



UNIVERSIDADE FEDERAL DE ITAJUBÁ

Banco de Dados II

COM 231

NoSQL

Vanessa Cristina Oliveira de Souza



UNIVERSIDADE FEDERAL DE ITAJUBÁ

NOSQL



NoSQL

■ Principais Características:

- Não utilizam o modelo relacional e nem a linguagem SQL;
- são projetados para rodar em *clusters* (sistemas distribuídos);
- tendem a ser *open source*;
- não possuem um esquema fixo, permitindo a persistência em qualquer registro
 - Não é SEM ESQUEMA
 - É ESQUEMA DINÂMICO



NoSQL

Persistência Poliglota

Utilizar diferentes armazenamentos de dados em diferentes circunstâncias.

Em vez de escolher o banco de dados relacional mais utilizado por todos, precisamos entender a natureza dos dados que estamos armazenando e como queremos manipulá-los.



NoSQL

- Um case real interessante é o da Netflix que utiliza 3 bancos NoSQL:
- SimpleDB
 - é uma implementação NoSQL chave-valor da Amazon, e por isso foi escolhido, já que a empresa utiliza o serviço AWS da Amazon.
- O HBase
 - é profundamente ligado ao Hadoop e pode escrever consultas em tempo real, apesar de sacrificar um pouco a disponibilidade pela consistência dos dados.
- Cassandra
 - foi a escolhida pela escalabilidade e pelo poder de replicar dados assincronamente através de múltiplas regiões geográficas. Pode escalar dinamicamente adicionando mais servidores sem a necessidade de re-shard ou reiniciar.



NoSQL

BASE – ACID alternative

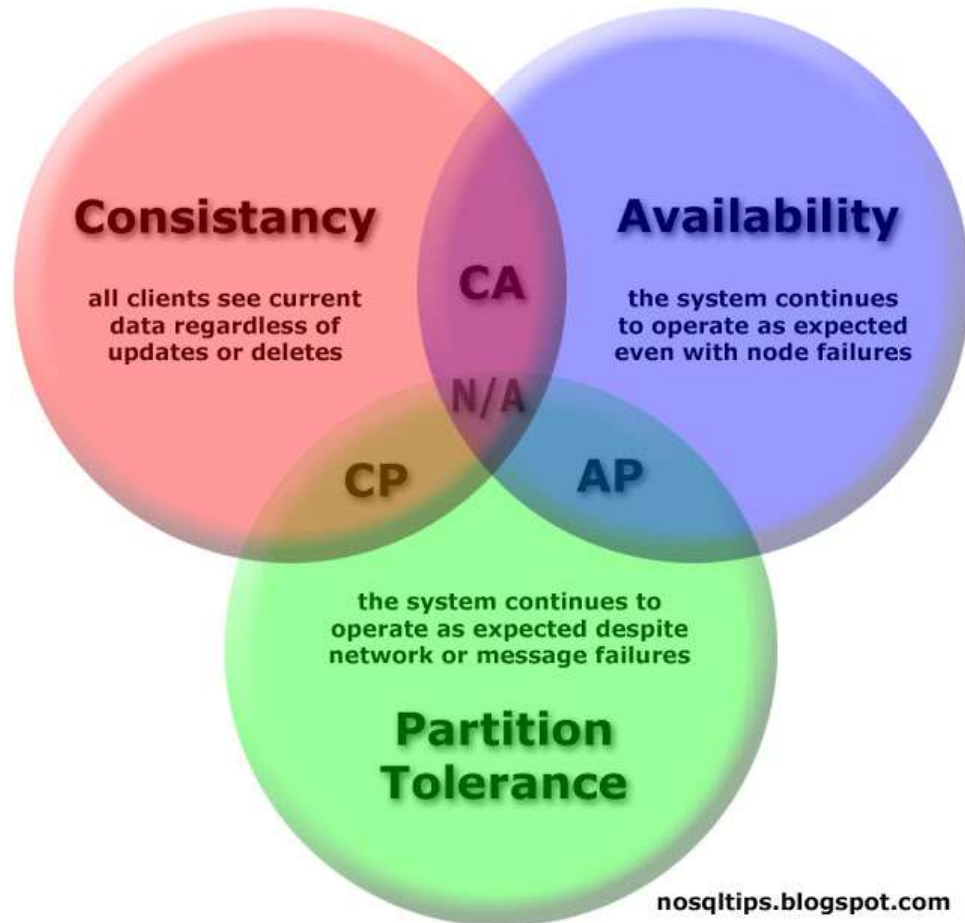
- **B**asically **a**vailable: Nodes in the a distributed environment can go down, but the whole system shouldn't be affected.
- **S**oft State (scalable): The state of the system and data changes over time, even without input. This is because of the eventual consistency model.
- **E**ventual Consistency: Given enough time, data will be consistent across the distributed system.



NoSQL

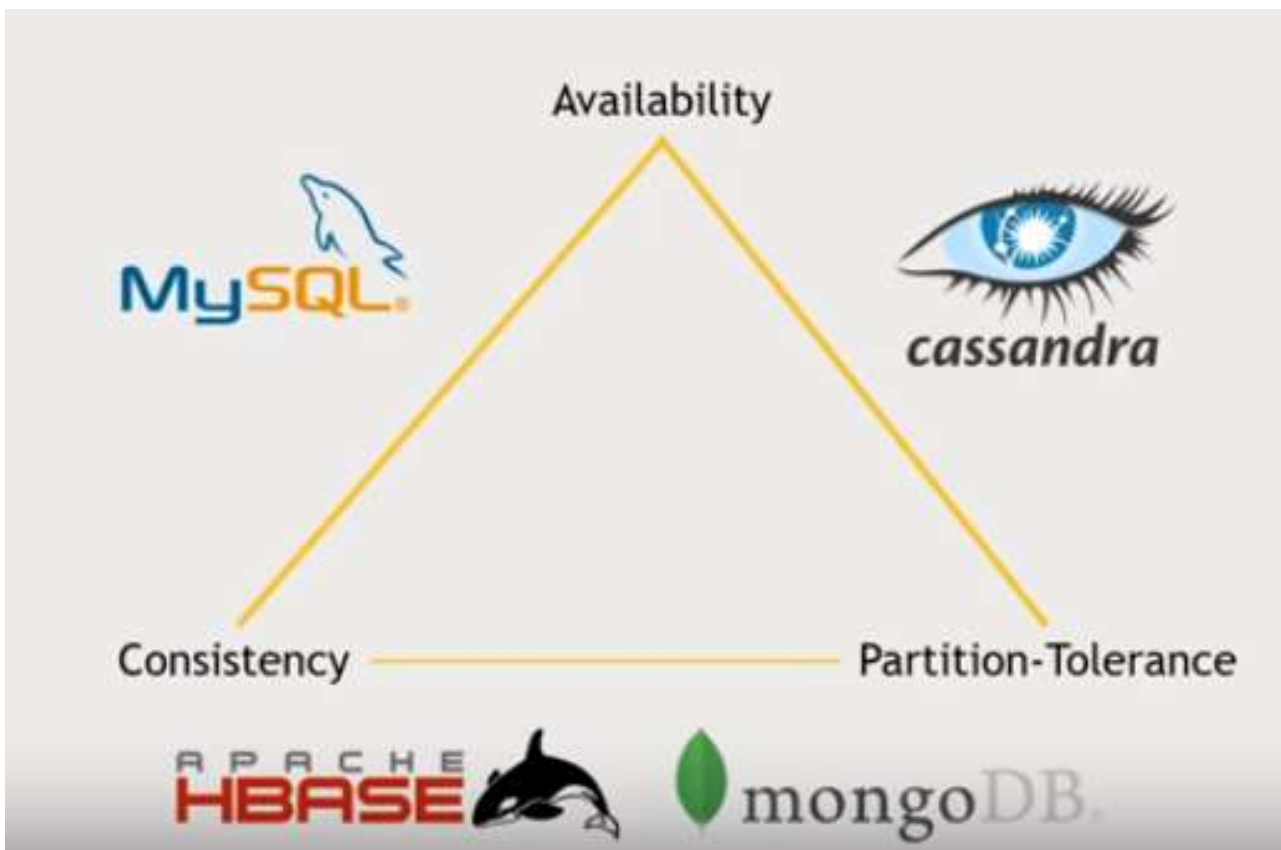
■ Teorema CAP

□ Brewer 2000





Modelos de Distribuição

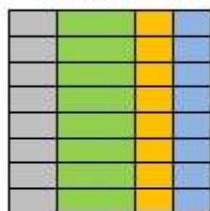




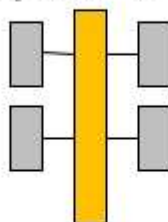
Modelos de Datos

After NoSQL

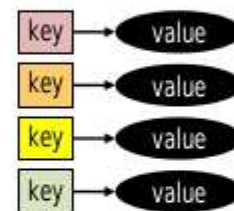
Relational



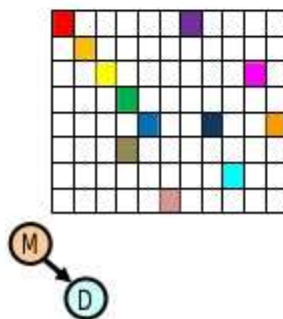
Analytical (OLAP)



