JSON

- JSON (JavaScript Object Notation - Notação de Objetos JavaScript) é uma formatação leve de troca de dados.
  - Para seres humanos, é fácil de ler e escrever.
  - Para máquinas, é fácil de interpretar e gerar.
- Constituído em duas estruturas:
  - Uma coleção de pares nome/valor.
  - Uma lista ordenada de valores.

Fonte: http://www.json.org

Exemplo JSON

```
{"widget": {
    "debug": "on",
    "window": {
        "title": "Sample Konfabulator Widget",
        "width": 500,
        "height": 500
    },
    "image": {
        "src": "Images/Sun.png",
        "hOffset": 250,
        "vOffset": 250,
        "alignment": "center"
    },
    "text": {
        "data": "Click Here",
        "size": 36,
        "style": "bold",
```

BSON

- Abreviação de Binary JSON, é a serialização binária de documentos JSON.
- Vantagens sobre JSON:
  - Tipos além do String
- byte, int32, int64, double, boolean, datetime, timestamp

Fonte: http://bson.org

Exemplo BSON

```
{"hello":
"world"}        →   "\x16\x00\x00\x00\x02hello\x00
                     \x06\x00\x00\x00world\x00\x00"

                    "\x31\x00\x00\x00\x04BSON\x00\x26\x00
{"BSON":             \x00\x00\x020\x00\x08\x00\x00
["awesome",     →    \x00awesome\x00\x011\x00\x33\x33\x33\x33\x33\x33
5.05, 1986]}         \x14\x40\x102\x00\xc2\x07\x00\x00
                     \x00\x00"
```

Modelo Documento

- Baseados em documentos (xml, json, bson...)
  - Não confundir com documentos de texto, planilhas, etc
- O documento é considerado um todo, evitando a divisão do dado.
- Forte modelo agregado.

Modelo Documento

JSON

```
{
    "firstname": "Pramod",
    "citiesvisited": [
        "Chicago",
        "London",
        "Pune",
        "Bangalore"
    ],
    "addresses": [
        {
            "state": "AK",
            "city": "DILLINGHAM",
            "type": "R"
        }, {
            "state": "MH",
            "city": "PUNE",
            "type": "R"
        }
    ],
    "lastcity": "Chicago"
}
```

XML

```
<person>
    <firstname>Pramod</firstname>
    <citiesvisited>
        <cityvisited>Chicago</cityvisited>
        <cityvisited>London</cityvisited>
        <cityvisited>Pune</cityvisited>
        <cityvisited>Bangalore</cityvisited>
    </citiesvisited>
    <addresses>
        <address>
            <state>AK</state>
            <city>DILLINGHAM</city>
            <type>R</type>
        </address>
        <address>
            <state>MH</state>
            <city>PUNE</city>
            <type>R</type>
        </address>
    </addresses>
    <lastcity>Chicago</lastcity>
</person>
```

Ranking Documento

31 systems in ranking, October 2014

Rank	Last Month	DBMS	Database Model	Score	Changes
1.	1.	MongoDB	Document store	240.41	-0.58
2.	2.	CouchDB	Document store	25.79	-0.18
3.	3.	Couchbase	Document store	20.00	+0.33
4.	4.	MarkLogic	Multi-model	7.87	+0.40
5.	5.	RavenDB	Document store	5.16	-0.20
6.	6.	Cloudant	Document store	2.34	+0.07
7.	7.	GemFire	Document store	2.13	+0.11
8.	8.	OrientDB	Multi-model	2.00	+0.07
9.	9.	RethinkDB	Document store	0.91	+0.09

Fonte: db-engines.com

MongoDB

- Nome derivado da expressão “hu**mongo**us“.
  - Imenso, gigantesco, monstruoso.
- Banco de dados *open-source,* modelo documento, multi-plataforma escrito em C++.
- Interpreta a linguagem JavaScript.
  - Utiliza o motor V8 da Google.

9

Componentes

- Server: **mongod**
  - Web console (porta 28017):
    --httpinterface
  - Suporte a REST:
    --rest
- *Shell*: **mongo**

10

Conceitos

- DB
  - Base de dados.
- Coleção
  - Coleção de documentos. Similar à “tabela”.
- Documento
  - Cada registro salvo.

11

Tipos de Dados

- String
- Numbers
  - double
  - int
  - long
- Datatime
- Custom

12

Operadores

Comparação
- $gt: maior que
- $gte: maior ou igual
- $in: dentro de um array
- $lt: menor que
- $lte: menor ou igual
- $ne: diferente
- $nin: não dentro do array
- $all: todos do array
- $type: o dado é do tipo

Lógicos
- $or: ou
- $and: e
- $not: negação
- $nor:

13

Mongo Console – Javascript

```
var x = 5;
print(x * 10);
for(i=0; i<10; i++) {
    print('hello');
};
var a = {age: 25};
var n = {name: 'Ed', languages:
['c', 'ruby', 'js']};
var student = {name: 'Jim', scores:
[75, 99, 87.2]};
```

14

Criando um documento

```
db.produtos.inse
rt({
      nome:"cami
sa",
      preco:10
});
```

```
db.produtos.inse
rt({
      nome:"calc
a",
      preco:20
});
```

```
db.produtos.insert({
   nome:"luva",
   tamanho:"p",
   preco:5.5
});
```

```
db.produtos.insert({
   nome: "bone",
   status: "esgotado"
});
```

