



# Retour d'expérience avec MongoDB et PHP

5 octobre 2013

# INTRODUCTION

## Julien BOURDIN



- ▶ Ingénieur Centrale Lyon
- ▶ Architecte PHP
- ▶ Expert Zend Framework
- ▶ Co-fondateur & Directeur Technique de WebTales

[julien.bourdin@webtales.fr](mailto:julien.bourdin@webtales.fr)

 @Sylfraor

WebTales, éditeur de solution open-source

Incubateur Centrale Paris  
■ ■ ■ ■ Accélérer votre réussite



Introduction

Pourquoi  
MongoDB

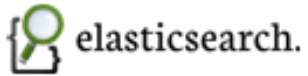
Comment ?

Points  
d'attention

Cas d'usage

Q&R

# Introduction



zend® Server



**RUBEDO**

[www.rubedo-project.org](http://www.rubedo-project.org)



Introduction

Pourquoi  
MongoDB ?

Comment ?

Points  
d'attention

Cas d'usage

Q&R



# Introduction

## NoSQL ?

**Introduction**

Pourquoi  
MongoDB ?

Comment ?

Points  
d'attention

Cas d'usage

Q&R

**POURQUOI MONGODB ?**



# Principe de MongoDB

- ▶ Base documentaire
  - L'unité élémentaire est le document
  - Les documents sont regroupés dans des collections
  - Une collection contient des documents hétérogènes et complets
  - Un document peut contenir des sous-documents
- ▶ La base n'est pas relationnelle
  - Les requêtes accèdent à un ou plusieurs documents filtrés par critères



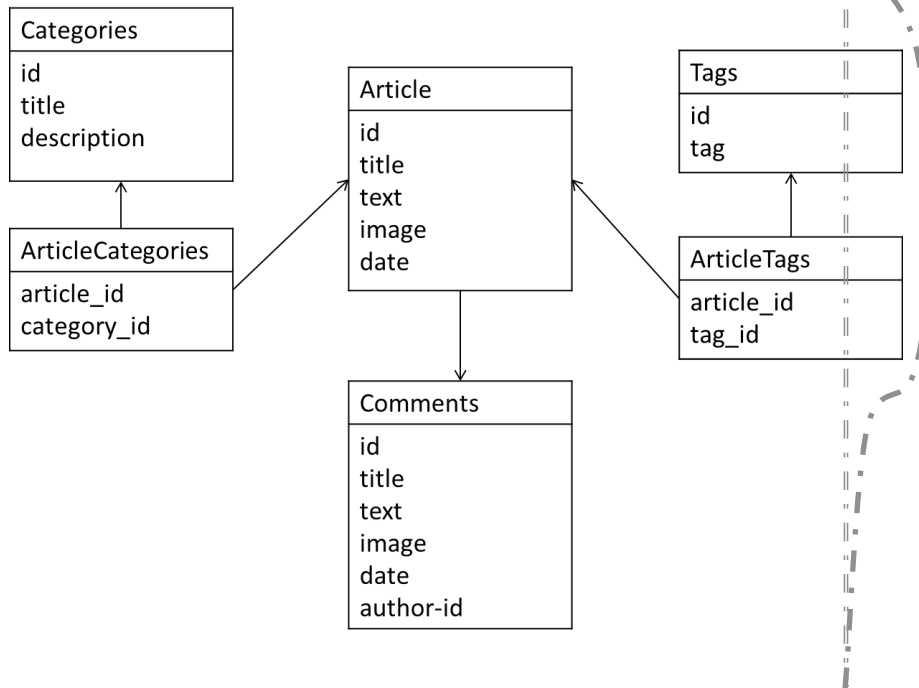
# Principe de MongoDB

- ▶ Base documentaire : un contenu = 1 objet Json

```
{  
  "email" : "admin@webtales.fr",  
  "login" : "admin",  
  "createTime" : 1376486993 ,  
  "createUser" : {"login" : "rubedo"}  
}
```



## Approche relationnelle type MySQL



- Pour un type de contenu : 6 tables
- Pour 10 types de contenus : 29 tables
- 1 requête unitaire = 6 tables et 2 jointures

## Approche NoSQL type MongoDB



- Pour un type de contenu : 1 collection
- Pour 10 types de contenus : 1 collection
- 1 requête unitaire : 1 collection

The logo consists of three overlapping, curved green shapes that resemble leaves or petals, arranged in a circular pattern. 

# Performance

- ▶ Un document est auto-suffisant
- ▶ Les données sont en mémoire
- ▶ Utilisation optimale des index
- ▶ Ecriture simple

## ▶ Replica Set

- Un serveur maître peut être répliqué
- Un groupe de serveur élit son maître

## ▶ Sharding

- Il est possible de répartir les documents sur plusieurs serveurs
  - Un document est autosuffisant
  - Une requête est poussée sur les fragments et le résultat recollé ensuite


- ▶ Un document n'est pas contraint
  - Il est possible d'ajouter des champs librement
  - Des index peuvent imposer des contraintes
  - Des opérateurs permettent de filtrer des contenus sur des champs optionnels
- ▶ Une collection peut être très hétérogène

Les CMS  
gèrent  
des  
Contenus  
structurés  
et  
classés

[Nouvelles technologies / Découvertes](#) ← Catégories

**Sortie de Rubedo, les premiers tests ...** ← Titre

Rubedo, le nouveau CMS d'entreprise Big Data, est enfin sorti des cartons de WebTales. Entièrement libre, il est téléchargeable sur GitHub et testable en ligne sur <http://www.webtales.fr> ← Texte