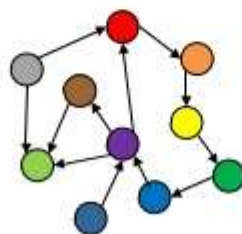
Key-Value



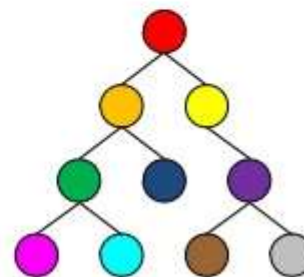
Column-Family



Graph

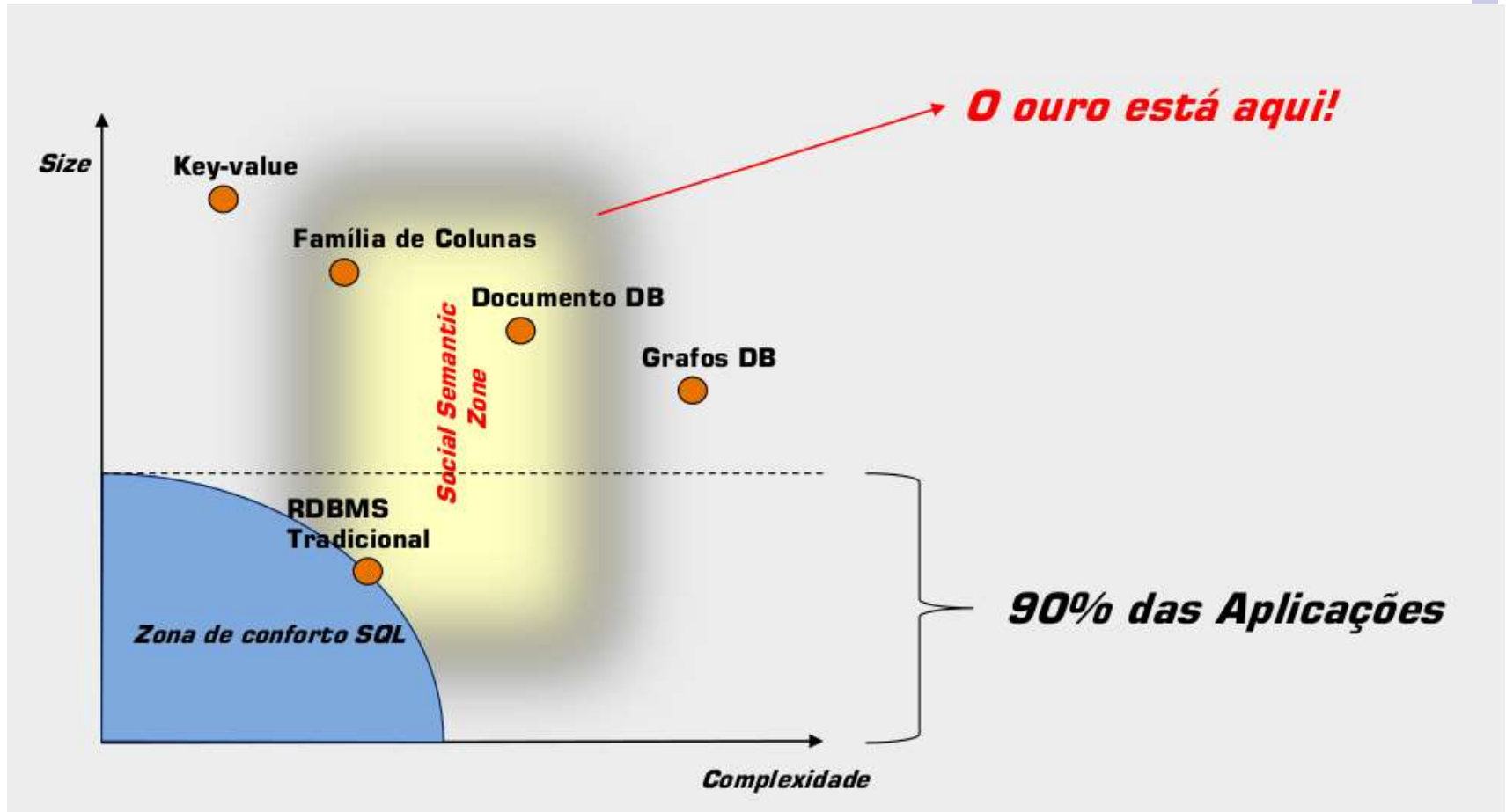


Document





Modelos NoSQL





UNIVERSIDADE FEDERAL DE ITAJUBÁ

ARQUITETURA EM SISTEMAS NOSQL



Modelos de Distribuição

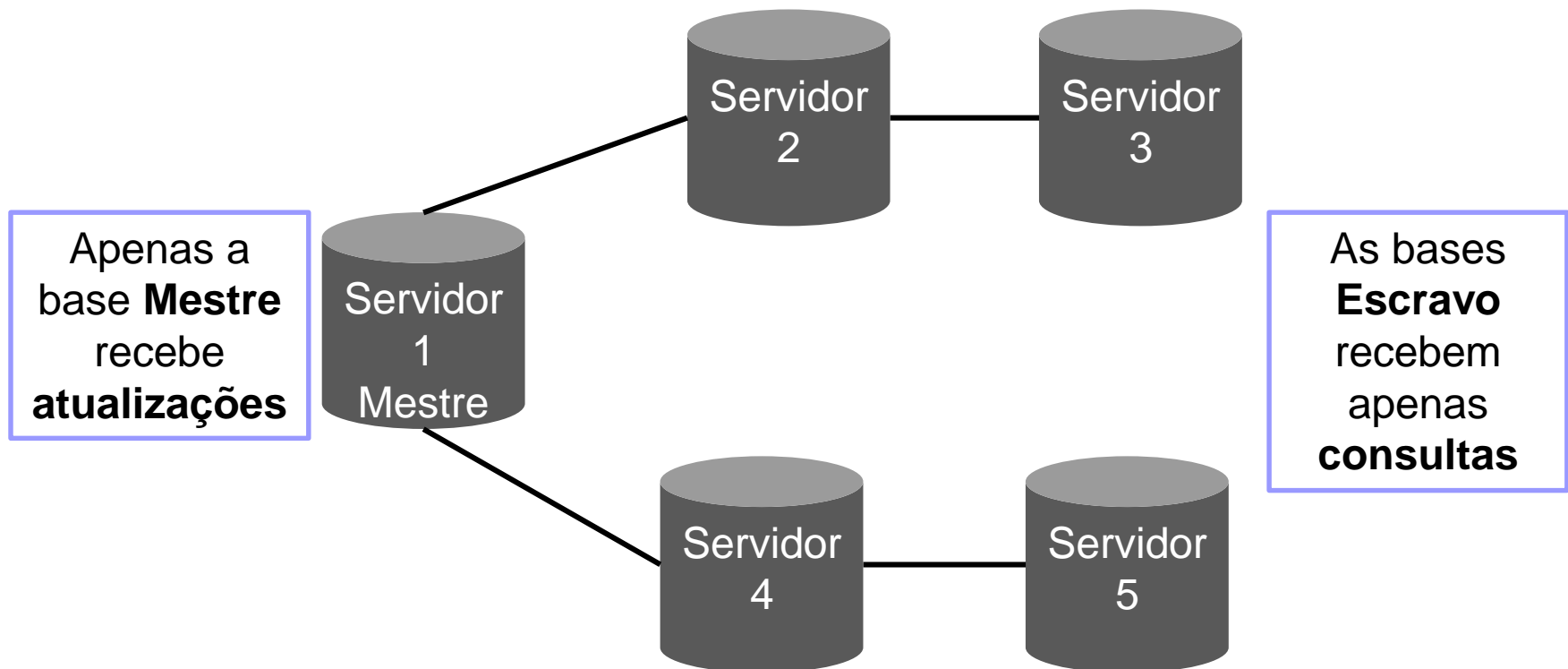
- Há dois caminhos para serem seguidos na distribuição e escalabilidade de dados:
 - Replicação
 - Mesmos dados em nós diferentes
 - Fragmentação
 - Dados diferentes em nós diferentes
 - Técnicas Ortogonais
 - Pode-se usar uma delas ou ambas



