

BBDD orientadas a documentos

MongoDB

Ampliación a Bases de Datos

Profesor: Pablo Ramos

pablo.ramos@u-tad.com

CONSULTAS SIMPLES: Consultas

- find devuelve un cursor al listado de documentos que cumplen los criterios de búsqueda. findOne devuelve un solo documento.
- Sintaxis:

```
db.collection.find(<criteria>, <projection>)  
db.collection.findOne(<criteria>, <projection>)
```

Mongodb / pymongo

Mongodb

```
db.students.find(  
    { "name.first": "Perico"},  
    { name: 1 }  
).sort( { name: 1}  
)
```

CONSULTAS SIMPLES: Operadores sobre consultas

- Comparación
 - \$gt : mayor que
 - \$gte : mayor o igual que
 - \$in : en la lista
 - \$lt : menor que
 - \$lte : menor o igual que
 - \$ne : diferente a
 - \$nin : no en la list

```
db.students.find({ age: { $gt: 18}})
```

```
db.students.find({ branch: {$in: ["Health", "Science"]}})
```

CONSULTAS SIMPLES: Operadores sobre consultas

- Lógicos
 - \$and
 - \$nor
 - \$not
 - \$or

```
db.students.find({ $and: [{ subject: "Math"},  
                           { subject: "Science"}]})
```

- Elemento
 - \$exists : el campo existe
 - \$type : el campo es de un determinado tipo

```
db.students.find({ class: {$exists: true}})
```

CONSULTAS SIMPLES: Operadores sobre consultas

- Evaluación
 - \$mod: realiza el modulo del campo y comprueba si coincide con el que se busca
 - \$regex: búsqueda con expresiones regulares
 - \$text: búsqueda de texto en índices de tipo texto
 - \$where: búsqueda con expresiones JavaScript

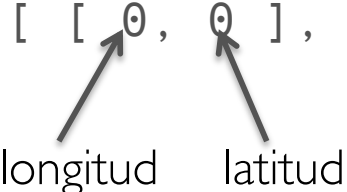
```
db.students.find({ $text: { $search: "Perico" } })
```

CONSULTAS SIMPLES: Operadores sobre consultas

- Geoespaciales
 - \$geoIntersects: documentos que intersecta una geometría especificada
 - \$geoWithin: documentos que se encuentran dentro de una geometría especificada
 - \$nearSphere: lista ordenada de documentos cercanos dentro del rango \$minDistance y \$maxDistance (distancia geodésica)
 - \$near: lista ordenada de documentos cercanos a un punto dentro del rango \$minDistance y \$maxDistance (distancia cartesiana).

```
db.students.find({ loc: {  
    $geoIntersects: {  
        $geometry: {  
            type: "Polygon",  
            coordinates: [  
                [ [ 0, 0 ], [ 3, 6 ], [ 6, 1 ], [ 0, 0 ] ]  
            ]  
        }  
    }  
})
```

<GeoJSON object type>



longitud latitud

CONSULTAS SIMPLES: Operadores sobre consultas

- Arrays
 - Consulta normal: documentos con un array donde al menos un elemento cumpla la consulta.
 - \$elemMatch: documentos con un array donde al menos un elemento cumpla una consulta multiple.
 - \$all: todos los elementos de la lista están en el array
 - \$size: tamaño del array coincide con el expecificado

```
db.students.find({ grades:  
    { $elemMatch:  
        { $gt: 8, $lte: 10 } }  
    })
```

PROYECCIONES

- Limitación en arrays
 - \$: limita el resultado de una consulta en documentos con arrays, devolviendo únicamente la primera coincidencia del array que cumpla la consulta.
 - \$elemMatch: limita el resultado de una consulta en documentos con arrays devolviendo únicamente la primera coincidencia del array en función de una condición especificada en la proyección. Puede devolver un documento con el array vacío.
 - \$slice: limita el resultado de una consulta en documentos con arrays devolviendo los documentos dentro de los índices especificados.

```
db.students.find({ subject:
                  { $all:["Biology", "Maths"]}},
                  {"subject.$": 1 })
```


