

# MongoDB and PHP

César D. Rodas

[crodas@ferpectos.com](mailto:crodas@ferpectos.com)

<http://cesarodas.com/>

Latinoware 2010

Foz do Iguaçu, Brasil

# Who is this fellow?

- ⊙ Paraguayan :-)
- ⊙ Zealot of
  - Open Source (BSD-license)
  - PHP
  - MongoDB
- ⊙ PHP/PECL Developer
- ⊙ Work for a cool company (btw, we're hiring)
- ⊙ ... and few other things

# MongoDB

- ⊙  $\langle \text{EN} \rangle \text{Mongo} \langle / \text{EN} \rangle \neq \langle \text{PT} \rangle \text{Mongo} \langle / \text{PT} \rangle$
- ⊙ Document oriented database
- ⊙ Fast, Scalable, Easy to use (devel-friendly)
- ⊙ Support Indexes
  - Simple
  - Compound
  - Geo spatial
- ⊙ Schemaless
- ⊙ Support sharding
- ⊙ Nested documents
- ⊙ PECL client

Documents?

It's just an `Array()`

In fact everything is an `Array()` in MongoDB

<http://bit.ly/mongodb-php>

# MongoDB - Operations

## ⊙ Select

- \$gt, \$lt, \$gte, \$lte, \$eq, \$neq: >, <, >=, <=, ==, !=
- \$in, \$nin, \$or
- \$size, \$exists
- \$where: Any javascript expression
- group()
- limit()
- skip()

## ⊙ Update

- \$set
- \$unset
- \$push
- \$pull
- \$inc
- findAndModify()



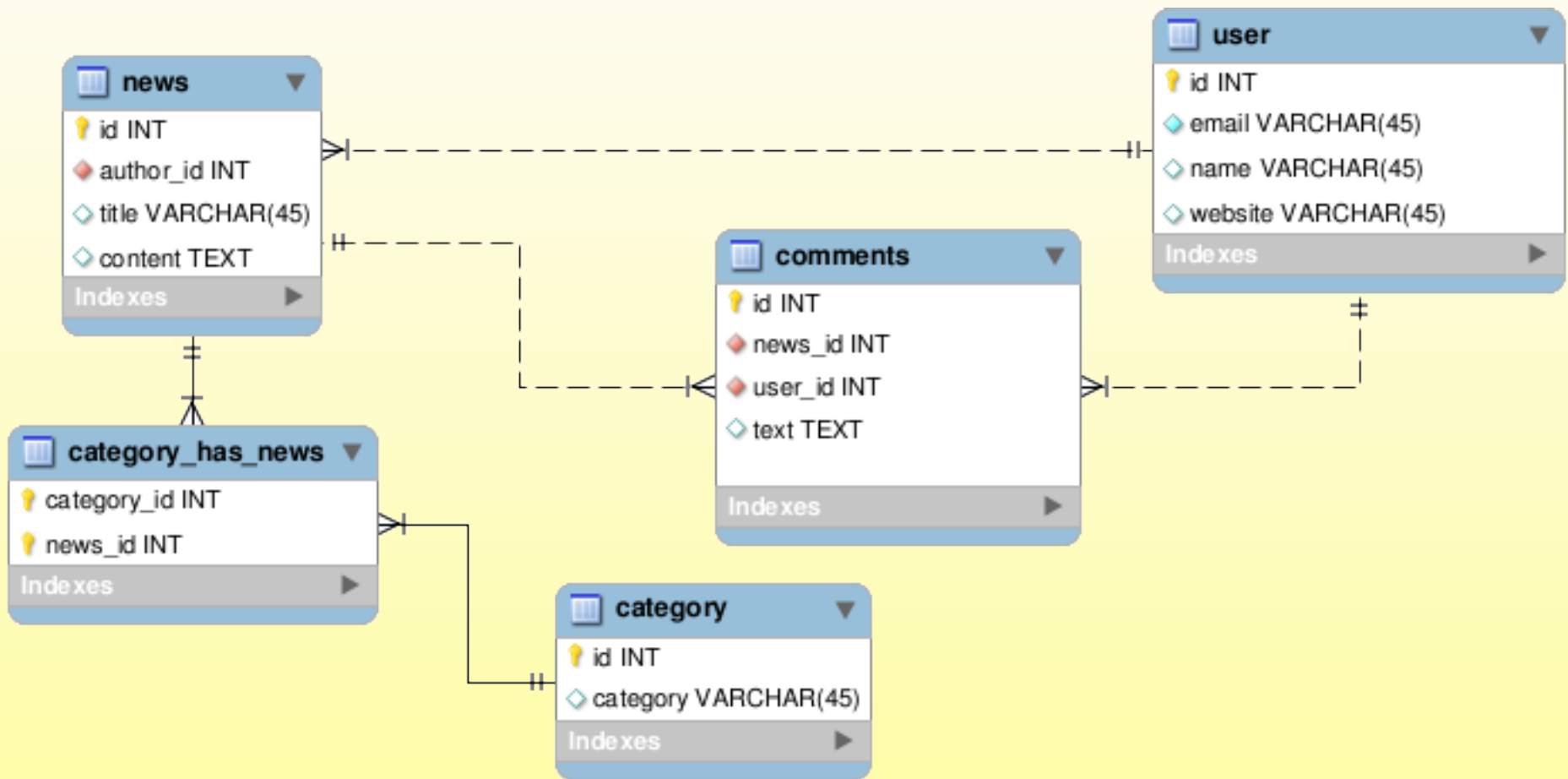
Is it better than MySQL (or Pg)?

