

MBA em IA e Big Data



Curso 01 - Linguagens e Ferramentas para Inteligência Artificial e Big Data (Python e SQL)

MongoDB - CRUD

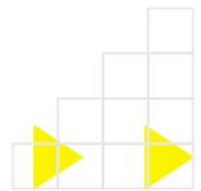
Jose Fernando Rodrigues Junior
ICMC-USP São Carlos

Objetivo: apresentar conceitos sobre sistemas não relacionais contrastando o modelo relacional com o sistema MongoDB



CRUD

Create, Read, Update, Delete



SQL to MongoDB



SQL Terms, Functions, and Concepts	MongoDB Aggregation Operators
WHERE	<code>\$match</code>
GROUP BY	<code>\$group</code>
HAVING	<code>\$match</code>
SELECT	<code>\$project</code>
ORDER BY	<code>\$sort</code>
LIMIT	<code>\$limit</code>
SUM()	<code>\$sum</code>
COUNT()	<code>\$sum</code>

Mapping Chart:

<http://docs.mongodb.org/manual/reference/sql-aggregation-comparison/>

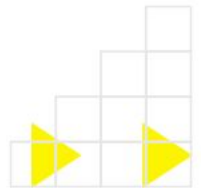


CRUD: Creation

Inserção de documentos/criação de novas coleções:

```
db.<collection>.insert(<document>)
```

```
INSERT INTO <table>  
VALUES(<attributevalues>);
```



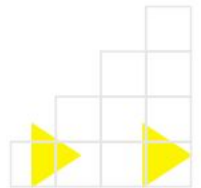


CRUD: Inserting Data

Inserir um documento:

```
db.<collection>.insert({<field>:<value>})
```

Para inserir múltiplos documentos de uma única vez, basta usar notação de array [<doc1,doc2,...>].





CRUD: Reading

- ❑ Recuperar todos os documentos: `db.<collection>.find()`

- ❑ Retorna um cursor para iterar sobre os primeiros 20 resultados

- ❑ Adicionar “.limit(<number>)” restringe o número de resultados

`db.<collection>.find().limit(2)`

SELECT * FROM <table>;

- ❑ Recuperar um documento: `db.<collection>.findOne()`, o primeiro encontrado no disco, usualmente o primeiro que foi inserido

CRUD: Reading

- Definindo um predicado:

```
db.<collection>.find({<field>:<value>})
```

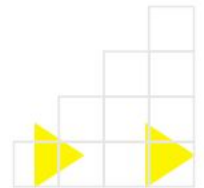
- Operador lógico “AND” se torna “,”:

```
db.<collection>.find({<field1>:<value1>,  
                     <field2>:<value2>  
                     })
```

SELECT *

FROM <table>

WHERE <field1> = <value1> AND <field2> = <value2>;





CRUD: Reading

- Operador lógico OR

```
db.<collection>.find({ $or: [  
    {<field>:<value1>},  
    {<field>:<value2>} ]  
})
```

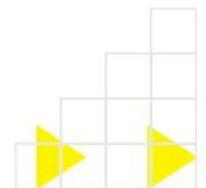
SELECT *
FROM <table>
WHERE <field> = <value1> OR <field> = <value2>;

Ex.: `db.Time.find({$or: [{nome:"Marcos"}, {saldo_gols:10}]})`

- Operador In (pertence) Checking for multiple values of a set:

```
db.<collection>.find({<field>: {$in: [<value>, <value>]}})
```

SELECT *
FROM <table>
WHERE <field> IN (<value>,<value>);





CRUD: Reading

- Seleccionando campos específicos:

```
db.<collection>.find({ }, {<field1>: 1})
```

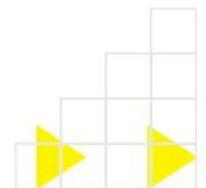
```
SELECT field1  
FROM <table>;
```

↓
0 false
>0 true

- Seleccionando campos específicos com predicado:

```
db.<collection>.find({<field1>:<value>}, {<field1>: 1})
```

```
SELECT field1  
FROM <table>  
WHERE <field1> = <value>;
```



CRUD: Updating

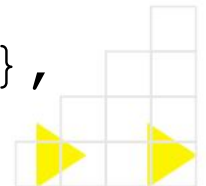


```
db.<collection>.update(  
  {<field1>:<value1>},           //predicado  
  {$set: {<field2>:<value2>}},    //novo valor  
  {multi:true} )                 //multiple docs
```

```
UPDATE <table>  
SET <field2> = <value2>  
WHERE <field1> = <value1>;
```

Exemplo:

```
db.Time.update({nome:"Marcos"}, {$set: {saldo_gols:20}},  
{multi:false})
```





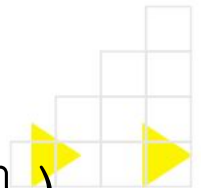
CRUD: Updating

- Removendo um atributo específico:

```
db.<collection>.update({<field>:<value>},  
                        {$unset: { <field>:1}})
```

```
ALTER TABLE DROP COLUMN <field>  
"WHERE field = value"
```

Exemplo: `db.Time.update({nome:"Marcos"},
 {$unset:{saldo_gols:1}})`





CRUD: Deleting

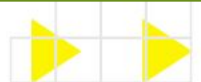
- Remover todos os documentos selecionados:

```
db.<collection>.remove({<field>:<value>})
```

```
DELETE FROM <table>  
WHERE <field> = <value>;
```

