Engrandeciendo Grails con MongoDB

Enrique Medina Montenegro
@emedinam

Agenda

- Quién soy
- Por qué estoy aquí
- NoSQL: qué está pasando ahí fuera
- MongoDB: por dónde empiezo
- Plugin MongoDB: qué puedo y qué no puedo hacer
- Conclusiones, ruegos y preguntas

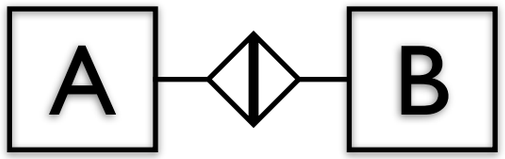
Quién soy

- Ingeniero en Informática (15 años de experiencia, no siempre en arquitecturas Java)
- Desarrollador con Groovy/Grails (3+ años)
- Creador del Observatorio de Grails y de cuestamenos.com
- Firme creyente y defensor del open-source
- Padre, liberal, autodidacta, y apasionado de la tecnología (¿geek?)

Por qué estoy aquí

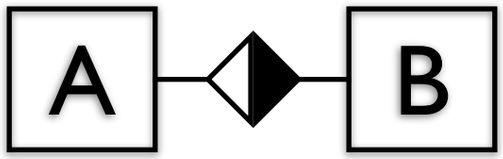
- Modelo relacional: es lo que todos conocemos (mapeador OR - Hibernate)
- Mapeo tradicional de asociaciones entre clases del dominio:

1:1



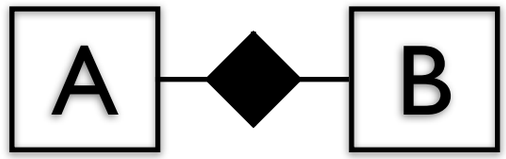
hasOne
(belongsTo)

1:M



hasMany &
belongsTo
(bidireccional)

M:N



hasMany &
belongsTo
(dueño)

CARGA PEREZOSA (LAZY LOADING)

- Modelo relacional **normalizado**: todo son 'joins', no duplicamos datos

Por qué estoy aquí

- ¿Dónde está el problema?
  - 1:M suele consistir en una colección de tipo Set (org.hibernate.collection.PersistentSet) que asegura la unicidad de sus elementos
  - Añadir un elemento a la colección implica que Hibernate debe cargar **TODA** la colección para asegurarse la unicidad
  - Aunque cambiemos a un tipo List, Hibernate debe cargar también **TODA** la colección para asegurar el orden, aún cuando normalmente insertamos al final de la misma
  - Y definir la *carga 'perezosa'* sólo retrasa el problema
- Entonces, ¿qué pasaría si trabajáramos con una colección de miles (o cientos de miles) de elementos? Da miedo pensarlo, ¿verdad?

Por qué estoy aquí

- Otros problemas que tenemos que conocer:
  - *Uso específico de 'eager' para ahorrar consultas*: mucho cuidado, puede ser peor el remedio que la enfermedad (muchos 'joins' son más lentos que las consultas por separado)
  - *Caché de consulta (query cache)*: inútil si estás trabajando con datos afectados por el tiempo (el parámetro 'tiempo' siempre será distinto)
  - *Caché de segundo nivel (2nd level cache)*: no está activa por defecto; hay que saber configurarla bien o comenzarán a aparecer errores de sincronización (stale data)
- Y la culpa de todo la tiene el DISEÑO relacional (y cuanto más normalizado, mucho peor...), así que **¡¡¡DESAPRENDE!!!**

NoSQL: qué está pasando

- **NoSQL**: la palabra en boca de todo el mundo. ¿Moda pasajera o viene a quedarse?
- **NoSQL** = no relacional, desestructurado, escalable, distribuido, eficiente
  - *Orientada a documentos*: CouchDB, MongoDB, SimpleDB
  - *Grafos*: Neo4j, OrientDB, FlockDB
  - *Clave/Valor*: Redis, Cassandra, Coherence, BigTable
  - *Objetual*: db4o, ObjectDBm ObjectStore
- El reto es saber **CÓMO** y **CUÁNDO** utilizar **QUÉ** tipo de NoSQL

NoSQL: qué está pasando

- Veamos si podemos ayudar un poco en este caos:
  - *Orientada a documentos*: Son las más 'parecidas' al modelo relacional, y por ello las más usadas (pero pueden ser un arma de doble filo)
  - *Grafos*: Verticalizan su ámbito a información gestionada mediante grafos, como redes sociales
  - *Clave/Valor*: Seguramente las más rápidas. Es como si en vez de SQL sólo pudieras usar mapas con 'get/put'
  - *Objetual*: Se esperaba mucho de ellas, pero no terminaron de cuajar

CONSEJO: ¿cuál es el diseño de tu modelo y la estrategia de acceso a los datos?

MongoDB: por dónde empiezo

- MongoDB (from "humongous") is a scalable, high-performance, open source, document-oriented database - 10gen, Inc (creadores de MongoDB)

Orientada a documentos (estilo JSON con esquemas dinámicos)

Soporte completo para indexación (optimización de búsquedas)

Replicación y Alta Disponibilidad (tolerancia a fallos)

Auto-Sharding (escalabilidad horizontal eficiente)

Modificadores atómicos (actualizaciones rápidas in-situ)

Lenguaje enriquecido de consultas (métodos propios)

Operaciones Map/Reduce (agregación y proceso de datos)

GridFS (almacenamiento avanzado de ficheros)

Pero, ¡¡¡no tiene TRANSACCIONALIDAD!!! ¿Uhhh?

