Tema 2 – Ejercicio MongoDB

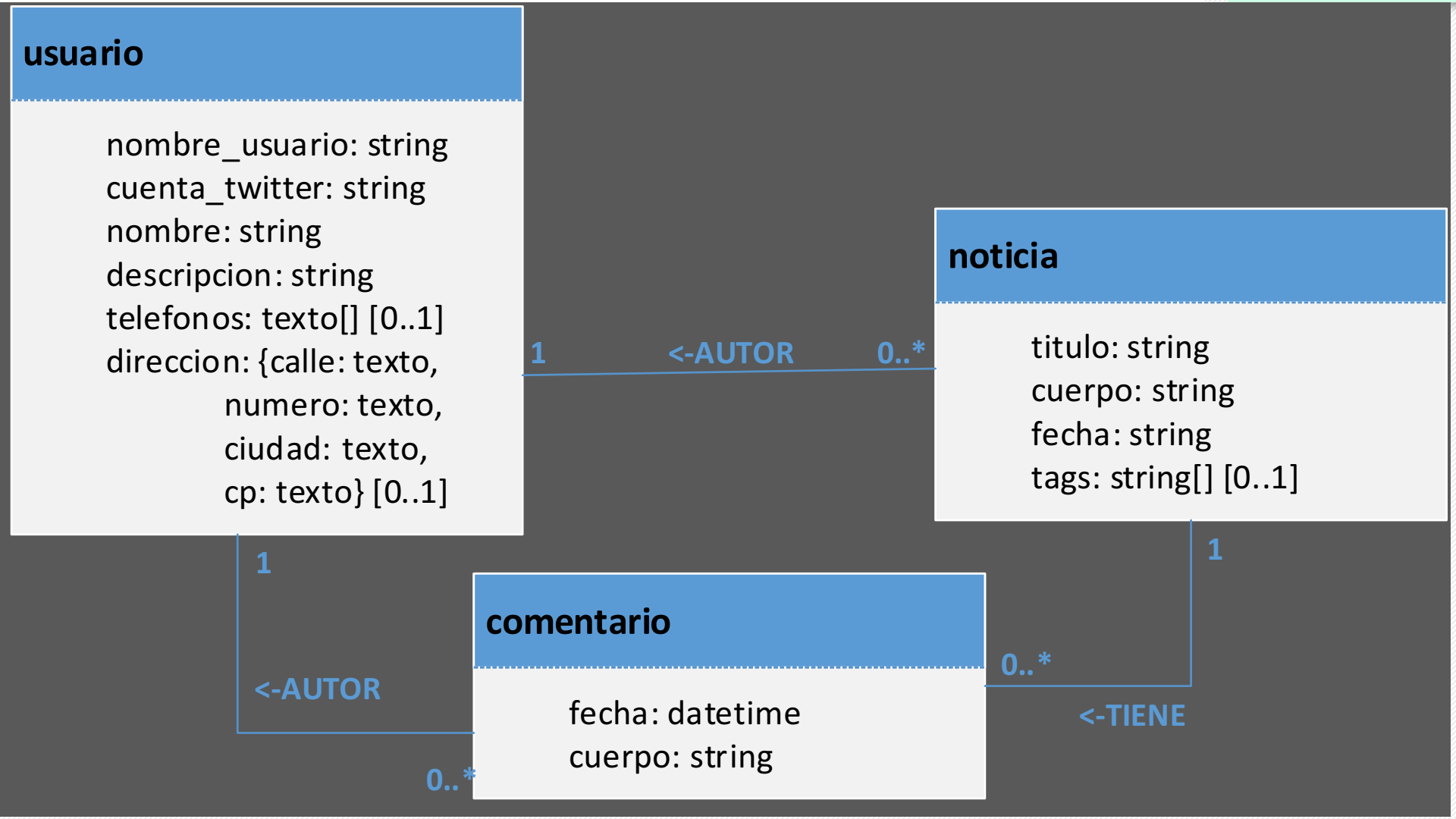
Marta Zorrilla – Diego García Saiz

Enero 2017

Enunciado

- Se precisa diseñar un blog de noticias donde los usuarios registrados pueda publicar sus comentarios:
  - Cada autor tiene un nombre, un nombre de usuario, una cuenta de Twitter y una descripción. Además, de forma opcional, los usuarios pueden proporcionar como datos su dirección postal (calle, número, puerta, C.P., ciudad) o sus teléfonos de contacto (pueden tener varios).
  - Las noticias tienen un título, un cuerpo y una fecha de publicación. Son publicadas por un autor y pueden contener o no, una lista de tags.
  - Las noticias reciben comentarios, quedando registrado la persona que lo escribió, el comentario escrito y el momento en el que lo hizo.

Diseño UML


usuario
nombre_usuario: string
cuenta_twitter: string
nombre: string
descripcion: string
telefonos: texto[] [0..1]
direccion: {calle: texto,
numero: texto,
ciudad: texto,
cp: texto} [0..1]
1
<-AUTOR
0..*
noticia
titulo: string
cuerpo: string
fecha: string
tags: string[] [0..1]
1
<-AUTOR
0..*
comentario
fecha: datetime
cuerpo: string
0..*
<-TIENE
1

Consultas frecuentes sobre la BD

- Consultas de los datos del usuario por nombre de usuario y por cuenta de Twitter:
  - Agrupación por código postal (contar el número de usuarios de cada C.P).
  - Consultas por número de teléfono.
- Consultas de noticias publicadas por usuarios
  - 10 últimas noticias publicadas ordenadas por fecha (de más reciente a más antigua).
- Número de comentarios por noticia, por día o por usuario.

