931},"count":2,"years":[1926,1931]}

Operador

- **$bucketAuto** Se puede utilizar dentro de una stage **$facet** para procesar varias canalizaciones de agregación en el mismo conjunto de documentos de entrada.

```
db.art.aggregate( [
 {
  $facet: {
   "price": [
    {
     $bucketAuto: {
      groupBy: "$price",
      buckets: 4 }
     }
   ],
   "year": [
    {
     $bucketAuto: {
      groupBy: "$year",
      buckets: 3 }
    }
   ]
  }}] )
```

Cada bucket genera en un documento de salida, **un atributo y un array con los documentos conteniendo los valores min, max, etc.**

Ejemplo de salida $bucketAuto y $facet

{ **"price"** : [
{ "_id" : { "min" : NumberDecimal("76.04"), "max" : NumberDecimal("159.00") }, "count" : 2 },
{ "_id" : { "min" : NumberDecimal("159.00"), "max" : NumberDecimal("199.99") }, "count" : 2 },
{ "_id" : { "min" : NumberDecimal("199.99"), "max" : NumberDecimal("385.00") }, "count" : 2 },
{ "_id" : { "min" : NumberDecimal("385.00"), "max" : NumberDecimal("483.00") }, "count" : 2 }
],
"year" : [
{ "_id" : { "min" : null, "max" : 1913 }, "count" : 3 },
{ "_id" : { "min" : 1913, "max" : 1926 }, "count" : 3},
{ "_id" : { "min" : 1926, "max" : 1931 }, "count" : 2}
]
}

Operador

- **$setWindowFields**

  Realiza operaciones en un intervalo especificado de documentos de una colección, conocido como **ventana**, y devuelve los resultados en función del **operador de ventana** elegido.

  Por ejemplo, se puede utilizar para generar lo siguiente:

  - Diferencia en las ventas entre dos documentos de una colección.
  - Rankings de ventas.
  - Totales de ventas acumuladas.
  - Análisis de información compleja de series temporales sin exportar los datos a una base de datos externa.

Operador

- **$setWindowFields**

Sintaxis

```
{
$setWindowFields: {
      partitionBy: <expression>,
      sortBy: {
      <sort field 1>: <sort order>,
      ...,
      <sort field n>: <sort order>
      },
      output: {
      <output field 1>: {
      <window operator>: <window operator parameters>,
      window: {
      documents: [ <lower boundary>, <upper boundary> ],
      range: [ <lower boundary>, <upper boundary> ],
      unit: <time unit>
}
}
```

Operador $setWindowFields

Operadores de ventana

Operadores de acumuladores:

$addToSet, $avg, $bottom, $bottomN, $count, $covariancePop, $covarianceSamp, $derivative, $expMovingAvg, $firstN, $integral, $lastN, $max, $maxN, $median, $min, $minN, $percentile, $push, $stdDevSamp, $stdDevPop, $sum, $top, $topN, .

Operadores de llenado de huecos: $linearFill y $locf.

Operadores de pedidos: $first, $last y $shift.

Operadores de rango: $denseRank, $documentNumber y $rank.

Operador $setWindowFields

Ejemplos con la colección cakesSales

{ _id: 0, type: "chocolate", state: "CA", price: 13, quantity: 120 },
{ _id: 1, type: "chocolate", state: "WA", price: 14, quantity: 140 },
{ _id: 2, type: "vanilla", state: "CA", price: 12, quantity: 145 },
{ _id: 3, type: "vanilla", state: "WA", price: 13, quantity: 104 },
{ _id: 4, type: "strawberry", state: "CA", price: 41, quantity: 162 },
{ _id: 5, type: "strawberry", state: "WA", price: 43, quantity: 134 }

```
db.cakeSales.aggregate( [
{
$setWindowFields: {
partitionBy: "$state",
sortBy: { quantity: -1 },
output: {
documentNumberForState: {
$documentNumber: {}
      }}}
}])
```

