UNIVERSIDADE FEDERAL DE SÃO PAULO
INSTITUTO DE CIÊNCIA E TECNOLOGIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

TRABALHO FINAL
HBASE

Nomes: Ana Júlia de Oliveira Bellini **RA:** 111774
Victor Silva dos Santos **RA:** 112264
Willian Dihanster Gomes de Oliveira **RA:** 112269

SÃO JOSÉ DOS CAMPOS
2019

Introdução

HBase é um banco de dados lançado no ano de 2008, pela Apache Software Foundation, sendo um SGBD modelado a partir do Google BigTable, outro serviço de banco de dados. Trata-se de um projeto *open source*, com códigos escritos em linguagem Java.

É um *database* NoSQL, do tipo tabular, orientado a colunas. Também é parte do ecossistema Hadoop, com capacidade para armazenamento de grandes quantidades de dados.

Suas principais características são descritas na tabela a seguir.

Tolerância a falhas	– Replicação em todo o centro de dados – Operações atômicas e fortemente consistentes a nível de linha – Alta disponibilidade através de failover automático – Balanços automáticos de corte e carga de tabela
Rápido	– Pesquisas em tempo real – Armazenamento na memória por meio de filtros de bloqueio e bloqueio de blocos – Processamento do lado do servidor via filtros e co-processadores
Utilizável	– Modelo de dados acomoda ampla gama de casos de uso – Exportações de métricas através de plugins de arquivo e ganglia – Fácil Java API, bem como Thrift e REST gateway APIs

Tabela 1: Descrição das características do HBase.

Algumas empresas de grande porte já fizeram uso do HBase anteriormente, como o Facebook e o Twitter.

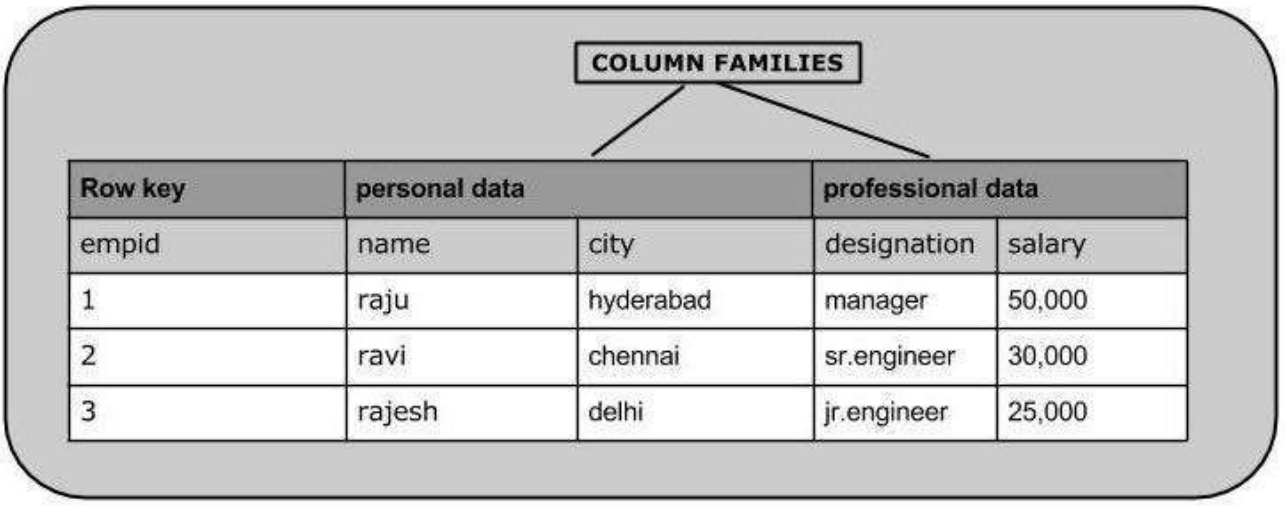
No caso do Facebook, HBase era utilizado para armazenamento de mensagens privadas dos usuários, enviadas e recebidas via Facebook Messenger, sendo adequado para aplicações pesadas, por conta de seu acesso aleatório e em tempo real a algum dado desejado. Outro ponto a se destacar é que a utilização do HBase pelo Facebook chegava a atingir 75 bilhões de operações de Read e Write por dia.

Já no caso do Twitter, o HBase é utilizado para realização de *backup* de todas as tabelas que a empresa possui em seu outro banco de dados MySQL. Utilizando as características do HBase para acessar dados para serem analisados.

Funcionamento

O banco de dados HBase, como já mencionado, é orientado a colunas, e ordenado com base em suas tuplas (ou *rows*). Ao contrário de outros SGBDs NoSQL, o HBase adota o conceito de *famílias*, dada por uma coleção de colunas que representam dados similares.

Um exemplo do modo de armazenamento do HBase pode ser visto na figura abaixo.



Row key	personal data		professional data	
empid	name	city	designation	salary
1	raju	hyderabad	manager	50,000
2	ravi	chennai	sr.engineer	30,000
3	rajesh	delhi	jr.engineer	25,000

Figura 1: Exemplo de armazenamento do HBase.
Fonte: https://www.tutorialspoint.com/hbase/hbase_quick_guide.htm

Como pode ser visto no exemplo, uma tabela pode conter qualquer número de famílias, e uma família pode ter várias colunas.

A Row Key funciona como uma chave única de cada tupla, tornando a pesquisa mais rápida e fácil. Ela pode ser comparada a Primary Key.

