

# Couchbase and Rails

Sergey Avseyev

[sergey@couchbase.com](mailto:sergey@couchbase.com)

@avsej



## CouchBase

# What is Couchbase

Simple. Fast. Elastic.

# Easy Scalability

Grow cluster without application changes, without downtime



**Simple. Fast. Elastic.**

# **Consistent High Performance**

Sub-millisecond read/write response & high throughput



**Simple. Fast. Elastic.**

# **Always On 24x365**

No downtime for software upgrades, hardware maintenance, etc





Simple. Fast. Elastic.

# Flexible Data Model

JSON document model with no fixed schema



gem i couchbase

## Basic K/V Operations

```
Couchbase.bucket.incr("user:#{id}:hits", :initial => 1)
Couchbase.bucket.set("user:#{id}:ip", "192.0.43.10")
Couchbase.bucket.get("user:#{id}:token", :ttl => 1.day)
```

# Rails Cache Store

```
config.cache_store = :couchbase_store,  
                     :bucket => 'cache',  
                     :expires_in => 1.hour
```

# Rails Session Store

```
config.session_store :couchbase_store,  
  :namespace => "session:",  
  :couchbase => {  
    :bucket => 'sessions',  
    :default_format => :json  
  }
```

gem i couchbase-model

# Generate Configuration

```
$ rails generate couchbase:config  
    create  config/couchbase.yml
```



## config/couchbase.yml

```
common: &common
  node_list:
    - example.com:8091
    - example.org:8091
    - example.net:8091
  username:
  password:

development:
  <<: *common
  bucket: example_development

production:
  <<: *common
  bucket: example_test
```

# Modeling

```
class Beer < Couchbase::Model
  attribute :name
  attribute :abv, :default => 0
  attribute :ibu, :default => 0
  attribute :category

  belongs_to :brewery

  view :all, :limit => 31

  before_save do |doc|
    doc.abv = doc.abv.to_f
    doc.ibu = doc.ibu.to_f
  end
end
```

## Views: Generator

```
$ rails generate couchbase:view beer avg_by_category  
  create  app/models/beer/avg_by_category/map.js  
  create  app/models/beer/avg_by_category/reduce.js
```

## Views: Map

Find average Alcohol by Volume (ABV) for each category

```
function(doc, meta) {  
  if (doc.type == "beer" && doc.category && doc.abv > 0) {  
    emit(doc.category, doc.abv);  
  }  
}
```

## Views: Reduce. The Hard Way

```
function (keys, values, rereduce) {  
  var ret = {sum: 0, count: 0};  
  if (rereduce) {  
    for (var i = 0; i < values.length; ++i) {  
      ret.sum += values[i].sum;  
      ret.count += values[i].count;  
    }  
  } else {  
    ret.sum = sum(values);  
    ret.count = values.length;  
  }  
  ret.avg = ret.sum / ret.count;  
  return ret;  
}
```

# Native Built-in Reducers

[http://www.couchbase.com/docs/couchbase-devguide-2.0/  
using-built-in-reducers.html](http://www.couchbase.com/docs/couchbase-devguide-2.0/using-built-in-reducers.html)

# Native Built-in Reducers

[http://www.couchbase.com/docs/couchbase-devguide-2.0/  
using-built-in-reducers.html](http://www.couchbase.com/docs/couchbase-devguide-2.0/using-built-in-reducers.html)

`_stats`

## Views: Reduce. Built-in \_stats

```
{  
  rows: [  
    {  
      key: "Belgian and French Ale",  
      value: {  
        sum: 1972.86,  
        count: 267,  
        min: 3.5,  
        max: 16,  
        sumsqr: 15772.540600000001  
      }  
    },  
    ....  
  ]  
}
```







## Views: Execution

```
Beer.avg_by_category.each do |beer|  
  avg = beer.value["sum"] / beer.value["count"]  
  printf("%s: %.2f\n", beer.key, avg)  
end
```

## Views: Execution

```
Beer.avg_by_category.each do |beer|  
  avg = beer.value["sum"] / beer.value["count"]  
  printf("%s: %.2f\n", beer.key, avg)  
end
```

Belgian and French Ale: 7.39

British Ale: 6.93

German Ale: 5.67

German Lager: 6.22

Irish Ale: 5.94

North American Ale: 6.79

North American Lager: 4.97

Other Lager: 4.55

Other Style: 5.82

## Views: Execution

```
Beer.avg_by_category.each do |beer|  
  avg = beer.value["sum"] / beer.value["count"]  
  printf("%s: %.2f\n", beer.key, avg)  
end
```

Belgian and French Ale: 7.39

British Ale: 6.93

German Ale: 5.67

German Lager: 6.22

Irish Ale: 5.94

North American Ale: 6.79

North American Lager: 4.97

Other Lager: 4.55

Other Style: 5.82

## Views: Execution

```
Beer.avg_by_category.each do |beer|  
  avg = beer.value["sum"] / beer.value["count"]  
  printf("%s: %.2f\n", beer.key, avg)  
end
```

Belgian and French Ale: 7.39

British Ale: 6.93

German Ale: 5.67

German Lager: 6.22

Irish Ale: 5.94

North American Ale: 6.79

North American Lager: 4.97

Other Lager: 4.55

Other Style: 5.82

# Views: Grouping

In this example I'll use compound key to demonstrate `group_level` option to the view.