CONSULTAS ACTUALIZACIÓN: Insert

- SINTAXIS:

```
db.collection.insertOne(<document or array of documents>)
```

- Devuelve la id del documento insertado o la lista de ids

- Ejemplos:

```
db.students.insertOne({ name: "Perico"})
```

```
db.students.insertMany(  
    [{ name: "Perico"},  
    { name: "Paquito"},  
    { name: "Luisito"}]  
)
```

CONSULTAS ACTUALIZACIÓN: Update

- SINTAXIS:

```
db.collection.updateOne(<query>,<document>,<options>)
```

```
db.collection.updateMany(<query>,<document>,<options>)
```

```
db.collection.replaceOne(<query>,<document>,<options>)
```

- Opciones:

- Upsert: si no existe lo inserta

- Devuelve el resultado de la operación

- Ejemplos:

```
db.students.updateOne( { name: "Perico"},  
                      {$set: {name: "Pablito", age: 21}}  
                      )
```

CONSULTAS ACTUALIZACIÓN: Operadores Update

- Operadores sobre campos:
 - \$currentDate: Introduce la fecha actual
 - \$inc: Incrementa el valor actual según el valor especificado
 - \$max: Actualiza si el campo es mayor que el especificado
 - \$min: Actualiza si el campo es menor que el especificado
 - \$mul: Multiplica el campo por el valor especificado
 - \$rename: Cambia el nombre del campo
 - \$setOnInsert: Inicializa o cambia el valor de un campo si se inserta un nuevo documento.
 - \$set: Inicializa o cambia el valor de un campo
 - \$unset: Elimina el campo del documento.
- Ejemplo:

```
db.students.updateOne({ name: "Perico"},  
                      { $setOnInsert: {freshmen: true}},  
                      {upsert: true})
```

CONSULTAS ACTUALIZACIÓN: Operadores Update

- Operadores sobre arrays:
 - \$: El operador afecta a la primera coincidencia de un campo lista.
 - \$[]: El operador afecta a todos los elementos de un campo lista.
 - \$addToSet: Añade un elemento al campo lista si todavía no existe.
 - \$pop: Elimina el primer o último elemento de un campo lista.
 - \$pullAll: Elimina todos los elementos de un campo lista que están en una lista especificada.
 - \$pull: Elimina todos los elementos de un campo lista que cumplen una condición
 - \$pushAll: Añade los elementos especificados en una lista a un campo lista.
 - \$push: Añade un elemento a un campo lista
- Ejemplo:

```
db.students.update({ name: "Perico", subjects:"Maths"},
                  { $set:
                    {"subjects.$": "Applied Maths"}}
                  )
```

CONSULTAS ACTUALIZACIÓN: Operadores Update

- Modificadores de arrays:
 - \$each: Añade cada uno de los elementos de una lista a un campo lista. Se usa con \$addToSet y \$push
 - \$position: Inserta cada uno de los elementos de una lista (\$push+\$each) en una posición específica de un campo lista.
 - \$slice: Limita el tamaño de un campo lista durante la inserción de elementos (\$push+\$each)
 - \$sort: Ordena los elementos de un campo lista durante la inserción de elementos (\$push+\$each)

- Ejemplo:

```
db.students.update({ name: "Perico"},  
                  { $push: { sport_preferences: {  
    $each: [ {name:"Football", priority: 2}  
              {name:"Basket", priority: 1}],  
    $sort: {priority: 1 } } } })
```

CONSULTAS ACTUALIZACIÓN: Operadores Update

- Otros:
 - \$bit: Para realizar operaciones binarias.
 - \$isolation: Atomiza una operación de modo que nadie puede acceder a los cambios hasta que la última operación se realiza
- Ejemplo:

```
db.students.updateOne( { freshmen: true, isolated: 1 },  
                        { year: 1 })
```

CONSULTAS ACTUALIZACIÓN: Find and modify

- SINTAXIS:

```
db.collection.findAndModify(query: <document>,  
                             update: <document>,  
                             remove: <Boolean>  
                             ...)
```

- Modifica/elimina y devuelve un único documento. Por defecto devuelve el documento sin modificar. Para devolver el objeto modificado usar `new: true`.