MongoDB: por dónde empiezo

- Llamemos a las cosas por su nombre:

MySQL term	Mongo term
database	database
table	collection
index	index
row	BSON document
column	BSON field
join	embedding and linking
primary key	_id field

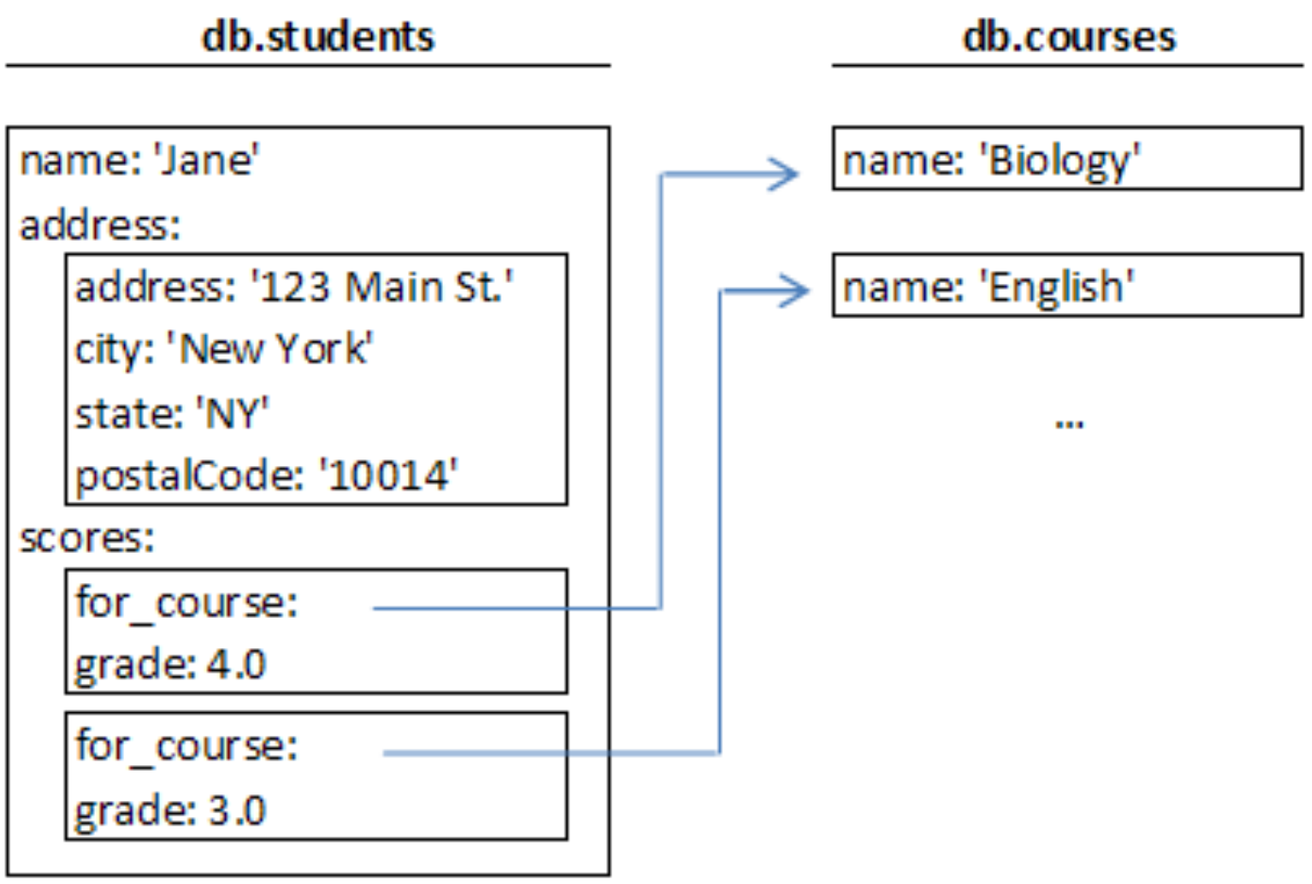
- ATENCIÓN: 'join' vs 'embedding and linking'

