UniRitter
Laureate International Universities®

Banco de Dados NoSQL

(NS0301)

Prof. Giovane Barcelos
giovane_barcelos@uniritter.edu.br

O que vamos aprender?

- Importância e características chaves do MongoDB
- Instalação
- *JSON* / BSON
- *CRUD* básico
- Pesquisas (*queries*)
- Agregação / *Map Reduce*
- Replicação / *Sharding*
- Expressão Regular *(RegEX)*
- Índices *(Index)*
- Validações
- Aplicação Prática com *NodeJS* e *Mongoose*
- Técnicas de Modelagem

Quem é o MongoDB? (*mongodb.com*)

- Banco de dados *NoSQL* do tipo documento mais utilizado
- *Startup* de *Big Data* que vale mais de 1.6B - http://bit.do/MongoValued
- Preferido dos *clouds* (*digitalocean, aws, compose, rosehosting, cloud mongo, objectrocket, mlab*, etc)
- Usado em muitos *Fullstacks* (mean.io, mern.io, etc)
- Número 1 entre os *NoSQL* - https://db-engines.com/en/ranking
- Possui oferta e procura constante por profissionais - https://www.indeed.com/jobtrends/q-mongodb.html
- Esta inserido em um mercado crescente - https://www.alliedmarketresearch.com/NoSQL-market?NoSQL-market
- *Benchmarkings* apresentam *MongoDB* com melhor performance que o *RDBMS* em geral entre 2 a 10x

Como vamos trabalhar com MongoDB?

- **Linha de comando do Mongo é muito desafiante**
- **Gerenciamento: MongoBooster, Robo3T (*open source*), NoSQL Manager e Studio3T**
- **Modelagem: DbSchema, Hackolad, Er Studio, ErWin, AquaData e KDM DataView**
- **Modelagem SQL: Sql Power Designer, Toad, entre outros**

  http://bit.do/ModelTool
- **Modelagem UML: Draw.io, LucidChart, GenMyModel, Umbrello, Papyrus, ArgoUml, UmlLet, PlantUml, UML Designer, Modelio, entre outros**

  http://bit.do/UmlTool
- **Backup e Restauração**

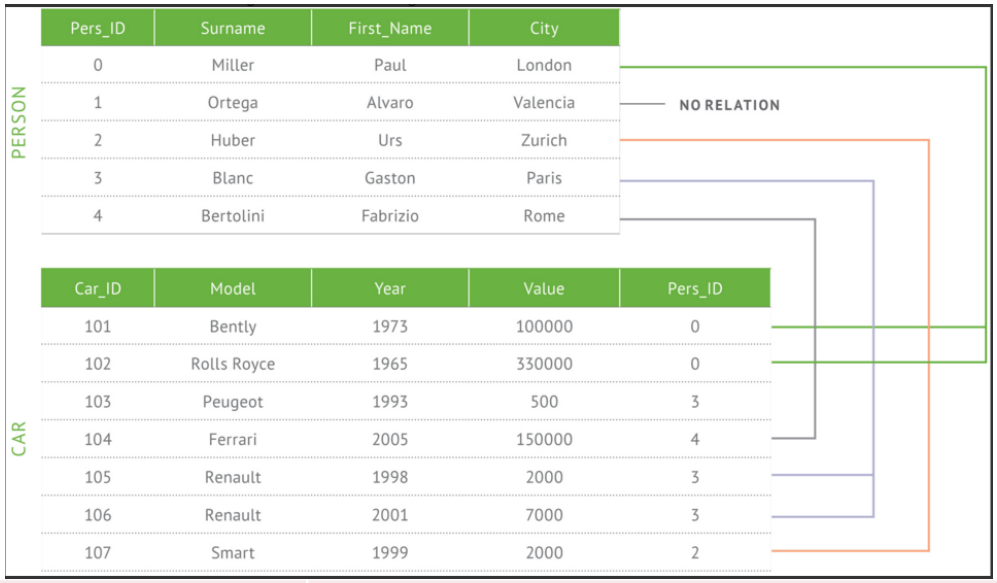
  mongodump, mongorestore, mongoexport* e *mongoimport

Características Técnicas do *MongoDB*

- Usa ***JSON** (JavaScript Object Notation)*
  http://json.org/ e http://jsoneditoronline.org/
- Grava no formato ***BSON** (Binary JSON)*
  https://json-bson-converter.appspot.com/
- **Não** tem **Esquema**
- **Não** possui **transações** (***MVCC** – Multiversion concurrency control*)
- Índice ***BTree*** do mesmo tipo do ***RDBMS***
- Possui vários conectores:
  R Project
  Hadoop
  Apache Spark
  ...