Modelos de Distribuição

■ Replicação

□ Mestre-Escravo

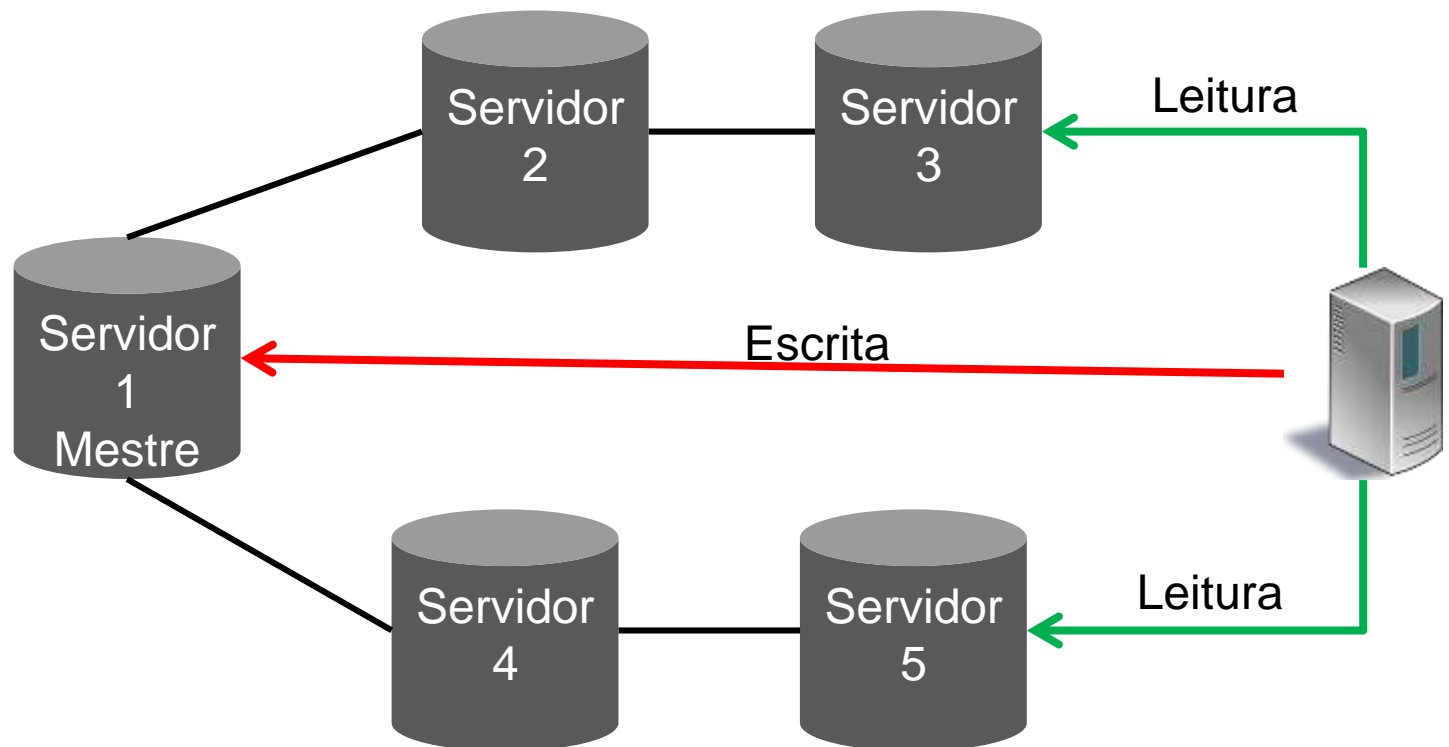




Modelos de Distribuição

■ Replicação

□ Mestre-Escravo

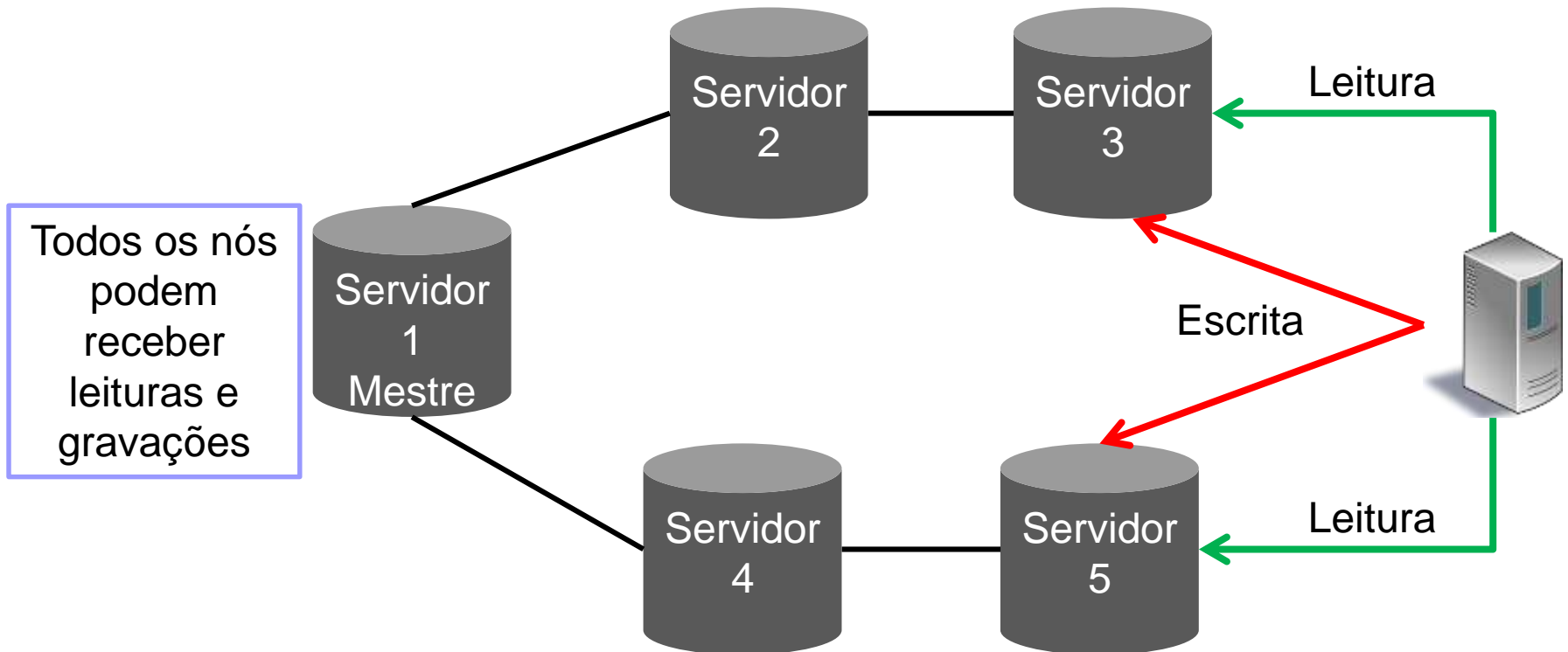




Modelos de Distribuição

■ Replicação

□ Ponto a Ponto





Modelos de Distribuição

■ Replicação

□ Mestre-Escravo

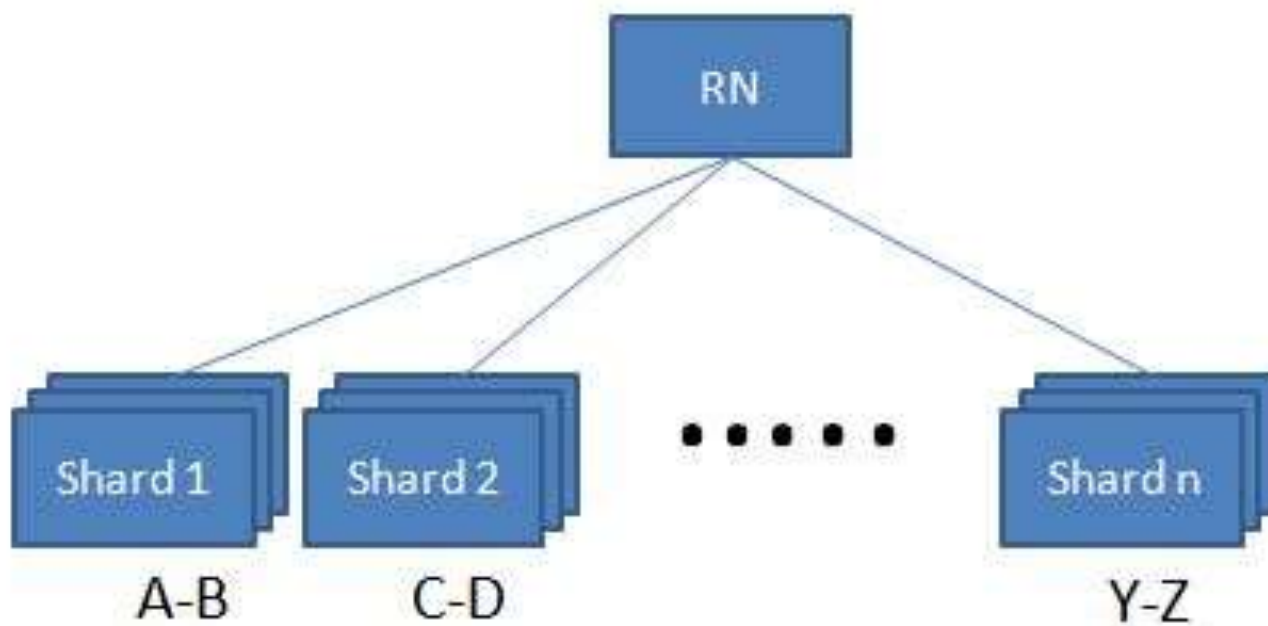
- Mais útil para a escalabilidade quando há muitas leituras
- Sistema limitado pela capacidade do mestre em processar as atualizações e sincronizá-las
- Reduz a chance de conflitos de atualização (consistência)

□ Ponto a Ponto

- Evita carregar todas as gravações em um único ponto de falha
- Gravações inconsistentes são eternas



Fragmentação





Fragmentação

- A fragmentação é valiosa para a performance porque pode melhorar o desempenho de leitura e gravação.
 - Permite a escalabilidade horizontal de gravação de dados
 - Embora os dados estejam em nós diferentes, uma falha em um nó torna indisponíveis os dados do fragmento, assim como acontece em um único servidor



Replicação + Fragmentação

- Replicação Mestre-Escravo + Fragmentação
 - Cada registro tem seu mestre e seus escravos



UNIVERSIDADE FEDERAL DE ITAJUBÁ

MODELO ORIENTADO A DOCUMENTOS MONGODB



Modelo Orientado a Documentos

- Não há registros, mas documentos
 - XML, JSON, BSON
- Não há relacionamentos
 - A redundância é até incentivada

*Um documento deve ser auto-contido e conter **todas** as informações de que necessita. Por que isto? Simples: porque assim você ao invés de fazer uma consulta com vários joins, como é o caso do modelo relacional, você executa uma única consulta, que já te retorna o documento **inteiro**.*