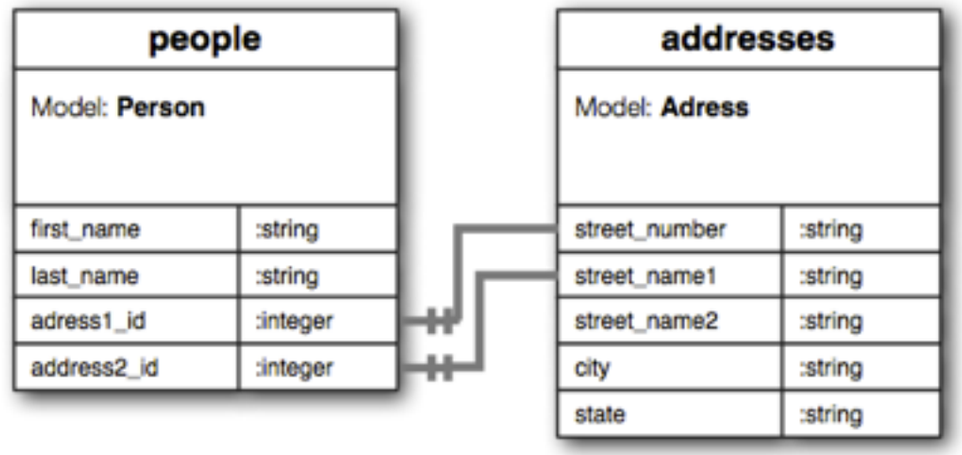
MongoDB: por dónde empiezo

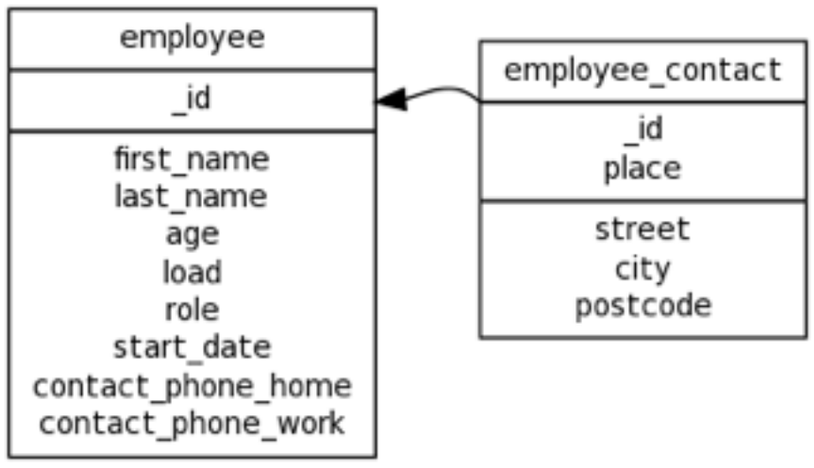
- Diseño del modelo: Embedding vs Linking
  - *Embedding* --> Incrustar un documento dentro de otro. Recomendado en asociaciones por identificación, maestro/detalle, padre/hijo, para evitar 'joins', etc.
  - *Linking* --> Es el homólogo al 'join' relacional, pero mucho más optimizado (DBRef).

MongoDB: por dónde empiezo

- Ejemplos de referencias (linking):







MongoDB: por dónde empiezo

- Ejemplos de incrustación (embedding):

db.students

name: ‘Jane’
address:
- address: ‘123 Main St.’
  city: ‘New York’
  state: ‘NY’
  postalCode: ‘10014’

scores:
- for_course:
  name: ‘Biology’
  grade: 4.0
- for_course:
  name: ‘English’
  grade: 3.0

db.people

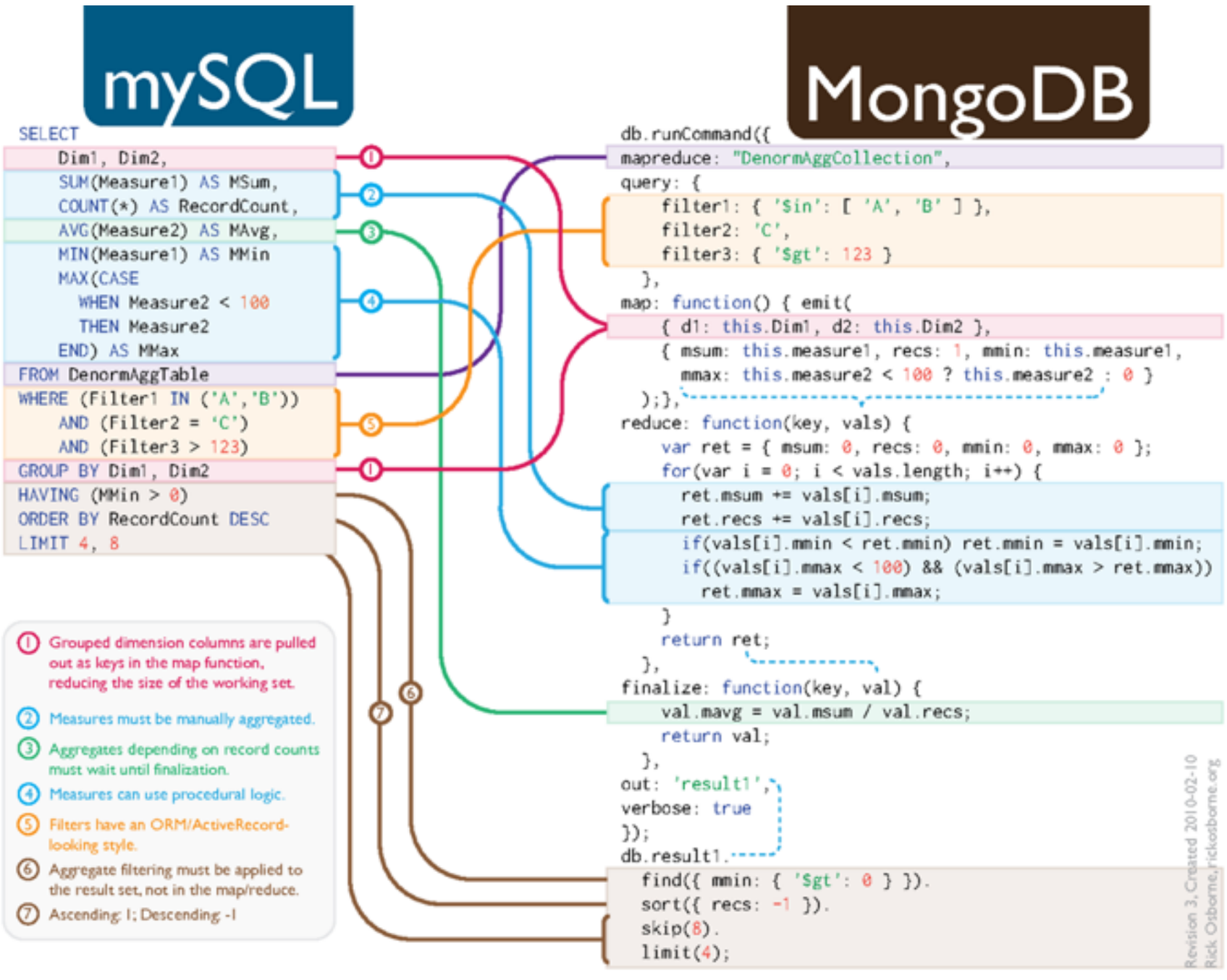
first_name: ‘Enrique’
last_name: ‘Medina’
addresses:
- address: ‘Calle 1’
  city: ‘Elche’
  state: ‘Alicante’
  postalCode: ‘03204’
- address: ‘Calle 2’
  city: ‘Madrid’
  state: ‘Madrid’
  postalCode: ‘28080’

db.employee

first_name: ‘Enrique’
last_name: ‘Medina’
age: 38
load: 2.0
role: ‘Manager’
start_date: 04/11/2010
contact:
- place: ‘Main building’
  address: ‘123 Main St.’
  city: ‘New York’
  postalCode: ‘10014’
  phone1: ‘900123456’
  phone2: ‘900789873’