## Views: Grouping

In this example I'll use compound key to demonstrate `group_level` option to the view.

```
function(doc, meta) {  
  if (doc.type == "brewery" && doc.country && doc.city) {  
    emit([doc.country, doc.city], 1);  
  }  
}
```



## Views: Grouping

In this example I'll use compound key to demonstrate `group_level` option to the view.

```
function(doc, meta) {  
  if (doc.type == "brewery" && doc.country && doc.city) {  
    emit([doc.country, doc.city], 1);  
  }  
}
```

`_count`

## Views: Grouping: group\_level = 2

```
# in this case group_level=2 behaves like group=true
Brewery.by_location(:group_level => 2).each do |doc|
  puts [doc.key, doc.value].inspect
end
```

## Views: Grouping: group\_level = 2

```
# in this case group_level=2 behaves like group=true
Brewery.by_location(:group_level => 2).each do |doc|
  puts [doc.key, doc.value].inspect
end
```

```
[["Argentina", "Mendoza"], 1]
[["Argentina", "San Martin"], 1]
[["Australia", "Adelaide"], 1]
[["Australia", "Camperdown"], 1]
[["Australia", "Canning Vale"], 1]
[["Belgium", "Blaugies"], 1]
[["Belgium", "Breendonk"], 2]
...
```

## Views: Grouping: group\_level = 2

```
# in this case group_level=2 behaves like group=true
Brewery.by_location(:group_level => 2).each do |doc|
  puts [doc.key, doc.value].inspect
end
```

```
[["Argentina", "Mendoza"], 1]
[["Argentina", "San Martin"], 1]
[["Australia", "Adelaide"], 1]
[["Australia", "Camperdown"], 1]
[["Australia", "Canning Vale"], 1]
[["Belgium", "Blaugies"], 1]
[["Belgium", "Breendonk"], 2]
...
```

## Views: Grouping: group\_level = 1

```
Brewery.by_location(:group_level => 1).each do |doc|  
  puts [doc.key, doc.value].inspect  
end
```

## Views: Grouping: group\_level = 1

```
Brewery.by_location(:group_level => 1).each do |doc|  
  puts [doc.key, doc.value].inspect  
end
```

```
[["Argentina"], 2]  
[["Australia"], 13]  
[["Austria"], 10]  
[["Belgium"], 98]  
[["Belize"], 1]  
[["Brazil"], 2]  
[["Canada"], 45]  
[["China"], 2]  
...
```

## Views: Grouping: group\_level = 1

```
Brewery.by_location(:group_level => 1).each do |doc|  
  puts [doc.key, doc.value].inspect  
end
```

```
[["Argentina"], 2]  
[["Australia"], 13]  
[["Austria"], 10]  
[["Belgium"], 98]  
[["Belize"], 1]  
[["Brazil"], 2]  
[["Canada"], 45]  
[["China"], 2]  
...
```

## Views: Grouping: group\_level = 0

```
# group_level=0 is default behaviour
Brewery.by_location(:group_level => 0).each do |doc|
  puts [doc.key, doc.value].inspect
end
```



## Views: Grouping: group\_level = 0

```
# group_level=0 is default behaviour
Brewery.by_location(:group_level => 0).each do |doc|
  puts [doc.key, doc.value].inspect
end
```

```
[nil, 1390]
```

## Views: Grouping: group\_level = 0

```
# group_level=0 is default behaviour
Brewery.by_location(:group_level => 0).each do |doc|
  puts [doc.key, doc.value].inspect
end
```

```
[nil, 1390]
```

- couchbase gem sources  
<https://github.com/couchbase/couchbase-ruby-client>
- couchbase issue tracker  
<http://couchbase.com/issues/browse/RCBC>
- couchbase-model gem sources  
<https://github.com/couchbase/couchbase-ruby-model>
- demo rails application  
<https://github.com/couchbaselabs/couchbase-beer.rb>

**And Even More**



# Query your geo data

[http://couchbase.com/docs/couchbase-manual-2.0/  
couchbase-views-writing-geo.html](http://couchbase.com/docs/couchbase-manual-2.0/couchbase-views-writing-geo.html)

```
function(doc, meta) {  
  if (doc.geo && doc.geo.lng && doc.geo.lat && doc.name) {  
    var key = {type: "Point",  
              coordinates: [doc.geo.lng, doc.geo.lat]};  
    var val = {name: doc.name, geo: doc.geo};  
    emit(key, val);  
  }  
}
```

# Rich queries with ElasticSearch

<http://couchbase.com/docs/couchbase-elasticsearch/>

Thank you

Questions?