Resultado: mais performance.



mongoDB

- Banco NoSQL, orientado a documentos
- É o mais famoso e mais usado banco de dados NoSQL
 - Facilidade de instalação
 - Ótima documentação
 - Diversos drivers para inúmeras linguagens de programação
 - Escalável
 - Esquema dinâmico
 - Alto desempenho
 - Código aberto escrito em C++
 - <https://www.mongodb.org/>



mongoDB

- Documentos são agrupados em **coleções** e um conjunto de coleções forma um **banco de dados**.

Relacional	MongoDB
Banco de Dados	Banco de Dados
Tabelas	Coleções
Registros	Documentos no formato JSON



{JSON}

■ JavaScript Object Notation

- Notação de Objetos JavaScript
- Douglas Crockford

■ Troca de dados

- Amigável aos seres humanos
- Fácil interpretação e geração para as máquinas

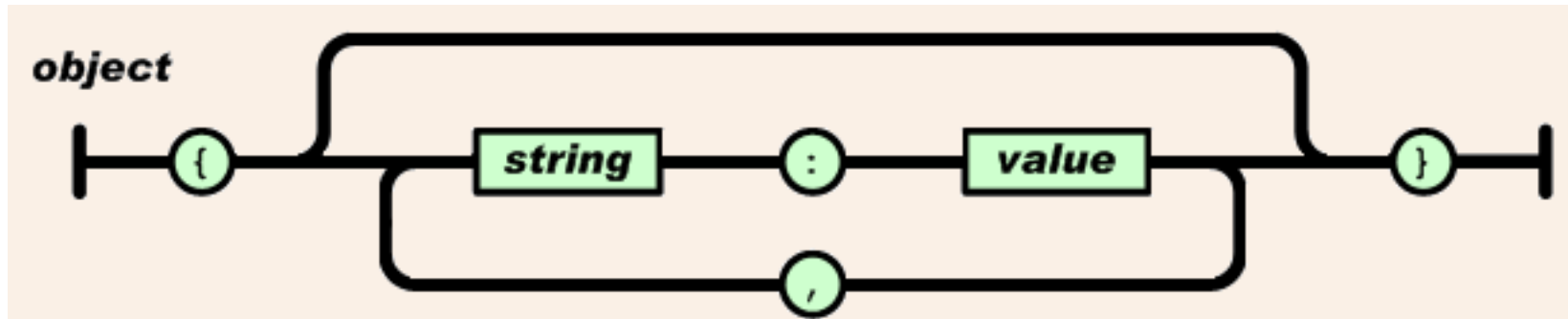
■ JSON possui duas estruturas:

- Uma coleção de pares nome/valor.
- Uma lista ordenada de valores.



{ JSON }

- Em JSON, os dados são apresentados desta forma:
 - Um objeto é um conjunto desordenado de pares nome/valor.
 - Um objeto começa com { (chave de abertura) e termina com } (chave de fechamento).
 - Cada nome é seguido por : (dois pontos)
 - Os pares nome/valor são seguidos por , (vírgula).





{JSON}

- A ideia é manter a simplicidade para transferir as informações, suportando tipos de dados bem simples:
 - Null
 - Boolean
 - Number
 - String
 - Object
 - um array não ordenado com itens do tipo chave-valor, onde todas as chaves devem ser strings distintas no mesmo objeto;
 - Array
 - lista ordenada de qualquer tipo, inteira entre colchetes e com cada elemento separado por vírgulas.



