MongoDB

OPERACIONES BÁSICAS

`db.createCollection('name')`	Crear nueva colección	`c.find({query},{projection:1})`	Consulta
`db.dropDatabase()`	Borrar bd	`c.distinct('field',{query})`	Consulta documentos distintos
`c.drop()`	Borra colección	`distinct.length`	Devuelve el número de documentos distintos
`c.insertOne(object)`	Insertar un solo objeto	`f.sort({field:1})`	Ordenar resultado
`c.insertMany(objects array)`	Insertar un array de objetos	`limit(num)`	Limitar el número de resultados
`c.deleteOne({query})`	Borra un registro	`size()`	Devuelve el número de documentos
`c.countDocuments({query})`	Contar documentos		

ÚTILES

`{field:{$eq:value}}`	Igual que...	`{field:{$all:[value,...]}}`	Contenga todos los valores
`{field:{$gt:value}}`	Mayor que...	`{$or:[{query1},{query2}...]}`	Or
`{field:{$gte:value}}`	Mayor o igual que...	`{$and:[{query1},{query2}...]}`	And
`{field:{$lt:value}}`	Menor que...	`{field:{$not:value}}`	No es...
`{field:{$lte:value}}`	Menor o igual que...	`{field:{$exists:true}}`	Campo existe
`{field:{$ne:value}}`	Distinto a...	`/^String/`	Comienza por...
`{field:{$in:[value,...]}}`	Está en una lista de valores	`/String/i`	Contiene... Ignorando mayúsculas
`{field:{$nin:[value,...]}}`	No está en una lista de valores	`/String$/`	Termina por...

ACTUALIZACIÓN

`c.updateOne({query},{$set:{field:change}},{upsert:true})`	Sintaxis básica de actualización. Upsert true inserta si no existe		
`c.updateOne({},{$rename:{field:'rename',field:'rename',...}})`	Sintaxis básica para renombrar campos		
`c.updateOne({},{$unset:{field:''}})`	Elimina campo	`c.updateOne({},{$inc:{field:1}})`	Incrementa/Decrementa campo

ARRAYS

`c.find({array_field:'value'})`	usca valores en un array	`c.updateOne({},{$pop:{array_field:-1}})`	Elimina último elemento
`c.updateOne({},{$push:{array_field:'value'}})`	Inserta aunque exista	`c.updateOne({},{$pull:{array_field:'value'}})`	Elimina ese elemento
`c.updateOne({},{$addToSet:{array_field:'value'}})`	Inserta si no existe	`c.find({array_field:{$elemMatch:{'field':{$gt:'value'}}}})`	
`c.updateOne({},{$push:{array_field:{$each:['value',..]}}})`	Inserta varios valores en un array		
`c.updateOne({array_field:'value'},{$set:{'array_field.$':'reValue'}})`	Actualiza el campo del array sin especificar el índice		
`c.updateOne({query},{$set:{'array_field.index':'reValue'}})`	Actualiza el campo del array especificando el índice		

PIPELINE

`c.agregate([{stage1},{stage2},...]`	Sintaxis básica de un Pipeline
`{$project:{newField:'$field'}`	Etapa que permite modificar la representación de los datos creando nuevos campos
`{$match:{query,query}}`	Etapa que permite filtrar documentos, se suele usar al principio del Pipeline
`{$group:{_id,newField:{$sum:'$field'}}}`	Etapa que permite agrupar, también se pueden crear nuevos campos
`{$sort:{field:1,field:-1}}`	Etapa que permite ordenar de manera ascendente (1) o descendente
`{$skip:num}`	Etapa que omite los X primeros documentos
`{$limit:num}`	Etapa que limita los resultados a X
`{$count:{"name"}}`	Crea un campo 'name' donde cuenta el número de resultados del pipeline
`newField:{$cond:[condition,valueIfTrue,valueIfFalse]}`	Condicional, se suele usar en la creación de nuevos campos
`newField:{$ifNull:[valueIfNotNull,valueIfNull]}`	Verifica si un campo es null o no, para asignarle un valor u otro
`newField:{$arrayElemAt:['$arrayField',index]}`	Devuelve el elemento en el índice especificado de un array.
`newField:{$concatArrays:['$array1','$array1'...]}`	Devuelve un array concatenado
`newField:{$isArray:{'$field'}}`	Devuelve true o false en función de si el campo es un array
`{$out:'nameOut'}`	Etapa que permite guardar el resultado de un Pipeline en una colección específica

