

Documentação de Teste – HBase Shell (via Docker)

1. Objetivo

Validar o ambiente **HBase** rodando em **container Docker**, realizando:

- acesso ao container;
- abertura do **HBase Shell**;
- criação de uma tabela com **famílias de colunas**;
- inserção de registros (put);
- consultas (get, scan, count);
- validação de operação básica do cluster.

2. Ambiente e contexto

- **Host:** Windows (PowerShell)
- **Acesso:** docker exec para entrar no container hbase-master
- **HBase:** versão **1.2.6** (apresentada ao abrir o shell)

2.1 Aviso apresentado (WARN)

Ao abrir o HBase Shell, apareceu:

WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...

Interpretação: é um aviso comum em ambientes containerizados/Java quando a lib nativa do Hadoop não está presente.

Impacto: não impede os testes básicos (o sistema usa classes Java “built-in”).

3. Procedimento executado (passo a passo)

3.1 Entrar no container

No PowerShell:

```
docker exec -it hbase-master bash
```

3.2 Abrir o HBase Shell

Dentro do container:

```
hbase shell
```

O shell confirmou a inicialização e mostrou a versão.

4. Verificação inicial: listar tabelas existentes

No HBase Shell:

```
list
```

Resultado: nenhuma tabela existia inicialmente (=> []).

5. Criação da tabela de teste

5.1 Criar tabela alunos com 3 famílias de colunas

Comando:

```
create 'alunos', 'dados', 'contato', 'sys'
```

5.2 Validar criação

```
list
```

```
describe 'alunos'
```

Resultado esperado:

- tabela alunos listada
- status ENABLED
- famílias de colunas: dados, contato, sys

Observação importante: no HBase, **é definido somente as famílias** na criação.

As “colunas” (qualifiers) são criadas dinamicamente no put, ex.: dados:nome, contato:email.

6. Inserção manual de um registro (validação rápida)

6.1 Inserir dados na rowkey ALU#0001

```
put 'alunos', 'ALU#0001', 'dados:nome', 'Ana Souza'
```

```
put 'alunos', 'ALU#0001', 'dados:curso', 'DS'
```

```
put 'alunos', 'ALU#0001', 'contato:email',  
'ana.souza@exemplo.com'
```

```
put 'alunos', 'ALU#0001', 'sys:created_at',  
Time.now.to_i.to_s
```

6.2 Consultar a linha inserida

```
get 'alunos', 'ALU#0001'
```

Resultado esperado: 4 colunas retornadas, como:

- dados:nome

- dados:curso
- contato:email
- sys:created_at

7. Tentativa de carga automática (massa pequena) e erro encontrado

7.1 Comando executado (com erro)

Foi definido o array de cursos:

```
cursos = ['DS', 'Redes', 'Mecatrônica', 'Eletrotécnica']
```

E depois tentou-se usar:

```
cursos.sample
```

Erro retornado:

- NoMethodError: undefined method 'sample' for Array

7.2 Motivo do erro

A versão do Ruby/JRuby embarcada no **HBase Shell 1.2.6** pode não suportar o método `Array#sample` (dependendo do runtime).

8. Correção recomendada para gerar a massa de dados

8.1 Versão corrigida (sem usar .sample)

Use seleção aleatória com `rand`:

```
cursos = ['DS', 'Redes', 'Mecatrônica', 'Eletrotécnica']
```

```
(1..15).each do |i|
```

```
  rk = format('ALU#%04d', i)
```

```
  curso = cursos[rand(cursos.length)]
```

```
  put 'alunos', rk, 'dados:nome', "Aluno#{i}"
```

```
  put 'alunos', rk, 'dados:curso', curso
```

```
  put 'alunos', rk, 'contato:email',  
  "aluno#{i}@exemplo.com"
```

```
  put 'alunos', rk, 'sys:created_at', Time.now.to_i.to_s  
end
```

8.2 Observação importante (sobrescrita da ALU#0001)

Como o loop começa em 1, ele vai inserir novamente ALU#0001 e **sobrescrever** algumas colunas (ex.: dados:nome pode virar Aluno1).