{ JSON }

```
{  
  "país": "Brasil",  
  "população": 201032714,  
  "PIB total em trilhões de dólares": 2.422,  
  "faz fronteira com": [  
    "Argentina",  
    "Bolívia",  
    "Colômbia",  
    "Guiana Francesa",  
    "Guiana",  
    "Paraguai",  
    "Peru",  
    "Suriname",  
    "Uruguai",  
    "Venezuela"  
  ],  
  "cidades": {  
    "capital": "Brasília",  
    "mais populosa": "São Paulo"  
  }  
}
```

Array

Objeto



{ JSON }

```
{
  hey: "guy",
  anumber: 243,
  - anobject: {
    whoa: "nuts",
    - anarray: [
      1,
      2,
      "thr<h1>ee"
    ],
    more: "stuff"
  },
  awesome: true,
  bogus: false,
  meaning: null,
  japanese: "明日がある。",
  link: http://jsonview.com,
  notLink: "http://jsonview.com is great"
}
```

Array

←



BSON

- Entretanto, a especificação JSON não padroniza o formato de data, ou como trabalhar com dados binários. Por esse motivo, o MongoDB trabalha com **BSON** (*Binary JSON*), que é uma extensão do JSON.
 - MinKey, MaxKey, Timestamp — tipos utilizados internamente no MongoDB;
 - BinData—array de bytes para dados binários;
 - ObjectId—identificador único de um registro do MongoDB;
 - Date—representação de data;
 - Expressões regulares.
- Os documentos do Mongo, após serem persistidos, serão transformados em BSON, um tipo de dado binário e serializado do próprio MongoDB.



BSON

```
{  
  "_id" : ObjectId("539e2idf1ff6b9fa95c3bdba"),  
  "Concurso" : 1,  
  "Data Sorteio" : "11/03/1996",  
  "1ª Dezena" : 4,  
  "2ª Dezena" : 5,  
  "3ª Dezena" : 30,  
  "4ª Dezena" : 33,  
  "5ª Dezena" : 41,  
  "6ª Dezena" : 52,  
  "Arrecadacao_Total" : 0,  
  "Ganhadores_Sena" : 0,  
  "Rateio_Sena" : 0,  
  "Ganhadores_Quina" : 17,  
  "Rateio_Quina" : "39158,92",  
  "Ganhadores_Quadra" : 2016,  
  "Rateio_Quadra" : "330,21",  
  "Acumulado" : "SIM",  
  "Valor_Acumulado" : "1714650,23",  
  "Estimativa_Prêmio" : 0,  
  "Acumulado_Mega_da_Virada" : 0  
}
```



Schema design

- Em um modelo relacional, é comum separar tudo em tabelas, já no MongoDB separa-se por “entidades”, onde todos os dados necessários (ou quase) estão juntos.
 - Modelagem baseada na aplicação
 - Quais são as consultas?
- AGREGADOS



Schema design

- O MongoDB usa esquemas dinâmicos.
- Você pode criar coleções, sem a definição da estrutura, ou seja, os campos ou os tipos de seus valores, dos documentos na coleção.
- Você pode alterar a estrutura de documentos simplesmente adicionando novos campos ou excluir os existentes.
- Os documentos em uma coleção não precisam ter um conjunto idêntico de campos.
 - Na prática, é comum para os documentos de uma coleção de ter uma estrutura muito homogênea.



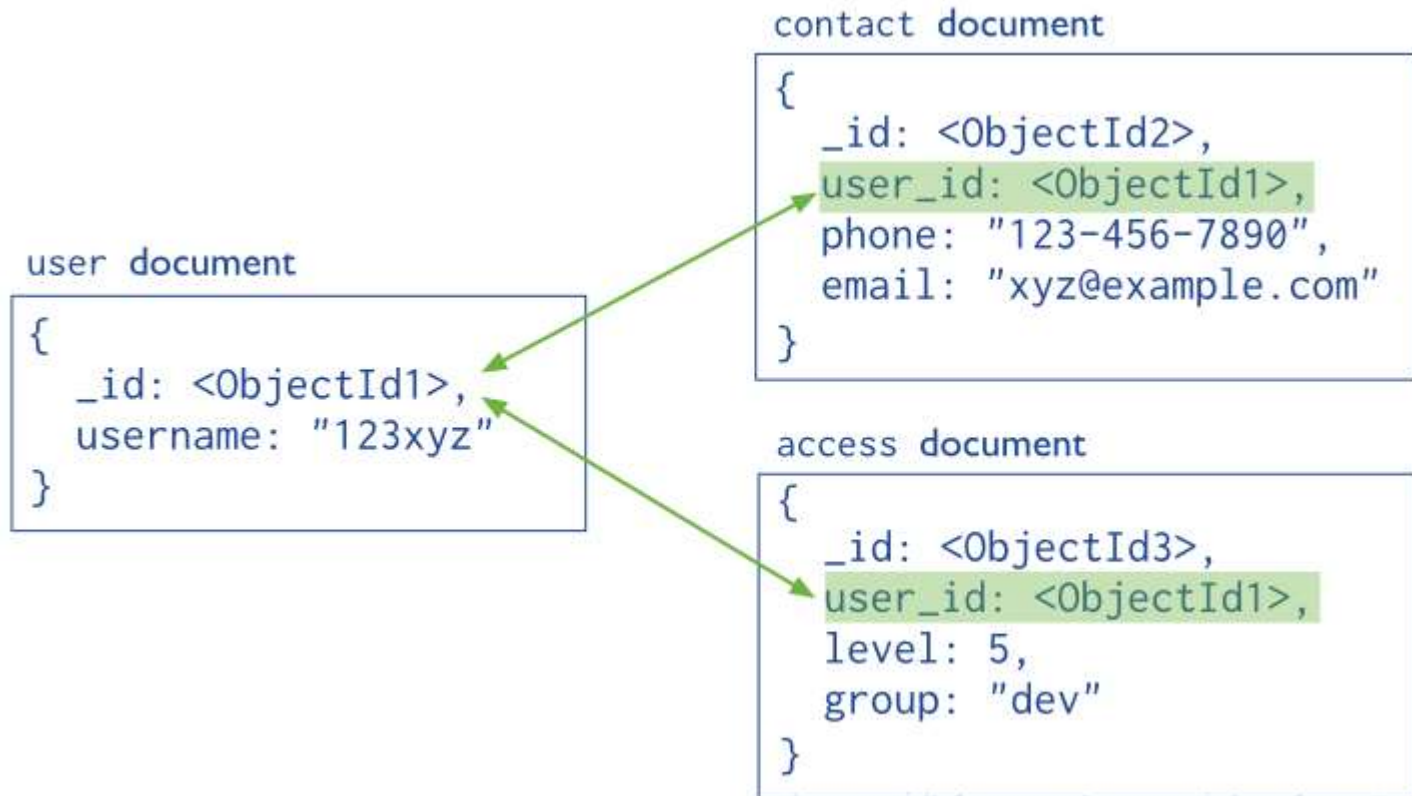
Schema design

- O MongoDB permite o uso de dois modelos:
- Modelo de Referência
 - Relacionamento entre documentos por meio de uma chave
 - Normalização -> menos redundância
- Modelo de Incorporação
 - Documentos integrados em um único documento de estrutura de dados



Schema design

■ Modelo de Referência





Schema design

■ Modelo de Incorporação

```
{
  _id: <ObjectId>,
  username: "123xyz",
  contact: {
    phone: "123-456-7890",
    email: "xyz@example.com"
  },
  access: {
    level: 5,
    group: "dev"
  }
}
```

Embedded sub-document

Embedded sub-document



Schema design

- Use o Modelo de Referência quando:
 - O modelo de incorporação resultar na duplicação de dados, mas sem vantagens suficientes de desempenho de leitura que compensem as implicações da duplicação.
 - Para representar relações mais complexas de muitos-para-muitos.
 - Para modelar grandes conjuntos de dados hierárquicos.



