03

# Atualizando dados

## Update, replace, set e multi

Nessa aula adicionaremos mais um aluno, o "Fernando", com suas respectivas informações, mas ao em vez de escrevermos que ele estuda "Sistemas de informação", faremos um erro proposital dizendo que ele estuda "Sistema de informação", retirando o "s" de "Sistemas". Assim, se efetuarmos uma busca, através do `find` em todos os alunos que cursam "Sistema de informação" não encontraremos Felipe, o aluno que faz "Sistemas", com o "s". Encontraremos apenas "Fernando". Veremos como consertar esse erro usando o *update*.

Primeiro, temos que adicionar o novo aluno, "Fernando". Digitaremos o `insert` no Editor e depois rodaremos no Terminal:

```
db.alunos.insert({
    nome : "Fernando",
    data_nascimento : new Date(1994, 03, 26),
    notas : [ 10, 4.5, 7],
    curso : {
        nome : "Sistema de informação"
    }
})
```

Para buscar os alunos que cursam "Sistema de informação" usaremos o `find` :

```
db.alunos.find({"curso.nome" : "Sistema de informação"})
```

Mas só aparece o Fernando! Isso aconteceu porque o Felipe está cadastrado como aluno de "Sistema de informação", no singular. Poderíamos remover o aluno "Fernando" e inseri-lo novamente e resolveríamos o problema. Fazer isso, entretanto, é pouco prático. Algo que podemos fazer é um `update` de todos os alunos que cursam "Sistema de informação" pedindo para que passem a ser "Sistemas de informação". O `update` funciona pedindo a condição que queremos modificar e pede também a condição que desejamos.

Atenção, o `update` não aceita o sinal de ".". Assim, não podemos escrever dentro das chaves `curso.nome` .

```
db.alunos.update(
    {"curso.nome" : "Sistema de informação"}
    {
        "curso.nome" : "Sistemas de informação"
    }
)
```

Também não podemos passar que a primeira condição seja `curso.nome` e na segunda apenas `nome` , a primeira condição apenas substituirá a segunda. Portanto, não podemos usar o `update` junto a "curso.nome":

```
db.alunos.update(
    {"curso.nome" : "Sistema de informação"}
    {
```

```
        "nome" : "Sistemas de informação"
    }
)
```

Temos que ter cuidado quando usamos o `update`. Ele não funciona da mesma maneira que no *SQL*. No *SQL* basta dizermos que desejamos "resetar" um determinado campo para outro, mas, no `db` não funciona dessa mesma maneira, ele não altera os demais campos. No *SQL* teríamos o seguinte:

```
UPDATE cursos SET nome = "Sistemas de informação" WHERE nome = "Sistema de informação"
```

Assim, no *SQL* altera-se apenas o campo `Sistema de informação`.