NO.

It is different.

# Differences with a rel. database

- ⊙ Denormalize data is not bad
  - It makes queries simplest and faster
  - Disk-space is much cheaper than CPU (and your user's time)
  - Much simple to distribute data across multiple nodes
- ⊙ No CPU wasting doing ORM
  - Objects in the programming language and in the database
  - No abstraction to generate SQL
- ⊙ No SQL injection :-)
- ⊙ No Joins (they are evil!)
- ⊙ Batch processing
- ⊙ No CREATE TABLE (or ALTER TABLE) needed



The fun part

# Data-structure

```
$user = array(  
    'name' => 'crodas',  
    'email' => 'crodas@ferpectos.com',  
    'website' => 'http://cesarodas.com',  
);
```

```
$category = array(  
    array('name' => 'Sport', 'description' => 'foo'),  
    array('name' => 'Politcs', 'description' => 'bar')  
);
```

```
$comment = array(  
    'news' => $news['_id'],  
    'text' => 'I like MongoDB',  
    // these are duplicated!  
    'name' => $user['name'], /* 'crodas' */  
    'website' => $user['website'], /* 'http://cesarodas.com', */  
);
```

```

$news = array(
    'title' => 'My Talk about MongoDB',
    'content' => 'MongoDB rules, bla, bla!',
    'author' => $user['_id'],
    // duplicated items
    'authorName' => $user['user'],
    'categories' => array(
        // copy all categories (incuding _id and name)
        array('id' => $category[0]['_id'], 'name' => $category[0]['name']),
    ),
    'comments' => array(
        // copy 10 comments (we show 10 comments and pagination buttons)
        $comment[0],
        $comment[1],
    ),
    // total comments for this news, to made easier pagination
    'totalComments' => count($comment),
);

```



# Select

// MySQL

```
"SELECT news.*, user.name FROM news  
INNER JOIN user ON user.id = news.author_id WHERE id = 1"
```

```
"SELECT category.category FROM category_has_news  
INNER JOIN category ON category WHERE news_id = 1"
```

```
"SELECT * FROM comments  
INNER JOIN user ON user.id = comments.user_id  
WHERE news_id = 1"
```

// In MongoDB

```
$mongo = new MongoDB;  
$db = $mongo->database;
```

```
$news = $db->news->find(array('_id' => 1));
```

# Select

// Get 10-most commented news

```
$db->news->find(array())->sort(array('totalComments' => -1))->limit(5);
```

// MySQL

// I'm bored to think in SQL

# Add new comment

```
$query = array('_id' => $_POST['news_id']);
$properties = array('totalComments' => true);
$news = $db->news->findOne($query, $properties);

if (empty($news)) throw new Exception("not valid news");

$new_comment = array(
    'name' => $_SESSION['user']['name'],
    'website' => $_SESSION['user']['website'],
    'text' => $_POST['text'], 'news' => $_POST['news_id'],
);
$db->comment->save($new_comment);

$update = array('$inc' => array('totalComments' => 1));
if ($news['totalComments'] < 10)
    $update['$push'] = array('comments' => $new_comment);
$db->news->update($news, $update);
```

What if the user changes its name?