MongoDB: por dónde empiezo

- Map/Reduce, la *bestia negra* de NoSQL:



Plugins MongoDB

- *MongoDB GORM v.1.0.0.M7* --> http://www.grails.org/plugin/mongodb
- *Grails MongoDB v.0.5.4* --> http://www.grails.org/plugin/gorm-mongodb
- *Alternative MongoDB GORM based on the Morphia library v.0.7.5* --> http://www.grails.org/plugin/mongodb-morphia
- *MongoDB Tools v.0.1.2* --> http://www.grails.org/plugin/mongodb-tools
- *Mongodb Multitenant v.0.2.2.6-BETA* --> http://www.grails.org/plugin/mongodb-multitenant
- *Spring MongoDB Datastore support for ZK v.1.1-M1* --> http://grails.org/plugin/zk-mongodb

Plugin MongoDB GORM

- Grails fue concebido con espíritu relacional (GORM con Hibernate)
- Pero, ¿y ahora que queremos usar MongoDB?

GORM	MongoDB
hasOne	Embed / *Reference*
hasMany	Embed / *Reference*
belongsTo	Embed / *Reference*
embedded	*Embed*

CONSEJO: No dejes que Grails condicione tu diseño MongoDB (NoSQL)

Plugin MongoDB GORM

- Diseccionando el plugin nos encontramos:
  - Integración con *GORM* (Spring Data - Document Stores)
  - Uso del *driver oficial Java* para MongoDB
  - Uso de la librería *GMongo* (API al estilo Groovy)
  - *Mapeador* de BSON a tipos Groovy/Java
  - Soporte de *métodos dinámicos*, *criterios* y *consultas nominadas* (pero no HQL, M:N, proyecciones en criterios, y más...)
  - *TRANSACCIONALIDAD* --> ¿Magia?

Plugin MongoDB GORM

- Mapeo de clases del dominio a colecciones Mongo:

Reference

```
class Person {
    String firstName
    String lastName

    static hasMany = [pets:Pet]
}
```

o bien

```
class Person {
    String firstName
    String lastName
}

class Pet {
    String name
    String breed

    Person person
}
```

Embedded

```
class Person {
    String firstName
    String lastName
    Address address
    List<Address> otherAddresses

    static embedded = ['address', 'otherAddresses']
}
```

Plugin MongoDB GORM

- Mapeo de clases del dominio a colecciones Mongo (cont.):

Identidad

```
import org.bson.types.ObjectId

class Person {
    ObjectId id

}
```

WriteConcern

```
import com.mongodb.WriteConcern

class Person {

    String name

    static mapping = {

        writeConcern WriteConcern.FSYNC_SAFE

    }

}
```

Índices

```
class Person {
    String name
    static mapping = {
        name index:true, indexAttributes: [unique:true, dropDups:true, sparse:true]
    }
}
```

Plugin MongoDB GORM

- Mapeo de clases del dominio a colecciones Mongo (cont.):

Geospacial

```
class Hotel {
    String name
    List location
    static mapping = {
        location geoIndex:true,
         indexAttributes:[min:-500, max:500]
    }

}

new Hotel(name:"Hilton", location:
     [lat: 40.739037, long: 73.992964]).save()

def h = Hotel.findByLocationNear([50, 60])
assert h.name == 'Hilton'


def box = [[40.73083, -73.99756], [40.741404,  -73.988135]]
def h = Hotel.findByLocationWithinBox(box)


def center = [50, 50]
def radius = 10
def h = Hotel.findByLocationWithinCircle([center, radius])
```

Atributos dinámicos

```
class Plant {
    boolean goesInPatch
    String name
}

def p = new Plant(name:"Pineapple")
p['color'] = 'Yellow'
p['hasLeaves'] = true
p.save()
p = Plant.findByName("Pineapple")

println p['color']

println p['hasLeaves']
```

Plugin MongoDB GORM

- Demo 1 - 1:1 / M:1 sin 'hasOne' ni 'belongsTo'

  Paso 1. Crear un maestro y un detalle

```
Thu Oct 27 17:46:59 [initandlisten] connection accepted from 192.168.0.102:53298 #54
Thu Oct 27 17:47:00 [conn54] remove prueba1.maestro  query: {} 20ms
Thu Oct 27 17:47:00 [conn54] query: prueba1.$cmd{ getlasterror: 1, w: 1, wtimeout: 0, fsync: true }
Thu Oct 27 17:47:00 [conn54] run command prueba1.$cmd { getlasterror: 1, w: 1, wtimeout: 0, fsync: true }
Thu Oct 27 17:47:00 [conn54] query prueba1.$cmd ntoreturn:1 command: { getlasterror: 1, w: 1, wtimeout: 0, fsync: true } reslen:106 33ms
Thu Oct 27 17:47:00 [conn54] remove prueba1.detalle  query: {} 0ms
Thu Oct 27 17:47:00 [conn54] query: prueba1.$cmd{ getlasterror: 1, w: 1, wtimeout: 0, fsync: true }
Thu Oct 27 17:47:00 [conn54] run command prueba1.$cmd { getlasterror: 1, w: 1, wtimeout: 0, fsync: true }
Thu Oct 27 17:47:00 [conn54] query prueba1.$cmd ntoreturn:1 command: { getlasterror: 1, w: 1, wtimeout: 0, fsync: true } reslen:106 100ms
Thu Oct 27 17:47:00 [conn54] insert prueba1.maestro 0ms
Thu Oct 27 17:47:00 [conn54] insert prueba1.detalle 0ms


> db.maestro.find()
{ "_id" : ObjectId("4ea97cf4a0ee87eee9573adf"), "nombre" : "Maestro1", "version" : 0 }

> db.detalle.find()
{ "_id" : ObjectId("4ea97cf4a0ee87eee9573ae0"), "maestro" : { "$ref" : "detalle", "$id" : ObjectId("4ea97cf4a0ee87eee9573adf") }, "nombre" :
"Detalle1", "version" : 0 }
```