Primer paso: crear una colección

- Recordemos:

"no es necesario crear una colección de forma explícita"

  - Se pueden insertar documentos y crear índices sobre campos sobre colecciones que aún no tengan documento alguno.
  - En MongoDB, no hay esquemas.
    - En cada colección, los documentos pueden tener diferentes campos.
    - Control por parte del cliente: la aplicación que accede a la base de datos MongoDB es responsable de cómo evolucione el esquema.

Modelo físico de usuarios

- Cuestiones a tener en cuenta en su “diseño” (por ahora nos olvidamos de que las noticias y los comentarios existen):
  - Índices secundarios: sólo los estrictamente necesarios para aumentar el rendimiento de las consultas frecuentes.
  - Teléfonos de contacto:
    - ¿Varios campos en un sub-documento?.
    - ¿Array de teléfonos?.
  - Dirección postal con varios campos:
    - ¿Sub-documento?.

Modelo físico de usuarios

Opción 1: Varios teléfonos en varios campos y dirección considerada como campos independientes.

```
{
        nombre_usuario: 'Frank_blog',
        nombre: 'Frank',
        cuenta_twitter: 'Frank_USE',
        descripcion: 'blogger aficionado',
        telefono1: '73128989',
        telefono2: '840932834',
        calle: 'Av. de los Castros',
        numero: '2256',
        cp: '39005',
        ciudad: 'Santander'
}
```

Requiere crear índices que garanticen unicidad sobre "nombre_usuario" y "cuenta_twitter"

Requeriría crear un índice para cada uno de los campos "telefonoX"… ¿cuántos?.

Los datos de dirección podrían agruparse como una unidad lógica y considerarse un sub-documento.