- Ejemplo:

```
db.students.findAndMmodify(query: {name: "Perico"},  
                           update: {branch: "Health"},  
                           new: true )
```

CONSULTAS ACTUALIZACIÓN: Save

- SINTAXIS:

```
db.collection.save(<document>)
```

- Actualiza un documento o lo inserta si no existe.

- Si se especifica una `_id`, se ejecuta update

- Si no existe la id, se ejecuta insert

- Si no se especifica un `_id`, se ejecuta insert

- Ejemplos:

```
#Insert
```

```
db.students.save({name: "Perico"})
```

```
#Update
```

```
db.students.save({_id: 1, name: "Perico"})
```


CONSULTAS ACTUALIZACIÓN: Remove

- SINTAXIS:

`db.collection.remove(<query>)`

- Elimina un elemento (si se especifica `justOne: true`) o varios que cumplan la consulta.

- Ejemplo:

`db.students.remove({ branch: "Health"}, justOne: true)`

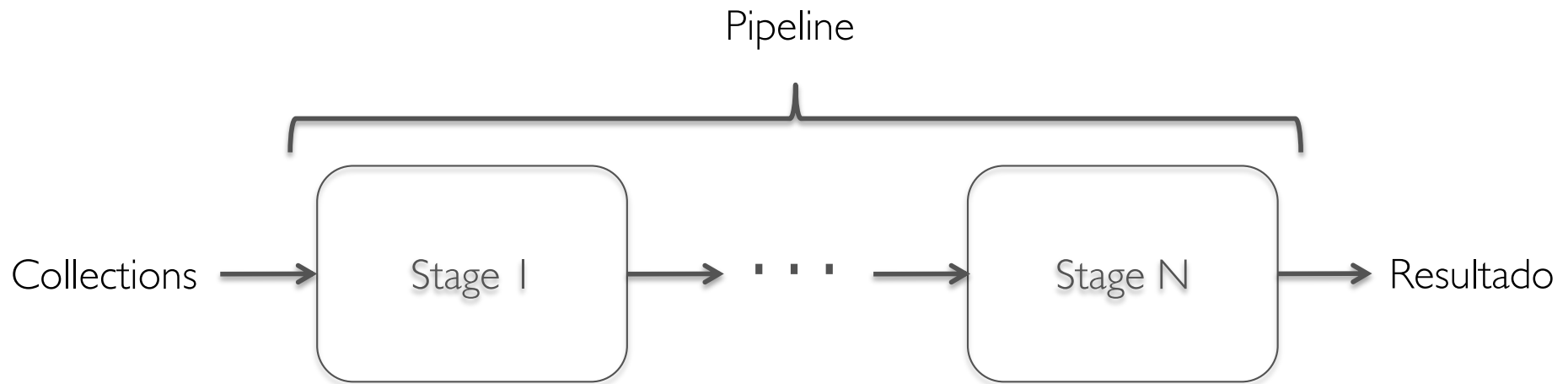
ÍNDICES

- `createIndex()`: Crea un índice en el campo especificado si este no ha sido creado ya
 - Tipos de índices:
 - Ascending (1), descending (-1), text
 - geo2d, geosphere
 - Avanzados: geohaystack, hashed
- `dropIndex()`: Elimina un índice
- `reindex()`: reconstruye los índices de una colección.

```
db.students.createIndex(  
    { name: 1 },  
    { unique: true }  
)
```

CONSULTAS AGREGADAS: Stages

- Devuelve un listado de los documentos resultado de las operaciones realizadas en cada etapa



```
db.collection.aggregate( [ { <stage> }, ... ] )
```

CONSULTAS AGREGADAS: \$match

- Filtra según condición especificada. Utilizar al principio del pipeline para mejorar el rendimiento de la consulta.
- La sintaxis es la misma utilizada para find y findOne. Se pueden utilizar los mismos operadores.

- SINTAXIS:

```
{ $match: {<query>} }
```

- Ejemplo:

```
db.students.aggregate([  
  { $match: { "name.first": "Perico",  
              "name.second": "García" }  
}] )
```

CONSULTAS AGREGADAS: \$lookup

- Realiza una consulta en la que combina registros de dos o mas colecciones (JOIN).