Como o *MongoDB* se compara a um *RDBMS?*

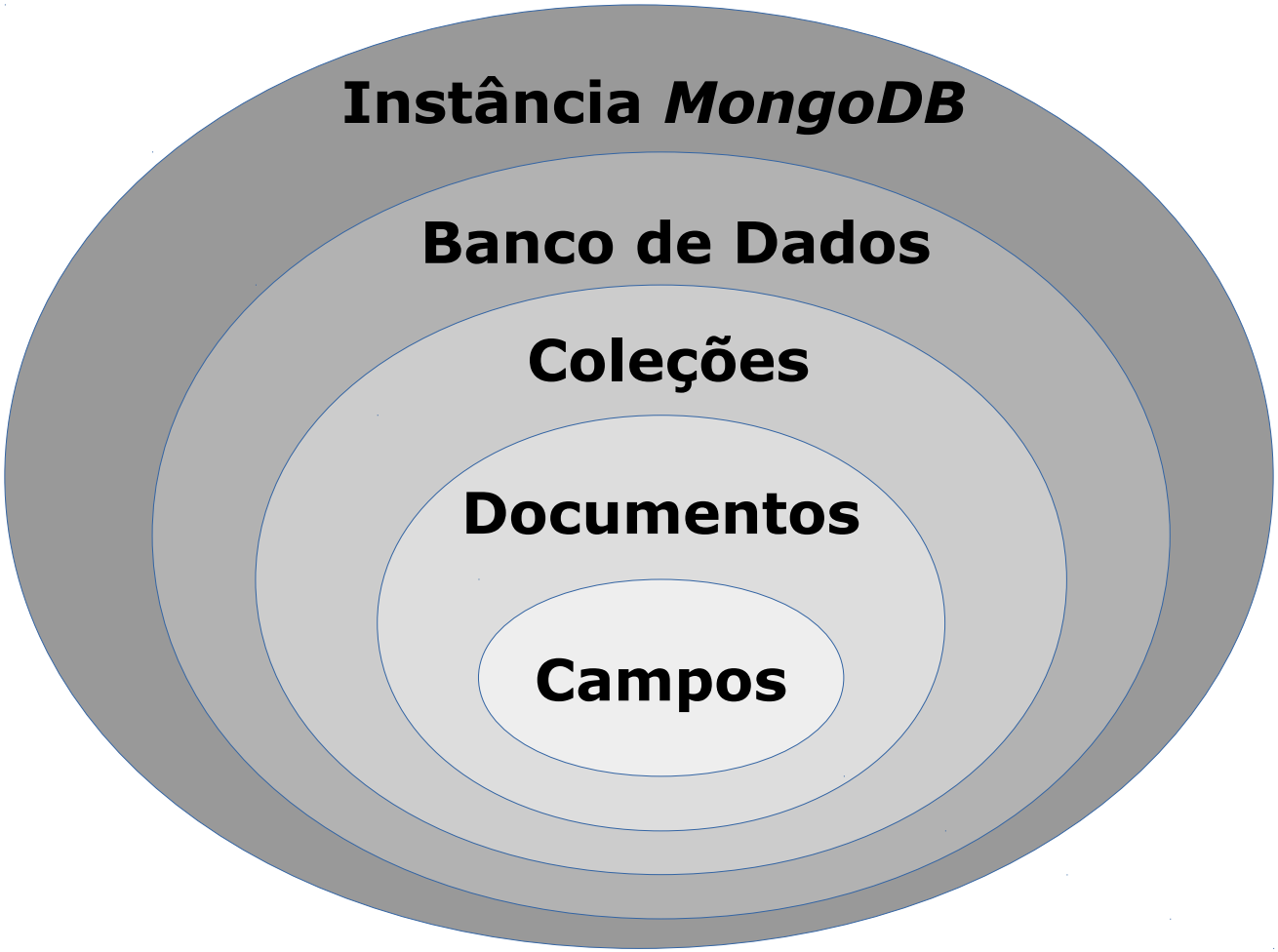
RDBMS	*MongoDB*
Banco de Dados	Banco de Dados
Tabela	Coleção
Linha	Documento (*JSON*/*BSON*)
Coluna	Campo
Índice	Índice
Junção (Join)	Documento Embutido
Chave Estrangeira	Referência
Particionamento	*Shard*




```
{ first_name: “Paul”,
  surname: “Miller”,
  city: “London”,
  location: [45.123,47.232],
  cars: [
    { model: “Bentley”,
      year: 1973,
      value: 100000, ….},
      { model: “Rolls Royce”,
        year: 1965,
        value: 330000, ….} ]
}
```