Salida

{ "_id" : 4, "type" : "strawberry", "state" : **"CA"**, "price" : 41, "quantity" : 162, "**documentNumberForState**" : **1** }
{ "_id" : 2, "type" : "vanilla", "state" : "CA", "price" : 12, "quantity" : 145, "documentNumberForState" : **2** }
{ "_id" : 0, "type" : "chocolate", "state" : "CA", "price" : 13, "quantity" : 120, "documentNumberForState" : **3** }
{ "_id" : 1, "type" : "chocolate", "state" : **"WA"**, "price" : 14, "quantity" : 140, "**documentNumberForState**" : **1** }
{ "_id" : 5, "type" : "strawberry", "state" : "WA", "price" : 43, "quantity" : 134, "documentNumberForState" : **2** }
{ "_id" : 3, "type" : "vanilla", "state" : "WA", "price" : 13, "quantity" : 104, "documentNumberForState" : **3** }

Operador $setWindowFields

Ejemplos con la colección cakesSales

{ _id: 0, type: "chocolate", state: "CA", price: 13, quantity: 120 },
{ _id: 1, type: "chocolate", state: "WA", price: 14, quantity: 140 },
{ _id: 2, type: "vanilla", state: "CA", price: 12, quantity: 145 },
{ _id: 3, type: "vanilla", state: "WA", price: 13, **quantity: 140 },**
{ _id: 4, type: "strawberry", state: "CA", price: 41, quantity: 162 },
{ _id: 5, type: "strawberry", state: "WA", price: 43, quantity: 134 }

Cambiamos este valor

```
db.cakeSales.aggregate( [
{
    $setWindowFields: {
    partitionBy: "$state",
    sortBy: { quantity: -1 },
    output: {
    denseRankQuantity: {
    $denseRank: {}
    }}}
}])
```

Salida

{ "_id" : 4, "type" : "strawberry", "state" : **"CA"**, "price" : 41, "quantity" : 162, "**denseRankQuantity**" : **1** }
{ "_id" : 2, "type" : "vanilla", "state" : "CA", "price" : 12, "quantity" : 145, "denseRankQuantity" : **2** }
{ "_id" : 0, "type" : "chocolate", "state" : "CA", "price" : 13, "quantity" : 120, "denseRankQuantity" : **3** }
{ "_id" : 1, "type" : "chocolate", "state" : **"WA"**, "price" : 14, "quantity" : 140, "**denseRankQuantity**" : **1** }
{ "_id" : 3, "type" : "vanilla", "state" : "WA", "price" : 13, "quantity" : 140, "denseRankQuantity" : **1** }
{ "_id" : 5, "type" : "strawberry", "state" : "WA", "price" : 43, "quantity" : 134, "denseRankQuantity" : **2** }

Operador $setWindowFields

Ejemplos con la colección cakesSales

{ _id: 0, type: "chocolate", state: "CA", price: 13, quantity: 120 },
{ _id: 1, type: "chocolate", state: "WA", price: 14, quantity: 140 },
{ _id: 2, type: "vanilla", state: "CA", price: 12, quantity: 145 },
{ _id: 3, type: "vanilla", state: "WA", price: 13, quantity: 104 **},**
{ _id: 4, type: "strawberry", state: "CA", price: 41, quantity: 162 },
{ _id: 5, type: "strawberry", state: "WA", price: 43, quantity: 134 }

```
db.cakeSales.aggregate( [
{
$setWindowFields: {
partitionBy: "$state",
sortBy: { orderDate: 1 },
output: {
cumulativeQuantityForState: {
$sum: "$quantity",
window: {
documents: ["unbounded", "current"]
}}}}}])
```

Salida

{ "_id" : 4, "type" : "strawberry", "state" : "CA", "price" : 41, "quantity" : 162, "**cumulativeQuantityForState**" : 162 }
{ "_id" : 0, "type" : "chocolate", "state" : "CA", "price" : 13, "quantity" : 120, "cumulativeQuantityForState" : 282 }
{ "_id" : 2, "type" : "vanilla", "state" : "CA", "price" : 12, "quantity" : 145, "cumulativeQuantityForState" : **427** }
{ "_id" : 5, "type" : "strawberry", "state" : "WA", "price" : 43, "quantity" : 134, "cumulativeQuantityForState" : 134 }
{ "_id" : 3, "type" : "vanilla", "state" : "WA", "price" : 13, "quantity" : 104, "cumulativeQuantityForState" : 238 }
{ "_id" : 1, "type" : "chocolate", "state" : "WA", "price" : 14, "quantity" : 140, "cumulativeQuantityForState" : **378** }