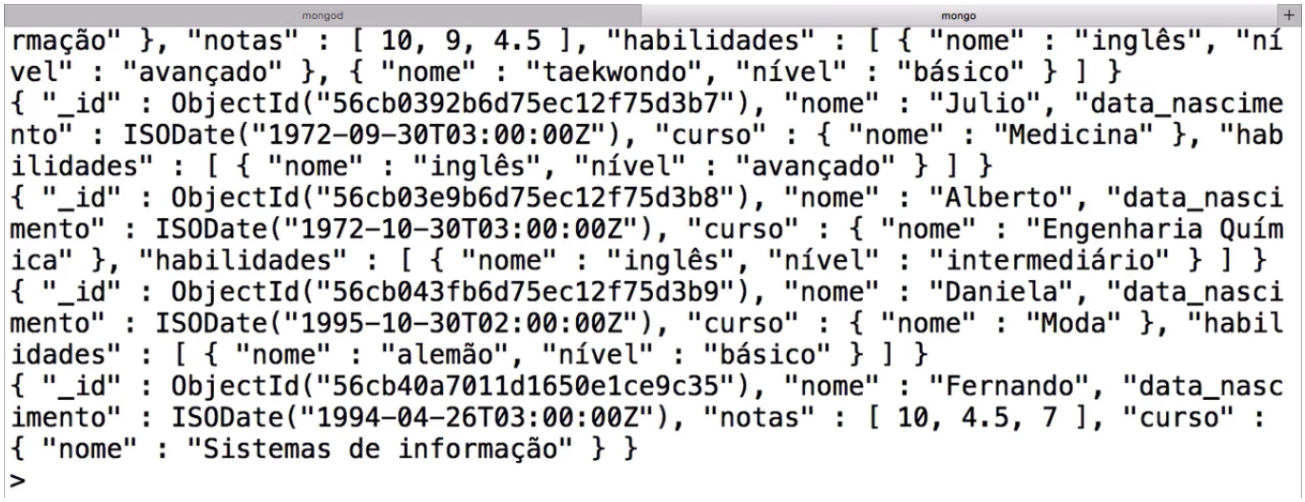
Voltando ao `db.aluno.update`, se não falarmos nada, se não usarmos operadores, estaremos dizendo, simplesmente, para simplesmente substituir uma coisa pela outra e esse foi o equívoco anterior. Vamos fazer um outro tipo de `update`, que não faça um simples cambio. Para isso, usaremos o "set" junto ao campo `curso.nome : Sistemas de informação`. Escrevemos o seguinte:

```
db.alunos.update(
    {"curso.nome" : "Sistema de informação"},
    {
        $set : {
            "curso.nome" : "Sistemas de informação"
        }
    }
)
```

Testando isso no Terminal ele nos responde:

```
WriteResult({ "nMatched" : 1 "nUpserted" : 0, "nModified" : 1 })
```

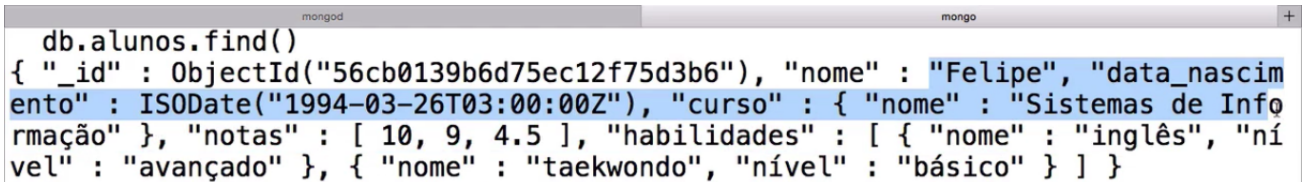
E ele nos diz que altera o curso do "Fernando" para, finalmente, "Sistemas de informação":




Temos um padrão dos nomes dos cursos, que é escreve-los com letras maiúsculas: "Moda", "Medicina", "Engenharia Química". Agora, falta padronizar o "Sistemas de informação". Já sabemos como fazer isso usando o `db.alunos.update`:

```
db.alunos.update(
    {"curso.nome" : "Sistemas de informação"},
    {
        $set : {
            "curso.nome" : "Sistemas de Informação"
        }
    }
)
```

Rodando isso no Terminal veremos que deu relativamente certo, pois, ele modifica, mas apenas um campo, o "Felipe" que agora cursa Sistemas de Informação, com letras maiúsculas!

