Upgrade hub

Modelado de datos

MongoDB Data Model

Quizás la gran duda a la hora de definir un esquema es:

¿Embeber subdocumentos, o hacer relaciones con otros documentos/colecciones?

MongoDB Data Model

Documentos Embebidos

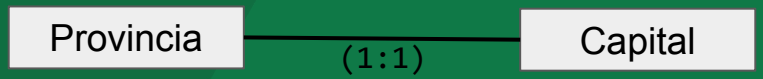
Se deberían usar cuando:

- No tienen suficiente entidad propia, es decir, carecen de sentido sin la información del documento padre.
- Son datos principalmente de lectura, no sufrirán muchos cambios.

MongoDB Data Model

Documentos Embebidos

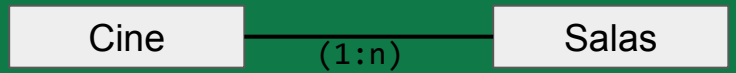
Relación **1:1**



En las relaciones 1:1 no tiene sentido que cada registro se guarde en un documento aparte.

En estos casos el documento principal será el registro con más entidad, y el embebido el que tenga menos importancia.

Relación **1:n** con mucha dependencia



Cuando se da una relación 1:n se embeberán los registros del lado "muchos" cuando no sean suficientemente importantes como para estar en un documento por sí solos.

MongoDB Data Model

```
{
  _id: ObjectId(“11”),
  nombreProv: “Badajoz”,
  población: 150000,
  gentilicio: “Pacense”,
  provincia: {
    nombre: “Badajoz”,
    capital: “Badajoz”,
    siglas: “BA”
  }
}
```

Embebemos la información básica de la Provincia que se necesitemos usar.

La cantidad de información a guardar dependerá de nuestra aplicación y el tipo de consultas a hacer.

MongoDB Data Model

```
{
  _id: ObjectId(“123”),
  empresa: “Kinépolis”,
  ciudad: “Pozuelo de Alarcón”,
  nombre: “Kinépolis Ciudad de la Imagen”,
  salas: [
    {
      nombre: “Sala 1”,
      capacidad: 204,
      pantalla: “6x14m”
    },
    {
      nombre: “Sala 25”,
      capacidad: 1016,
      pantalla: “10x25m”
    }
  ]
}
```

La información de las salas debería ser la mínima necesaria que nos sea útil.

Si se mete mucha información corremos el riesgo de alcanzar el tamaño máximo del documento (16MB).

En caso de necesitar tener mucha información habría que considerar usar referencias.

MongoDB Data Model

MongoDB, al igual que las bases de datos documentales en general, es muy flexible a la hora de modelar la estructura de datos. Sin embargo no deja de ser una tarea compleja encontrar la estructura idónea para nuestro proyecto.

Como cualquier base de datos, hay que tener en cuenta dos factores:

- **LECTURAS**. Hay que optimizar las operaciones de lecturas más frecuentes, no sólo con índices, también organizando los datos para que sean fácilmente accesibles.
- **ESCRITURAS.** Intentar agilizar las escrituras más frecuentes para que sean lo menos bloqueante posible. Tener siempre en mente el mantenimiento de los índices.

MongoDB Data Model

Documentos Relacionados

Se deberían usar cuando:

- Cuando el mismo dato esté repartido por varios documentos (habría varios documentos embebidos iguales en varios documentos -> problemas en actualizaciones.
- Para representar relaciones más complejas.
- Cuando el subdocumento es grande (existe limitación en el tamaño de un documento a 16MB).

MongoDB Data Model

Documentos Relacionados

Referencias Manuales: cuando sólo se trabaja con otra colección, o se sabe a ciencia cierta hacia dónde va la referencia.