Plugin MongoDB GORM

- Demo 1 - 1:1 / M:1 sin ‘hasOne’ ni ‘belongsTo’

  Paso 2. Buscar un maestro e imprimir su detalle

```
Thu Oct 27 17:50:47 [initandlisten] connection accepted from 192.168.0.102:53403 #55
Thu Oct 27 17:50:47 [conn55] query: prueba1.maestro{ nombre: "Maestro1" }
Thu Oct 27 17:50:47 [conn55]    used cursor: 0x100f5c520
Thu Oct 27 17:50:47 [conn55] query prueba1.maestro ntoreturn:1 reslen:92 nreturned:1 0ms
Thu Oct 27 17:50:47 [conn55] query: prueba1.detalle{ maestro.$id: ObjectId('4ea97cf4a0ee87eee9573adf') }
Thu Oct 27 17:50:47 [conn55]    used cursor: 0x100f53d80
Thu Oct 27 17:50:47 [conn55] query prueba1.detalle ntoreturn:1 reslen:141 nreturned:1 0ms
```

Plugin MongoDB GORM

- Demo 2 - 1:1 bidireccional con belongsTo

  Paso 1. Crear un maestro y un detalle

```
Thu Oct 27 18:14:10 [initandlisten] connection accepted from 192.168.0.102:53795 #63
Thu Oct 27 18:14:10 [conn63] remove prueba2.maestro  query: {} 0ms
Thu Oct 27 18:14:10 [conn63] query: prueba2.$cmd{ getlasterror: 1, w: 1, wtimeout: 0, fsync: true }
Thu Oct 27 18:14:10 [conn63] run command prueba2.$cmd { getlasterror: 1, w: 1, wtimeout: 0, fsync: true }
Thu Oct 27 18:14:10 [conn63] query prueba2.$cmd ntoreturn:1 command: { getlasterror: 1, w: 1, wtimeout: 0, fsync: true } reslen:106 0ms
Thu Oct 27 18:14:10 [conn63] remove prueba2.detalle  query: {} 0ms
Thu Oct 27 18:14:10 [conn63] query: prueba2.$cmd{ getlasterror: 1, w: 1, wtimeout: 0, fsync: true }
Thu Oct 27 18:14:10 [conn63] run command prueba2.$cmd { getlasterror: 1, w: 1, wtimeout: 0, fsync: true }
Thu Oct 27 18:14:10 [conn63] query prueba2.$cmd ntoreturn:1 command: { getlasterror: 1, w: 1, wtimeout: 0, fsync: true } reslen:106 95ms
Thu Oct 27 18:14:10 [conn63] insert prueba2.detalle 0ms
Thu Oct 27 18:14:10 [conn63] insert prueba2.maestro 0ms


> db.maestro.find()
{ "_id" : ObjectId("4eb41fc8744edce76bfb9339"),
        "detalle" : { "$ref" : "maestro", "$id" : ObjectId("4eb41fc8744edce76bfb933a") }, "nombre" : "Maestro2", "version" : 0 }

> db.detalle.find()
{ "_id" : ObjectId("4eb41fc8744edce76bfb933a"),
        "maestro" : { "$ref" : "detalle", "$id" : ObjectId("4eb41fc8744edce76bfb9339") }, "nombre" : "Detalle2", "version" : 0 }
```

Plugin MongoDB GORM

- Demo 2 - 1:1 bidireccional con belongsTo

  Paso 2. Buscar un maestro e imprimir su detalle

```
Thu Oct 27 17:50:47 [initandlisten] connection accepted from 192.168.0.102:53403 #55
Thu Oct 27 17:50:47 [conn55] query: prueba2.maestro{ nombre: "Maestro2" }
Thu Oct 27 17:50:47 [conn55]    used cursor: 0x101dab800
Thu Oct 27 17:50:47 [conn55] query prueba2.maestro ntoreturn:1 reslen:141 nreturned:1 0ms
Thu Oct 27 17:50:47 [conn55] query: prueba2.detalle{ _id: ObjectId('4eb420c2744e2fe7f81c87be') }
Thu Oct 27 17:50:47 [conn55] query prueba2.detalle ntoreturn:1 idhack  reslen:141 0ms
```

Plugin MongoDB GORM

- Demo 3 - 1:M con ‘hasMany’

  Paso 1. Crear un maestro y varios detalles

```
Thu Oct 27 18:37:23 [conn78] insert prueba3.detalle 0ms
Thu Oct 27 18:37:23 [conn78] insert prueba3.maestro 0ms
Thu Oct 27 18:37:23 [conn78] update prueba3.maestro  query: { _id: ObjectId('4ea988c3a0ee8ac0c9c3cab7') } update: { _id:
ObjectId('4ea988c3a0ee8ac0c9c3cab7'), nombre: "Maestro3", version: 0, detalles: [ { $ref: "maestro", $id: ObjectId('4ea988c3a0ee8ac0c9c3cab8') },
{ $ref: "maestro", $id: ObjectId('4ea988c3a0ee8ac0c9c3cab9') }, { $ref: "maestro", $id: ObjectId('4ea988c3a0ee8ac0c9c3caba') } ] } byid  0ms

> db.maestro.find()
{ "_id" : ObjectId("4ea988c3a0ee8ac0c9c3cab7"), "nombre" : "Maestro3", "version" : 0, "detalles" : [
        {
                "$ref" : "maestro",“$id" : ObjectId("4ea988c3a0ee8ac0c9c3cab8")
        },
        {
                "$ref" : "maestro",“$id" : ObjectId("4ea988c3a0ee8ac0c9c3cab9")
        },
        {
                "$ref" : "maestro", "$id" : ObjectId("4ea988c3a0ee8ac0c9c3caba")
        }
] }
> db.detalle.find()
{ "_id" : ObjectId("4ea988c3a0ee8ac0c9c3cab8"), "nombre" : "Detalle3.2", "version" : 0 }
{ "_id" : ObjectId("4ea988c3a0ee8ac0c9c3cab9"), "nombre" : "Detalle3.1", "version" : 0 }
{ "_id" : ObjectId("4ea988c3a0ee8ac0c9c3caba"), "nombre" : "Detalle3.3", "version" : 0 }
```