- Remover apenas o primeiro documento

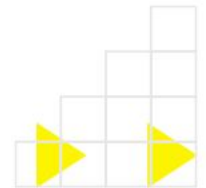
```
db.<collection>.remove({<field>:<value>}, true)
```





Pipelines: leitura avançada

- Provê:
 - filtros aplicados sequencialmente
 - transformações nos documentos para produzir nova informações
- Ferramentas para:
 - agrupar e ordenar
 - operar sobre arranjos (arrays) de dados



Agregação



- **Somatório do saldo de gols:**

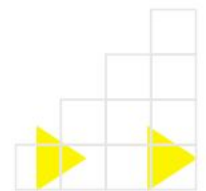
```
db.Time.aggregate({$group:{_id:"group_gols",total_gols:{$sum:"$saldo_gols"}}});
```

```
SELECT SUM(SALDO_GOLS) AS TOTAL_GOLS  
FROM TIME
```

- **Somatório do saldo de gols por estado:**

```
db.Time.aggregate({$group:{_id:"$estado",total_gols:{$sum:"$saldo_gols"}}});
```

```
SELECT ESTADO, SUM(SALDO_GOLS) AS TOTAL_GOLS  
FROM TIME  
GROUP BY ESTADO
```



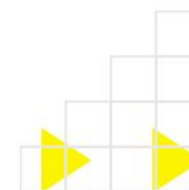
Agregação com ordenação



- **Somatório do saldo de gols por estado com ordenação decendente:**

```
db.Time.aggregate(  
  {$group: {_id: "$estado",  
            total_gols: {$sum: "$saldo_gols"}}},  
  {$sort: {total_gols: -1}}  
);
```

```
SELECT ESTADO, SUM(SALDO_GOLS) AS TOTAL_GOLS  
FROM TIME  
GROUP BY ESTADO  
ORDER BY TOTAL_GOLS DESC
```



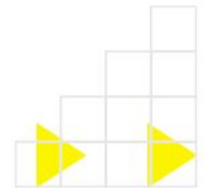
Agregação com ordenação



- **Somatório do saldo de gols por estado com ordenação decendente, top 2:**

```
db.Time.aggregate(  
  {$group: {_id: "$estado",  
            total_gols: {$sum: "$saldo_gols"}}},  
  {$sort: {total_gols: -1}},  
  {$limit: 2}  
);
```

```
SELECT * FROM(  
  SELECT ESTADO, SUM(SALDO_GOLS) AS TOTAL_GOLS  
  FROM TIME  
  GROUP BY ESTADO  
  ORDER BY TOTAL_GOLS DESC)  
WHERE ROWNUM <=2
```





Isolamento

- Por padrão, todas as escritas são atômicas com relação a documentos isolados
- A atualização de múltiplos documentos pode ocorrer em alternância (interleaving) quando as operações ocorrem em concorrência
- Pode-se isolar uma coleção inteira adicionando-se **“\$isolated:1”**:

```
db.foo.update(  
  { status : "A" , $isolated : 1 },  
  { $inc : { count : 1 } },  
  { multi: true }  
)
```

← predicado

← update é o incremento

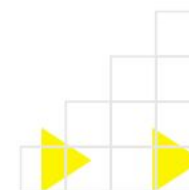
- **“\$isolated:1” faz com que outros clientes aguardem leitura/escrita até que o comando seja concluído**
- A partir da versão 4.0, suporte a isolamento de múltiplos documentos





Contras

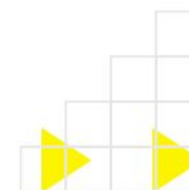
- Sem esquema \Rightarrow Sem projeto: é tentador começar o projeto sem pensar nele antes
- Sem projeto \Rightarrow Gerenciamento de aplicações torna-se mais custoso
- Junções acabam sendo necessárias, afinal \Rightarrow Bases relacionais fazem isso melhor
- [Porque nunca usar MongoDB?](#)
- [Is MongoDB Really A Good Fit Data Solution for Your Business?](#)





Contras

- MongoDB não é um substituto universal para bancos de dados;
- Diferentes paradigmas têm aplicações (nichos) distintas;
- Bancos de dados relacionais são universais e já bem consolidados, com tecnologia aperfeiçoada ao longo de décadas;
 - mas tem limitações com escalabilidade, custos, e curva de aprendizado.
- Cada sistema, uma aplicação diferente:
 - leitura intensiva;
 - escrita intensiva;
 - muitas entidades conceituais interagindo, estrutura complexa de dados;
 - segurança;
 - escalabilidade;
 - mobilidade.
- É necessário analisar cada caso \Rightarrow mais detalhes no Curso 03.





Comparação com outros bancos de dados

MongoDB vs MySQL Differences

<https://www.mongodb.com/compare/mongodb-mysql>

Cassandra vs MongoDB - What are the Differences?

<https://phoenixnap.com/kb/cassandra-vs-mongodb>

PostgreSQL vs MongoDB:

<https://www.mongodb.com/compare/mongodb-postgresql>



Mais informações



Resource	Location
MongoDB Downloads	mongodb.com/download
Free Online Training	education.mongodb.com
Webinars and Events	mongodb.com/events
White Papers	mongodb.com/white-papers
Case Studies	mongodb.com/customers
Presentations	mongodb.com/presentations
Documentation	docs.mongodb.org
Additional Info	info@mongodb.com

⇒ [Instalação do MongoDB](#)

Mongodb: The Definitive Guide: Powerful and Scalable Data Storag,

By Kristina Chodorow and Mike Dirolf

Published: 9/24/2019

Pages: 514

Language: English

Publisher: O'Reilly Media, CA