Se desejar **preservar** a ALU#0001 manual (Ana Souza), tem que rodar o loop a partir de 2:

```
(2..15).each do |i|  
  rk = format('ALU#%04d', i)  
  curso = cursos[rand(cursos.length)]  
  
  put 'alunos', rk, 'dados:nome', "Aluno#{i}"  
  put 'alunos', rk, 'dados:curso', curso  
  put 'alunos', rk, 'contato:email',  
  "aluno#{i}@exemplo.com"  
  
  put 'alunos', rk, 'sys:created_at', Time.now.to_i.to_s  
end
```

9. Validações pós-carga (conferência)

9.1 Contagem de linhas

```
count 'alunos'
```

9.2 Leitura parcial (amostra)

```
scan 'alunos', { LIMIT => 5 }
```

9.3 Consulta de uma linha específica

```
get 'alunos', 'ALU#0007'
```

No teste, ALU#0007 retornou vazio porque o loop **falhou** antes de inserir as linhas (por causa do erro do sample).

10. Remoção de coluna (teste de delete)

Testou:

```
delete 'alunos', 'ALU#0007', 'contato:email'
```

```
get 'alunos', 'ALU#0007'
```

Resultado: continuou vazio, porque a linha não existia.

Para validar o delete de verdade, tem que fazer uma row que exista (ex.: ALU#0001 ou outra criada após corrigir o loop).

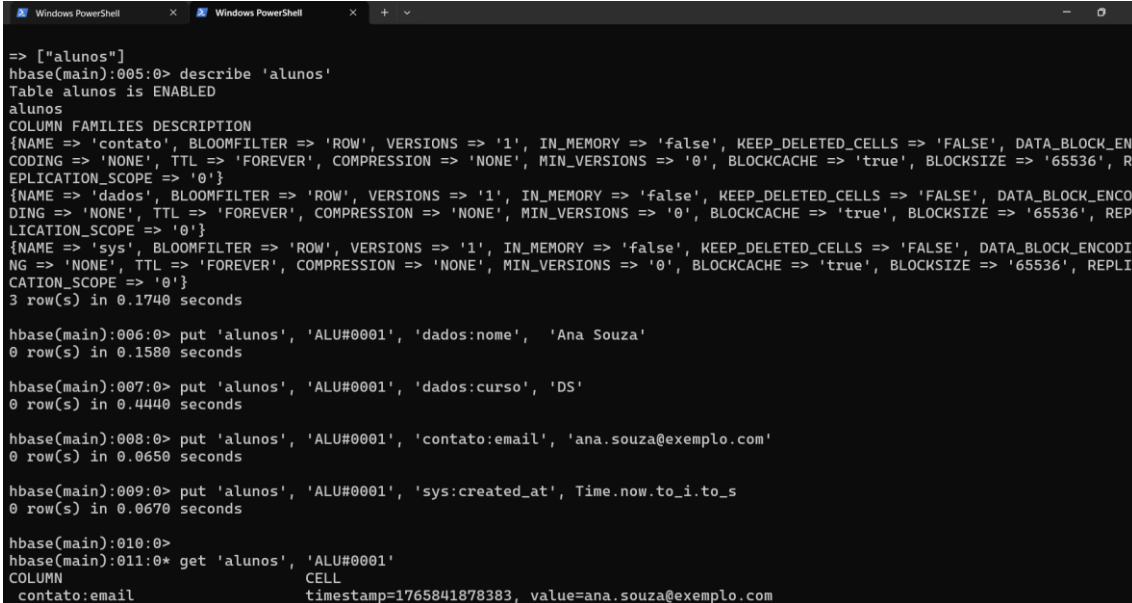
Exemplo:

```
delete 'alunos', 'ALU#0001', 'contato:email'
```

```
get 'alunos', 'ALU#0001'
```