Insertado de datos en usuarios

M.Zorrilla - D.García

- Insertemos algunos datos en la colección:

```
db.usuario.insert({
        nombre_usuario: 'Frank_blog',
        nombre: 'Frank',
        cuenta_twitter: 'Frank_USA',
        descripcion: 'blogger aficionado',
        telefono1: '73128989',
        telefono2: '111111111',
        calle: 'Av. de los Castros',
        numero: '2256',
        cp: '39005',
        ciudad: 'Santander'
})
```

```
db.usuario.insert({
        nombre_usuario: 'Peter_blog',
        nombre: 'Peter',
        cuenta_twitter: 'Pete',
        descripcion: 'blogger aficionado',
        telefono1: '808080',
        telefono2: '4323424',
        calle: 'Av. de los Castros',
        numero: '289s',
        cp: '39005',
        ciudad: 'Santander'
})
```

Índices sobre usuarios

- Creemos a continuación los índices:
  - Índice único para los campos nombre_usuario y cuenta_twitter:

```
db.usuario.createIndex({“nombre_usuario”: 1}, {unique: true})
db.usuario.createIndex({“cuenta_twitter”: 1}, {unique: true})
```

  - Índice para el campo cp (código postal). Lo hacemos *sparse* para que no se indicen por este campo aquellos documentos que no lo tengan definido:

```
db.usuario.createIndex({“cp”: 1}, {sparse: true})
```

  - Varios índices para los teléfonos (tantos como campos haya):

```
db.usuario.createIndex({“telefono1”: 1})
db.usuario.createIndex({“telefono2”: 1})
...
```

Consultas sobre usuarios

- Retornar al usuario con nombre de usuario “Frank_blog”.

```
db.usuario.find({nombre_usuario: “Frank_blog”})
```

- Retornar los usuarios llamados “Peter”.

```
db.usuario.find({nombre: “Peter”})
```

- Contar el número de usuarios con CP= “39005”.

```
db.usuario.find({cp: “39005”})
```

- Retornar a un usuario con número de teléfono: “111111111”.

```
db.usuario.find({$or: [{telefono1: “111111111”}, {telefono2: “111111111”},
… ]})
```

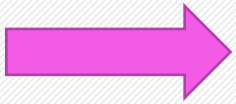
- Retornar nombre_usuario y cuenta_twitter, sin _id, de usuarios con CP igual o mayor que “39005”.

```
db.usuario.find({cp: {$gte: “39005”}},{nombre_usuario: 1,
cuenta_twitter: 1, _id:0})
```

Modelo físico de usuarios

- **Opción 2**: teléfonos en un *array*, subdocumento para dirección.

```
{
        nombre_usuario: 'Frank_blog',
        nombre: 'Frank',
        cuenta_twitter: 'Frank_USE',
        descripcion: 'blogger aficionado',
        telefonos: ['73128989','43278944']
        direccion:{
                calle: 'Av. de los Castros',
                numero: '2256',
                cp: '39005',
                ciudad: 'Santander'
                }
}
```



Índice sobre *Array*: si este crece mucho, puede no ser efectivo.

Conceptualmente mejora el diseño para almacenar la dirección. El índice se puede seguir creando sobre CP. No hay diferencia en el rendimiento con respecto a crear el índice sobre un campo fuera del subdocumento.

Índices sobre usuarios

- Creemos a continuación los índices:
  - Como paso previo, borraremos los datos introducidos con anterioridad:

```
db.usuario.remove()
```

  - Índice único para los campos nombre_usuario y cuenta_twitter:

```
db.usuario.createIndex({“nombre_usuario”: 1}, {unique: true})
db.usuario.createIndex({“cuenta_twitter”: 1}, {unique: true})
```

  - Índice para el campo cp (código postal). Lo hacemos *sparse* para que no se indicen por este campo aquellos documentos que no lo tengan definido. Dado que es un campo contenido en un subdocumento, utilizaremos la notación con punto:

```
db.usuario.createIndex({“dirección.cp”: 1}, {sparse: true})
```

  - Índice para los teléfonos. Dado que es un array, se crea el índice para cada valor contenido en el mismo:

```
db.usuario.createIndex({“telefonos”: 1})
```

