

PHP and mongoDB

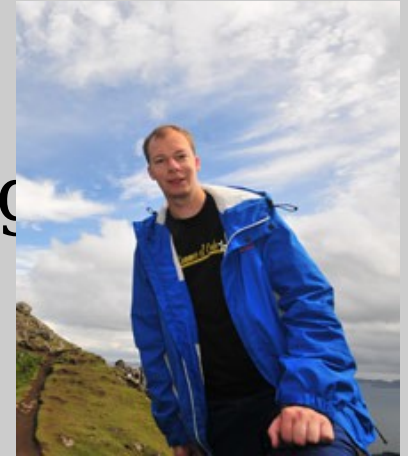
PHP Benelux - Edegem, Belgium - Jan 28th, 2012

Derick Rethans - derick@10gen.com - twitter:
@derickr

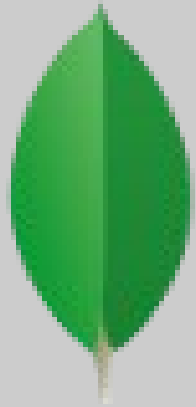
<http://joind.in/4756>

Derick Rethans

- Dutchman living in London
- PHP MongoDB driver maintainer for 10g (10gen, the company behind MongoDB)
- Author of Xdebug
- Author of the `mcrypt`, `input_filter`, `dbus`, `translit` and `date/time` extensions



What is mongoDB?



mongoDB

- mongoDB is a document storage and retrieval engine
- It requires almost no configuration to set-up a high available and high performant cluster of database servers
- Each document is stored into a collection, which is stored into a database, which is stored in a database server.

Installation by downloading from :

```
wget http://fastdl.mongodb.org/linux/mongodb-linux-x86_64-2.0.2.tgz
tar xvzf mongodb-linux-x86_64-2.0.2.tgz
cd mongodb-linux-x86_64-2.0.2/bin
mkdir ../../data
./mongod --dbpath ../../data --logpath /tmp/mongod.log --fork
tail -f /tmp/mongod.log
```

Installation through apt-get:

```
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv 7F0CEB10
$ sudo echo "deb http://downloads-distro.mongodb.org/repo/debian-sysvinit dist 10gen" >> /etc/apt/sources.list
# sudo apt-get update
$ sudo apt-get install mongodb-10gen
```

Tweak config in `/etc/mongodb.conf` if you must.
mongo (the shell) is a useful application to try out things with.



- Maintained and supported by 10gen (well, Kristina and me really)
- Can be installed with pecl: `pecl install mongo`
- Add `extension=mongo.so` to `php.ini`

Terminology

- JSON Document: the data (row)
- Collection: contains documents (table, view)
- Index
- Embedded Document (~join)

No tables or collections have to do be explicitly created

```
<?php
$m = new Mongo();
$database = $m->demo;
$collection = $database->testCollection;
?>
```

Different connection strings:

- `$m = new Mongo("mongodb://localhost");`
- `$m = new Mongo("localhost:27017");`
- `$m = new Mongo("mongodb://localhost:29000");`
- `$m = new Mongo("mongodb://mongo.example.com");`
- `$m = new Mongo("mongodb://mdb1.example.com,mdb2.example.com");`

- Stored as BSON (Binary JSON)
- Can have embedded documents
- Have a unique ID (the `_id` field)

Simple document:

```
{
  "_id" : ObjectId("4cb4ab6d7addf98506010001"),
  "id" : NumberLong(1),
  "desc" : "ONE"
}
```

Document with embedded documents:

```
{
  "_id" : "derickr",
  "name" : "Derick Rethans",
  "articles" : [
    {
      "title" : "Profiling PHP Applications",
      "url" : "http://derickrethans.nl/talks/profiling-phptourlille11.pdf",
    },
    {
      "title" : "Xdebug",
      "url" : "http://derickrethans.nl/talks/xdebug-phpbcn11.pdf",
    }
  ]
}
```

Inserting a document

```
<?php
$document = array(
    "_id" => "derickr",
    "name" => "Derick Rethans",
    "articles" => array(
        array(
            "title" => "Profiling PHP Applications",
            "url" => "http://derickrethans.nl/talks/profiling-phptourlille11.pdf",
        ),
        array(
            "title" => "Xdebug",
            "url" => "http://derickrethans.nl/talks/xdebug-phpbcn11.pdf",
        )
    )
);

$m = new Mongo();
var_dump( $m->demo->articles->insert( $document ) );
?>
```

mongoDB supports many types

- null
- boolean
- integer (both 32-bit and 64-bit, `MongoInt32`, `MongoInt64`)
- double
- string (UTF-8)
- array
- associative array
- `MongoRegex`
- `MongoId`
- `MongoDate`
- `MongoCode`
- `MongoBinData`

