Del mismo modo que en los ejemplos anteriores, restauramos la BBDD commerceMulti para realizar los ejemplos con índices de texto. A modo de ejemplo de búsquedas complejas y para verificar el buen funcionamiento de los índices, vamos a realizar una búsqueda de zapatos hechos de madera o hierro que sean “Awesome” (increíbles) o “Handcrafted” (artesanos):

```
> db.commerces.find({item:{$regex:/Shoes/} ,$or:[{mat:"Wooden"},{mat:"Steel"}],cat:{$in:["Awesome","Handcrafted"]}}).explain("executionStats")
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "commerceMulti.commerces",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "$and" : [
                                {
                                        "$or" : [
                                                {
                                                        "mat" : {
                                                                "$eq" : "Wooden"
                                                        }
                                                },
                                                {
                                                        "mat" : {
                                                                "$eq" : "Steel"
                                                        }
                                                }
                                        ]
                                },
                                {
                                        "item" : {
                                                "$regex" : "Shoes"
                                        }
                                },
                                {
                                        "cat" : {
                                                "$in" : [
                                                        "Awesome",
                                                        "Handcrafted"
                                                ]
                                        }
                                }
                        ]
                },
                "winningPlan" : {
                        "stage" : "COLLSCAN",
                        "filter" : {

                                "$and" : [
                                        {
                                                "$or" : [
                                                        {
                                                                "mat" : {
                                                                        "$eq" : "Wooden"
```

```
                                                }
                                        },
                                        {
                                                "mat" : {
                                                        "$eq" : "Steel"
                                                }
                                        }
                                ]
                        },
                        {
                                "item" : {
                                        "$regex" : "Shoes"
                                }
                        },
                        {
                                "cat" : {
                                        "$in" : [
                                                "Awesome",
                                                "Handcrafted"
                                        ]
                                }
                        }
                ]
        },
        "direction" : "forward"
},
"rejectedPlans" : [ ]
},
"executionStats" : {
"executionSuccess" : true,
"nReturned" : 32,
"executionTimeMillis" : 28,
"totalKeysExamined" : 0,
"totalDocsExamined" : 12000,
"executionStages" : {
        "stage" : "COLLSCAN",
        "filter" : {
                "$and" : [
                        {
                                "$or" : [
                                        {
                                                "mat" : {
                                                        "$eq" : "Wooden"
                                                }
                                        },
                                        {
                                                "mat" : {
                                                        "$eq" : "Steel"
                                                }
                                        }
                                ]
                        },
                        {
                                "item" : {
                                        "$regex" : "Shoes"
                                }
                        },
                        {
                                "cat" : {
```

```
                                        "cat" : {
                                                "$in" : [
                                                        "Awesome",
                                                        "Handcrafted"
                                                ]
                                        }
                                }
                        ]
                },
                "nReturned" : 32,
                "executionTimeMillisEstimate" : 30,
                "works" : 12002,
                "advanced" : 32,
                "needTime" : 11969,
                "needYield" : 0,
                "saveState" : 93,
                "restoreState" : 93,
                "isEOF" : 1,
                "invalidates" : 0,
                "direction" : "forward",
                "docsExamined" : 12000
        }
    },
    "serverInfo" : {
            "host" : "7ea9850856d3",
            "port" : 27017,
            "version" : "3.4.2",
            "gitVersion" : "3f76e40c105fc223b3e5aac3e20dcd026b83b38b"
    },
    "ok" : 1
}
```


La búsqueda devuelve 32 resultados y se toma en total 28 milisegundos para ser ejecutada en su totalidad. De cara a mejorar la eficiencia para este ejemplo, vamos a realizar una serie de comprobaciones que determinen el correcto funcionamiento de los índices de texto:

1. Creamos un índice simple en item y realizamos la query con explain.

```
db.commerces.createIndex({item:1})
```

2. Hacemos la misma query buscando para resultados exactos con explain (observar que sigue examinando los 12000 documentos, esto es por la expresión regular).

```
db.commerces.find({item:{$regex:/ /},$or:[{mat:"Wooden"},{mat:"Steel"}],cat:{$in:["Awesom
e","Handcrafted"]}}).explain("executionStats")
```