http://bit.do/RDBMSxMongo

Estrutura do *MongoDB*



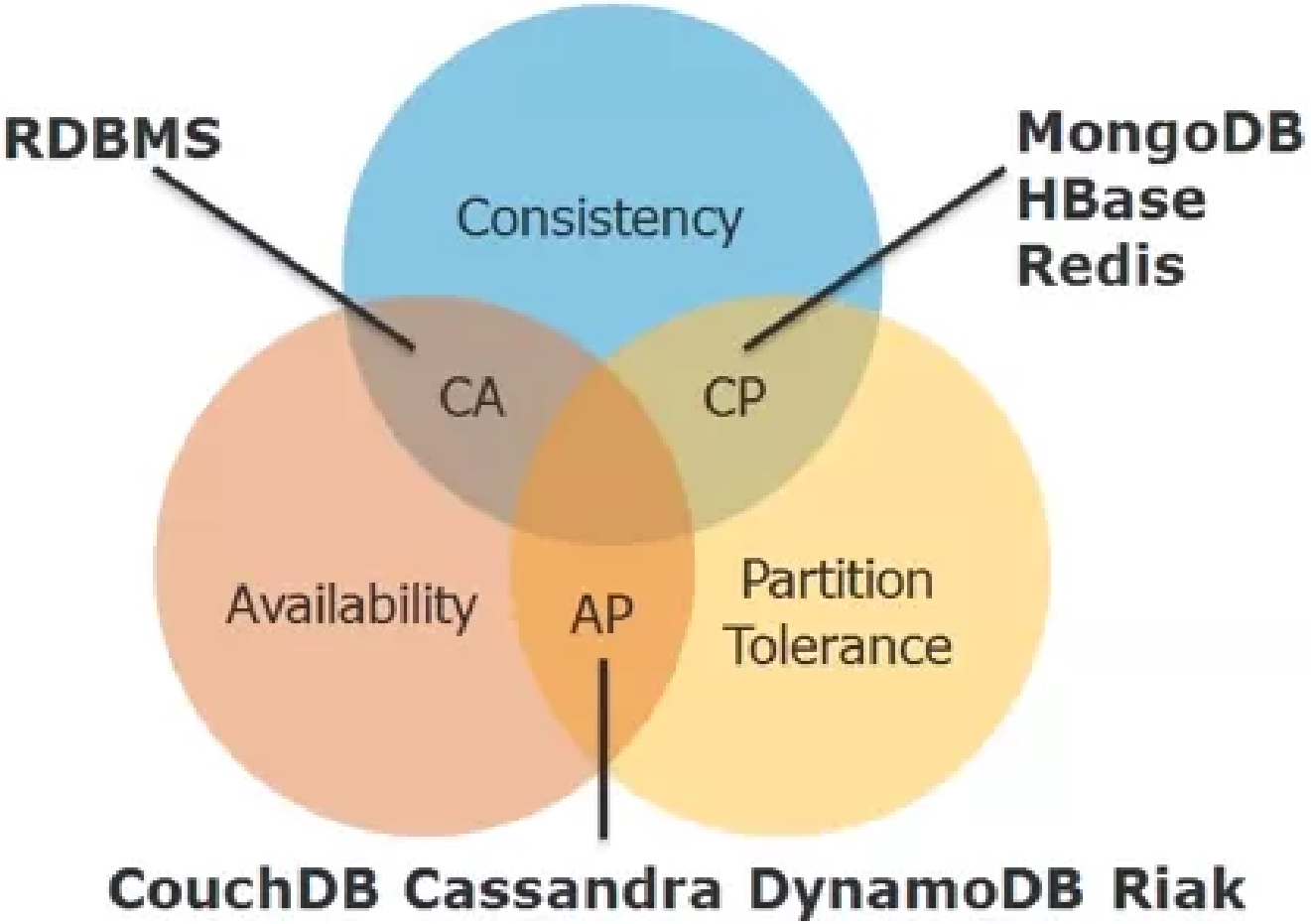
Outras Características Técnicas do MongoDB

- **Todos os documentos tem uma chave primária _id do tipo *ObjectId* automática por hash de 12 bytes hexa que evitam colisões**
  - ✔ **4B Unix Epoch + 3B Id Máquina + 2B Id Processo + 3B Contador**
- ***MongoDB* possui limitações**
  http://bit.do/MongoLimits
- **Possibilidade de Agregações e MapReduce**
  http://bit.do/MongoAgregacao

Quais são os teoremas/propriedades dos bancos?

- *ACID (MongoDB* somente no nível de documento*)*
  - ✔ Atomicidade (tudo ou nada)
  - ✔ Consistência (estado válido/consistente)
  - ✔ Isolamento (sem interferência)
  - ✔ Durabilidade (salvou, não perde mais)
- *CAP (MongoDB* garante CP*)*
  - Consistência (estado válido/consistente)
  - Availability / Disponibilidade (sempre disponível)
  - Particionamento Tolerante (falhou nodo, continua)
- *BASE (MongoDB* é B*)*
  - *Basically Available* (responde com espera)
  - *Soft State* (não é 100% consistente/válido)
  - *Eventually Consistent* (cedo ou tarde)

Do ponto de vista do CAP



http://www.w3resource.com/mongodb/nosql.php

Quais são os Prós e Contras do *MongoDB*?