- Every document needs an ID
- IDs need to be unique
- You can set your own, or let them be auto generated

```
<?php
$m = new Mongo();
$c = $m->demo->articles;

$d = array( 'name' => 'Derick', '_id' => 'derickr' );
$c->insert( $d );
var_dump( $d );

$d = array( 'name' => 'Derick' );
$c->insert( $d );
var_dump( $d );
?>
```

So far, we have not checked for whether inserts worked.

```
<?php
$m = new Mongo;
$c = $m->demo->articles;

$c->insert( array( '_id' => 'derickr' ) );
$c->insert( array( '_id' => 'derickr' ) );
?>
```

"Safe" inserts:

```
<?php
$m = new Mongo;
$c = $m->demo->articles;

try {
    $c->insert(
        array( '_id' => 'derickr' ), // document
        array( 'safe' => true )     // options
    );
} catch ( Exception $e ) {
    var_dump( $e->getMessage() );
}
?>
```

Checking for errors (2)

Safeness levels:

- Confirm change recorded in memory: `array('safe' => true);`
- Confirm inclusion in journal: `array('j' => true);`
- Confirm committed to disk: `array('fsync' => true);`

```
<?php
$m = new Mongo();
$c = $m->demo->articles;

$c->insert(
    array( '_id' => 'derickr' ), // document
    array( 'j' => true )       // options
);

$c->insert(
    array( '_id' => 'mongodb' ), // document
    array( 'fsync' => true )     // options
);
?>
```

Querying

```
<?php
$m = new Mongo();
$record = $m->demo->articles->findOne();

$record = $m->demo->articles->findOne( array( '_id' => 'derickr' ) );

$record = $m->demo->articles->findOne( array( 'articles.title' => 'Xdebug' ) );

$record = $m->demo->articles->findOne(
    array( 'articles.title' => 'Xdebug' ),
    array( 'name' => true, 'articles.url' => true )
);
var_dump( $record );
?>
```

Querying and looping

Finding multiple documents is done with the `find()`, and returns an iterator in the form of a **MongoCursor** object.

```
<?php
$m = new Mongo();
$c = $m->demo->articles;

$cursor = $c->find();
var_dump( $cursor );

foreach ( $cursor as $r )
{
    var_dump( $r );
}
?>
```

Sorting

```
<?php
$m = new Mongo();
$c = $m->demo->articles;

$cursor = $c->find()->sort( array( '_id' => -1 ) );

foreach ( $cursor as $r )
{
    var_dump( $r );
}
?>
```

Limit and skip

```
<?php
$m = new Mongo();
$c = $m->demo->articles;

$cursor = $c->find();
$cursor->sort( array( '_id' => 1 ) )
    ->limit( 3 )
    ->skip( 3 );

foreach ( $cursor as $r )
{
    var_dump( $r );
}
?>
```

Advanced querying operators

Besides testing for exact matches, MongoDB also has operators. Operators start with a '\$', so make sure to use single quotes!

Compare:

- \$lt, \$lte, \$gt, \$gte (<, <=, >, >=)
- \$ne (not equal)
- \$exists (field exists)
- \$mod (modulo)

Logical:

- \$or (one needs to match)
- \$and (all need to match)
- \$nor (not or)
- \$not

Array:

- \$in (value needs to be one of the elements of the given array)
- \$nin (not in)
- \$all (value needs to match all elements of the given array)
- \$size (exact size of array)
- \$elemMatch (element in an array matches the specified match expression)

Others:

- \$type (check for type number, see)

Querying: \$gte

```
<?php
$m = new Mongo;
$c = $m->demo->circus;

$c->insert( array(
    '_id' => 'circ1',
    'name' => 'Diaspora',
    'performers' => 43,
    'elephpants' => array (
        array ( 'name' => 'Annabelle', 'colour' => 'pink', 'year' => 1964 ),
        array ( 'name' => 'Chunee', 'colour' => 'blue', 'year' => 1826 )
    )
) );

$c->insert( array(
    '_id' => 'circ2',
    'name' => 'Sensible',
    'performers' => 27,
    'elephpants' => array (
        array ( 'name' => 'Kandula', 'colour' => 'pink', 'year' => 2001 ),
        array ( 'name' => 'Kolakolli', 'colour' => 'blue', 'year' => 2006 )
    )
) );
?>
```