Além disso, cada célula desta tabela possui um *timestamp* associado a ela, gerado automaticamente pelo próprio SGBD, no momento da inserção deste dado no banco.

Para o funcionamento do SGBD, são necessários quatro componentes principais, sendo eles:

- HBase Master;
- Region Server;
- Regiões; e
- ZooKeeper.

Uma região é composta por faixas contínuas de Row Keys. Elas são divididas em outras regiões quando atingir um tamanho pré-fixado, o tamanho default é de 256Mb porém esse número pode aumentar caso necessário.

Os Region Servers são responsáveis de gerir e servir as suas regiões. Uma de funções seria, por exemplo, a de dividir as regiões que ela gerencia quando estas atingirem o tamanho máximo determinado.

O HBase Master realiza todas as operações de administração, como criar e deletar uma tabela, além de gerenciar e manter a consistência das Region Servers.

O ZooKeeper monitora toda a estrutura. Ele coordena o HBase Master, detecta pontos de falhas no sistema e além disso possui acesso ao Meta Table, arquivo de configuração na qual tem acesso a localização de todos os dados.

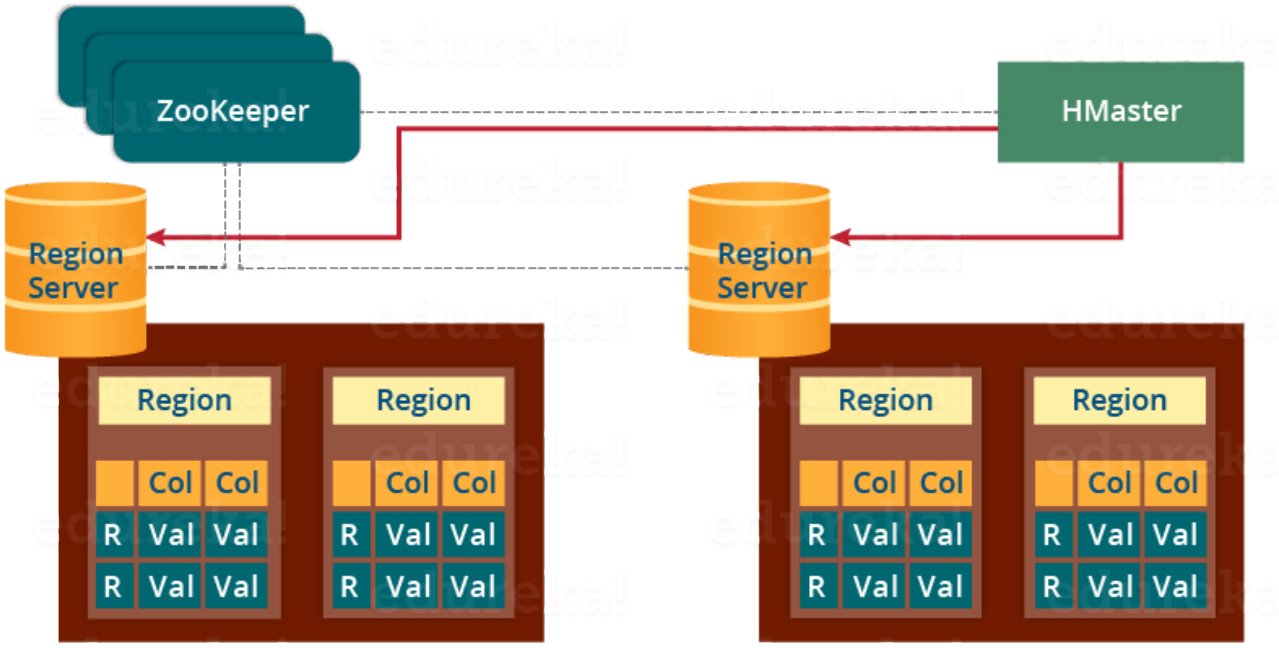


Figura 2: Fluxo de funcionamento do HBase.
Fonte: www.edureka.co/blog/hbase-architecture/

O processo de leitura e escrita são bem parecidos. O cliente requisita o acesso a um dado, podendo ser para ler, modificar ou deletar.

O ZooKeeper, que tem acesso a localização de todos os dados, recebe essa requisição do cliente onde é passado a Row Key da tupla. Após o ZooKeeper localizá-lo utilizando a Meta Table, ele repassa a informação para o cliente. Assim, ele se conecta ao Region Server que fica responsável por repassar todas as ações para a região onde está o dado.