11. Conclusão do teste

- O ambiente HBase no Docker está **funcionando** (criação de tabela, put, get, scan OK).
- O aviso do Hadoop nativo não impediu a execução.
- A carga automática falhou por incompatibilidade do método Array#sample no runtime do shell.
- A correção com rand resolve e permite gerar massa pequena para teste.



```
=> ["alunos"]
hbase(main):005:0> describe 'alunos'
Table alunos is ENABLED
alunos
COLUMN FAMILIES DESCRIPTION
{NAME => 'contato', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
{NAME => 'dados', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
{NAME => 'sys', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
3 row(s) in 0.1740 seconds

hbase(main):006:0> put 'alunos', 'ALU#0001', 'dados:nome', 'Ana Souza'
0 row(s) in 0.1580 seconds

hbase(main):007:0> put 'alunos', 'ALU#0001', 'dados:curso', 'DS'
0 row(s) in 0.4440 seconds

hbase(main):008:0> put 'alunos', 'ALU#0001', 'contato:email', 'ana.souza@exemplo.com'
0 row(s) in 0.0650 seconds

hbase(main):009:0> put 'alunos', 'ALU#0001', 'sys:created_at', Time.now.to_i.to_s
0 row(s) in 0.0670 seconds

hbase(main):010:0>
hbase(main):011:0> get 'alunos', 'ALU#0001'
COLUMN CELL
contato:email timestamp=1765841878383, value=ana.souza@exemplo.com
```

```

Windows PowerShell x Windows PowerShell + v
hbase(main):011:0> get 'alunos', 'ALU#0001'
COLUMN
contato:email timestamp=1765841878383, value=ana.souza@exemplo.com
dados:curso timestamp=1765841877911, value=DS
dados:nome timestamp=1765841877796, value=Ana Souza
sys:created_at timestamp=1765841878465, value=1765841878
4 row(s) in 0.0300 seconds

hbase(main):012:0> cursos = ['DS', 'Redes', 'Mecatrônica', 'Eletrotécnica']
=> ["DS", "Redes", "Mecatr\303\264nica", "Eletrot\303\251cnica"]
hbase(main):013:0>
hbase(main):014:0> (1..15).each do |i|
hbase(main):015:1> rk = format('ALU#%04d', i)
hbase(main):016:1> put 'alunos', rk, 'dados:nome', "Aluno#{i}"
hbase(main):017:1> put 'alunos', rk, 'dados:curso', cursos.sample
hbase(main):018:1> put 'alunos', rk, 'contato:email', "aluno#{i}@exemplo.com"
hbase(main):019:1> put 'alunos', rk, 'sys:created_at', Time.now.to_i.to_s
hbase(main):020:1> end
0 row(s) in 0.0130 seconds

NoMethodError: undefined method `sample' for ["DS", "Redes", "Mecatr\303\264nica", "Eletrot\303\251cnica"]:Array

hbase(main):021:0> count 'alunos'
1 row(s) in 0.0650 seconds

=> 1
hbase(main):022:0> scan 'alunos', { LIMIT => 5 }
ROW
ALU#0001 column=contato:email, timestamp=1765841878383, value=ana.souza@exemplo.com
ALU#0001 column=dados:curso, timestamp=1765841877911, value=DS
ALU#0001 column=dados:nome, timestamp=1765841884840, value=Aluno1
ALU#0001 column=sys:created_at, timestamp=1765841878465, value=1765841878
1 row(s) in 0.0230 seconds