- SINTAXIS:

```
{ $lookup:
  { from: <collection to join>,
    localField: <field from the input documents>,
    foreignField: <field from the documents of the
"from" collection>,
    as: <output array field> }
```

- Ejemplo:

```
db.university.aggregate([
  { $lookup: {from: "students" ,
              localfield: "name",
              foreignField: "university",
              as: "students"}
  }])
```

CONSULTAS AGREGADAS: \$group

- Agrupa documentos en función de una expresión especificada.

- SINTAXIS:

```
{ $group: { _id: <expression>,  
  <field1>: { <accumulator1>: <expression1> }, ... }
```

- Ejemplo:

```
db.students.aggregate([  
  { $match: {branch: "Health"}}  
  { $group: {_id: "$branch"  
    students_num: {$sum: 1}  
  }}  
])
```

CONSULTAS AGREGADAS: Operadores expresiones

- Acumuladores
 - \$sum: suma de campos de los documentos agrupados o números
 - \$avg: media de variables de los documentos agrupados o números
 - \$first: devuelve el valor de un campo del primer documento agrupado
 - \$last: devuelve el valor de un campo del último documento agrupado
 - \$max: devuelve el valor máximo de un campo de los documentos agrupados.
 - \$min: devuelve el valor mínimo de un campo de los documentos agrupados.
 - \$push: devuelve una lista con el campo especificado de cada uno de los documentos agrupados
 - \$addToSet: devuelve un conjunto con el campo especificado de cada uno de los documentos agrupados

- Ejemplo:

```
db.students.aggregate([
  { $group: { _id: "$branch",
              students_num: {$sum: 1},
              total_avg_grade: {$avg: "$avg_score"},
              degree: {$addToSet: "$degree"}}
  }])
```

CONSULTAS AGREGADAS: \$unwind

- Descompone una lista de uno o varios documentos creando tantos documentos como elementos del lista

- SINTAXIS:

```
{ $unwind: <field path> }
```

- Ejemplo:

```
db.students.aggregate([  
    { $match: {"name.first": "Perico"}}  
    { $unwind: "$subjects" } ])
```


CONSULTAS AGREGADAS: \$sort

- Ordena los documentos.
- SINTAXIS:

```
{ $sort: { <field1>: <sort order>, ... } }
```
- Ejemplo:

```
db.students.aggregate([  
  { $sort: {avg_grade: 1, fails: -1}}  
])
```

CONSULTAS AGREGADAS: \$limit

- Limita el numero de documentos
- SINTAXIS:
`{ $limit: <positive integer> }`
- Ejemplo:
`db.students.aggregate([
 { $limit: 15 }])`

CONSULTAS AGREGADAS: \$skip

- Se salta los primeros documentos y devuelve el resto.
- SINTAXIS:

```
{ $skip: <positive integer> }
```
- Ejemplo:

```
db.students.aggregate([  
  { $skip: 5 } ])
```

CONSULTAS AGREGADAS: \$geoNear

- Ordena los documentos del más cercano a más lejano de un punto especificado

- SINTAXIS:

```
{ $geoNear: { <geoNear options> } }
```

- Ejemplo:

```
db.students.aggregate([{$geoNear: {  
    near: { type: "Point",  
            coordinates: [ -73.99279 , 40.719296 ] },  
    maxDistance: 9,  
    query: { branch: "health" },  
    distanceField: "dist.calculated",  
    includeLocs: "dist.location",  
    num: 5,  
    spherical: true  
    }  
  ]})
```

CONSULTAS AGREGADAS: \$out

- Escribe el resultado de la consulta agregada en una colección. Debe ir el último del pipeline.
- Si no existe una colección con ese nombre la crea, sino la sobrescribe.