Insertado de datos en usuarios

M.Zorrilla - D.García

- Insertemos algunos datos en la colección:

```
db.usuario.insert({
        nombre_usuario: 'Frank_blog',
        nombre: 'Frank',
        cuenta_twitter: 'Frank_USA',
        descripcion: 'blogger aficionado',
        telefono1: ['73128989', '111111111'],
        direccion: {
            calle: 'Av. de los Castros',
            numero: '2256',
            cp: '39005',
            ciudad: 'Santander'
        }
})
```

```
db.usuario.insert({
        nombre_usuario: 'Peter_blog',
        nombre: 'Peter',
        cuenta_twitter: 'Pete',
        descripcion: 'blogger aficionado',
        telefono1: ['808080', '4323424'],
        direccion: {
            calle: 'Av. de los Castros',
            numero: '289s',
            cp: '39005',
            ciudad: 'Santander'
        }
})
```

Consultas sobre usuarios

- Retornar al usuario con nombre de usuario “Frank_blog”.

```
db.usuario.find({nombre_usuario: “Frank_blog”})
```

- Retornar los usuarios llamados “Peter”.

```
db.usuario.find({nombre: “Peter”})
```

- Contar el número de usuarios con CP= “39005”.

```
db.usuario.find({“direccion.cp”: “39005”})
```

- Retornar a un usuario con número de teléfono: “111111111”.

```
db.usuario.find({telefono1: “111111111”})
```

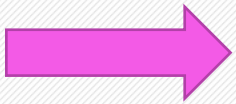
- Retornar nombre_usuario y cuenta_twitter, sin _id, de usuarios con CP igual o mayor que “39005”.

```
db.usuario.find({“direccion.cp”: {$gte:
    “39005”}},{nombre_usuario: 1, cuenta_twitter: 1, _id:0})
```

Modelo físico de usuarios

- **Opción 3**: que el _id sea un campo “nombre_usuario”:

```
{
        _id: 'Frank_blog',
        nombre: 'Frank',
        cuenta_twitter: 'Frank_USE',
        descripcion: 'blogger aficionado',
        telefonos: ['73128989',’43278944’]
        direccion:{
                calle: 'Av. de los Castros',
                numero: '2256',
                cp: '39005',
                ciudad: 'Santander‘
                }
}
```



El _id es humanamente comprensible.

Ya está indizado (por defecto).

Insertado de datos en usuarios

- Previo borrado de los datos anteriormente insertados y de los índices creados, añadimos algunos datos en la colección:

```
db.usuario.insert({
        _id: 'Frank_blog',
        nombre: 'Frank',
        cuenta_twitter: 'Frank_USA',
        descripcion: 'blogger aficionado',
        telefono1: ['73128989', '11111111'],
        direccion: {
            calle: 'Av. de los Castros',
            numero: '2256',
            cp: '39005',
            ciudad: 'Santander'
        }
})
```

```
db.usuario.insert({
        _id: 'Peter_blog',
        nombre: 'Peter',
        cuenta_twitter: 'Pete',
        descripcion: 'blogger aficionado',
        telefono1: ['808080', '4323424'],
        direccion: {
            calle: 'Av. de los Castros',
            numero: '289s',
            cp: '39005',
            ciudad: 'Santander'
        }
})
```

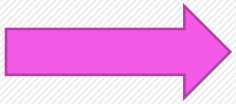
- ¿Qué pasaría si se tratase de insertar datos con el _id repetido?. ¿Y sin dar valor al campo _id?. Prueba ambas operaciones.

Modelo físico de las noticias

M.Zorrilla - D.García

- **Opción 1**: Datos de los usuarios en la noticia

```
{
        titulo: “Noticia de impacto”,
        cuerpo: “CUERPO DE LA NOTICIA”,
        fecha: ISODate(“2014-10-21”),
        tags: [“A”,”B”],
        {
        nombre_usuario: 'Frank_blog',
        nombre: 'Frank',
        cuenta_twitter: 'Frank_USE',
        descripcion: 'blogger aficionado',
        telefonos: ['73128989',’43278944’]
                direccion:{
                        calle: 'Av. de los Castros',
                        numero: '2256',
                        cp: '39005',
                        ciudad: 'Santander‘
                        }
        }
}
```



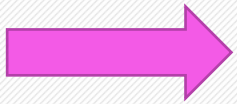
Los datos de los usuarios se repiten y puede llevar a inconsistencias.