Prós	
Alta performance	Alta disponibilidade (replicação)
Alta escalabilidade (*sharding*)	Dinâmico (esquema livre)
Dados heterogêneos	Facilita documentos embutidos
Representação JSON/BSON	Fácil de integrar (Ex: Hadoop)
Contras	
Não é ACID	Transações são complexas
Sem *stored procedure*	Ocupa mais espaço em disco
Relacionamentos múltiplos	Sem Joins

Onde usar?

Usar	
Catálogo de produtos *e-commerce*	Blogs e CMS
Análise e log em tempo real, *caching*, e alta escalabilidade	Gerenciador de configuração
Dados Geoespaciais	Mobilidade e Redes sociais*
Requisitos instáveis e com baixo acoplamento	Dados em evolução
Não Usar	
Sistemas altamente transacionais (Ex: *ERP*)	Sistemas fortemente acoplados
Muitos relacionamentos	Alta segurança de autorização

Quais são as operações de *CRUD*?

- ***Create*** **(Criar)**
  - **db.collection.insert(<document>)**
  - **db.collection.save(<document>)**
  - **db.collection.update(<query>, <update>, { upsert: true })**
- ***Read*** **(Ler)**
  - **db.collection.find(<query>, <projection>)**
  - **db.collection.findOne(<query>, <projection>)**
- ***Update*** **(Modificar)**
  - **db.collection.update(<query>, <update>, <options>)**
- ***Delete*** **(Excluir)**
  - **db.collection.remove(<query>, <justOne>)**
- **Operações CRUD do MongoDB:**

  https://docs.mongodb.com/manual/crud/
  https://docs.mongodb.com/manual/reference/operator/update/

Exemplo de *CRUD*

- db.aluno.insertMany(
  [{'matricula': '20170701',
  'nome': 'Penelope Charmosa' },
  { 'matricula': '20170801',
  'nome': 'Dick Vigarista'}]);
- db.aluno.find({'nome': /.*Vigarista.*/});
- db.aluno.update({'matricula': '20170801'},
  {$set: {'sexo': 'M'}});
- db.aluno.find({});
- db.aluno.remove({'matricula': '20170801'});
- db.aluno.find({});
- Operadores *Update*:
  https://docs.mongodb.com/manual/reference/operator/update/

Exemplo de Pesquisas

- db.aluno.find({$and:[{'matricula': /.*2017.*/},
  {'nome': /.*Char.*/ }]})

- Operadores de pesquisa *(query)*:
  https://docs.mongodb.com/manual/reference/operator/query/

- Aprendizado e Teste de expressão regular:
  https://regex101.com e http://regexr.com/

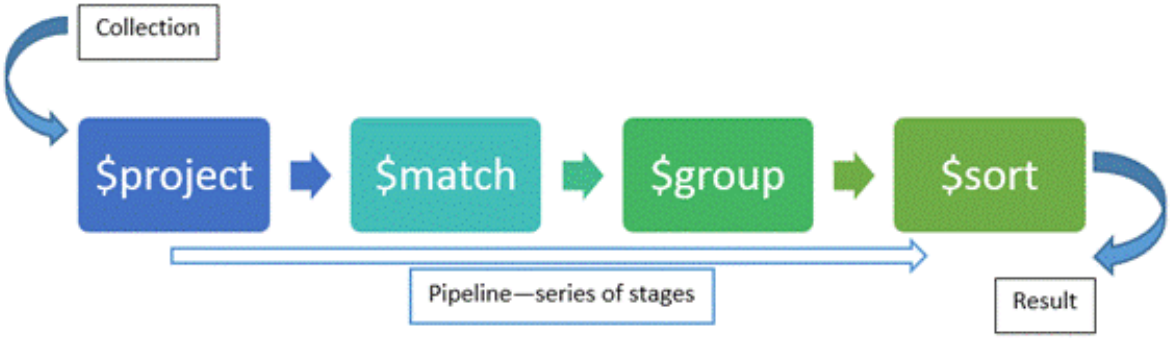
- Expressão Regular (ER) é utilizada para validar, verificar e limpar dados