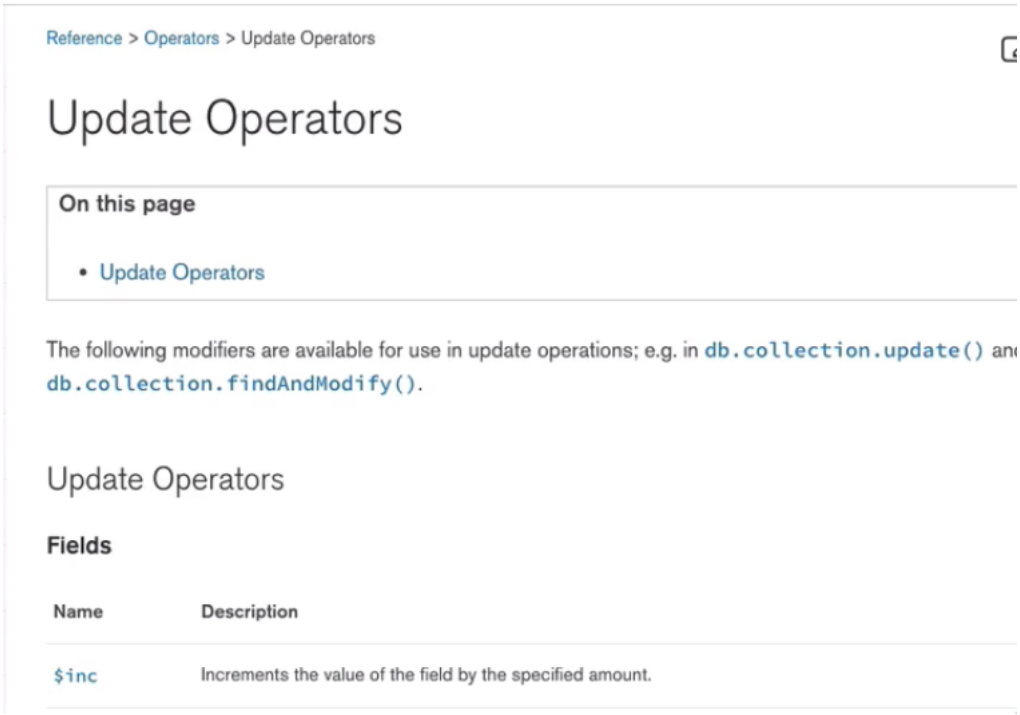



O curso do Fernando, entretanto, aparece escrito da mesma maneira. Isso ocorre pois o `update`, por padrão, só executa um comando para o primeiro documento que ele encontra. O `update`, por padrão, troca apenas um pedacinho do conteúdo. Para que ele entenda que isso tem que ser feito para várias coisas é preciso passar um `multi : true`, que, por padrão é `false`. Como queremos passar isso para "múltiplas linhas" escreveremos o seguinte:

```
db.alunos.update(
    {"curso.nome" : "Sistemas de informação"},
    {
        $set :
           {"curso.nome" : "Sistemas de Informação"}
        },
      {
        multi : true
      }
)
```

Executando isso no terminal ele vai mostrar tudo certinho e todos os alunos que cursam sistemas estarão escritos da forma padrão, "Sistemas de Informação", com letras maiúsculas:

Aprendemos a usar o `update` para fazer uma troca completa e para "setar" apenas um pedacinho. Se quisermos usar outros operadores podemos buscar informações a cerca na documentação:




## Update com arrays através do push e each

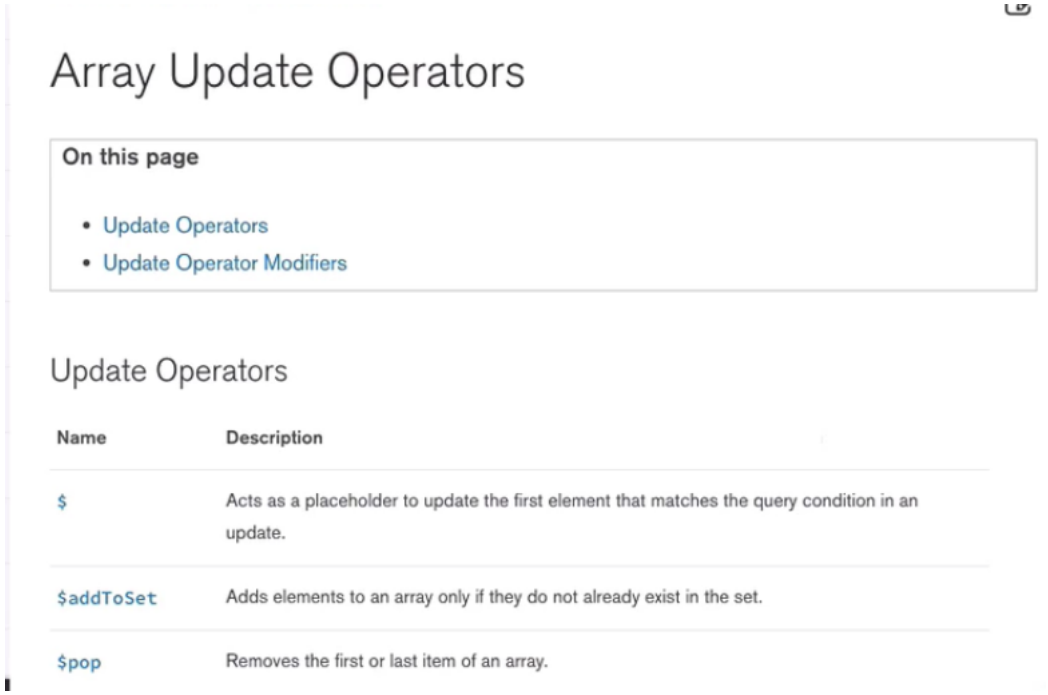
Vamos dar mais um passo com nossos alunos. Se dermos um `db.alunos.find()` veremos a lista dos alunos que temos, dentre eles está "Felipe", cujo `id` é `ObjectId("56cb0002b6d75ec12f75d3b5")`, vamos copiar seu `id` e buscar o aluno "Felipe". Para isso digitamos:

```
db.alunos.find({"_id" : ObjectId("56cb0002b6d75ec12f75d3b5")}).pretty()
```

Podemos observar as notas de "Felipe". Imagine que "Felipe" realizou mais uma prova e que gostaríamos de atualizar sua nota, suponhamos que a nota seja "8,5". Podemos usar o `update`, o `set`, o `id` daquilo que queremos modificar e as novas notas:

```
db.alunos.update(
    {"_id" : ObjectId("56cb0002b6d75ec12f75d3b5")},
    {$set : {
        {
            "notas" : [
                10,
                9,
                4.5,
                8.5
            ]
        }
    }
}
```

Isso é válido! Mas vejamos algo que pode ser mais funcional e simples! Vamos adicionar um valor dentro de uma *Array*, para confirmar isso, podemos, inclusive, consultar a documentação buscando *Array Update Operators*:



Array Update Operators

On this page

- Update Operators
- Update Operator Modifiers

Update Operators

| Name | Description |
|---|---|
| $ | Acts as a placeholder to update the first element that matches the query condition in an update. |
| $addToSet | Adds elements to an array only if they do not already exist in the set. |
| $pop | Removes the first or last item of an array. |

A ideia, aqui, é buscar uma otimização dessa tarefa! Usando uma *Array* e utilizando o `$push` empurramos algo dentro da *Array*. Escreveremos `$push` , colocaremos o campo "notas" e dentro dele o valor da nota. Teremos o seguinte:

```
db.alunos.update(
    {"_id" : ObjectId("56cb0002b6d75ec12f75d3b5")},
    {
        $push : {
            notas : 8.5
        }
    }
)
```