Plugin MongoDB GORM

- Demo 3 - 1:M con 'hasMany'

  Paso 2. Buscar un maestro e imprimir sus detalles

```
Thu Oct 27 18:40:40 [conn78] query: prueba3.maestro{ nombre: "Maestro3" }
Thu Oct 27 18:40:40 [conn78]    used cursor: 0x101d33da0
Thu Oct 27 18:40:40 [conn78] query prueba3.maestro ntoreturn:1 reslen:236 nreturned:1 0ms
Thu Oct 27 18:40:40 [conn78] query: prueba3.detalle{ _id: { $in: [ ObjectId('4ea988c3a0ee8ac0c9c3cab8'),
ObjectId('4ea988c3a0ee8ac0c9c3cab9'), ObjectId('4ea988c3a0ee8ac0c9c3caba') ] } }
Thu Oct 27 18:40:40 [conn78]    used cursor: 0x101d34140
Thu Oct 27 18:40:40 [conn78] query prueba3.detalle reslen:210 nreturned:3 0ms
```

Plugin MongoDB GORM

- Demo 4 - 1:M con detalles embebidos

  Paso 1. Crear un maestro y varios detalles

Thu Oct 27 19:11:06 [conn94] insert prueba4.maestro 0ms

```
> db.maestro.find()
{ "_id" : ObjectId("4ea990aaa0ee155609d17314"), "detalles" : [
        {
                "nombre" : "Detalle4.1", "_embeddedClassName" : "prueba4.Detalle"
        },
        {
                "nombre" : "Detalle4.2", "_embeddedClassName" : "prueba4.Detalle"
        },
        {
                "nombre" : "Detalle4.3", "_embeddedClassName" : "prueba4.Detalle"
        }
], "nombre" : "Maestro4", "version" : 0 }

> db.detalle.find()
```

(No hay resultados porque 'detalle' no tiene entidad propia, sino que es un documento embebido en 'maestro')

Plugin MongoDB GORM

- Demo 4 - 1:M con detalles embebidos

  Paso 2. Buscar un maestro e imprimir sus detalles

```
Thu Oct 27 19:12:09 [conn95] query: prueba4.maestro{ nombre: "Maestro4" }
Thu Oct 27 19:12:09 [conn95]    used cursor: 0x100f4a9e0
Thu Oct 27 19:12:09 [conn95] query prueba4.maestro ntoreturn:1 reslen:320 nreturned:1 0ms
```

Plugin MongoDB GORM

- De nuevo la pregunta del millón, ¿*embedded* o *reference*?
  - *Depende* del dominio de tu aplicación
  - Para *1:M* opta en realidad por un *M:1*

<1>.get<M>() { <M>.findAllBy<1>(this) }

  - Usa *embedded* cuando la entidad 'embebida' carece de identidad propia fuera de su 'contenedor'
  - Pero si usas *embedded*, Grails tiene sus limitaciones :-(
  - Trees --> http://www.mongodb.org/display/DOCS/Trees+in+MongoDB

Plugin MongoDB GORM

- Limitación 1 - Indexación de propiedades embebidas

```
class Maestro implements Serializable {

    static mapWith = 'mongo'

    ObjectId id

    String nombre
    List<Detalle> detalles = []

    static embedded = ['detalles']

    static constraints = {
        nombre blank: false
    }

    static mapping = {
        nombre index: true
        'detalles.nombre' index: true

        writeConcern com.mongodb.WriteConcern.FSYNC_SAFE
    }

}
```

```
class BootStrap {

    def init = { servletContext ->
        Maestro.collection.ensureIndex("detalles.nombre")
    }

    def destroy = {
    }

}
```

```
> db.system.indexes.find()
{ "name" : "_id_", "ns" : "detalle", "key" : { "_id" : 1 }, "v" : 0 }
{ "name" : "nombre_1", "ns" : "detalle", "key" : { "nombre" : 1 }, "v" : 0 }
{ "name" : "_id_", "ns" : "maestro", "key" : { "_id" : 1 }, "v" : 0 }
{ "name" : "nombre_1", "ns" : "maestro", "key" : { "nombre" : 1 }, "v" : 0 }
{ "name" : "detalles.nombre_1", "ns" : "maestro", "key" : { "detalles.nombre" : 1 }, "v" : 0 }
```

Plugin MongoDB GORM

- Limitación 1 - Indexación de propiedades embebidas (cont.)

```
> db.maestro.find({"detalles.nombre": "Detalle4.2"})
{
    "_id" : ObjectId("4ea990dfa0ee155609d17315"),
    "detalles" : [
        {
            "nombre" : "Detalle4.1",
            "_embeddedClassName" : "Detalle"
        },
        {
            "nombre" : "Detalle4.2",
            "_embeddedClassName" : "Detalle"
        },
        {
            "nombre" : "Detalle4.3",
            "_embeddedClassName" : "Detalle"
        }
],
    "nombre" : "Maestro4",
    "version" : 0 }
```

```
> db.maestro.find({"detalles.nombre": "Detalle4.2"}).explain()
{
        "cursor" : "BtreeCursor detalles.nombre_1",
        "nscanned" : 1,
        "nscannedObjects" : 1,
        "n" : 1,
        "millis" : 24,
        "nYields" : 0,
        "nChunkSkips" : 0,
        "isMultiKey" : true,
        "indexOnly" : false,
        "indexBounds" : {
                "detalles.nombre" : [
                        [
                                "Detalle4.2",
                                "Detalle4.2"
                        ]
                ]
        }
}
```