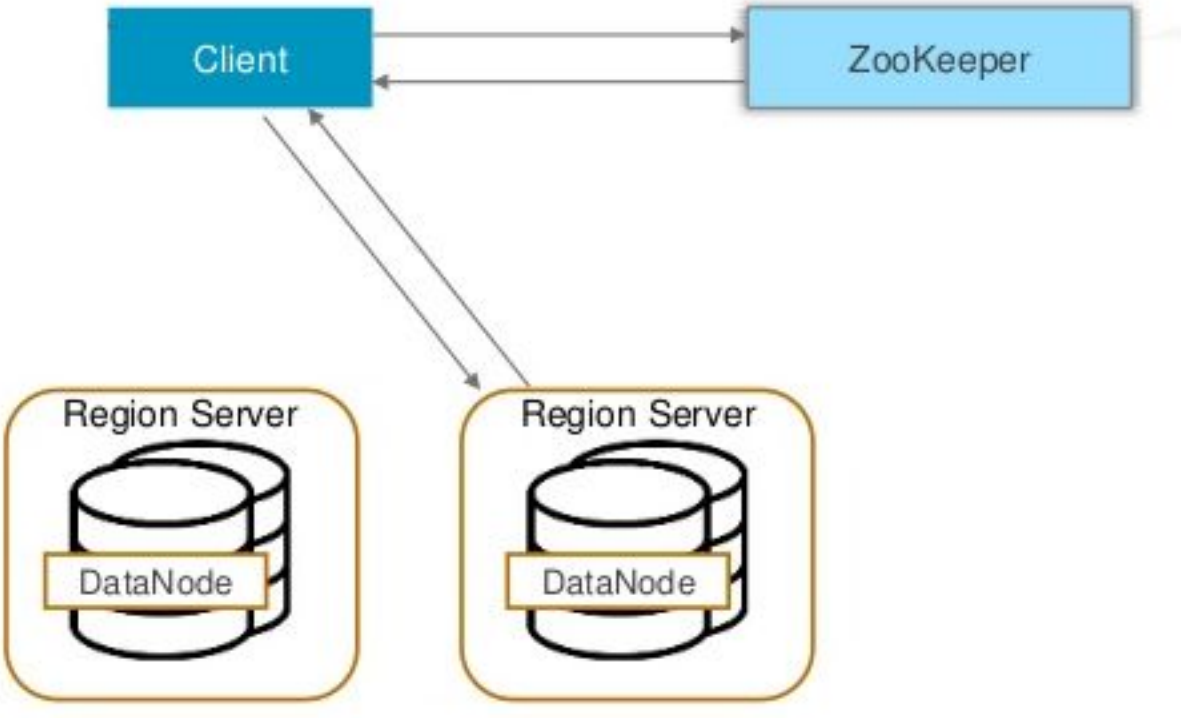


Figura 3: Operações de leitura e escrita.

Sintaxe

A seguir, a sintaxe dos principais comandos em HBase são apresentados. Nota-se que o HBase é mais um *datastore*, e com isso, não oferece consultas muito sofisticadas, tendo disponíveis apenas comandos bem básicos.

Para gerenciamento de tabelas, temos os seguintes comandos:

- Criação de tabelas
  - create <tablename>, <columnfamilyname>
- Listar tabela
  - list <tablename>
- Descrever tabela (Número de linhas, tamanho em disco)
  - describe <table name>
- Desativar tabela
  - disable <table name>
- Ativar tabela
  - enable <table name>
- Deletar tabela
  - drop <table name>
- Deletar todas tabelas
  - drop all
- Checar se a tabela existe
  - exists <table name>
- Alterar alguma informação da tabela
  - alter <tablename>, NAME=>'<column familyname>', VERSIONS=><new version no>

Já para os comandos de manipulação de dados, temos:

- Count (Retorna número de linhas da tabela)
  - count <'tablename'>, CACHE =>1000
- Put (Insere um dado na tabela)
  - put <'tablename'>, <'rowname'>, <'columnvalue'>, <'value'>
- Get (Retorna um dado de uma tabela)
  - get <'tablename'>, <'rowname'>, {< Additional parameters>}
- Delete (Deletar um dado)
  - delete <'tablename'>, <'row name'>, <'column name'>
- Delete all (Deleta uma linha)
  - delete all <'tablename'>, <'rowname'>
- Truncate (Desativar e criar uma nova tabela com o mesmo nome)
  - truncate <tablename>
- Scan (Faz a pesquisa na tabela de acordo com os parâmetros passados)
  - scan <'tablename'>, {Optional parameters}

Objetivos

O principal objetivo com a realização deste trabalho é o estudo de um banco NoSQL.

Neste caso, como objetivo específico, temos o estudo teórico do HBase, implementar e criar um banco, além de consultas com o banco utilizado.

Metodologia

Versões utilizadas:

- Hadoop: 2.8.0
- HBase: 1.4.9

Base de Dados

Para a base de dados, foi simulado o cenário dos ingressantes do ICT UNIFESP. Assim, utilizando o conceito de famílias do HBase, temos duas famílias de colunas: Dados Pessoais e Dados das Matérias. Sendo assim, foram gerados 300 exemplos com dados gerados aleatoriamente pelo Planilhas Google. Portanto, temos:

Row Key:

- RA;

Dados Pessoais:

- Nome;
- Sexo;
- Turno;
- Idade;
- Endereço;
- Telefone.

Matérias (notas entre 0 e 10):

- Cálculo em uma Variável (CUV);
- Ciência, Tecnologia e Sociedade (CTS);
- Fundamentos de Biologia Moderna (FBM);
- Lógica de Programação (LP);
- Química Geral (QG).

Uma visualização dos dados gerados pode ser conferida na Figura 4, onde é apresentada uma captura de tela sobre a planilha de dados dos ingressantes construída.

