```
{
    "title": "MongoDB ir jos ypatumai",
    "author":
    {
        "name": "Arunas",
        "surname": "Smaliukas"
    },
    "date":
    {
        "year": 2013,
        "month": "April",
        "day": 04
    }
}
```

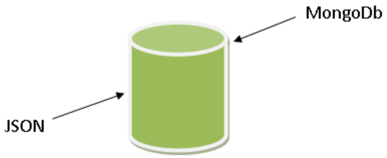
What is MongoDB?

- Non-relational database

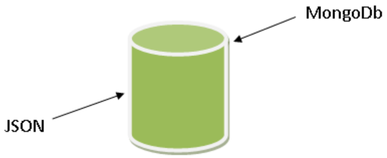
What is MongoDB?

- Non-relational database
- Schemaless database

Non-relational database



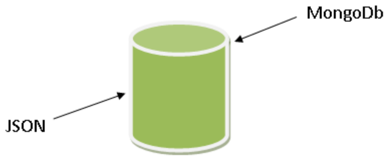
Non-relational database



JSON

- Arrays - list of items

Non-relational database



JSON

- Arrays - list of items
- Dictionaries - associative maps

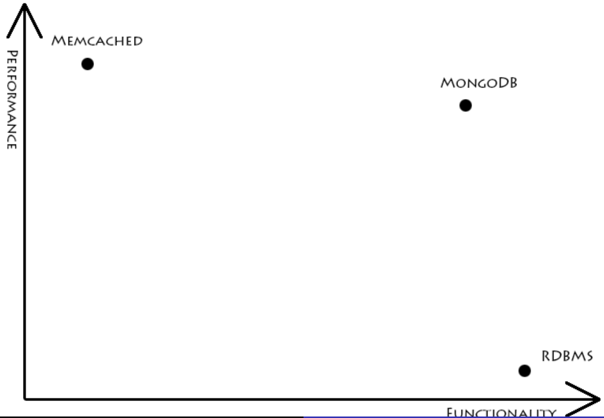
Schemaless database

- No tables

Schemaless database

- No tables
- {a:1,b:2},... {a:1,b:2,c:3}

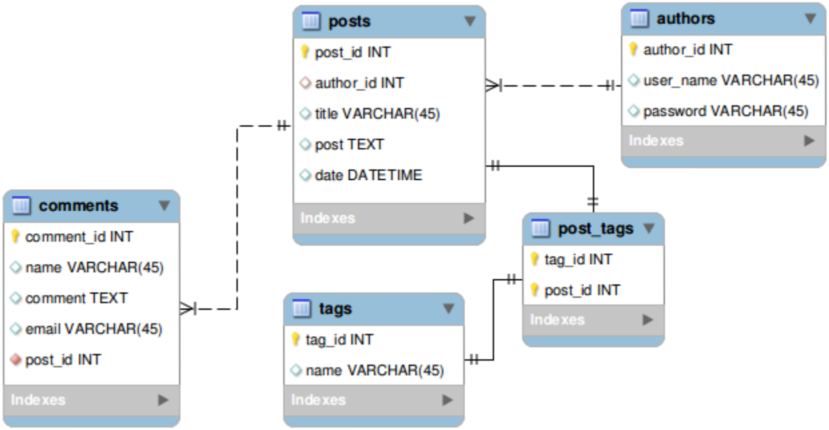
MongoDB vs RDBMS



Blog example

- Posts
- Comments
- Tags
- Votes

Blog in relational tables



Blog in Documents: posts

```
{
    "title": "Sample post",
    "body": "...",
    "author": {
        "name": "Arunas Smaliukas",
        "username": "arunas"
    },
    "date": "...",
    "comments": [{
            "name": "Commenter",
            "email": "...",
            "date": "...",
            "comment": "..."
        }, ...],
    "tags": ["tag1", "tag2", "tag3"]
}
```

Blog in Documents: authors

```
{
    "_id": "arunas",
    "password": "..."
}
```

Post votes

```
{
    ...
    "votes": 3,
    "voters": ["arunas", "...", "..."]
}
```

Post votes

```
{
        ...
        "votes": 3,
        "voters": ["arunas", "...", "..."]
}
```

```
db.posts.update(
        {"_id":"...", "voters":{$ne:"arunas"}},
        {
                $push: {"voters": "arunas"},
                $inc: {"votes": 1}
        }
)
```

Insert

- db.collection.insert({"name":"arunas", "surname":"..."});

Find

- db.collection.find({"name":"arunas"});

Find

- db.collection.find({"name":"arunas"});
- $lt, $gt, $lte, $gte. Example: {"votes":{$gte:3}};

Find

- db.collection.find({"name":"arunas"});
- $lt, $gt, $lte, $gte. Example: {"votes":{$gte:3}};
- $regex, $exsits, $type. Example: {"profession":{$exsits:true}};

Find

- db.collection.find({"name":"arunas"});
- $lt, $gt, $lte, $gte. Example: {"votes":{$gte:3}};
- $regex, $exsits, $type. Example: {"profession":{$exsits:true}};
- $or, $and, $in, $all. Example: { friends : { $all: ["Joe" , "Bob"] }, favorites : { $in : ["running" , "pickles"] } };