GROUP

`$sum:'$field'`	Suma
`$avg:'$field'`	Media
`$first:'$field'`	Devuelve el primer valor del grupo
`$last:'$field'`	Devuelve el último valor del grupo
`$max:'$field'`	Devuelve el valor máximo
`$min:'$field'`	Devuelve el valor mínimo

CADENAS

`$concat:[]`	Concatena varias cadenas
`$substr:['$field',start,end]`	Extrae una subcadena
`$toLower:'$field'`	Convierte en minúsculas
`$toUpper:'$field'`	Convierte en mayúsculas

ARITMÉTICAS

`$abs:'$field'`	Devuelve el valor absoluto
`$add:['$field',num]`	Suma de números o milis a fechas
`$ceil:'$field'`	Redondeo hacia arriba
`$floor:'$field'`	Redondeo hacia abajo
`$divide:['$field',num]`	Divide
`$mod:['$field',num]`	Resto
`$multiply:['$field',num]`	Multiplica
`$pow:['$field',num]`	Potencia
`$sqrt:'$field'`	Raíz cuadrada
`$substract:['$field1','$field2']`	Resta
`$trunc:['$field',num]`	Trunca

XML

XPATH

Expresiones comunes

`e1//e2`	**E2** descendiente cualquiera
`//e1`	**E1** ubicado a cualquier nivel
`e1[filter]`	Filtro
`@Attribute`	Atributo
`*`	Cualquier elemento
`@*`	Nodo padre
`.`	This
`..`	Nodo padre

Funciones varias

`/text(),/string(),/data()`	Contenido textual
`number()`	Trata como número
`round(),abs(),floor(),ceiling()`	Funciones numéricas, todas 1 parámetro
`count(),avg(),max(),min(),sum()`	Funciones de agregado
`position()`	Devuelve posición
`element[n]`	Devuelve el elemento de esa posición
`last()-i`	Último, penúltimo....
`name()`	Nombre del nodo actual
`root()`	Elemento raíz
`node()`	Nodos descendientes del actual

Funciones de cadenas

`substring('Value',1,4)='Valu'`	Subcadena
`starts-with('Value','V')=true`	Comienza por...
`contains('Value','Va')=true`	Contiene
`concat('Value','Va')=ValueVa`	Concatena
`translate('Value','Va','Wa')=Walue`	Cambia carácteres
`string-length('Value')=5`	Longitud
`upper-case('Value')=VALUE`	Cadena a mayúsculas, también **lower-case()**

Operadores

`and,or,not,=,!=` `>,<,>=,<=`	Booleanos
`+ ,-,*,div,mod`	Aritméticos, **div y mod** no llevan paréntesis
`(node1 | node2)`	Devuelve conjuntos de nodos

XQUERY

`for $var in (path || (1 to 5))`	Estructura básica de un for
`for $var at $i in path`	El **AT** sirve para enumerar
`for $var in distinct_values(path)`	Valores distintos
`return <label> {$var} <label>`	Return con etiquetas modificadas
`return element {label}{$var}`	Return con etiquetas modificadas 2
`let $var := count(path)`	Crear y asignar valor a una variable
`where`	Condición normal
`order by $path ascending`	Ordena, poner antes del **return**.
`if () then else`	Estructura **IF**, se puede usar en variables
`rename node path as "value"`	Renombrar 1 nodo
`for... return rename node...`	Renombrar varios nodos
`insert node xml code`	Insertar nodo
`(before || after) /path[position]`	**Opc.1 -** Antes o despues de una posición
`as (first || last) into /path`	**Opc.2** - Al principio o al final del **xml**
`replace value of node path[position] with`	Actualizar un valor
`for... return replace...`	Actualizar varios valores
`delete node path[position]`	Elimina un nodo
`for... return delete node...`	Elimina varios nodos