3. Definimos item como un índice de texto.

```
db.commerces.createIndex({item:"text"}
```

4. Realizamos de nuevo la búsqueda usando el comando $text y explain (observar cómo decrece el tiempo de ejecución y sobre todo el número de documentos examinados).

```
db.commerces.find({$text:{$search:"Shoes"} ,$or:[{mat:"Wooden"},{mat:"Steel"}],cat:{$in:
["Awesome","Handcrafted"]}}).explain("executionStats")
```

5. Definimos índices de texto en item y cat y un índice simple en mat (ya que serán valores discretos y podremos hacer que la búsqueda siempre sea exacta)

```
 db.commerces.getIndexes()
> db.commerces.dropIndex("item_text")
> db.commerces.dropIndex("item_1")
> db.commerces.ensureIndex({item:"text",cat:"text",mat:1})
```

6. Ver con getIndexes() las propiedades de los índices definidos

```
 db.commerces.getIndexes()
```

7. Buscamos en commerce por resultados que tengan la palabra X en item pero no la palabra Y

```
 db.commerces.find({$text:{$search:"Shoes"},$or:[{mat:"Wooden"},{mat:"Steel"}],cat:{$in:
["Awesome","Handcrafted"]}}).count()

> db.commerces.find({$text:{$search:"Shoes -Wooden"},$or:[{mat:"Wooden"},{mat:"Steel"}],ca
t:{$in:["Awesome","Handcrafted"]}}).count()
```

8. Buscamos por varias palabras

```
> db.commerces.find({$text:{$search:"Shoes Wooden"} ,$or:[{mat:"Wooden"},{mat:"Steel"}],ca
t:{$in:["Awesome","Handcrafted"]}}).count()
```

9. Buscamos con caseSensitive

```
> db.commerces.find({$text:{$search:"shoes wooden",$caseSensitive:true} ,$or:[{mat:"Woode
n"},{mat:"Steel"}],cat:{$in:["Awesome","Handcrafted"]}}).count()

> db.commerces.find({$text:{$search:"shoes wooden",$caseSensitive:false} ,$or:[{mat:"Woode
n"},{mat:"Steel"}],cat:{$in:["Awesome","Handcrafted"]}}).count()
```

10. Buscamos por puntuación del resultado (no habiendo definido previamente los pesos)

```
> db.commerces.find({$text:{$search:"shoes wooden",$caseSensitive:false} ,$or:[{mat:"Woode
n"},{mat:"Steel"}],cat:{$in:["Awesome","Handcrafted"]}},{item:1,_id:0,score: {$meta: "text
Score"}}).sort({score:{$meta:"textScore"}}).pretty()
```

11. Buscamos por frase completa

```
db.commerces.find({$text:{$search:"\"Wooden Shoes\""} ,$or:[{mat:"Wooden"},
{mat:"Steel"}],cat:{$in:["Awesome","Handcrafted"]}}).count()
```

12. Cambiamos el peso de las puntuaciones para hacer búsquedas y puntuaciones mejores

```
> db.commerces.dropIndex("item_text_cat_text_mat_1")
> db.commerces.createIndex({item:"text",cat:"text",mat:1},{weights: {item:10,cat:5}})
```

13. Repetimos búsqueda de puntuaciones

```
>db.commerces.find({$text:{$search:"shoes wooden Awesome
Handcrafted",$caseSensitive:false} ,$or:[{mat:"Wooden"},{mat:"Steel"}]},{item:1,_id:0,scor
e: {$meta: "textScore"}}).sort({score:{$meta:"textScore"}}).pretty()
```

Después de realizar todos los ejercicios, es obvio que emplear índices de texto supone un salto de eficiencia en cuanto a rendimiento de la BBDD cuando manejemos colecciones con mucho contenido de este tipo de datos. Además es vital realizar la búsquedas empleando el comando `$text` .