```

```

Windows PowerShell x Windows PowerShell + v
hbase(main):016:1> put 'alunos', rk, 'dados:nome', "Aluno#{i}"
hbase(main):017:1> put 'alunos', rk, 'dados:curso', cursos.sample
hbase(main):018:1> put 'alunos', rk, 'contato:email', "aluno#{i}@exemplo.com"
hbase(main):019:1> put 'alunos', rk, 'sys:created_at', Time.now.to_i.to_s
hbase(main):020:1> end
0 row(s) in 0.0130 seconds

NoMethodError: undefined method `sample' for ["DS", "Redes", "Mecatr\303\264nica", "Eletrot\303\251cnica"]:Array

hbase(main):021:0> count 'alunos'
1 row(s) in 0.0650 seconds

=> 1
hbase(main):022:0> scan 'alunos', { LIMIT => 5 }
ROW
ALU#0001 column=contato:email, timestamp=1765841878383, value=ana.souza@exemplo.com
ALU#0001 column=dados:curso, timestamp=1765841877911, value=DS
ALU#0001 column=dados:nome, timestamp=1765841884840, value=Aluno1
ALU#0001 column=sys:created_at, timestamp=1765841878465, value=1765841878
1 row(s) in 0.0230 seconds

hbase(main):023:0> get 'alunos', 'ALU#0007'
COLUMN
CELL
0 row(s) in 0.0090 seconds

hbase(main):024:0> delete 'alunos', 'ALU#0007', 'contato:email'
0 row(s) in 0.0540 seconds

hbase(main):025:0> get 'alunos', 'ALU#0007'
COLUMN
CELL
0 row(s) in 0.0020 seconds

hbase(main):026:0> |

```

```
Windows PowerShell x Windows PowerShell x + v
0 row(s) in 0.0060 seconds
=> 1..15
hbase(main):039:0> count 'alunos'
15 row(s) in 0.0260 seconds

=> 15
hbase(main):040:0> scan 'alunos', { LIMIT => 5 }
ROW
ALU#0001      column=contato:email, timestamp=1765842737532, value=aluno1@exemplo.com
ALU#0001      column=dados:curso, timestamp=1765842737515, value=DS
ALU#0001      column=dados:nome, timestamp=1765842737499, value=Aluno1
ALU#0001      column=sys:created_at, timestamp=1765842737550, value=1765842737
ALU#0002      column=contato:email, timestamp=1765842737581, value=aluno2@exemplo.com
ALU#0002      column=dados:curso, timestamp=1765842737573, value=Redes
ALU#0002      column=dados:nome, timestamp=1765842737562, value=Aluno2
ALU#0002      column=sys:created_at, timestamp=1765842737589, value=1765842737
ALU#0003      column=contato:email, timestamp=1765842737615, value=aluno3@exemplo.com
ALU#0003      column=dados:curso, timestamp=1765842737605, value=DS
ALU#0003      column=dados:nome, timestamp=1765842737597, value=Aluno3
ALU#0003      column=sys:created_at, timestamp=1765842737624, value=1765842737
ALU#0004      column=contato:email, timestamp=1765842737646, value=aluno4@exemplo.com
ALU#0004      column=dados:curso, timestamp=1765842737639, value=DS
ALU#0004      column=dados:nome, timestamp=1765842737633, value=Aluno4
ALU#0004      column=sys:created_at, timestamp=1765842737653, value=1765842737
ALU#0005      column=contato:email, timestamp=1765842737676, value=aluno5@exemplo.com
ALU#0005      column=dados:curso, timestamp=1765842737669, value=DS
ALU#0005      column=dados:nome, timestamp=1765842737661, value=Aluno5
ALU#0005      column=sys:created_at, timestamp=1765842737682, value=1765842737
5 row(s) in 0.0400 seconds
hbase(main):041:0> |
```

```
hbase(main):041:0> get 'alunos', 'ALU#0007'
COLUMN      CELL
contato:email      timestamp=1765842737737, value=aluno7@exemplo.com
dados:curso        timestamp=1765842737729, value=Redes
dados:nome         timestamp=1765842737723, value=Aluno7
sys:created_at     timestamp=1765842737743, value=1765842737
4 row(s) in 0.0130 seconds
hbase(main):042:0> |
```

```
hbase(main):042:0> delete 'alunos', 'ALU#0001', 'contato:email'
0 row(s) in 0.0140 seconds

hbase(main):043:0> get 'alunos', 'ALU#0001'
COLUMN      CELL
dados:curso        timestamp=1765842737515, value=DS
dados:nome         timestamp=1765842737499, value=Aluno1
sys:created_at     timestamp=1765842737550, value=1765842737
3 row(s) in 0.0140 seconds
hbase(main):044:0> |
```