- SINTAXIS:

```
{ $out: "<output-collection>" }
```

- Ejemplo:

```
db.students.aggregate([  
    { $match: {"branch": "Health"}}  
    { $out: "health_students" } ])
```

CONSULTAS AGREGADAS: \$project

- Remodela documentos (añadiendo o quitando campos). Por cada documento que entra sale uno.
- SINTAXIS:

```
{ $project: { <specifications> } }
  <field>: 1/0 o true/false
  <field>: <expression>
```

 - El campo `_id` se añade por defecto, el resto de campos hay que especificarlos.
 - Se pueden utilizar operadores para crear los nuevos campos.
- Ejemplo:

```
db.students.aggregate([
  { $project : { "name.first":1,
                 branch:1,
                 abbrev_subjects: { $substr: ["$subjects", 0, 3] } } } ] )
```

CONSULTAS AGREGADAS: Operadores expresiones

- Operadores booleanos (true o false):
 - \$and: y
 - \$or: o
 - \$not: no
- Operadores de comparación (true o false):
 - \$cmp: devuelve 0 si son iguales, 1 si el primer es mayor que el 2 y -1 si el primer valor es menor que el segundo
 - \$eq: es igual
 - \$gt : mayor que
 - \$gte : mayor o igual que
 - \$in : en la lista
 - \$lt : menor que
 - \$lte : menor o igual que
 - \$ne : diferente a

CONSULTAS AGREGADAS: Operadores expresiones

- Operadores sobre sets (no funcionan con arrays anidados):
 - \$setEquals: devuelve true si son iguales, false en caso contrario
 - \$setIntersection: devuelve los elementos comunes de dos o más arrays
 - \$setUnion: devuelve la unión de dos o más arrays
 - \$setDifference: devuelve los elementos que aparecen en el primer array pero no en el segundo
 - \$setIsSubset: devuelve true si el primer array es un subconjunto del segundo array
 - \$anyElementTrue: devuelve true si cualquier elemento del array es true
 - \$allElementsTrue: : devuelve true ningún elemento del array es false

CONSULTAS AGREGADAS: Operadores expresiones

- Operadores aritméticos:
 - \$add: suma dos campos/números o una variable/número a una fecha.
 - \$subtract: resta dos campos/números o una variable/número a una fecha.
 - \$multiply: multiplica dos campos/números
 - \$divide: divide dos campos/números
 - \$mod: devuelve el resto al dividir dos campos/números

CONSULTAS AGREGADAS: Operadores expresiones

- Operadores sobre cadenas:
 - \$concat: concatena dos o más campos/cadenas.
 - \$substr: devuelve una subcadena del campo/cadena especificada.
 - \$toLower: cambia a minúsculas un campo/cadena
 - \$toUpper: cambia a mayúsculas un campo/cadena
 - \$strcasecmp: realiza una comparación entre dos cadenas (no tiene en cuenta mayúsculas y minúsculas). Devuelve cero si las cadenas son iguales, 1 si la primera cadenas e mayor que la segunda y -1 en caso contrario.

CONSULTAS AGREGADAS: Operadores expresiones

- Operadores sobre listas:
 - \$size: devuelve el tamaño de un lista.
- Operadores sobre variables
 - \$map: aplica una expresión a cada uno de los elementos de un lista y devuelve un lista con los resultados.
 - \$let: permite aplicar expresiones sobre variables creadas dinámicamente.
- Operadores sobre literales
 - \$literal: devuelve un valor sin analizarlo. Se usa cuando no quieres que se evalúe una cadena que puede confundirse con una expresión.

CONSULTAS AGREGADAS: Operadores expresiones

- Operadores sobre fechas:
 - \$dayOfYear: 1-366
 - \$dayOfMonth: 1-31
 - \$dayOfWeek: 1 (domingo)-7
 - \$year
 - \$month: 1-12
 - \$week: 0-53
 - \$hour: 0-23
 - \$minute: 0-59
 - \$second: 0-59
 - \$millisecond: 0-999

CONSULTAS AGREGADAS: Operadores expresiones

- Expresiones condicionales:
 - \$cond: realiza un "if-then" con una expresión booleana. Si se cumple la expresión, se devuelve el primer valor especificado, sino se cumple lo contrario.
 - \$ifnull: evalúa expresión y si es null, devuelve un valor especificado