Row Key	Dados pessoais						Matérias				
RA	Nome	Sexo	Turno	Idade	Endereço	Telefone	CUV	LP	FBM	QG	CTS
1	Agatha	F	I	20	Alta Floresta D'Oeste	996411709	6	7	8	9	10
2	Agatha	F	N	19	Ariquemes	987290624	9	7	6	4	2
3	Alice	F	I	18	Cabixi	987732944	2	10	0	8	1
4	Alice	F	I	18	Cacoal	987104795	3	9	1	3	1
5	Alícia	F	N	19	Cerejeiras	993009071	6	9	6	10	4
6	Alícia	F	N	19	Colorado do Oeste	987661942	8	6	0	3	5
7	Álvaro	M	I	20	Corumbiara	990620370	3	10	10	2	0
8	Álvaro	M	N	18	Costa Marques	986393871	5	5	8	7	4
9	Ana Clara	F	I	18	Espigão D'Oeste	991453763	3	5	10	0	1
10	Ana Clara	F	I	20	Guajará-Mirim	997757730	7	5	2	10	9
11	Ana Júlia	F	I	20	Jaru	991338717	0	2	5	1	0
12	Ana Laura	F	I	19	Ji-Paraná	996103608	2	6	1	5	9
13	Ana Laura	F	I	20	Machadinho D'Oeste	984914994	7	6	2	4	5
14	Ana Luiza	F	I	20	Nova Brasilândia D'Oeste	993657082	4	6	10	5	8
15	André	M	I	20	Ouro Preto do Oeste	993130652	9	9	3	4	0
16	André	M	I	19	Pimenta Bueno	994075035	4	6	6	9	7
17	Anthony	M	I	19	Porto Velho	995325752	6	8	4	2	3

Figura 4: Base de dados criada para simulação do problema.

Criação das tabelas

Utilizando a sintaxe apresentada anteriormente, criou-se a tabela do problema modelo (dados e notas dos ingressantes). Assim, a tabela criada deve possuir duas famílias de colunas: dados pessoas e dados das matérias. Na Figura 5, é apresentada a criação da tabela 'Alunos', com as famílias de colunas 'DadosPessoais' e 'DadosMaterias'. Também verificou-se com o comando scan 'Alunos' que a tabela está vazia.

```
hbase(main):005:0> create 'Alunos', 'DadosPessoais', 'DadosMaterias'
0 row(s) in 1.2550 seconds

=> Hbase::Table - Alunos
hbase(main):006:0> list 'Alunos'
TABLE

Alunos

1 row(s) in 0.0220 seconds

=> ["Alunos"]
hbase(main):007:0> scan 'Alunos'
ROW                                    COLUMN+CELL

0 row(s) in 0.0270 seconds
```

Figura 5: Criação da tabela de Alunos no terminal.

Em seguida, verificamos o uso de comandos como o exists, onde apenas para testes dos comandos, verificamos se de fato a tabela existia. Também foi testado a deleção da tabela, assim utilizou o comando *disable* 'Alunos' e *drop* 'Alunos' para deletar a tabela e confirmou-se essa exclusão com o comando exists, novamente, onde dessa vez retornou que a tabela não existe, como o esperado. Esses comandos e resultados podem ser visualizados na Figura 6.

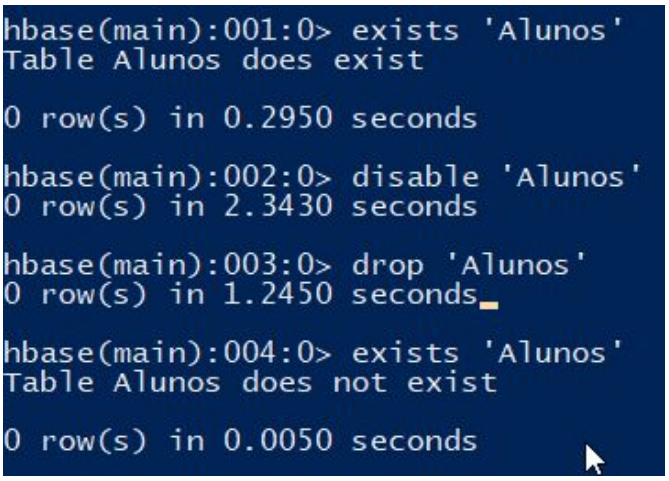


Figura 6: Verificação de existência e deleção da tabela de Alunos.

Inserções

Para inserir a base de dados construída no HBase foram tentadas a leitura diretamente por *csv* e por conexão com a API de Java, mas sem sucesso. Dessa forma, foi criado um *script* em Python, que faz a leitura do arquivo .*csv* e transforma cada atributo em código de inserção de acordo com a sintaxe do HBase. Uma visualização desse *script* pode ser vista na Figura 7.

```
for i in range(0, 300):
    print('put \'Alunos\',', i, ',' #('\'' + str(i) + '\','),
          '\'DadosPessoais:Nome\',', ('\'' + planilha.values[i][1] + '\''))
    print('put \'Alunos\',', i, ',' #('\'' + str(i) + '\','),
          '\'DadosPessoais:Sexo\',', ('\'' + planilha.values[i][2] + '\''))
    print('put \'Alunos\',', i, ',' #('\'' + str(i) + '\','),
          '\'DadosPessoais:Turno\',', ('\'' + planilha.values[i][3] + '\''))
    print('put \'Alunos\',', i, ',' #('\'' + str(i) + '\','),
          '\'DadosPessoais:Idade\',', ('\'' + str(planilha.values[i][4]) + '\''))
    print('put \'Alunos\',', i, ',' #('\'' + str(i) + '\','),
          '\'DadosPessoais:Endereco\',', ('\'' + planilha.values[i][5] + '\''))
    print('put \'Alunos\',', i, ','#('\'' + str(i) + '\','),
          '\'DadosPessoais:Telefone\',', ('\'' + str(planilha.values[i][6]) + '\''))
    print('put \'Alunos\',', i, ','#('\'' + str(i) + '\','),
          '\'DadosMaterias:CUV\',', ('\'' + str(planilha.values[i][7]) + '\''))
    print('put \'Alunos\',', i, ','#('\'' + str(i) + '\','),
          '\'DadosMaterias:LP\',', ('\'' + str(planilha.values[i][8]) + '\''))
    print('put \'Alunos\',', i, ','#('\'' + str(i) + '\','),
          '\'DadosMaterias:FBM\',', ('\'' + str(planilha.values[i][9]) + '\''))
    print('put \'Alunos\',', i, ','#('\'' + str(i) + '\','),
          '\'DadosMaterias:QG\',', ('\'' + str(planilha.values[i][10]) + '\''))
    print('put \'Alunos\',', i, ','#('\'' + str(i) + '\','),
          '\'DadosMaterias:CTS\',', ('\'' + str(planilha.values[i][11]) + '\''))
```

Figura 7: Script para geração de código na sintaxe do HBase.