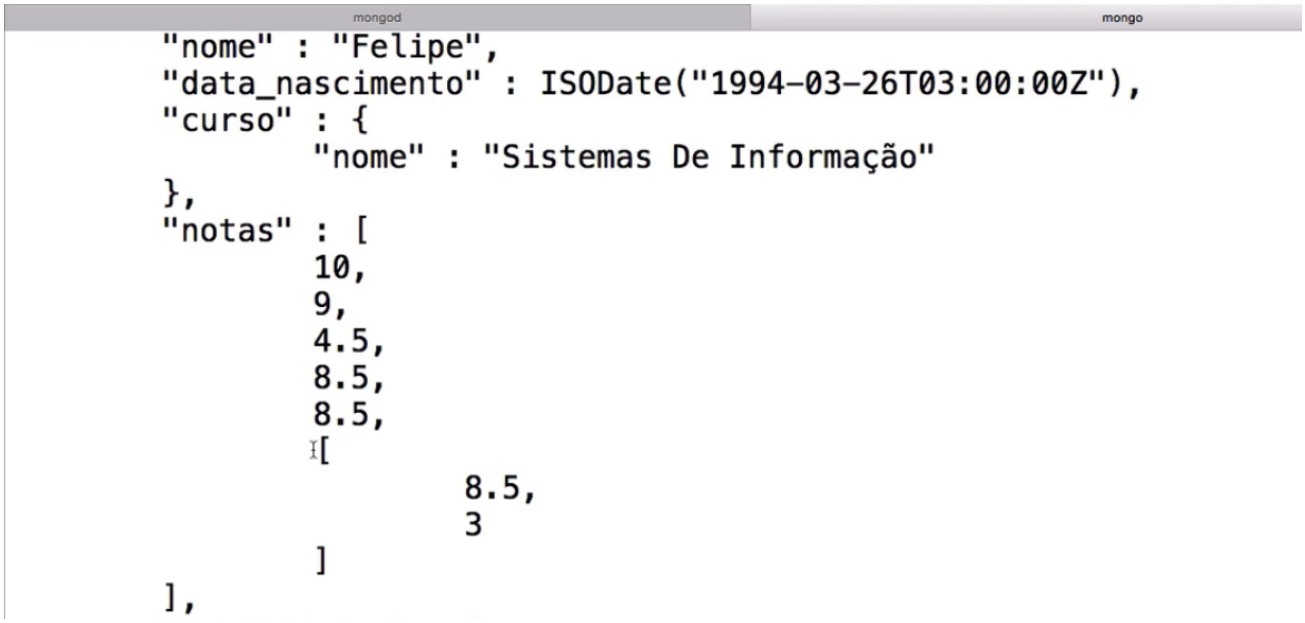
Vamos rodar isso no Terminal e dar um `db.alunos.find({ "nome": "Felipe"}).pretty()` . Ele nos mostrará o aluno "Felipe" com a nota alterada.


Nós usamos o `$push` , mas existem outros métodos de fazermos isso. Por exemplo, temos o `$addToSet` que pode ser usado apenas se a *Array* não existe e é utilizado quando não queremos elementos repetidos. Já o `$push` pode ser usado quando queremos elementos repetidos. Portanto, o uso de um ou de outro depende do que buscamos.

Para introduzir duas notas diferentes você pode estar pensando que basta escrever da seguinte maneira:

```
db.alunos.update(
    {"_id" : ObjectId("56cb0002b6d75ec12f75d3b5")},
    {
        $push : {
            notas : [8.5, 3]
        }
    }
)
```

Nesse caso, se rodarmos isso em nosso Terminal teremos as notas que já foram inseridas, uma *Array* e as notas 8,5 e 3. Observe:




Fica estranho! Temos que tomar muito cuidado quando queremos introduzir mais de um valor no `$push` .

Já que introduzimos as notas de maneira errada teremos que fazer um `db.alunos.update` e "setar", novamente, as notas. Após fazer isso e dando um `find` encontraremos tudo como queríamos!

Vamos tentar refazer o `$push` acrescentando dois valores, mas, dessa vez, da maneira correta. Basta falarmos que para cada um desses colegas, seja feito algo, isto é, acrescentaremos um `each` :

```
db.alunos.update(
    {"_id" : ObejctId("56cb0139b6d75ec12f75d3b6")},
    {
        $push : {
            "notas" : {$each : [8.5, 3] }
        }
    }
)
```

Observe que temos uma escrita mais complexa. Rodando isso veremos que as notas foram alteradas:


Além da operação `$set` , que substitui um pedaço, temos diversos operadores muito interessantes, inclusive, que usam *Array*. Para observá-los basta consultar a documentação.

É importante observar que algumas *Array* são usadas como conjuntos matemáticos. Nesse caso dos conjuntos, os elementos não se repetem, em outros como no `$push` os elementos se repetem. Outra questão importante de lembrarmos é o uso do `each` quando queremos alterar mais de um elemento.