Tipos de datos

Levantamos contenedor y entramos a mongo

```
var date = new Date()
date
var array = ["el1", 2, true, null]
array
```

1 a 1

Cuando la relación entres dos entidades sea uno a uno, como por ejemplo entre un libro y su autor, normalmente la opción a la hora de diseñar la colección será embeber documentos, de tal forma que toda la información quede recogida en el mismo objeto.

Esto aplicará cuando ambos recursos (autor y libro) no sean requeridos de manera recurrente por otras colecciones, como por ejemplo si un autor hubiera escrito 1 millón de libros. Sería más eficiente en cuanto a espacio en disco escribir el autor una única vez y referenciarlo desde los libros (si el espacio no fuera algo finito, sería más eficiente para la aplicación embeber el documento siempre).

Es importante llegar a una solución de compromiso para según qué caso estemos resolviendo y que requerimientos tengamos para nuestra aplicación, que será lo que marque el diseño de la BBDD.

```
use oneToOne
db.articulos.insert({titulo:"Cien años de soledad", pages:550, autor:{name:"Gabriel García Márq
uez", location:"Colombia"}})
db.articulos.find().pretty()
```

1 a n

Cuando la relación entre dos entidades es 1 a n, normalmente se aplicará la misma norma que para los documento 1 a 1 vista con anterioridad, es decir, se embeberan los documentos salvo que el acceso a ellos sea recurrente desde otras areas de la aplicación.

Imaginemos un blog en el que tenemos artículos y tenemos comentarios dentro de cada artículo, normalmente los comentarios aparecen de manera exclusiva en ese artículo, luego en este caso, lo ideal será embeber los comentarios en el artículo.

```
> use oneToN
switched to db oneToN
> db.articulos.insert({Title:"Aprendiendo mongo con Pablo Campos", Content:"OpenWebinars mola
 :) !", comentarios: [{usuario:"pcampos",contenido:"Mola tu curso!"},{usuario:"Paquito", conten
ido:"La verdad es que me entero de todo"}]})
WriteResult({ "nInserted" : 1 })
> db.articulos.find().pretty()
{
        "_id" : ObjectId("58a20397c89278b620355926"),
        "Title" : "Aprendiendo mongo con Pablo Campos",
        "Content" : "OpenWebinars mola :) !",
        "comentarios" : [
                {
                        "usuario" : "pcampos",
                        "contenido" : "Mola tu curso!"},{
                        "usuario" : "Paquito",
                        "contenido" : "La verdad es que me entero de todo"}]}
```

Sin embargo, si cada comentario incluyera información del usuario que lo hace (avatar, nombre, puesto de trabajo y edad) y la repetición de comentarios por parte de los mismos usuarios en muchos artículos fuera algo generalizado, sería más eficiente referenciar en cada comentario al autor del mismo para evitar redundar información.

```
> use oneToNReference
switched to db oneToNReference
> db.comentarios.insert([{username:"pablo",contenido:"De lujo"},{username:"paco",contenido:"Muy
 bien"},{username:"ignacio",contenido:"Regular"}])
> db.comentarios.find().pretty()
{       "_id" : ObjectId("58a204c9c89278b620355927"),
        "username" : "pablo",
        "contenido" : "De lujo"}{
        "_id" : ObjectId("58a204c9c89278b620355928"),
        "username" : "paco",
        "contenido" : "Muy bien"}{
        "_id" : ObjectId("58a204c9c89278b620355929"),
        "username" : "ignacio",
        "contenido" : "Regular"}
>db.articulos.insert({Titulo:"Ejemplo referenciando", comentarios:[ObjectId("58a204c9c89278b620
355927"),ObjectId("58a204c9c89278b620355928"),ObjectId("58a204c9c89278b620355929")]})
WriteResult({ "nInserted" : 1 })
> db.articulos.find().pretty()
{
        "_id" : ObjectId("58a20509c89278b62035592a"),
        "Titulo" : "Ejemplo referenciando",
        "comentarios" : [
                ObjectId("58a204c9c89278b620355927"),
                ObjectId("58a204c9c89278b620355928"),
                ObjectId("58a204c9c89278b620355929")]}
```

N a n

Cuando la relación entre las entidades es de muchos a muchos, se aplica la misma lógica que en los casos anteriores, la referenciación o no de documentos dependerá de la repetición que se de de los mismos y de la operatividad de la aplicación.

Imaginemos que tenemos un blog donde muchos autores distintos escriben (incluso de manera conjunta) muchos artículos cada uno. Para este caso la información relativa a los autores se repetirá con asiduidad y será más conveniente referenciar la misma.

```
> use NToN
switched to db NToN
> db.autores.insert([{username:"JaviSant", edad:21},{username:"pcampos",edad:27}])

> db.autores.find().pretty()
{
        "_id" : ObjectId("58a20673c89278b62035592b"),
        "username" : "JaviSant",
        "edad" : 21
}
{
        "_id" : ObjectId("58a20673c89278b62035592c"),
        "username" : "pcampos",
        "edad" : 27
}
> db.articulos.insert({Titulo:"La casa por la venta", autores:[ObjectId("58a20673c89278b62035592b"), ObjectId("58a20673c89278b62035592c")]})
WriteResult({ "nInserted" : 1 })
> db.articulos.insert({Titulo:"La casa por la ventaNA II", autores:[ObjectId("58a20673c89278b62035592b"), ObjectId("58a20673c89278b62035592c")]})
WriteResult({ "nInserted" : 1 })
> db.articulos.insert({Titulo:"La casa por la ventaNA III - El retorno", autores:[ObjectId("58a20673c89278b62035592b"), ObjectId("58a20673c89278b62035592c")]})
WriteResult({ "nInserted" : 1 })
Además, para el perfil de cada autor podríamos querer ver cuántos y cuáles artículos ha escrito…
> db.articulos.find()
{ "_id" : ObjectId("58a206a1c89278b62035592d"), "Titulo" : "La casa por la venta", "autores" : [ ObjectId("58a20673c89278b62035592b"), ObjectId("58a20673c89278b62035592c") ] }
{ "_id" : ObjectId("58a206a7c89278b62035592e"), "Titulo" : "La casa por la ventaNA II", "autores" : [ ObjectId("58a20673c89278b62035592b"), ObjectId("58a20673c89278b62035592c") ] }
{ "_id" : ObjectId("58a206afc89278b62035592f"), "Titulo" : "La casa por la ventaNA III - El retorno", "autores" : [ ObjectId("58a20673c89278b62035592b"), ObjectId("58a20673c89278b62035592c") ] }
> db.autores.update({username:"JaviSant"},{$set:{articulos:
[ObjectId("58a206a1c89278b62035592d"),ObjectId("58a206a7c89278b62035592e"),ObjectId("58a206afc89278b62035592f")]}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.autores.find().pretty()
{
        "_id" : ObjectId("58a20673c89278b62035592b"),
        "username" : "JaviSant",
```

```
        "edad" : 21,
        "articulos" : [
                ObjectId("58a206a1c89278b62035592d"),
                ObjectId("58a206a7c89278b62035592e"),
                ObjectId("58a206afc89278b62035592f")
        ]
}
{
        "_id" : ObjectId("58a20673c89278b62035592c"),
        "username" : "pcampos",
        "edad" : 27
}
```