Imposibilidad de crear índices eficientes de búsqueda sobre usuarios.

Modelo físico de las noticias

- **Opción 2**: Referencia a las noticias en un array dentro de la colección de los usuarios

```
{
        nombre_usuario: 'Frank_blog',
        nombre: 'Frank',
        cuenta_twitter: 'Frank_USE',
        descripcion: 'blogger aficionado',
        noticias: [<REF_notica>, <REF_noticia>, …]
        telefonos: ['73128989',’43278944’]
        direccion: {
                calle: 'Av. de los Castros',
                numero: '2256',
                cp: '39005',
                ciudad: 'Santander‘
                }
}
```

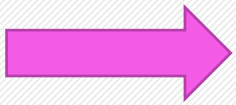


Si los usuarios publican muchas noticias, habrá un crecimiento excesivo del array: búsquedas poco eficientes, incluso con índices sobre el array, y escrituras pesadas al tener que actualizar los índices del array.

Modelo físico de las noticias

- **Opción 3**: Referencia a los usuarios en las noticias utilizando el nombre de usuario:

```
{
        titulo: “Noticia de impacto”,
        cuerpo: “CUERPO DE LA NOTICIA”,
        fecha: ISODate(“2014-10-21”),
        tags: [“A”,”B”],
        nombre_usuario: “Frank_blogger”
}
```



Si el _id del usuario es su nombre de usuario, es más sencillo de almacenar (no necesitas buscarlo en otras colecciones)

Documentos “menos pesados”.

No hay crecimiento de arrays “sin control”.

Se pueden crear índices sobre el campo “nombre_usuario” en la colección de noticias.

Insertando datos en noticias e indexando

- Insertar varias noticias (al menos, 10 documentos) con diferentes valores en los campos y que, al menos, 3 documentos tengan varios valores en el array de tags, y otros que no tengan valor en este campo.
- Queda aún por implementar los mecanismos para que se faciliten esta consulta:
  - 10 últimas noticias publicadas ordenadas por fecha (de más reciente a más antigua).
- Para ello, crearemos un índice descendente por el campo fecha y consultaremos mediante rangos o funciones de agregación sobre este campo

```
db.noticias.createIndex({“fecha”: -1})
```

Sharding sobre noticias y usuarios

M.Zorrilla - D.García

- Imaginemos que queremos hacer Sharding de la colección de noticias. ¿Qué tipo de *Shard Key* (*hashed* o *ranged*) crearías? ¿Sobre que campo?. ¿Por qué?.
  - Podríamos crear una *Ranged Shard Key* sobre el campo fecha, que ya ha sido previamente indizado, de la siguiente forma:

```
sh.shardCollection("db.noticias", {fecha: -1}
```

  - De esta forma, tendríamos que la colección noticas estaría repartida en varios *Shards*…
    - … ¿se garantiza de esta forma que siempre se devuelvan las 10 últimas noticias ordenadas por fecha?.

- ¿Qué tipo de *Shard Key* crearías si se desease hacer *Sharding* sobre la colección de usuarios?. ¿Sobre qué campos?. ¿Podrías crearla habiendo dos índices únicos ya previamente definidos?.

Practicar realizando los siguientes ejercicios

- Realizar las siguientes consultas:
  - Número de noticias publicadas por usuario.
  - 10 últimas noticias publicadas.
  - Noticias que no tienen el campo tag.
  - Noticias publicadas en un periodo de fechas. ¿Se podrían realizar consultas por año, mes y día sobre el campo de tipo *ISOdate*?. ¿Cómo?.
- Ampliar la base de datos para almacenar los datos de los comentarios que los usuarios pueden dejar sobre las noticias, según la especificación indicada en las transparencias 2, 3 y 4.