CRUD SQL no *MongoDB*

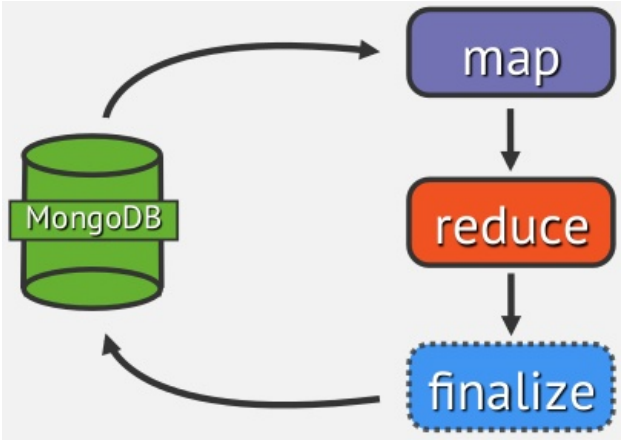
- ***CRUD*** **possui** **equivalências** **com** ***SQL***
  http://bit.do/MongoDBxSQL

O que é Agregação e MapReduce?

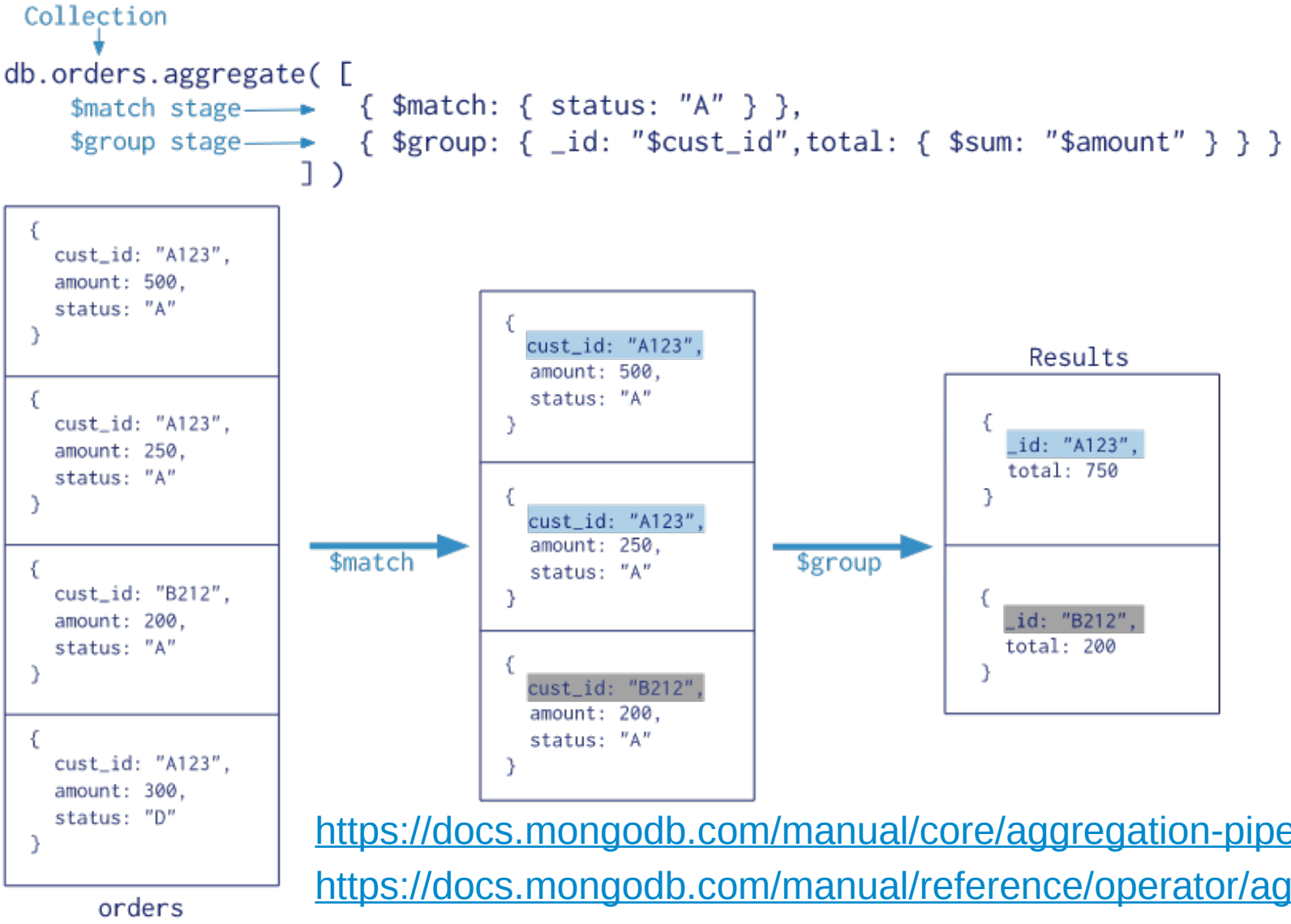
- **Formas de gerar tarefas que realizam média, somas, agrupamentos e remodelagens**
- **Pipeline de Agregação:**