Schema design

- Use o Modelo de Incorporação quando:
 - Existir relacionamentos do tipo contém entre as entidades (1-1)
 - Existir relacionamentos 1-n entre as entidades



Schema design

If the address data is frequently retrieved with the name information, then with referencing, your application needs to issue multiple queries to resolve the reference. The better data model would be to embed the address data in the patron data, as in the following document:

```
{
  _id: "joe",
  name: "Joe Bookreader"
}

{
  patron_id: "joe",
  street: "123 Fake Street",
  city: "Faketon",
  state: "MA",
  zip: "12345"
}
```

```
{
  _id: "joe",
  name: "Joe Bookreader",
  address: {
    street: "123 Fake Street",
    city: "Faketon",
    state: "MA",
    zip: "12345"
  }
}
```



UNIVERSIDADE FEDERAL DE ITAJUBÁ

ARQUITETURA MONGODB



Replicação

■ Arquitetura Mestre-Escravo

- ☐ Prioriza Consistência ao invés da Disponibilidade
- ☐ Há um banco central e outros '*backups*' ativos
- ☐ O MongoDB faz a replicação automática dos dados entre os escravos
- ☐ Se o mestre cair um escravo pode ser eleito como mestre rapidamente



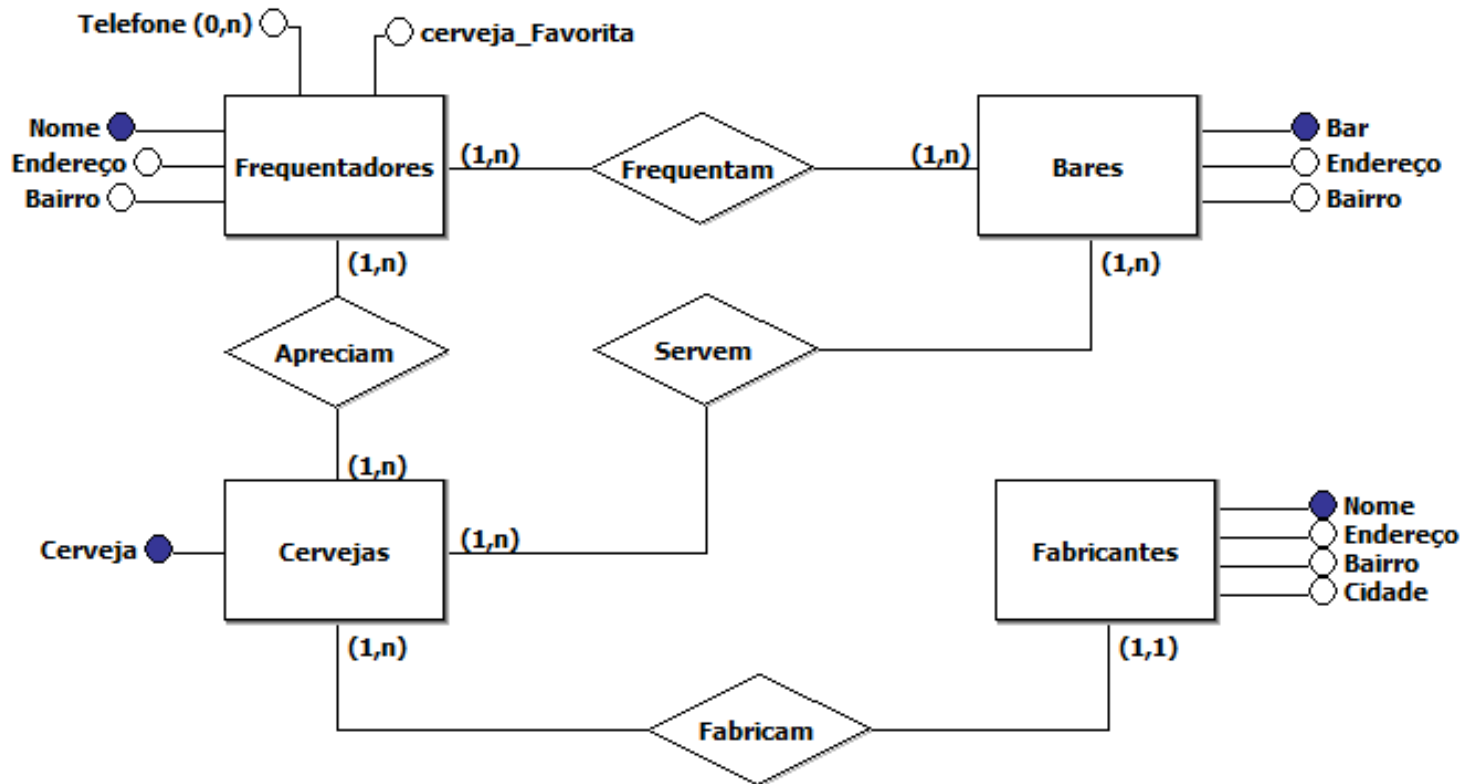
Schema design

- **Projete uma base de dados para a seguinte descrição do problema das “Cervejas, Bares e Restaurantes” [Ullman97]:**
 - Frequentadores têm nomes e endereços unívocos. Eles apreciam uma ou mais cervejas e frequentam um ou mais bares, mas cada frequentador tem sua cerveja favorita. Eles possuem telefone, usualmente um, mas às vezes vários ou nenhum.
 - Bares têm nomes e endereços unívocos. Eles servem uma ou mais cervejas e são frequentados por um ou mais frequentadores.
 - Cervejas têm nomes e fabricantes unívocos. São servidas por um ou mais bares e são apreciadas por um ou mais frequentadores.
 - Fabricantes têm nomes e endereços unívocos e podem fabricar mais de uma cerveja.



Schema design

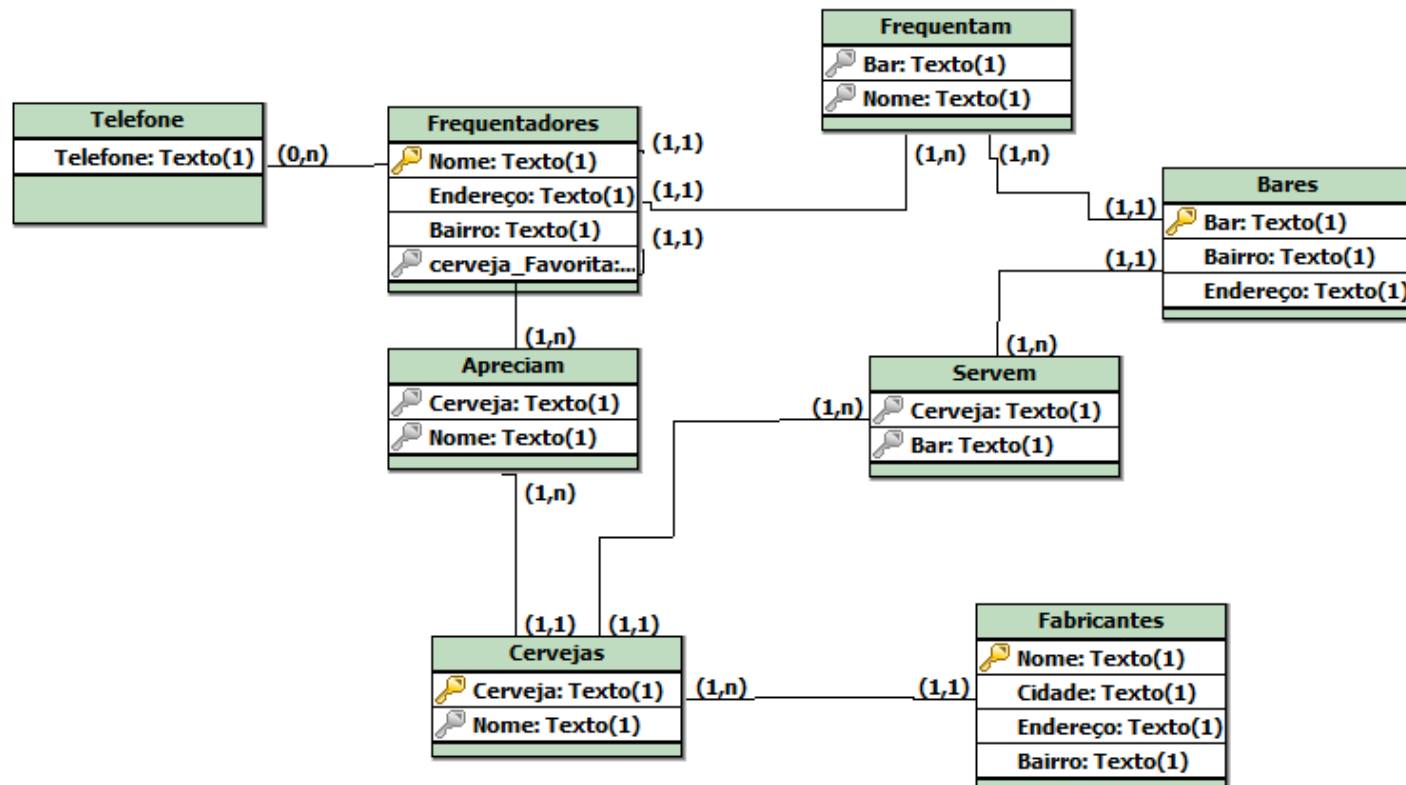
- Projete uma base de dados para a seguinte descrição do problema das “Cervejas, Bares e Restaurantes” [Ullman97]:





Schema design

- Projete uma base de dados para a seguinte descrição do problema das “Cervejas, Bares e Restaurantes” [Ullman97]:





Schema design

■ Como representar esse modelo no MongoDB?