Plugin MongoDB GORM

- Limitación 2 - Indexación de referencias (DBRef)

```
class Detalle implements Serializable {

    static mapWith = 'mongo'

    ObjectId id

    String nombre
    Maestro maestro

    static constraints = {
        nombre blank: false
        maestro nullable: false
    }

    static mapping = {
        nombre index: true
        maestro index: true

        writeConcern WriteConcern.FSYNC_SAFE
    }

}
```

```
> db.system.indexes.find()
{ "name" : "_id_", "ns" : "detalle", "key" : { "_id" : 1 }, "v" : 0 }
{ "name" : "nombre_1", "ns" : "detalle", "key" : { "nombre" : 1 }, "v" : 0 }
{ "name" : "_id_", "ns" : "maestro", "key" : { "_id" : 1 }, "v" : 0 }
{ "name" : "nombre_1", "ns" : "maestro", "key" : { "nombre" : 1 }, "v" : 0 }
{ "name" : "maestro_1", "ns" : "detalle", "key" : { "maestro" : 1 }, "v" : 0 }
```

Plugin MongoDB GORM

- Limitación 2 - Indexación de referencias (cont.)

```
def maestro = Maestro.findByNombre("Maestro1")
def detalle = Detalle.findByMaestro(maestro)
```

```
Wed Nov  2 12:58:44 [conn1] query: detalle{ maestro.$id: ObjectId('4ea97cf4a0ee87eee9573adf') }
Wed Nov  2 12:58:44 [conn1]     used cursor: 0x101e02ac0
Wed Nov  2 12:58:44 [conn1] query detalle reslen:141 nreturned:1 0ms
```

Plugin MongoDB GORM

- Limitación 2 - Indexación de referencias (cont.)

```
> db.detalle.find({"maestro.$id": "4ea97cf4a0ee87eee9573adf"})
{
    "_id" : ObjectId("4ea97cf4a0ee87eee9573ae0"),
    "maestro" : {
            "$ref" : "detalle",
            "$id" : ObjectId("4ea97cf4a0ee87eee9573adf")
    },
    "nombre" : "Detalle1", "version" : 0 }
```

```
> db.detalle.find({"maestro.$id": "4ea97cf4a0ee87eee9573adf"}).explain()
    {
            "cursor" : "BasicCursor",
            "nscanned" : 1,
            "nscannedObjects" : 1,
            "n" : 1,
            "millis" : 0,
            "nYields" : 0,
            "nChunkSkips" : 0,
            "isMultiKey" : false,
            "indexOnly" : false,
            "indexBounds" : {}
    }
```

Plugin MongoDB GORM

- Limitación 2 - Indexación de referencias (cont.)

```
> db.detalle.find({'maestro': new DBRef('detalle', ObjectId("4ea97cf4a0ee87eee9573adf"))})
{
    "_id" : ObjectId("4ea97cf4a0ee87eee9573ae0"),
    "maestro" : { "$ref" : "detalle", "$id" : ObjectId("4ea97cf4a0ee87eee9573adf") },
    "nombre" : "Detalle1", "version" : 0 }
```

```
> db.detalle.find({'maestro': new DBRef('detalle', ObjectId("4ea97cf4a0ee87eee9573adf"))}).explain()
    {
            "cursor" : "BtreeCursor maestro_1",
            "nscanned" : 1,
            "nscannedObjects" : 1,
            "n" : 1,
            "millis" : 0,
            "indexBounds" : {
                    "maestro" : [ [
                            {
                                    "$ref" : "detalle", "$id" : ObjectId("4ea97cf4a0ee87eee9573adf")
                            },
                            {
                                    "$ref" : "detalle", "$id" : ObjectId("4ea97cf4a0ee87eee9573adf")
                            }
                    ] ] } }
```

Plugin MongoDB GORM

- Limitación 2 - Indexación de referencias (DBRef)

```
class Detalle implements Serializable {

    static mapWith = 'mongo'

    ObjectId id
    String nombre
    Maestro maestro

    static constraints = {
        nombre blank: false
        maestro nullable: false
    }

    static mapping = {
        nombre index: true
        maestro index: true

        writeConcern WriteConcern.FSYNC_SAFE
    }

}
```

```
class BootStrap {

    def init = { servletContext ->
        Detalle.collection.ensureIndex('maestro.$id')
    }

    def destroy = {
    }

}
```

```
> db.system.indexes.find()
{ "name" : "_id_", "ns" : "detalle", "key" : { "_id" : 1 }, "v" : 0 }
{ "name" : "nombre_1", "ns" : "detalle", "key" : { "nombre" : 1 }, "v" : 0 }
{ "name" : "_id_", "ns" : "maestro", "key" : { "_id" : 1 }, "v" : 0 }
{ "name" : "nombre_1", "ns" : "maestro", "key" : { "nombre" : 1 }, "v" : 0 }
{ "name" : "maestro.$id_1", "ns" : "detalle", "key" : { "maestro.$id" : 1 }, "v" : 0 }
```

Plugin MongoDB GORM

- Limitación 2 - Indexación de referencias (cont.)

```
> db.detalle.find({"maestro.$id": "4ea97cf4a0ee87eee9573adf"})
{
    "_id" : ObjectId("4ea97cf4a0ee87eee9573ae0"),
    "maestro" : {
            "$ref" : "detalle",
            "$id" : ObjectId("4ea97cf4a0ee87eee9573adf")
    },
    "nombre" : "Detalle1", "version" : 0 }
```

```
> db.detalle.find({"maestro.$id": "4ea97cf4a0ee87eee9573adf"}).explain()
    {
            "cursor" : "BtreeCursor maestro.$id_1",
            "nscanned" : 1,  "nscannedObjects" : 1,
            "n" : 1, "millis" : 0,
            "indexBounds" : {
                    "maestro.$id" : [
                            [
                                    ObjectId("4ea97cf4a0ee87eee9573adf"), ObjectId("4ea97cf4a0ee87eee9573adf")
                            ]
                    ]
            }
    }
```