- **Processo do *MapReduce*:**



Exemplo de Agregação

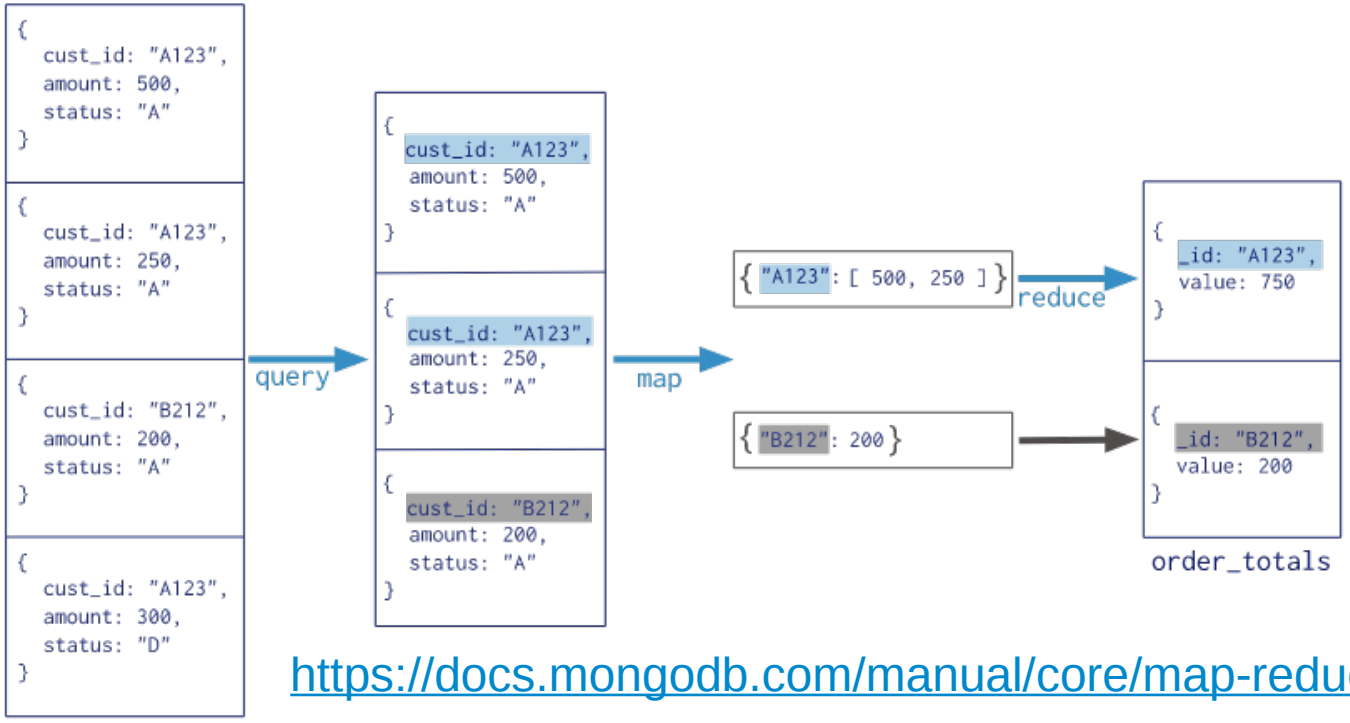


https://docs.mongodb.com/manual/core/aggregation-pipeline/

https://docs.mongodb.com/manual/reference/operator/aggregation/

Exemplo de *MapReduce*

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



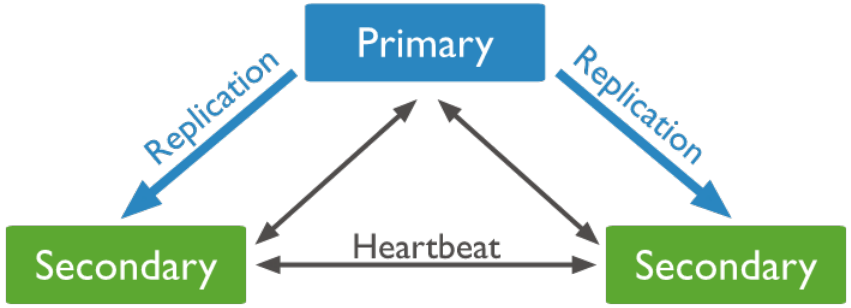
https://docs.mongodb.com/manual/core/map-reduce/