Find circuses with 40 or more performers.

```
<?php
$m = new Mongo;
$c = $m->demo->circus;
$r = $c->findOne(
    array( 'performers' => array( '$gte' => 40 ) ),
    array( 'name' => true, 'performers' => true )
);
var_dump( $r );
?>
```

Find circuses which have blue elephants after 1900.

```
<?php ini_set( 'xdebug.var_display_max_depth', 4 );
$m = new Mongo;
$c = $m->demo->circus;
$r = $c->find(
    array( 'elephants.colour' => 'blue', 'elephants.year' => array( '$gt' => 1900 ) )
);
foreach( $r as $c ) {
    echo $c['name'], ': ';
    foreach( $c['elephants'] as $elephant ) {
        echo "\n- ", join( ", ", $elephant );
    }
    echo "\n\n";
}
?>
```

Querying: matching array elements (\$elemMatch)

```
<?php
$m = new Mongo;
$c = $m->demo->circus;
$r = $c->find(
    array (
        'elephpants' =>
            array( '$elemMatch' => array ( 'colour' => 'blue', 'year' => array( '$gt' => 1900 ) )
        )
    ) );
foreach( $r as $c ) {
    echo $c['name'], ': ';
    foreach( $c['elephpants'] as $elephpant ) {
        echo "\n- ", join( ", ", $elephpant );
    }
    echo "\n\n";
}
?>
```

Updating documents

```
<?php
$m = new Mongo;
$c = $m->demo->elephpants;
$c->remove();

$c->insert( array( '_id' => 'e42', 'name' => 'Kamubpo' ) );
var_dump( $c->findOne( array( '_id' => 'e42' ) ) );

$c->update( array( '_id' => 'e42' ), array( 'name' => 'Bo Tat' ) );
var_dump( $c->findOne( array( '_id' => 'e42' ) ) );

$c->update( array( 'name' => 'Bo Tat' ), array( 'age' => '17' ) );
var_dump( $c->findOne( array( '_id' => 'e42' ) ) );
?>
```

update() replaces the document matching criteria entirely with a new object.

Modifying documents

```
<?php
$m = new Mongo;
$c = $m->demo->elephants;
$c->remove();

$c->insert( array( '_id' => 'e43', 'name' => 'Dumbo' ) );
var_dump( $c->findOne( array( '_id' => 'e43' ) ) );

$c->update( array(
    'name' => 'Dumbo' ), // criteria
    array(
        '$set' => array ( 'age' => '17' )
    )
);
var_dump( $c->findOne( array( '_id' => 'e43' ) ) );
?>
```

Update only updates the first document it finds by default.

```
<?php
$m = new Mongo;
$c = $m->demo->elephpants;
$c->drop();

$c->insert( array( '_id' => 'e42', 'name' => 'Kamubpo', 'age' => 17 ) );
$c->insert( array( '_id' => 'e43', 'name' => 'Denali', 'age' => 17 ) );

$c->update( array( 'age' => 17 ), array( '$inc' => array( 'age' => 1 ) ) );

var_dump( iterator_to_array( $c->find() ) );
?>
```

You can set an option to get all matching documents to be updated

```
<?php
$m = new Mongo;
$c = $m->demo->elephpants;
$c->drop();

$c->insert( array( '_id' => 'e42', 'name' => 'Kamubpo', 'age' => 17 ) );
$c->insert( array( '_id' => 'e43', 'name' => 'Denali', 'age' => 17 ) );

$c->update(
    array( 'age' => 17 ),           // criteria
    array( '$inc' => array( 'age' => 1 ) ), // update spec
    array( 'multiple' => true )    // options
);

var_dump( iterator_to_array( $c->find() ) );
?>
```

Upserting documents

upsert: if the record(s) do not exist, insert one.

```
<?php
$m = new Mongo;
$c = $m->demo->elephants;
$c->drop();

function birthday( $c, $name )
{
    $c->update(
        array( 'name' => $name ),           // criteria
        array( '$inc' => array( 'age' => 1 ) ), // update spec
        array( 'upsert' => true )         // options
    );
    var_dump( $c->findOne( array( 'name' => 'Santon' ) ) );
}

birthday( $c, 'Santon' );
birthday( $c, 'Santon' );
?>
```