# Update

- ⦿ We're duplicating the user name in several places
- ⦿ Update the name in every collection
- ⦿ MongoDB can update multiple documents at once

# Update

// Catch the update event somewhere in your app and run this:

// update comments

```
$update = array(  
    '$set' => array('name' => $new_name)  
);  
$opt = array('multiple' => true);  
$db->comments->update(array('news' => 1), $update, $opt);
```

// update news (and embed comments)

```
$update = array( '$set' => array('comments.name' => $new_name ) );  
$db->news->update(array('comments._id' => $user_id), $update, $opt);
```

```
$update = array( '$set' => array('authorName' => $new_name) );  
$db->news->update(array('author' => $user_id), $update, $opt);
```

What about schema migration?

# Schemaless

- We don't have constraints in the DB
- Update the name in every collection
- MongoDB can update multiple documents at once



# Update Schema

// MySQL

```
"ALTER TABLE news ADD COLUMN url VARCHAR(250) NOT NULL"
```

// MongoDB (simple version – one node)

```
foreach ($db->news->find() as $news) {  
    $query = array('_id' => $news['_id']);  
    $update = array('$set' =>  
        array('url' => get_url_from_title($news['title'])))  
    );  
    $db->news->update($query, $update);  
}
```

# Update Schema

```
// better approach (can run multiple instances safely)
while (true) {
    $news = $db->command(array(
        'findAndModify' => 'news',
        // where url doesn't exists (much better than $exists => false)
        'query' => array('url' => null),
        // set a new value for url, diff than null
        'update' => array('$set' => array('url' => ' ')),
    ));
    if ($news['ok'] != 1) break;
    $query = array('_id' => $news['value']['_id']):
    $update = array('$set' => array('url' => get_url_from_title($news['value']['title'])));
    $db->news->update($query, $update);
}
```

# Beyond SQL!

# Store files

- ① No need of any distributed File system
- ① Each file has a checksum
- ① 100% customizable
  - Open data structure
  - You can extend it to support locking
  - You can save as much meta data as you want
- ① Works in a sharded-environment
- ① Native extension for NGINX

# Store files

```
$grid = $db->getGridFS();
```

```
$metadata = array(  
    "whatever" => "metadata",  
    "path" => "/foo",  
    "download" => 0  
);
```

```
$grid->storeFile($filename, $metadata);
```

```
// or
```

```
$grid->storeBytes($bytes, $metadata);
```