Laboratório do MongoDB

http://bit.do/Lab0301MongoDB

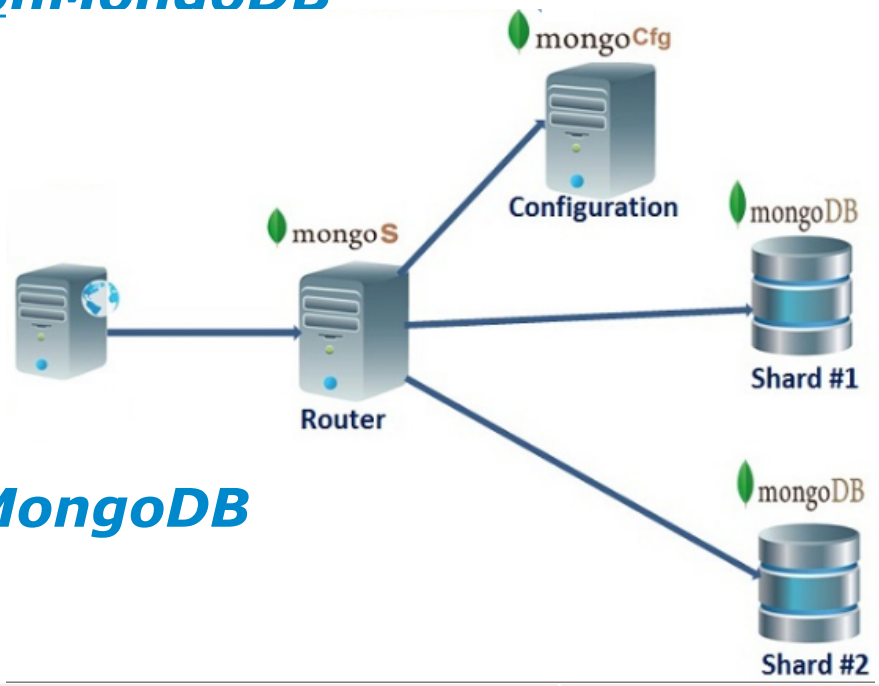
O que é *Replication* e *Sharding*?

- ***Replica***: é uma **replicação *master/slave*** do ***MongoDb*** que mantém exatamente os mesmos dados em diferentes instâncias do servidor



http://bit.do/SetupReplicationMongoDB

- ***Sharding***: método de **distribuição** dos dados em diferentes instâncias do servidor baseado em uma ***shard key*** que tem o limite de **512B**



http://bit.do/SetupShardingMongoDB

Como funcionam os Índices?

- *MongoDB* implementa índice do tipo *Btree+*
- Um índice de chave primária *(PK)* é criado automaticamente para o campo _id
- É possível criar outros índices inclusive únicos
- Suporta índices compostos que podem ser ordenados
- Exemplo:

```
db.aluno.insert([{ 'matricula': '0001',
                   'nome': 'Dick Vigarista', 'curso': 1},
                 { 'matricula': '0002',
                   'nome': 'Penelope Charmosa', 'curso': 2},
                 { 'matricula': '0003',
                   'nome': 'Medinho', 'curso': 2}]);
// Curso e nome ascendentes (1)
db.aluno.createIndex({curso: 1, nome: 1})
// Curso descendente (-1)
db.aluno.createIndex({curso: -1})
```

Como funciona a validação um documento?

- *MongoDB* 3.2+ valida documentos durante a inserção e atualização com a opção *validator*
- Pode ser utilizado qualquer operador de query, com a exceção $near, $nearSphere, $text, e $where.
- Exemplo:

```
db.createCollection("contato",
   { validator: { $or: [
      { fone:  { $type: "string"}},
      { email: { $regex: /@uniritter\.edu\.br$/}},
      { status: {$in: ["Desconhecido", "Incompleto"]}} ] }});

db.getCollectionInfos({'name': 'contato'});

db.contato.insert({nome: "Muttley", status: "Atualizado"});
```

E a segurança do *MongoDb*?

Item	Características
Autenticação	SHA-SCRAM Challenge / Response – padrão Certificados x.509 LDAP* e Kerberos*
Autorização	Usuários e Roles para ações e recursos Regras podem ser nível de campo
Auditoria	Log de Auditoria* (*DML* e *DDL*)
Criptografia	Rede: SSL/TLS (com FIPS 140-2*) Disco: motor de criptografia* (3.2+)