Find

- db.collection.find({"name":"arunas"});
- $lt, $gt, $lte, $gte. Example: {"votes":{$gte:3}};
- $regex, $exsits, $type. Example: {"profession":{$exsits:true}};
- $or, $and, $in, $all. Example: { friends : { $all: ["Joe" , "Bob"] }, favorites : { $in : ["running" , "pickles"] } };
- Queries with dot notation. Example: {"reviews.rating":{$gte:3}};

Update

- db.foo.update({"_id":"..."}, {"population":30000000},{<options>});

Update

- db.foo.update({"_id":"..."}, {"population":30000000},{<options>});
- $set, $unset. Example: {$set:{"population":30000000}};

Update

- db.foo.update({"_id":"..."}, {"population":30000000},{<options>});
- $set, $unset. Example: {$set:{"population":30000000}};
- $push, $pop, $pull, $pushAll, $pullAll, $addToSet. Example: {$pushAll:{"interests": ["skydiving" , "skiing"]}};

Update

- db.foo.update({"_id":"..."}, {"population":30000000},{<options>});
- $set, $unset. Example: {$set:{"population":30000000}};
- $push, $pop, $pull, $pushAll, $pullAll, $addToSet. Example: {$pushAll:{"interests": ["skydiving" , "skiing"]}};
- Options: upsert, multi. Example: {upsert:true}

Insert

- db.collection.remove({"score":{$lt:60}});

Indexes

- db.foo.ensureIndex({comments.author:1});

Indexes

- db.foo.ensureIndex({comments.author:1});
- Multikey index: db.foo.ensureIndex({a:1, b:1});

Indexes

- db.foo.ensureIndex({comments.author:1});
- Multikey index: db.foo.ensureIndex({a:1, b:1});
- Geospacial Spherical Indexes.

Aggregation Framework

```
{
    "_id" : ObjectId("..."),
    "name" : "Nexus 7",
    "category" : "Tablets",
    "manufacturer" : "Google",
    "price" : 199
}
```

Aggregation Framework

```
{
    "_id" : ObjectId("..."),
    "name" : "Nexus 7",
    "category" : "Tablets",
    "manufacturer" : "Google",
    "price" : 199
}
```

```
db.products.aggregate (
    [{
        $group: {
            "_id":"$category",
            "num_products":{"$sum":1}
        }
    }])
```

The Aggregation Pipeline

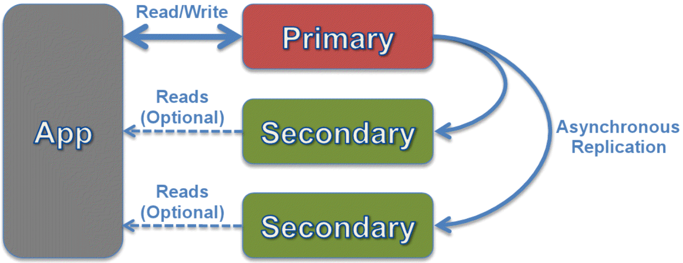
- $project (1:1);
- $match - filter (n:1);
- $group (n:1);
- $sort (1:1);
- $skip (n:1);
- $limit (n:1);
- $unwind (1:n);

Example

```
db.inventory.insert({'name':"TShirt", 'sizes':
    ["Small", "Medium", "Large", "XLarge"], '
    colors':['navy', "black", 'orange', 'red']
    })

db.inventory.aggregate([
     {$unwind: "$sizes"},
     {$unwind: "$colors"},
     {$group:
       {
            '_id': {'size':'$sizes', 'color':'
               $colors'},
            'count' : {'$sum':1}
       }
     }
])
```

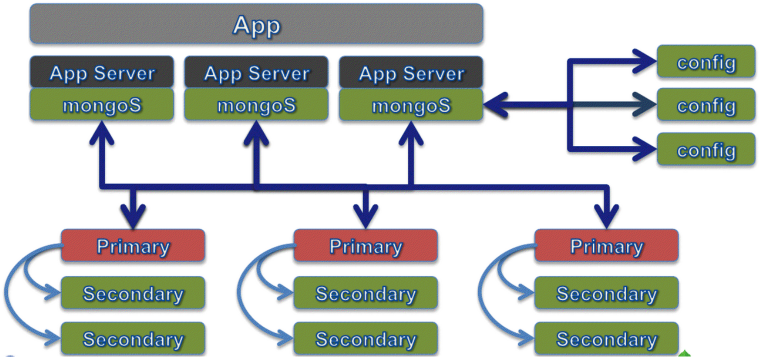
Replica sets



Replica sets

- Automatic Failover
- Automatic Recovery
- All writes to primary node
- Rolling Outages, zero downtime

Sharding


App
App Server
App Server
App Server
mongoS
mongoS
mongoS
config
config
config
Primary
Secondary
Secondary
Primary
Secondary
Secondary
Primary
Secondary
Secondary