15

Buscas

```
db.produtos.find();

db.produtos.find().sort({nome:1});

db.produtos.find({preco: 20});

db.produtos.find({preco:{$gt: 15}});

db.produtos.find({tamanho:{$in: ["p", "m"]}});

db.produtos.find({$or: [{tamanho: "p"}, {tamanho: "m"}]});

db.produtos.find({tamanho: {$exists: false}});
```

16

Busca com Cursor

```
var c = db.produtos.find();
while (c.hasNext()) {
  printjson(c.next());
}
```

17

Índices

- Um atributo ascendente:
  - `db.produtos.ensureIndex({nome: 1});`
- Um atributo descendente:
  - `db.produtos.ensureIndex({nome: -1});`
- Um atributo único:
  - `db.produtos.ensureIndex({nome: 1}, {unique: true});`
- Remove
  - `db.produtos.dropIndex({nome: 1});`

18

Alterando um documento

```
db.produtos.update(
     {nome: "camisa}, // qual
     {$set: {preco: 12}} // o que
);

db.produtos.update({
     {nome: "calca},
     {$inc: {preco: 5}}
);
```

19

Removendo um documento

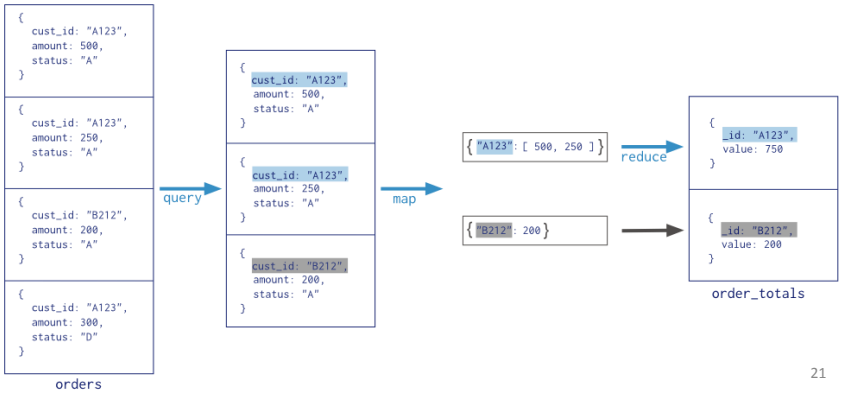
```
db.produtos.remove({nome: "luva"});

// Remove apenas 1
db.produtos.remove({nome: "bone"}, true);
```

20

Map-Reduce

Collection

```
db.orders.mapReduce(
    map       →  function() { emit( this.cust_id, this.amount ); },
    reduce    →  function(key, values) { return Array.sum( values ) },
    query     →  {
    output    →    query: { status: "A" },
                   out: "order_totals"
                 }
               )
```

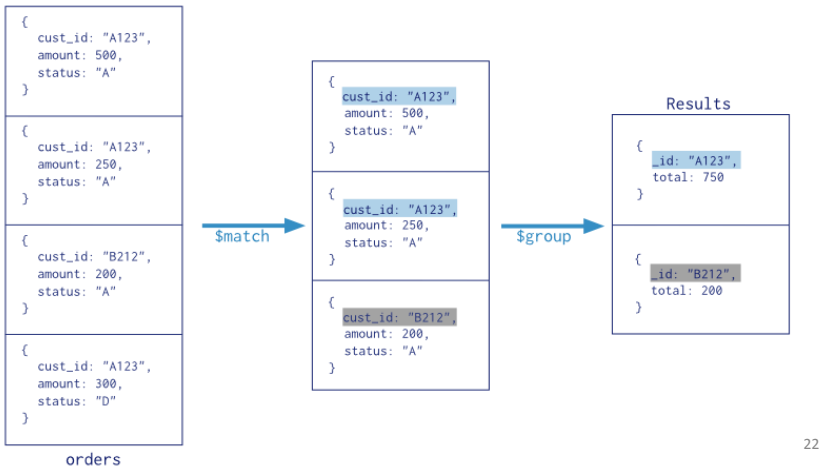


21

Aggregate

Collection

```
db.orders.aggregate( [
    $match phase  →  { $match: { status: "A" } },
    $group phase  →  { $group: { _id: "$cust_id",total: { $sum: "$amount" } } }
                  ] )
```



22

Java com MongoDB

- Necessário uso do driver
  - mongo-java-driver-<version>.jar
- Não é um driver JDBC

23

Java com MongoDB

- Conecta a um server:

```
MongoClient mongoClient = new
MongoClient("<host>");
```

- Usa uma base:

```
DB db = mongoClient.getDB("mydb");
```

- Busca um ou mais coleções:

```
Set<String> colls =
db.getCollectionNames();
DBCollection coll =
db.getCollection("testCollection");
```

24

Incluindo um documento

```
DBObject data = new BasicDBObject();
data.put(“key1”, “value”); // string value
data.put(“key2”, 10); // int value
data.put(“key3”, 10.5); // double value
data.put(“key4”, new Date()); // datatime value
data.put(“key5”, true); // boolean value
coll.insert(data);
```

25

Busca um documento

```
BasicDBObject query = new BasicDBObject("i", 71);

cursor = coll.find(query);

try {
   while(cursor.hasNext()) {
       System.out.println(cursor.next());
    }
} finally {
   cursor.close();
}
```

26

Busca com Cursor

```
DBCursor cursor = coll.find();
try {
   while(cursor.hasNext()) {
System.out.println(cursor.next());
    }
} finally {
   cursor.close();
}
```

27