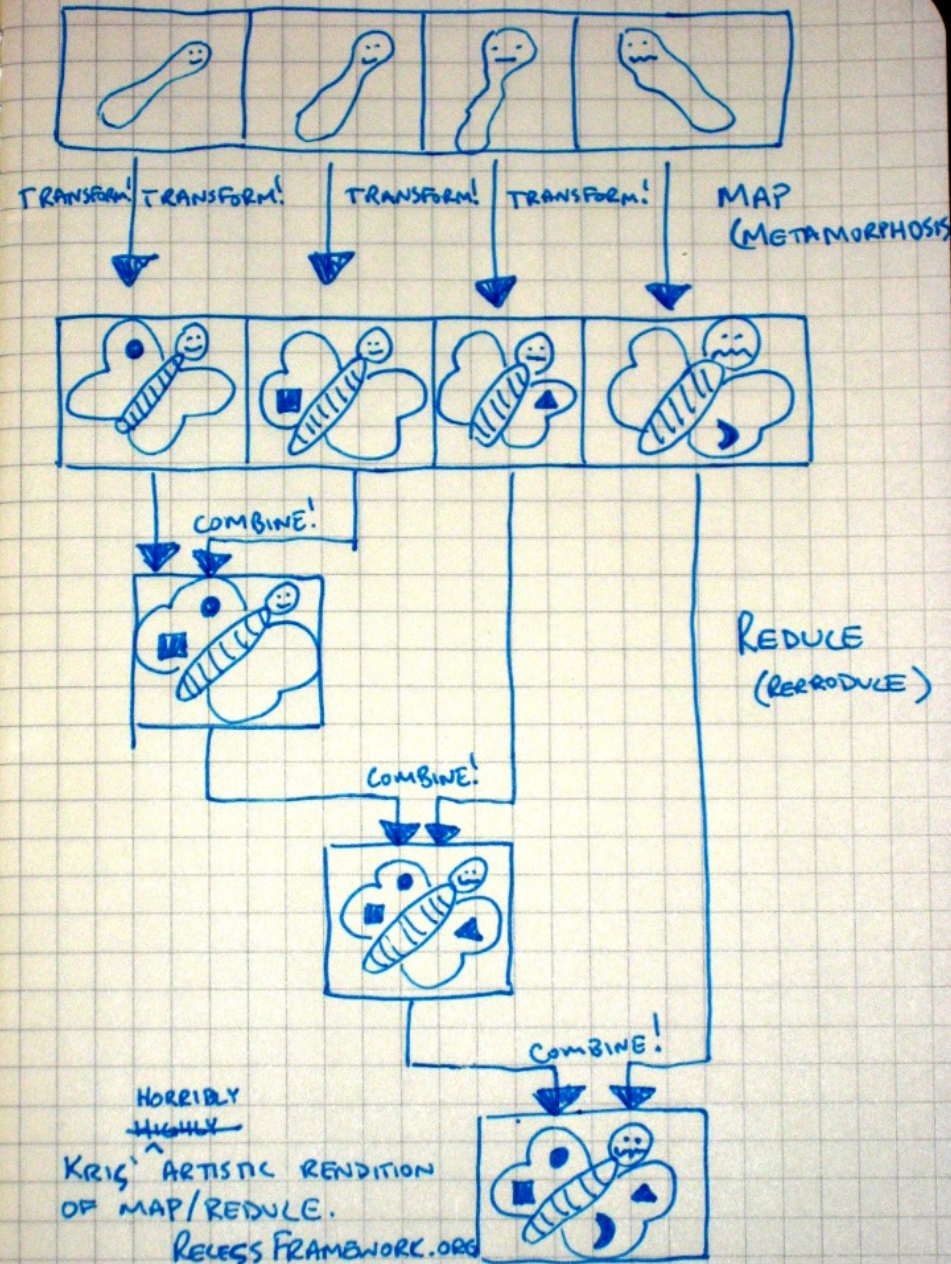
```
// Or (save $_FILE['foo'])
```

```
$grid->storeUpload('foo', $metadata);
```

# Read file

```
$grid = $db->getGridFS();  
  
$file = $grid->findOne(array('path' => '/foo'));  
  
// update download  
$update = array('$inc' => array('download' => 1));  
$id = array('_id' => $file->file['_id']);  
$grid->update($id, $update);  
  
// print it  
echo $file->Bytes();  
  
// or (a bit better)  
$tmp = '/tmp/apache/' . $file->file['_id'];  
if (!is_file($tmp)) $file->write($tmp);  
virtual($tmp);
```

# Map/Reduce





# Map/Reduce

- Well known (Google's) map/reduce approach to process tons of data
- Move the computation where the data is
- GROUP/BY like operations, but more powerful (Javascript)
- One function is applied to each news (map)
- Map() produces 0 to N new values (key, value)
- MongoDB sort everything by key.
- Reduce is called for each (key, values) and return 1 value

# Map/Reduce

```
// This method will be executed for each news
var $map = function() {
  // if the document doesn't have categories
  if (!this.categories) {
    // return nothing
    return;
  }
  for ($index in this.categories) {
    // return (several times) a new document
    // with _id: category_name, and value: 1
    emit(this.categories[$index].name, 1);
  }
}
```

# Map/Reduce

```
// we get (category.name, [1, 1, 1, 1]) and we
// return the sum to get {_id: category.name, value: 4}
var $reduce = function($key, $values) {
    var $count = 0;
    for ($index in $values) {
        $count += $values[$index];
    }
    return $count;
}
```

Questions?

Thanks

@crodas

crodas.org