Se guardará el campo _id de un documento como referencia en otro documento.

```
{id_movie: ObjectId(“345”)}
```

Suponemos que la colección destino será **Movies** y buscamos en ella el identificador “345”

MongoDB Data Model

Documentos Relacionados

DBRef: cuando hay que resolver la referencia dinámicamente. Es un documento embebido compuesto por:

$ref: colección destino
$id: identificador del documento destino
$db: (opcional) base de datos destino

Se crea:

```
{
  student: {
    $ref: “students”,
    $id: “9999”,
    $db: “college”
  }
}
```

Se muestra “la llamada”:

```
{
  student: DBRef(“students”,“999”)
}
```

MongoDB Data Model

Documentos Relacionados - EJEMPLO

```
{
        "_id": ObjectId("514d920b44ae16d201a3ff51),
        "nombre":"Capucchino",
        "precio":20,
        "marca":"Nescafé",
        "comentarios":
        [
                DBRef("Comentario", ObjectId("514d91a644ae16d201a3ff50"))
        ]
}
```

Documento de PRODUCTO con referencia a un documento de COMENTARIO

MongoDB Data Model

```
db.actors.insertOne({name: "qwerty usa",
pais: "USA", _id: ObjectId("2233")})

db.movies.insert(
  {actor:
    {
      "$ref": "actors",
      "$id": ObjectId("2233"),
      "$db": "upflix"
    }
  }
)

db.movies.find(
  {
    "actor.$id": ObjectId("2233")
  }
)
```

El DBRef se crea así, después se muestra como:

DBRef("actors", ObjectId("2233"), "upflix")

El orden es importante. Siempre tiene que ser ($ref, $id, $db). $db es opcional, por defecto es la misma base de datos.

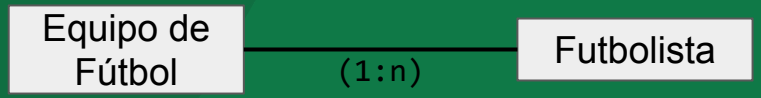
Así se realiza una búsqueda sobre un campo del DBRef.

Se están buscando todas las películas en donde aparezca el actor con identificador "2233"

MongoDB Data Model

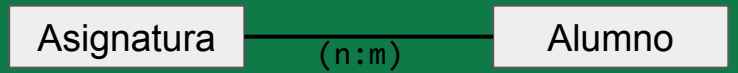
Documentos Relacionados

Relación **1:n**



Cuando el campo con múltiple valores pueda tener muchos tipos de consultas y actualizaciones, es mejor separarlo en un documento aparte, y referenciarlo donde se necesite.

Relación **n:n**



En las relaciones “muchos a muchos” se suelen usar siempre referencias, ya que ambas entidades seguramente serán objeto de todo tipo de consultas.

MongoDB Data Model

Colección Equipos:

```
{
  _id: ObjectId(“11”),
  name: “Extremadura UD”,
  league: “La Liga 123”,
  stadium: “Francisco de la Hera”,
  players:[
    DBRef(“players”,“22”),
    DBRef(“players”,“25”),
    DBRef(“players”,“26”)
  ]
}
```

Listado de referencias a los futbolistas.

Con este esquema, aquí sólo habría que añadir o quitar referencias, todos los datos del futbolista estarían en su colección particular.

MongoDB Data Model

Se pueden hacer referencias cruzadas si fuera necesario

colección Asignaturas / Subjects

```
{
  _id: ObjectId(“99”),
  name: “Bases de Datos”,
  centre: “Escuela Politécnica”,
  career: “Grado en Ing del Software”,
  course: 2,
  students:[
    DBRef(“students”,“622”),
    DBRef(“students”,“650”),
    DBRef(“students”,“712”),
    DBRef(“students”,“720”)
  ]
}
```

colección Alumnos / Students

```
{
  _id: ObjectId(“622”),
  name: “Juan López”,
  birthday: ISODate(“30/02/1998”),
  career: “Grado en Ing del Software”,
  subjects:[
    DBRef(“subjects”,“78”),
    DBRef(“subjects”,“98”),
    DBRef(“subjects”,“99”),
    DBRef(“subjects”,“100”),
    DBRef(“subjects”,“102”)
  ]
}
```

MongoDB Data Model

Replicación de datos

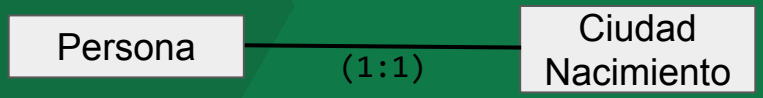
- En general no es una práctica recomendable
- Se trata de duplicar datos a lo largo de la base de datos
- Esta técnica se usa para optimizar las búsquedas: el objetivo ideal es que un documento tenga toda la información necesaria
- Sólo hacerlo en datos con pocas inserciones y muy raras actualizaciones

No es siempre tan sencillo, hay que pensar qué tipo de operaciones se realizarán, y optimizarlas.