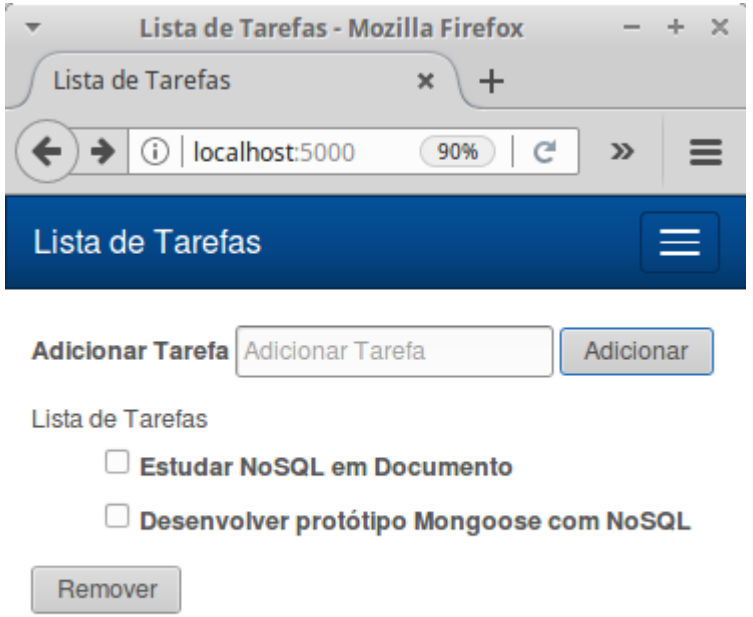
- **Exemplos de Comandos:** db.createUser, db.dropUser, db.createRole, db.grantRole, db.revokeRolesFromUser, db.changeUserPassword, etc

ORM (Object-Relational Mapping) MongoDB

- ***Mongoose*** (http://mongoosejs.com/) **mais utilizado**
- **Operações de modelagem, validação e *CRUD* completa**
- **Aplicação exemplo no Linux:**

```
# git clone https://github.com/giovanebarcelos/tarefasmongo
# cd tarefasmongo
# npm install
# node app.js
```

No navegador: http://localhost:5000

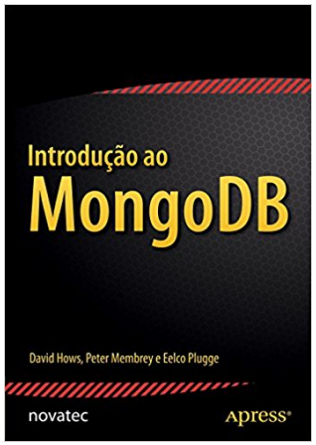


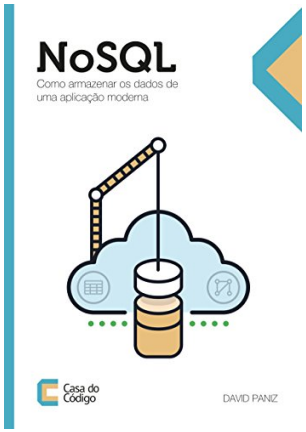
Como modelar utilizando *MongoDB*?

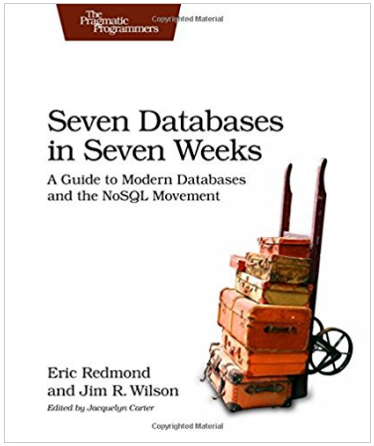
- A **desnormalização** e **agregação** são as **regra**
- Utilize alguma ferramenta de modelagem de classes *UML* ou *DbSchema*, *Hackolad*, *Er Studio*, *ErWin*, ...
- **Evite** modelar com **Joins**, ou seja, deixe o máximo possível das **coleções** de forma **embutida**
- Se for necessário **transações** que envolvam **mais** de uma coleção existe **padrão** definido para **implementação**: http://bit.do/Pattern2PhaseMongo e http://bit.do/Pattern2PhaseMongo1
- **Artigo** com as principais **técnicas** de **modelagem** NoSQL: http://bit.do/NoSQLTecnicasModelagem

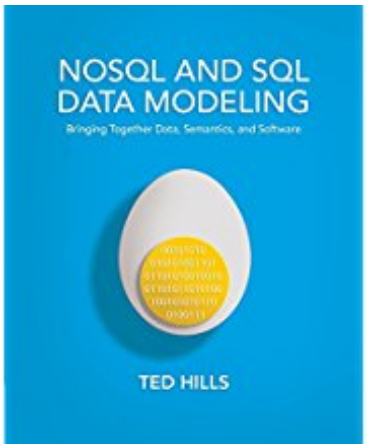
Dicas, Livros e Informações ...

- https://docs.mongodb.com/
- https://university.mongodb.com
- https://www.tutorialspoint.com/mongodb/
- http://www.graphdatamodeling.com/
- https://www.mongodb.com/blog/









Lembre-se

" No fim tudo dá certo, e se não deu certo é porque ainda não chegou ao fim. "

Fernando Sabino