Plugin MongoDB GORM

- Limitación 3 - Búsquedas sobre propiedades embebidas

```
Maestro.collection.find(['detalles.nombre': 'Detalle4.2'])
```

```
Thu Oct 27 20:51:23 [conn99] query: maestro{ detalles.nombre: "Detalle4.2" }
Thu Oct 27 20:51:23 [conn99]    used cursor: 0x101d108d0
Thu Oct 27 20:51:23 [conn99] query maestro reslen:320 nreturned:1 0ms
```

Pero luego, hay que obtener el detalle en cuestión...

```
def maestro = Maestro.collection.find(['detalles.nombre': 'Detalle4.2'])[0]
if (maestro) {
    def detalle = maestro.detalles.find { it.nombre == 'Detalle4.2' }
}
```

Plugin MongoDB GORM

- Limitación 4 - Búsquedas embebidas con múltiples coincidencias

```
Factura.collection.find(['articulos.descripcion': '~/(?i)iPhone/'])
```

```
Thu Oct 27 20:53:45 [conn99] query: factura{ articulos.descripcion: "~/(i)iPhone/" }
Thu Oct 27 20:53:45 [conn99]    used cursor: 0x101f108e1
Thu Oct 27 20:53:45 [conn99] query factura reslen:932 nreturned:1 0ms
```

Pero como puede haber varias líneas que coincida la descripción de artículo...

```
def factura = Factura.collection.find(['articulos.descripcion': '~/(?i)iPhone/'])[0]
if (factura) {
    def codigos = factura.articulos.findAll { it.descripcion =~ /(?i)iPhone/ }.collect { it.codigo }
    def articulos = Articulo.findAllByCodigoInList(codigos)
}
```

Plugin MongoDB GORM

- Limitación 5 - Búsquedas por existencia en campos embebidos (e.j. facturas sin cobros)

```
Factura.collection.find(['cobros': ['$exists': false]])
```

```
Thu Oct 27 20:55:45 [conn99] query: factura{ cobros: { $exists: false } }
Thu Oct 27 20:55:45 [conn99]    used cursor: 0x2314508a1
Thu Oct 27 20:55:45 [conn99] query factura reslen:320 nreturned:15 0ms
```

Plugin MongoDB GORM

- Limitación 6 - Búsquedas con $where (expresiones Javascript)

```
Factura.collection.find(['$where': 'this.total > 1000'])
```

```
Thu Oct 27 21:55:45 [conn99] query: factura{ $where: "this.total > 1000" }
Thu Oct 27 21:55:45 [conn99]    used cursor: 0x101d42e20
Thu Oct 27 21:55:45 [conn99] query factura reslen:92 nreturned:13 0ms
```

Plugin MongoDB GORM

- Limitación 7 - Búsquedas 'low API' en referencias (sin plugin)

```
def refMaestro = new DBRef(null, Detalle.collection.name, new ObjectId("4ea97cf4a0ee87eee9573adf"))
Detalle.collection.find(['maestro': refMaestro])
```

```
Thu Oct 27 21:56:23 [conn99] query: detalle{ maestro: { $ref: "detalle", $id: ObjectId('4ea97cf4a0ee87eee9573adf') } }
Thu Oct 27 21:56:23 [conn99]    used cursor: 0x10302cba0
Thu Oct 27 21:56:23 [conn99] query factura reslen:141 nreturned:1 0ms
```

Plugin MongoDB GORM

- Limitación 8.1 - Cálculos mediante Map/Reduce (servidor)

```
def db = mongo.getDB("test")

def words = ['foo', 'bar', 'baz']
def rand  = new Random()

1000.times {
    db.words << [word: words[rand.nextInt(3)]]
}

assert db.words.count() == 1000
```

```
def result = db.words.mapReduce(
    """
    function map() {
        emit(this.word, {count: 1})
    }
    """,
    """
    function reduce(key, values) {
        var count = 0
        for (var i = 0; i < values.length; i++)
            count += values[i].count
        return {count: count}
    }
    """,
    "mrresult",
    [:] // No Query
)

assert db.mrresult.count() == 3
assert db.mrresult.find()*.value*.count.sum() == 1000
```

Plugin MongoDB GORM

- Limitación 8.2 - Cálculos mediante agrupación (db.group)

```
Click.collection.insert(day: 1, total: 10)
Click.collection.insert(day: 1, total: 14)
Click.collection.insert(day: 2, total: 45)
Click.collection.insert(day: 1, total:  9)
Click.collection.insert(day: 3, total: 32)
Click.collection.insert(day: 2, total: 11)
Click.collection.insert(day: 3, total: 34)

def keyf = "function(clicks) { return clicks.day % 2 ? { odd: true } : { even: true } }"

def command = ['$keyf': keyf, cond: [:], initial: [count: 0], $reduce: "function(doc, out) { out.count += doc.total }"]

def result = Click.collection.group(command)

println result
```

[[odd:true, count:99.0], [even:true, count:56.0]]

Plugin MongoDB GORM

- Limitación 8.3 - Procedimientos almacenados (db.eval)

```
function my_erase() {
  db.things.find().forEach( function(obj) {
                delete obj.foo;
                db.things.save(obj);
              } );
}

my_erase();
```

Para evitar cargar en cliente el contenido entero de la colección...

```
db.eval(my_erase);
```

Conclusiones

- Lecciones aprendidas en mi experiencia de desarrollo:
  - Piensa muy bien el diseño de tu repositorio MongoDB
  - Si embebes colecciones, prevé si necesitarás identificar los elementos
  - Aprende *GMongo:* se convertirá en tu mejor aliado
  - No te dejes influenciar por lo que ofrece o no ofrece el plugin
  - MongoDB no es sólo CRUD: investiga, aprende, disfruta

Preguntas

&

Respuestas