MongoDB Data Model

¿Cuándo y cómo replicamos los datos?

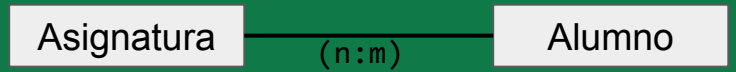
Son datos con poca entidad por sí misma, y van a tener pocas o ninguna actualización.



Ejemplo: Cada persona tiene un subdocumento con su ciudad de nacimiento con los campos: **Cod_Ciudad, Nombre_Ciudad, Provincia.**

Estos datos estarán duplicados (en una ciudad nacen muchas personas), pero normalmente esos datos no se modificarán.

No es necesario replicar todos los datos, se podría replicar sólo los necesarios, y referenciar al documento original para obtener el resto.



Ejemplo: Cada alumno tendrá un listado de asignaturas con datos básicos: **Nombre_Asign, Calificación, Curso, Referencia_Asignatura.**

Para sacar el resto de datos de la asignatura cuando sea necesario, se tiene que resolver la referencia.

MongoDB Data Model

- La mezcla de Documentos embebidos y Referencias no es exclusiva en datos replicados
- Se puede usar una estructura normal de datos.

Si quisiéramos saber más de alguno de los jugadores, deberíamos buscar segçun su _id devuelto en la tabla jugadores

colección de equipos

```
{
  _id: ObjectId(“1234”),
  “name”: “C.D. Badajoz”,
  “league”: “Segunda Divisón B”,
  “stadium”: “Nuevo Vivero”,
  “players”: [
    {
        “player_id”: ObjectId(“11”),
        “shortName”: “Guzmán”,
        “position”: “Extremo derecho”
    },
    {
        “player_id”: ObjectId(“22”),
        “shortName”: “César Morgado”,
        “position”: “Defensa Central”
    }
    ...
  ]
}
```

MongoDB Data Model

- La mezcla de Documentos embebidos y Referencias no es exclusiva en datos replicados
- Se puede usar una estructura normal de datos.

colección de jugadores/player

```
{
  _id: ObjectId(“11”),
  “shortName”: “Guzmán”,
  “fullName”: “Guzmán Casaseca Lozano”,
  “position”: “Extremo Derecho”,
  “birthDate”: ISODATE(“1993-01-26”),
  “statistics”: {
    “matches”: 13,
    “goals”: 0,
    “assists”: 1
  }
},
{
  _id: ObjectId(“22”),
  “shortName”: “César Morgado”,
  “fullName”: “César Morgado Ortega”,
  “position”: “Defensa Central”,
  “birthDate”: ISODATE(“1984-12-26”),
  ...
},
```

MongoDB Data Model

Por lo general, para embeber o referenciar puedes tener en cuenta estas pautas para tomar la decisión correcta:

Embeber es mejor para:

- Pequeños subdocumentos.
- Datos que no cambian regularmente.
- Cuando la consistencia final es aceptable.
- Documentos que crecen en pequeñas cantidades.
- Datos que necesitarás a menudo para realizar una segunda consulta para obtener lecturas rápidas.

MongoDB Data Model

Referenciar es mejor para:

- Subdocumentos grandes.
- Datos volátiles.
- Cuando es necesaria la consistencia inmediata.
- Documentos que crecen una gran cantidad.
- Datos que a menudo excluye de los resultados.
- Escrituras rápidas.

MongoDB Data Model

Ejercicio 1: Diseño de un Blog - Funcionalidades

(a resolver entre todos → brainstorming + justificación + debate) GUIADO

- Un usuario se puede darse de alta, modificar sus datos, y darse de baja.
- Un usuario puede escribir tantos Posts como desee.
- Cada Post tiene información básica: título, texto e imágenes.
- Los Posts se categorizan con tags.
- Existirá una sección con los Posts más leídos.
- Existirá una sección con los tags más populares (que tengan los Posts más leídos).
- Existirá una sección con los Usuarios más valorados (con mejores Posts).
- Un usuario puede suscribirse a otro Usuario para recibir notificaciones de sus nuevos Posts.
- Un usuario puede suscribirse a un tag.
- Un usuario puede buscar Posts por su título.
- La *home* del usuario será una combinación personalizada de Posts según sus suscripciones a tags y/o usuarios.
- Un usuario puede comentar un Post.
- Existirá una sección con los Posts más comentados.