← Image

**RUBEDO**

NoSQL → Innovation → CMS ← Tags

[Commentaires \(8\)](#) ← Commentaires



## Cas du CMS

- ▶ Un contenu est un document
  - L'ensemble des données sont dans le document
  - L'hétérogénéité permet de gérer une infinité de type de contenu
  - La consultation est toujours faite sur une liste de contenus ou un contenu seul
  - L'intégrité est limité au document

**MISE EN ŒUVRE EN PHP**



## Le driver PHP

- ▶ Un drive Natif est proposé en PHP (extension PECL Mongo)

```
$m = new MongoClient(); // connexion  
$db = $m->selectDB("exemple");
```

- ▶ Un objet représente une collection et autorise les requêtes

```
$collection = $m->selectDB("foo")->selectCollection("bar");  
$collection = $m->foo->bar;
```

- ▶ Un document est un array PHP

```
$document = $collection->findOne(array('critere' => 'valeur'));
```





## Exemple de requêtes

- ▶ Read

```
$document = $collection->findOne(array('critere' => 'valeur'));
```

- ▶ Create

```
$obj = array('x' => 1);  
$collection->save($obj);
```

- ▶ Update

```
$newdata = array('$set' => array("address" => "1 Smith Lane"));  
$collection->update(array("firstname" => "Bob"), $newdata);
```

- ▶ Delete

```
$collection->remove(array('type' => 94), array("justOne" => true));
```



## Identifiant unique

- ▶ Un ObjectId est un type BSON sur 12 octets
  - 4 octets représentent le temps (timestamp Unix),
  - 3 octets identifient la machine,
  - 2 octets identifient le processus,
  - 3 octets sont formés par un dernier compteur (incrément)
- ▶ A priori, un ID mongo est universel
- ▶ N'importe quelle machine peut générer un ID, pas seulement le serveur MongoDB
  
- ▶ En PHP
  - `$m1 = new Mongoid('51b14c2de8e185801f000006');`
  - `$m2 = new Mongoid();`



## Opérateur et filtres

- ▶ Les requêtes prennent en argument des filtres

```
{ "_id": {"$id": "520b8643c1c3dad506000003"} }
{ "$and": [{"locale": "fr"}, {"active": true}] }
```
- ▶ En PHP, ces filtres sont des array

```
$filter = array('locale' => 'Fr', 'active' => true);
$cursor = $collection->find($filter);
```
- ▶ Ces filtres sont utilisés pour filtrer les lectures et les écritures
- ▶ Des opérateurs existent
  - \$and, \$or, \$not, \$exists
  - \$gt, \$lte, \$in
  - \$geoWithin



## Opérateur et filtres

- ▶ Pour manipuler des filtres comme des objets :
  - <https://github.com/WebTales/MongoFilters>

```
$filter = Filter::Factory('In')  
    ->setName('param2')  
    ->setValue(array('value1', 'value2'));
```



## Mode d'interaction

- ▶ Ecriture : acquittement de plusieurs réplica
- ▶ Ecriture sans acquittement : Fire and Forget
- ▶ Upsert
- ▶ Update multiple ou unitaire
- ▶ Ces options sont généralement un argument supplémentaire des requêtes
  - `$collection->save($obj,array('w'=>false)); //fire and forget`



- ▶ MongoDB propose un stockage en base de fichier
  - La taille n'est pas limité
  - Les fichiers peuvent être répartis sur un cluster

```
$images = $m->my_db->getGridFS('images');
```

```
$image = $images->findOne('mongo.png');
```

```
header('Content-type: image/png;');
```

```
$stream = $image->getResource();
```

```
while (!feof($stream)) {  
    echo fread($stream, 8192);  
}
```

# POINTS D'ATTENTION



## Les relations

- ▶ Ce n'est pas une base relationnelle !
- ▶ Les relations sont gérées au niveau de l'application
  - Stockage d'un ID dans un champs du document
  - Requête secondaire pour les données
- ▶ Choisir le format le plus adapté aux cas d'usage
  - Filtrage de contenus par auteurs
  - Taxonomies





# Arborescence de données

- ▶ Cas d'usage :
  - arborescence de page
  - Hiérarchie de classement
  
- ▶ Parent Id
  - Possibilité de requêtes successives
  - Requête unique et classement
  - Stockage de rootline (lignée de parents)



# Méta-données

- ▶ Un document peut toujours être enrichis de méta-données
- ▶ Ce sont des champs supplémentaires
  
- ▶ Exemples
  - Ajouter la date de dernière modification
  - L'auteur
  - Des commentaires
  - Une note



## Souplesse vs cohérence

- ▶ Attention à ne pas perdre la cohérence !
- ▶ La souplesse de la base ne doit pas être un problème
  - Cohérence au niveau de l'application
  - Résilience aux incohérences

# LES CAS D'USAGE



## Versionning de contenus

- ▶ Une collection contenant les anciennes versions des documents
- ▶ Ecriture de l'archive lors de la publication du nouveau contenu
- ▶ Utilisation d'une « capped collection » pour maîtriser le volume des archives



## Gestion des traductions

- ▶ Un document contient chacune de ses variantes

```
{  
  metadata : ...  
  i18n: {  
    fr: {'titre':'un document'}  
    en: {'titre':'a document'}  
  }  
}
```



# Statistiques

- ▶ Utilisation du mode « Fire & Forget »
- ▶ Utilisation d'un logger d'action vers une collection
- ▶ Possibilité d'agrégation de log hétérogènes
- ▶ Traitement a posteriori avec les fonctions d'agrégation MongoDB



## Intégration avec Elasticsearch

- ▶ Un moteur de recherche à facette
- ▶ Documentaire également
- ▶ Gère les types hétérogènes
- ▶ Branchement au niveau de l'application :
  - Ecriture dans mongoDB déclenche une mise à jour de l'index



**Q & R**