☐ Entidades -> Agregados

■ Quais serão os agregados?

- ☐ Frequentadores
- ☐ Bares
- ☐ Cervejas
- ☐ Fornecedores





Schema design

■ Como representar esse modelo no MongoDB?

☐ Entidades -> Agregados

■ Quais serão os agregados?

- ☐ Frequentadores
- ☐ Bares
- ☐ Cervejas
- ☐ Fornecedores



Depende da aplicação!!!



Schema design

- Como representar esse modelo no MongoDB?
 - Entidades -> Agregados
 - Solução mais simples
 - Um único agregado

Bares	
🔑	Id(nomeBar): String
	Endereço: String
	Bairro: String
<Cervejas : Array de Object>	
🔑	Id(nomeCerveja): Número(4)
	Fabricante: String
	Endereço: String
	Bairro: String
	Cidade: String
<Frequentadores : Array de Object>	
🔑	Id(nomeFrequentador): String
	Endereço: String
	Bairro: String
	Cerveja_Favorita: String
	Telefones: Array



Schema design

■ Como representar esse modelo no MongoDB?

□ Entidades -> Agregados

■ Outra Solução

□ Dois Agregados

Frequentedores
🔑 Id(nome): Texto(1)
Endereço: Texto(1)
Bairro: Texto(1)
Cerveja_Favorita: String
<Telefones : Array>
Telefone: String
<Cervejas : Array>
Nome_Cerveja: String
<Bares : Object>
Id(nomeBar): String
Endereço: String
Bairro: String

Cervejas
🔑 Id(nome): Texto(1)
<Fabricante : Object>
id (nomeFabricant...
Endereço: String
Bairro: String
Cidade: String
<Bares : Array>
Bares: String



Schema design

■ Como representar esse modelo no MongoDB?

□ Entidades -> Agregados

■ Outra Solução

□ Dois Agregados

Depende da aplicação!!!

Frequentedores
🔑 Id(nome): Texto(1)
Endereço: Texto(1)
Bairro: Texto(1)
Cerveja_Favorita: String
<Telefones : Array>
Telefone: String
<Cervejas : Array>
Nome_Cerveja: String
<Bares : Object>
Id(nomeBar): String
Endereço: String
Bairro: String

Cervejas
🔑 Id(nome): Texto(1)
<Fabricante : Object>
id (nomeFabricant...
Endereço: String
Bairro: String
Cidade: String
<Bares : Array>
Bares: String



UNIVERSIDADE FEDERAL DE ITAJUBÁ

PRÁTICA NO MONGODB CRUD



MongoDB

■ Comandos básicos para o **Banco de Dados**

☐ show dbs

- Listar os bancos de dados do servidor

☐ use nome_do_banco

- Utilizar um banco já existente ou criar um novo

☐ db.dropDatabase()

- Apaga o banco de dados corrente

☐ **Prática**

- Listar os bancos do servidor
- Criar o banco 'baresCervejas'



MongoDB

■ Comandos básicos para **Coleção**

☐ show collections

- Lista as coleções de um banco de dados

☐ db.createCollection('nome_da_coleção')

- Cria uma nova coleção

☐ db.nome_da_coleção.drop()

- Deleta uma coleção do banco de dados

☐ db.nome_da_coleção.delete()

☐ **Prática**

- Criar a coleção 'bares'



MongoDB

- Utilizando a modelagem e dados abaixo, criar 3 documentos JSON.

<http://www.jsoneditoronline.org/>

Bar 1				
Rua X, 12				
Centro				
	Cerveja 1	Cerveja 3	Cerveja 9	Cerveja 6
	XXX	YYY	XXX	KKK
	Rua k, 16	Rua 28, 16	Rua k, 16	Rua EST, 16
	Perimetral	Distrito Industrial	Perimetral	Cohab
	Delfim Moreira	Lavras	Delfim Moreira	Belo Horizonte
	João	Maria		
	Rua Q, 90	Rua P, 35		
	Centro	Pinheirinho		
	Cerveja 3	Cerveja 1		
		3777-8888, 6666-9587		




Bares	
🔑	Id(nomeBar): String
	Endereço: String
	Bairro: String
<Cervejas : Array de Object>	
🔑	Id(nomeCerveja): Número(4)
	Fabricante: String
	Endereço: String
	Bairro: String
	Cidade: String
<Frequentadores : Array de Object>	
🔑	Id(nomeFrequentador): String
	Endereço: String
	Bairro: String
	Cerveja_Favorita: String
	Telefones: Array



MongoDB

- Utilizando a modelagem e dados abaixo, criar 3 documentos JSON.

“_id”

Bares	
	Id(nomeBar): String
	Endereço: String
	Bairro: String
<Cervejas : Array de Object>	
	Id(nomeCerveja): Número(4)
	Fabricante: String
	Endereço: String
	Bairro: String
	Cidade: String
<Frequentadores : Array de Object>	
	Id(nomeFrequentador): String
	Endereço: String
	Bairro: String
	Cerveja_Favorita: String
	Telefones: Array



MongoDB

- Utilizando a modelagem e dados abaixo, criar 3 documentos JSON.

Bar 2			
Rua Bela, 35			
Varginha			
Cerveja 3	Cerveja 9		
YYY	XXX		
Rua 28, 16	Rua k, 16		
Distrito Industrial	Perimetral		
Lavras	Delfim Moreira		
Maria	Paulo	Karen	
Rua P, 35	Rua Nova, 69	Rua Marciano, 85	
Pinheirinho	Centro	Centro	
Cerveja 1	Cerveja 3		
3777-8888, 6666-9587	5555-6666, 9999-7474		

Bares	
🔑	Id(nomeBar): String
	Endereço: String
	Bairro: String
<Cervejas : Array de Object>	
🔑	Id(nomeCerveja): Número(4)
	Fabricante: String
	Endereço: String
	Bairro: String
	Cidade: String
<Frequentadores : Array de Object>	
🔑	Id(nomeFrequentador): String
	Endereço: String
	Bairro: String
	Cerveja_Favorita: String
	Telefones: Array



MongoDB

- Utilizando a modelagem e dados abaixo, criar 3 documentos JSON.