Document update modifiers: Single value manipulation

- `$set` (sets a field to a new value)
- `$unset` (removes a field)
- `$inc` (increments the value in a field)

```
<?php
$m = new Mongo;
$c = $m->demo->circus;
$c->remove();

$c->insert( array( '_id' => 'circ3', 'name' => 'Humberto', 'performers' => 12 ) );
$c->update( array( 'name' => 'Humberto' ), array( '$inc' => array( 'performers' => 4 ) ) );
var_dump( $c->findOne( array( 'name' => 'Humberto' ) ) );
?>
```

Document update modifiers: Array manipulation

- `$push/$pushAll` (adds elements to an array)
- `$addToSet` (like `$push`, but without duplicates in the array)
- `$pop` (removes the first or last element of an array)
- `$pull/$pullAll` (removes elements from an array)

```
<?php
$m = new Mongo;
$c = $m->demo->circus; $c->remove();
$c->insert( array( '_id' => 'circ5', 'name' => 'Funny Led' ) );

$c->update(
    array( '_id' => 'circ5' ),
    array( '$push' => array(
        'elephants' => array( 'name' => 'Bobo', 'year' => 2012 ) )
    )
);
var_dump( $c->findOne( array( 'name' => 'Funny Led' ), array( 'elephants' => true ) ) );
?>
```

Document update modifiers: Misc

- \$rename (renames fields)
- \$bit (bitwise updates of field)

```
<?php
$m = new Mongo;
$c = $m->demo->bits;
$c->remove();

$c->insert( array( 'bitmask' => 0b1001 ) );
echo decbin( $c->findOne()['bitmask'] ), "\n";

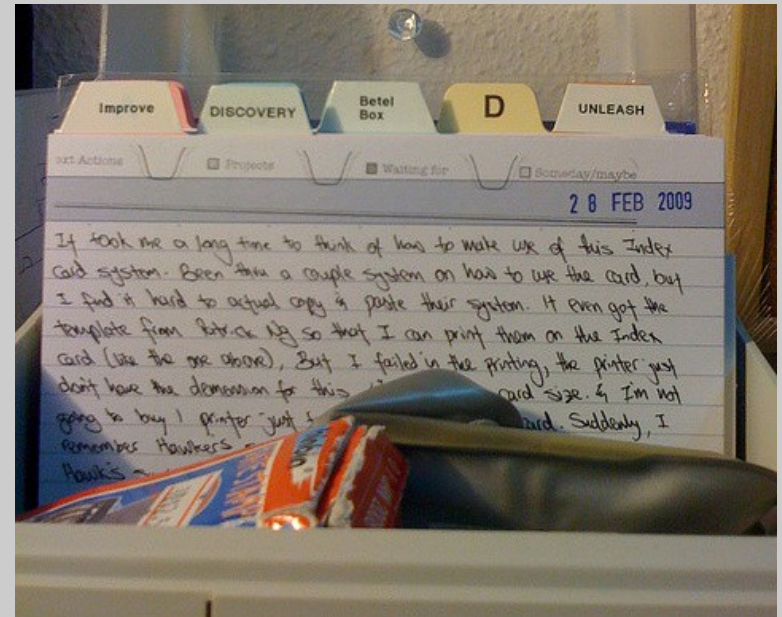
$c->update( array(), array( '$bit' => array( 'bitmask' => array( 'and' => 0b0101 ) ) ) );
echo decbin( $c->findOne()['bitmask'] ), "\n";

$c->update( array(), array( '$bit' => array( 'bitmask' => array( 'or' => 0b10100 ) ) ) );
echo decbin( $c->findOne()['bitmask'] ), "\n";

// this is not in yet ;- )
$c->update( array(), array( '$bit' => array( 'bitmask' => array( 'xor' => 0b01110 ) ) ) );
echo decbin( $c->findOne()['bitmask'] ), "\n";
?>
```

Indexes

- Just like a relational database, MongoDB also benefits from indexes.
- Every collection has (automatically) an index on `_id`.
- Indexes can be on single or multiple fields.
- `MongoCursor->explain()`.