Já o arquivo gerado pelo gerador de código HBase pode ser visualizado na Figura 8.

```
put 'Alunos', 0 ,'DadosPessoais:Nome', 'Agatha'
put 'Alunos', 0 ,'DadosPessoais:Sexo', 'F'
put 'Alunos', 0 ,'DadosPessoais:Turno', 'I'
put 'Alunos', 0 ,'DadosPessoais:Idade', '20'
put 'Alunos', 0 ,'DadosPessoais:Endereco', 'Alta Floresta D'Oeste'
put 'Alunos', 0 ,'DadosPessoais:Telefone', '992261720'
put 'Alunos', 0 ,'DadosMaterias:CUV', '6'
put 'Alunos', 0 ,'DadosMaterias:LP', '7'
put 'Alunos', 0 ,'DadosMaterias:FBM', '8'
put 'Alunos', 0 ,'DadosMaterias:QG', '9'
put 'Alunos', 0 ,'DadosMaterias:CTS', '10'
put 'Alunos', 1 ,'DadosPessoais:Nome', 'Agatha'
put 'Alunos', 1 ,'DadosPessoais:Sexo', 'F'
put 'Alunos', 1 ,'DadosPessoais:Turno', 'N'
put 'Alunos', 1 ,'DadosPessoais:Idade', '19'
put 'Alunos', 1 ,'DadosPessoais:Endereco', 'Ariquemes'
put 'Alunos', 1 ,'DadosPessoais:Telefone', '992080489'
put 'Alunos', 1 ,'DadosMaterias:CUV', '5'
put 'Alunos', 1 ,'DadosMaterias:LP', '10'
put 'Alunos', 1 ,'DadosMaterias:FBM', '8'
put 'Alunos', 1 ,'DadosMaterias:QG', '5'
put 'Alunos', 1 ,'DadosMaterias:CTS', '3'
```

Figura 8: Código gerado para inserir no HBase.

Assim, para inserir os dados, apenas copiou-se esse textos nas linhas de comando do terminal para inserir os 300 dados no HBase, conforme Figura 9.

```
hbase(main):3287:0> put 'Alunos', 298 ,'DadosMaterias:FBM', '2'
0 row(s) in 0.0020 seconds

hbase(main):3288:0> put 'Alunos', 298 ,'DadosMaterias:QG', '10'
0 row(s) in 0.0010 seconds

hbase(main):3289:0> put 'Alunos', 298 ,'DadosMaterias:CTS', '2'
0 row(s) in 0.0010 seconds

hbase(main):3290:0> put 'Alunos', 299 ,'DadosPessoais:Nome', 'Yasmin'
0 row(s) in 0.0000 seconds

hbase(main):3291:0> put 'Alunos', 299 ,'DadosPessoais:Sexo', 'F'
0 row(s) in 0.0010 seconds

hbase(main):3292:0> put 'Alunos', 299 ,'DadosPessoais:Turno', 'N'
0 row(s) in 0.0000 seconds

hbase(main):3293:0> put 'Alunos', 299 ,'DadosPessoais:Idade', '18'
0 row(s) in 0.0020 seconds

hbase(main):3294:0> put 'Alunos', 299 ,'DadosPessoais:Endereco', 'Cutias'
0 row(s) in 0.0000 seconds

hbase(main):3295:0> put 'Alunos', 299 ,'DadosPessoais:Telefone', '984263473'
0 row(s) in 0.0000 seconds

hbase(main):3296:0> put 'Alunos', 299 ,'DadosMaterias:CUV', '7'
0 row(s) in 0.0010 seconds

hbase(main):3297:0> put 'Alunos', 299 ,'DadosMaterias:LP', '8'
0 row(s) in 0.0010 seconds

hbase(main):3298:0> put 'Alunos', 299 ,'DadosMaterias:FBM', '5'
0 row(s) in 0.0010 seconds

hbase(main):3299:0> put 'Alunos', 299 ,'DadosMaterias:QG', '3'
0 row(s) in 0.0010 seconds

hbase(main):3300:0> put 'Alunos', 299 ,'DadosMaterias:CTS', '3'
0 row(s) in 0.0020 seconds
```

Figura 9: Inserção dos dados no HBase.

Também é possível agora visualizar e confirmar que os dados foram inseridos, conforme Figura 10. Com o comando scan ‘Alunos’, foram retornadas todas as 300 linhas, como esperado.