Bar 3		
Rua Feia, 53		
Centro		
	Cerveja 6	Cerveja 1
	KKK	XXX
	Rua EST, 16	Rua k, 16
	Cohab	Perimetral
	Belo Horizonte	Delfim Moreira
	Karen	João
	Rua Marciano, 85	Rua Q, 90
	Centro	Centro
		Cerveja 3

Bares	
🔑	Id(nomeBar): String
	Endereço: String
	Bairro: String
<Cervejas : Array de Object>	
🔑	Id(nomeCerveja): Número(4)
	Fabricante: String
	Endereço: String
	Bairro: String
	Cidade: String
<Frequentadores : Array de Object>	
🔑	Id(nomeFrequentador): String
	Endereço: String
	Bairro: String
	Cerveja_Favorita: String
	Telefones: Array



MongoDB

- Inserir os documentos na coleção

- `db.bares.insert(documento)`

```
writeResult({ "nInserted" : 1 })
```

- Ao final das inserções

- `db.bares.count()`

- `db.bares.stats()`



MongoDB

■ Consultas Simples

□ `db.bares.find()`

- Lista todos os documentos da coleção
- `db.bares.find().pretty()` para visualizar de forma mais amigável

□ `db.bares.findOne()`

- Retorna o primeiro registro da consulta



MongoDB

■ Consultas Simples

- O comando find pode receber parâmetros para filtrar o resultado

```
db.collection.find(query, projection)
```

■ Query é baseada no Criteria

```
db.<nome-da-collection>.find(<campo1>:<valor1>,  
<campo2>:<valor2>,...);
```

■ Consultar os dados do Bar 3

- `db.bares.find({_id : "Bar 3"})`



MongoDB

■ Consultas Simples

□ Consultar os bares frequentados pela Karen

- `db.bares.find({"Frequentadores.nomeFrequentador" : "Karen"})`

□ Consultar os bares localizados no centro

- `db.bares.find({Bairro : "Centro"})`



MongoDB

`db.collection.find(query, projection)`

■ Consultas Simples

- O comando find por padrão exibe todas as colunas.
- Para restringir os campos, devemos informar conforme a sintaxe:

```
db.<nome-da-collection>.find(  
    {<campo1>:<valor1>,  
      <campo2>:<valor2>,  
      ...},  
    {<campoParaExibir>:<exibeOuNaoExibe>,  
      <campoParaExibir>:<exibeOuNaoExibe>,  
      ...});
```

- O valor de `exibeOuNaoExibe` pode ser em dois formatos:
 - 1 ou true exibem o campo;
 - 0 ou false, não.



MongoDB

■ Consultas Simples

□ Consultar os bairros dos bares frequentados pela Karen

- `db.bares.find({"Frequentadores.nomeFrequentador" : "Karen"}, {Bairro : true})`

```
{ "_id" : "Bar 2", "Bairro" : "Varginha" }  
{ "_id" : "Bar 3", "Bairro" : "Centro" }
```

- `db.bares.find({"Frequentadores.nomeFrequentador" : "Karen"}, {Bairro : true, _id : false})`



MongoDB

■ Consultas Simples

□ Operadores

<http://docs.mongodb.org/manual/reference/operator/query/>

■ Prática

□ Selecionar os bares frequentados pela Karen e pelo João

```
db.bares.find({ $and: [  
    {"Frequentadores.nomeFrequentador" : "Karen"},  
    {"Frequentadores.nomeFrequentador" : "Joao"}]  
},  
    {_id : true}))
```



MongoDB

■ Consultas Simples

☐ Operadores

<http://docs.mongodb.org/manual/reference/operator/query/>

■ Prática

- ☐ Selecionar os bares frequentados pela Karen ou pelo João



MongoDB

■ Apagar Documento

☐ Comando remove

```
db.<nome-da-collection>.remove(  
  { <critérioDeBusca1>:<valor1>,...}  
);
```

☐ Remover o Bar 2 da coleção

- db.bares.remove({_id : "Bar 2"})
- db.bares.count()

☐ Inserir o Bar 2 novamente na coleção



MongoDB

■ Alterar Documento

□ Comando update

```
db.<nome-da-collection>.update(  
  {<critérioDeBusca1>:<valor1>,...},  
  {<campoParaAtualizar1>:<novoValor1>,...})  
));
```

- Além de atualizar registros, com ele é possível também alterar a estrutura de uma coleção e até remover colunas.

■ CUIDADO!!!

```
{ "_id" : ObjectId("53cdccc474f51f1e837600c0"),  
  "Concurso" : 99999,  
  "CPF" : 12345678900,  
  "Nome" : "Coffin Joe" }
```

```
>db.ganhadores.update(  
  { "_id" : ObjectId("53cdccc474f51f1e837600c0") },  
  { "Nome:" : "Zé do caixão" });
```

```
> db.ganhadores.find(  
  { "_id" : ObjectId("53cdccc474f51f1e837600c0") });  
{ "_id" : ObjectId("53cdccc474f51f1e837600c0"),  
  "Nome:" : "Zé do caixão" }
```



MongoDB

■ Alterar Documento

□ Comando update

```
db.<nome-da-collection>.update(  
  {<critérioDeBusca1>:<valor1>,...},  
  {<campoParaAtualizar1>:<novoValor1>,...})  
);
```

- Este é o comportamento padrão do comando update: substituir as colunas informadas no comando pelas linhas encontradas. Se a coluna não foi especificada, ela será eliminada.
- Existe, porém, uma opção que se assemelha ao comando SQL, que é uso do set no update:

```
db.<nome-da-collection>.update(  
  { <critérioDeBusca1>:<valor1>,...},  
  { $set: { <campoParaAtualizar1>:<novoValor1>,...} })  
);
```



MongoDB

■ Alterar Documento

□ Comando update

- Existe, porém, uma opção que se assemelha ao comando SQL, que é uso do set no update:

```
> db.ganhadores.update(  
  {"_id" : ObjectId("53cdccc474f51f1e837600c0")},  
  { $set: {"Nome": "Zé do caixão"}});
```



MongoDB

- Alterar Documento

- Comando update

- Prática

- Alterar o bairro do Bar 2 para "Medicina"



MongoDB

■ Alterar Documento

- Comando update

- Prática

- Alterar o bairro do Bar 2 para “Medicina”

```
db.bares.update(  
    { _id : "Bar 2"},  
    { $set : {Bairro : "Medicina"}}  
)
```



MongoDB

■ Alterar Documento

□ Comando update

- Outro ponto do comando update que é importante saber é que, por padrão, ele atualiza apenas o **primeiro registro** que obedecer ao critério de busca especificado.
- Para que ele altere todos os registros, é preciso adicionar mais um parâmetro booleano, o **multi**, que por padrão é false.

```
db.ganhadores.update({},  
{ $set: {"CPF": 55555555555 },{multi:true}});
```



MongoDB

■ Alterar Documento

- ☐ Comando update

- ☐ Prática

- Alterar o endereço do fornecedor da Cerveja 3 para: Rua 35, 87



MongoDB

■ Alterar Documento

- Comando update

- Prática

- Alterar o endereço do fornecedor da Cerveja 3 para: Rua 35, 87

```
db.bares.update(  
    {"Cervejas.nomeCerveja" : "Cerveja 3"},  
    {$set : {"Cervejas.$.Endereço" : "Rua 35, 87"}},
```

- Consultar os registros do banco



MongoDB

■ Alterar Documento

- Comando update

- Prática

- Alterar o endereço do fornecedor da Cerveja 3 para: Rua 35, 87

```
db.bares.update(  
    {"Cervejas.nomeCerveja" : "Cerveja 3"},  
    {$set : {"Cervejas.$.Endereço" : "Rua 35, 87"}}  
)
```

- Consultar os registros do banco



MongoDB

■ Alterar Documento

☐ Comando update

- Inserir o campo 'Rendimento_Anual' em todos os bares:

- ☐ Bar 1 : 1.500.000
- ☐ Bar 2 : 450.000
- ☐ Bar 3 : 790.000



MongoDB

■ Alterar Documento

□ Comando update

- Para remover um atributo do documento, usar o update + \$unset

```
db.bares.update(  
    {},  
    {$unset : {Rendimento_Anual : ""}},  
    {multi : true})
```

- Remove o campo Rendimento_Anual de todos os documentos da coleção.



MongoDB

■ Alterar Documento

□ Comando update

- Para alterar o nome de um atributo do documento, usar o update + \$rename

```
db.bares.update(  
  {},  
  {$rename : {Rendimento_Anual2 : "Rendimento_Anual"}},  
  {multi : true})
```

- Altera o nome do campo Rendimento_Anual2 para Rendimento_Anual de todos os documentos da coleção.



MongoDB

■ Alterar Documento

- ☐ Comando update

<http://docs.mongodb.org/manual/reference/operator/update/>



Referências

- NoSQL – um guia conciso para o Mundo Emergente da Persistência Poliglota.
 - Sadalage & Fowler
 - Editora Novatec
 - 2013

- MongoDB. Construa novas aplicações com novas tecnologias
 - Fernando Boaglio
 - Casa do Código

- MongoDB Manual 3.0
 - <http://docs.mongodb.org/manual/reference/>