Indexes

```
<?php ini_set('xdebug.var_display_max_depth', 1);  
$m = new Mongo;  
$c = $m->demo->elephants;  
$c->drop();  
  
$c->insert( [ '_id' => 'ele1', 'name' => 'Jumbo' ] );  
$c->insert( [ '_id' => 'ele2', 'name' => 'Tantor' ] );  
$c->insert( [ '_id' => 'ele3', 'name' => 'Stampy' ] );  
  
var_dump( $c->find( [ 'name' => 'Jumbo' ] )->explain() );  
?>
```

Indexes

```
<?php ini_set('xdebug.var_display_max_depth', 1);
$m = new Mongo;
$c = $m->demo->elephants;
$c->drop();

$c->ensureIndex( [ 'name' => 1 ] );

$c->insert( [ '_id' => 'ele1', 'name' => 'Jumbo' ] );
$c->insert( [ '_id' => 'ele2', 'name' => 'Tantor' ] );
$c->insert( [ '_id' => 'ele3', 'name' => 'Stampy' ] );

var_dump( $c->find( [ 'name' => 'Jumbo' ] )->explain() );
?>
```

Helps you with finding locations (pubs!) in a 2D space

```
<?php
$m = new Mongo; $c = $m->demo->pubs; $c->drop();

$c->ensureIndex( array( 'location' => '2d' ) );

$c->insert( [ '_id' => 'pub1', 'name' => 'Mrs Betsy Smith', 'location' => [ -0.1933, 51.5375 ] ] );
$c->insert( [ '_id' => 'pub2', 'name' => 'North London Tavern', 'location' => [ -0.2025, 51.5455 ] ] );

$closest = $m->demo->command( [
    'geoNear' => 'pubs',
    'near' => [ -0.198, 51.538 ],
    'spherical' => true,
] );
//var_dump( $closest['stats'] );

foreach ( $closest['results'] as $res ) {
    printf( "%s: %.2f km\n", $res['obj']['name'], $res['dis'] * 6378 );
}
?>
```

Commands

- Are run on the database
- Do not return a cursor
- Return one document
- Some commands have helpers in the drivers and shell

Distinct

```
<?php
$m = new Mongo; $c = $m->demo->pubs; $c->drop();

$c->insert( [ '_id' => 'pub1', 'name' => 'Mrs Betsy Smith', 'city' => 'London' ] );
$c->insert( [ '_id' => 'pub2', 'name' => 'North London Tavern', 'city' => 'London' ] );
$c->insert( [ '_id' => 'pub3', 'name' => 'Lammars', 'city' => 'Manchester' ] );
$c->insert( [ '_id' => 'pub4', 'name' => 'Weatherspoons', 'city' => 'Coventry' ] );

$r = $m->demo->command( [
    'distinct' => 'pubs',
    'key' => 'city',
    'query' => [ 'name' => [ '$ne' => 'Weatherspoons' ] ]
] );

var_dump( $r['values'] );
?>
```

Powerfull framework for running multiple operations in a pipeline

Operations are:

- `$match`: for matching documents, à la `find()`.
- `$project`: for "rewriting" documents, but more powerful with computed expressions: adding fields (`$add`), conditions (`$ifNull`), string functions (`$toLower`), etc
- `$unwind`: hands out array elements each at a time
- `$group`: aggregates items into buckets defined by key
- `$sort`
- `$limit` / `$skip`

Aggregation Example

```
<?php
$m = new Mongo; $d = $m->demo; $c = $d->articles; $c->drop();

$c->insert( [
    "title" => "Profiling PHP Applications",
    "url" => "http://derickrethans.nl/talks/profiling-phptourlille11.pdf",
    "tags" => [ "php", "profiling" ]
] );
$c->insert( [
    "title" => "Xdebug",
    "url" => "http://derickrethans.nl/talks/xdebug-phpbcn11.pdf",
    "tags" => [ "php", "xdebug" ]
] );

$m = $d->command( [
    'aggregate' => 'articles',
    'pipeline' => [
        [ '$match' => [ 'tags' => 'php' ] ],
        [ '$unwind' => '$tags' ],
        [ '$project' => [
            'title' => [ '$add' => [ '$title', ': ', '$url' ] ],
            'tag' => '$tags'
        ] ]
    ]
] );

foreach( $m['result'] as $res ) { echo $res['title'], ' (', $res['tag'], ")\n"; }
?>
```

- Ghengis: single file mongoDB management app (a la phpMyAdmin)
- RockMongo: mongoDB management app (a la phpMyAdmin)
- MMS: hosted application for instrumentation