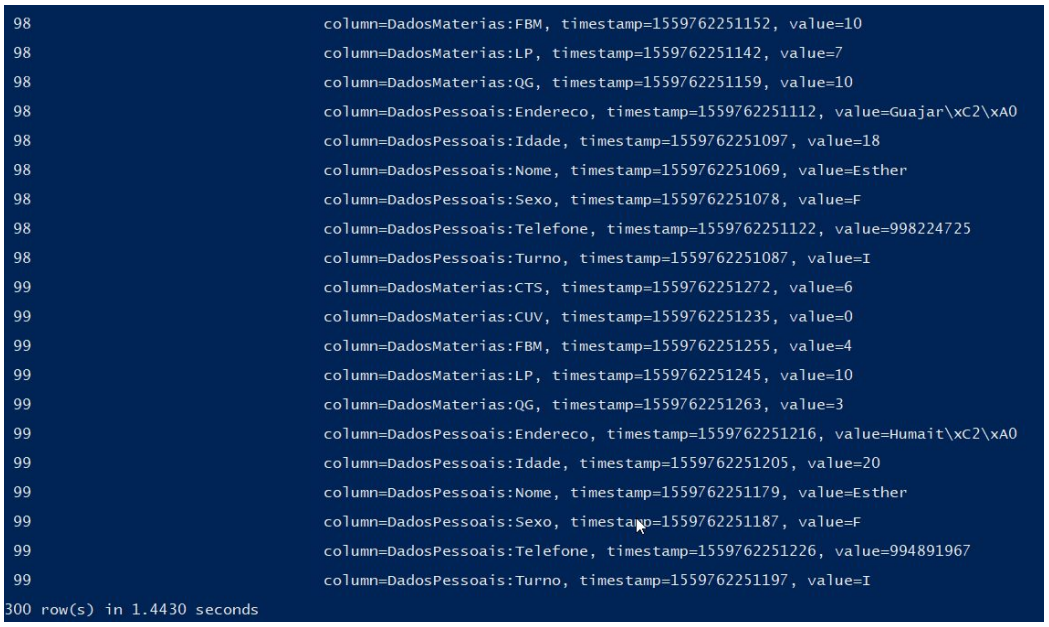


Figura 10: Visualização dos dados inseridos no HBase através do scan Alunos.

Consultas

Foram realizadas cinco consultas diferentes ao banco de dados implementado, feitas diretamente no HBase Shell, sendo elas:

1. Aprovados na disciplina de CUV;
2. Ingressantes do sexo feminino;
3. Alunos do período noturno;
4. Reprovados na disciplina de LP; e
5. Alunos com mais de 25 anos.

O comando e o resultado de cada uma das consultas pode ser visto a seguir:

- **Aprovados na disciplina de CUV**
  Para a primeira consulta, onde deseja-se visualizar os alunos com nota maior ou igual a 6 na disciplina de CUV, o comando utilizado é:

  ```
  scan 'Alunos', {COLUMNS => 'DadosMaterias:CUV', LIMIT => 300, FILTER
  => "(ValueFilter(>=, 'binary:6') OR ValueFilter(=, 'binary:10'))"}
  ```

  Gerando o seguinte resultado no HBase Shell:

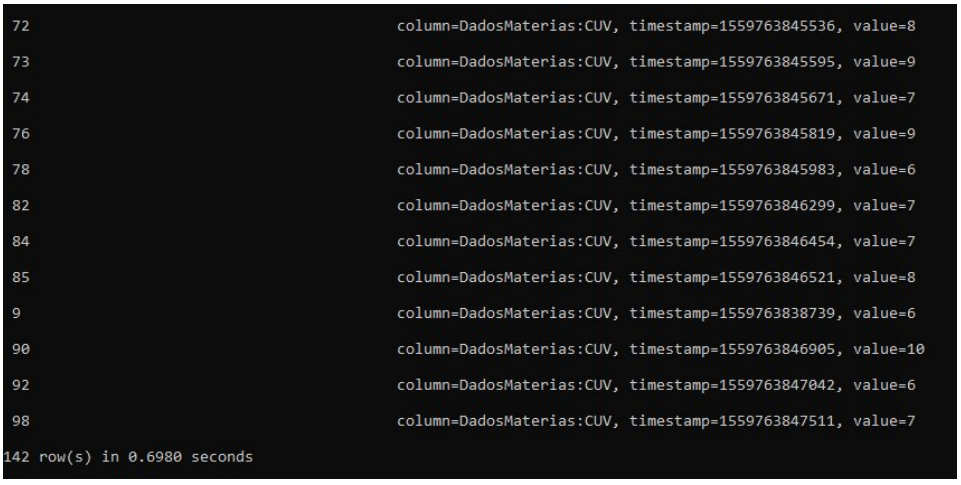


Figura 11: Consulta aos aprovados na disciplina de CUV.

- **Ingressantes do sexo feminino**
  Para este caso, onde deseja-se obter uma lista das mulheres do campus, o comando desenvolvido é:

```
scan 'Alunos', {COLUMNS => 'DadosPessoais:Sexo', LIMIT => 300, FILTER
=> "ValueFilter(=, 'regexstring:F')"}
```

  Onde o resultado obtido se dá por:

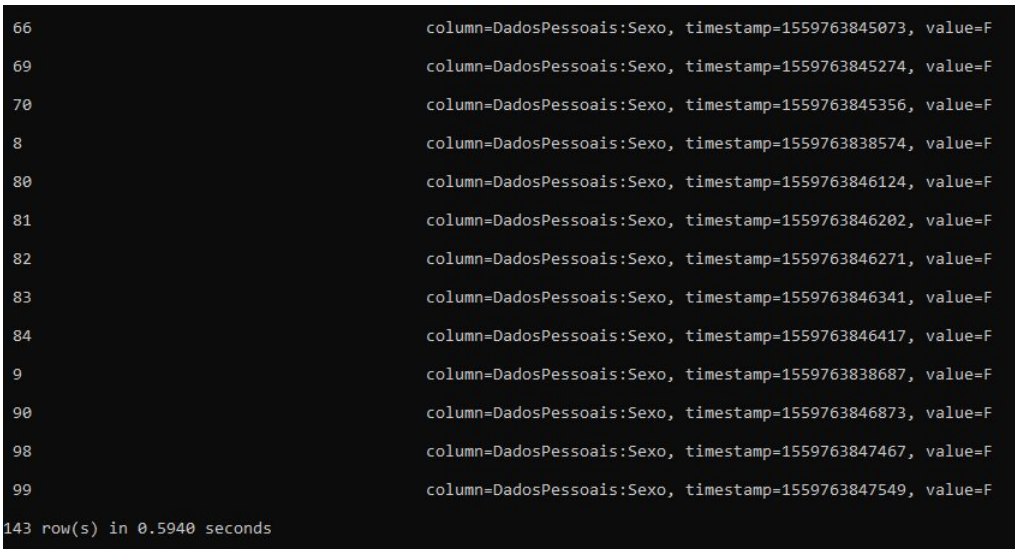


Figura 12: Consulta aos ingressantes do sexo feminino.

- **Alunos do período noturno**
  Para a listagem de todos os alunos que estudam à noite, utilizou-se o seguinte comando:

```
scan 'Alunos', {COLUMNS => 'DadosPessoais:Turno', LIMIT => 300,
FILTER => "ValueFilter( =, 'regexstring:N')"}
```

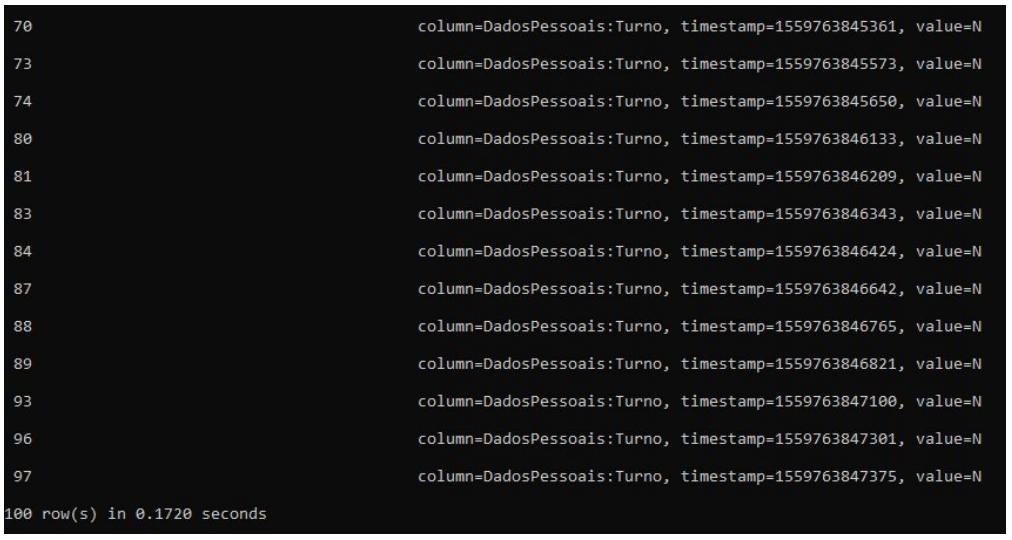


Figura 13: Consulta aos alunos do período noturno.

- **Reprovados na disciplina de LP**
  A quarta consulta realizada destina-se a listar todos os alunos com nota abaixo de 6 em LP, e é dada pelo seguinte comando:

  ```
  scan 'Alunos', {COLUMNS => 'DadosMaterias:LP', LIMIT => 300, FILTER
  => "ValueFilter(<, 'binary:6') AND ValueFilter(!=, 'binary:10')"}
  ```

  Retornando o seguinte resultado:

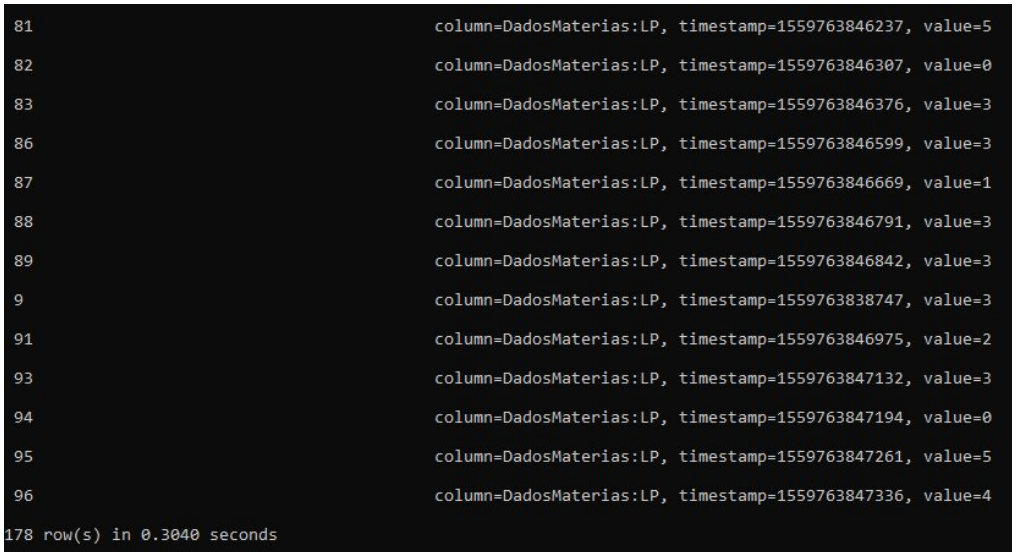


Figura 14: Consulta aos reprovados na disciplina de LP.