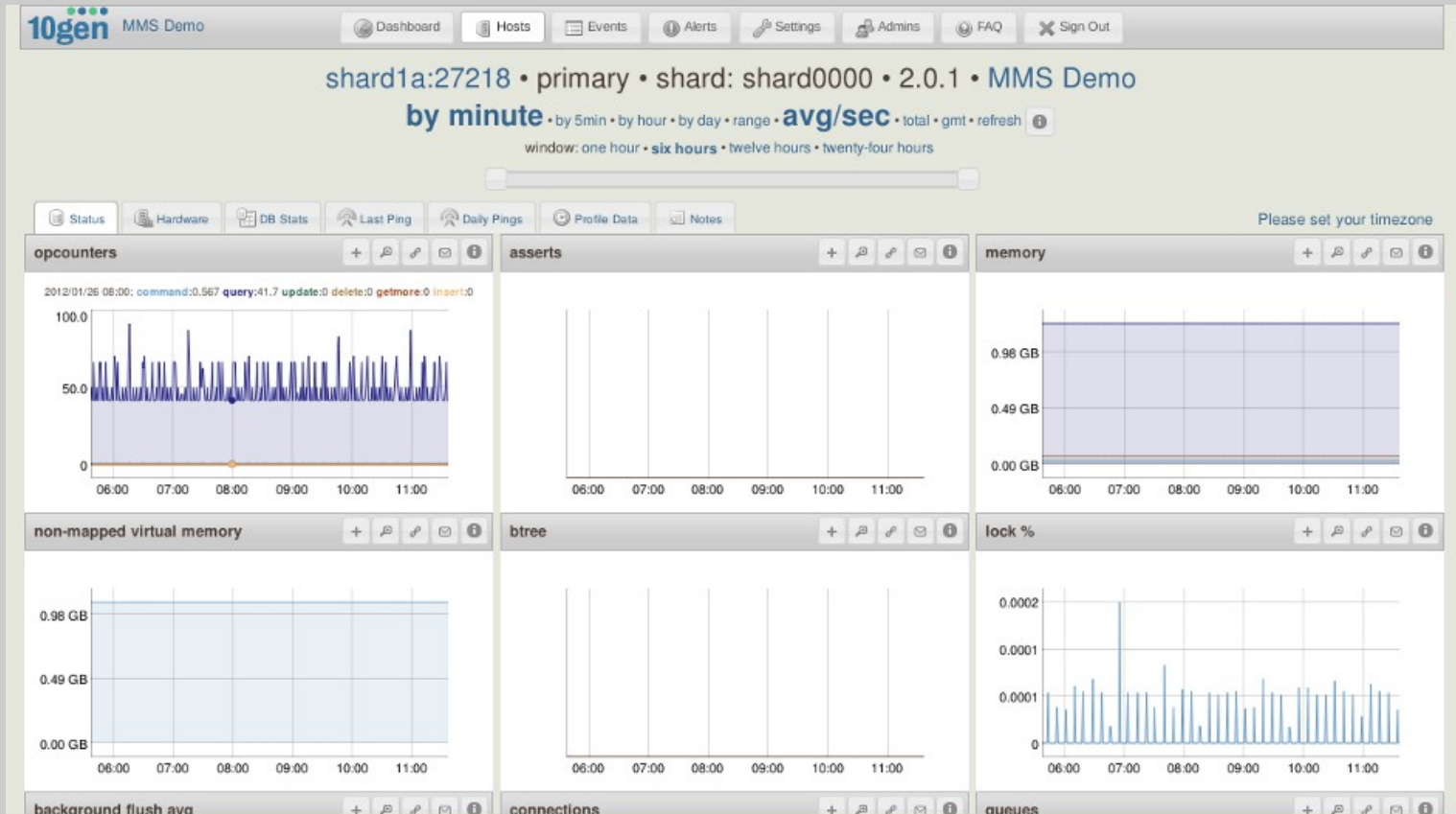
Collections

name	documents	indexes	
articles	2	1	
bits	1	1	
circus	1	1	
elephants	3	2	remove
profiles	2	1	
pubs	4	1	
test	1	1	

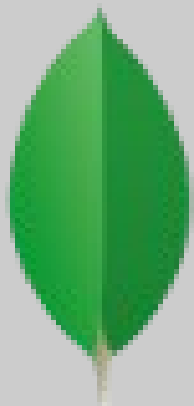
2 Indexes

- {_id: 1}
- {name: 1}

Add collection



- Download MongoDB:
<http://www.mongodb.org/downloads>
- BSON Spec: <http://bsonspec.org/>
- These slides: <http://derickrethans.nl/talks.html>
- Contact me: Derick Rethans: @derickr,
derick@10gen.com
- Feedback: <http://joind.in/4756>



mongoDB