- **Alunos com mais de 25 anos**
  Por fim, com a última consulta, desejamos saber quem são os alunos com idade maior que 25 anos, com base no comando abaixo:

  ```
  scan 'Alunos', {COLUMNS => 'DadosPessoais:Idade', LIMIT => 300,
  FILTER => "ValueFilter(>, 'binary:25')"}
  ```

Onde obtém-se o seguinte resultado:

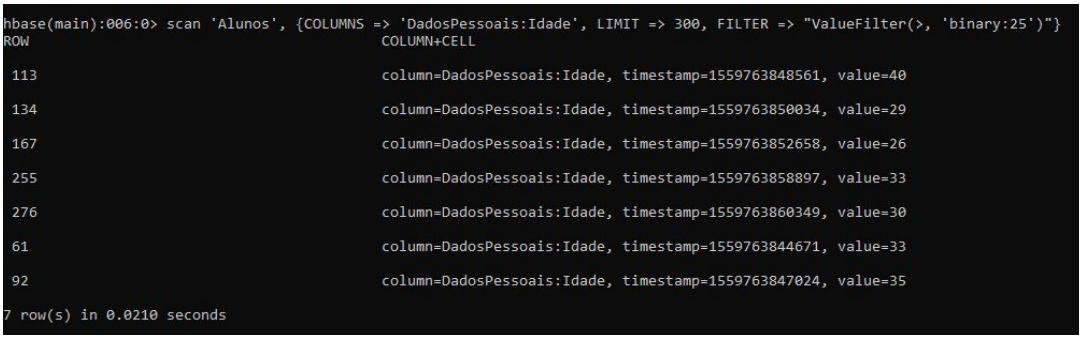


Figura 15: Consulta aos alunos com mais de 25 anos.

Considerações Finais

Com a realização deste trabalho, foi possível ver na prática como o “paradigma” NoSQL funciona. O HBase é um exemplo de abordagem NoSQL do tipo orientado a coluna. Com ele, novos conceitos também foram apresentados como o de famílias de colunas, o que pode ser muito interessante para algumas aplicações.

O HBase é bastante indicado para uma grande quantidade de dados e é facilmente integrado ao Hadoop, que é uma plataforma de software em Java de computação distribuída voltada para clusters e processamento, com atenção à tolerância de falhas. Dito isso, pode ser utilizado para aplicações de *Big Data*, que é uma área que vem se tornado cada vez mais presente.

Porém, o HBase ainda possui pouco uso comercial e com isso, carece de informações e tutoriais na internet. Dessa forma, houveram alguns problemas durante o desenvolvimento deste trabalho. Para a instalação foram gastos quases 2 dias seguindo tutoriais que sempre resultavam em erro em algum momento, devido a versionamentos, incompatibilidades, *etc*. Após a instalação, também ocorreram outros erros, como na hora de fazer a leitura de uma arquivo *.csv* para inserção de dados no banco, o que nos levou à criação de um *script* que gerasse código na sintaxe do HBase.

Outro ponto a se ressaltar é o fato de que grande parte de sua utilização é feita através de janela de comandos, ou por códigos em Java, o que torna o sistema não muito intuitivo, pela falta de uma interface para facilitar seu uso, como existe para outros SGBDs, como MySQL (com Workbench) e PostgreSQL (interface pelo pgAdmin).

Por fim, recomenda-se o HBase para aplicações onde o volume de dados é muito grande, mas que não seja necessário fazer muitas consultas, pois o HBase serve mais como um *datastore*, fornecendo apenas alguns serviços de consultas básicos.

Referências Bibliográficas

FUNDAMENTOS do HBase. [*S. l.*], [201-]. Disponível em: https://www.ibm.com/support/knowledgecenter/pt-br/SSPT3X_4.1.0/com.ibm.swg.im.infosphere.biginsights.analyze.doc/doc/hbaseConcepts.html. Acesso em: 1 jun. 2019.

FILIPA, Sara. **Apache HBase: O que é, Conceitos e Definições**. [*S. l.*], [201-]. Disponível em: https://www.cetax.com.br/blog/o-que-e-o-apache-hbase/. Acesso em: 1 jun. 2019.

PINTO, Pedro. **Apache HBase – Base de dados NoSQL do ecossistema Hadoop**. [*S. l.*], 2018. Disponível em: https://pplware.sapo.pt/informacao/apache-hbase-not-only-sql-nosql/. Acesso em: 2 jun. 2019.

HBASE - Quick Guide. [*S. l.*], 2018. Disponível em: https://www.tutorialspoint.com/hbase/hbase_quick_guide.htm. Acesso em: 2 jun. 2019.

FERREIRA, Felipe. **Conhecendo Apache HBase**. [*S. l.*], 2014. Disponível em: https://www.infoq.com/br/presentations/conhecendo-apache-hbase/. Acesso em: 2 jun. 2019.

BAHIA, Milton. **HBase**. [*S. l.*], 2017. Disponível em: https://www.slideshare.net/MiltonBahia/hbase-81317238?. Acesso em: 2 jun. 2019.