

Banco de Dados NoSQL

(NS0501)

Prof. Giovane Barcelos
giovane_barcelos@uniritter.edu.br

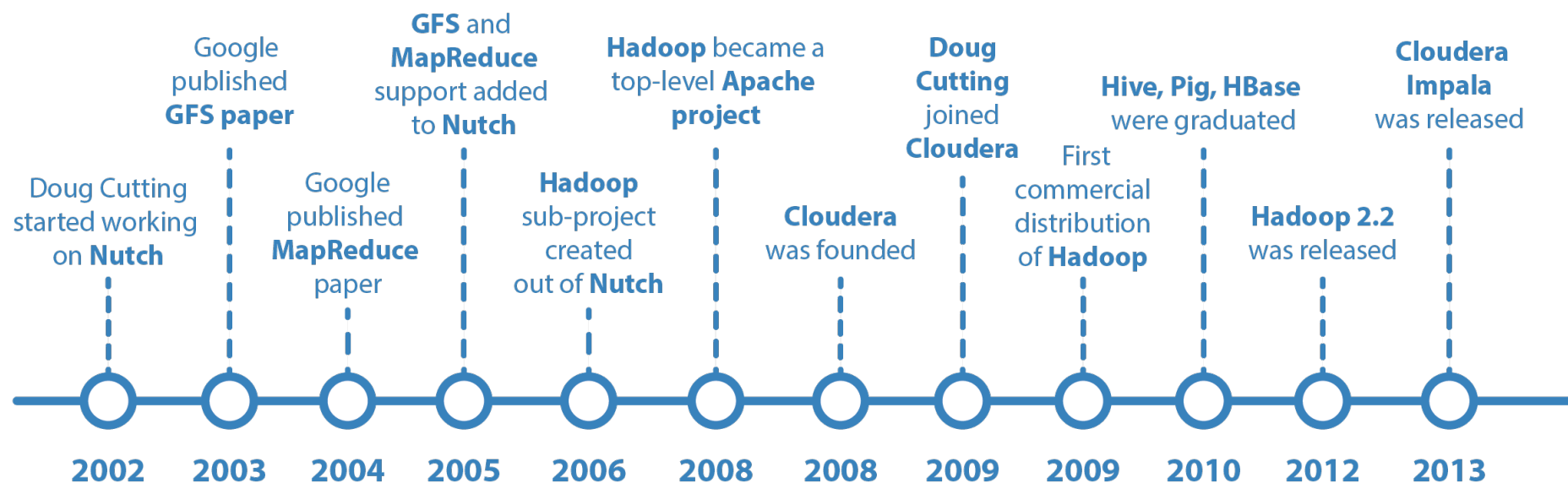
O que vamos aprender?

- ***HBase e Hadoop***
- **Tutoriais com as seguinte ferramentas / tecnologias:**
 - ✓ ***Sqoop***
 - ✓ ***Impala***
 - ✓ ***Avro***
 - ✓ ***Parquet***
 - ✓ ***Hive***
 - ✓ ***Hue***
 - ✓ ***Flume***
 - ✓ ***Spark***
 - ✓ ***Solr***
 - ✓ ***MorphLines***
- ***SQL***

Banco de Dados em Coluna *HBase* e *Hadoop*

O que é o *Hadoop*?

- **Framework** mantido pela fundação **Apache** que auxilia no processamento de **grandes massas de dados** (\geq **petabytes**)
- É um canivete suíço dos dados com **muitos componentes** padrões e outros plugáveis
- Breve **história** do *Hadoop*:




Banco de Dados em Coluna *HBase* e *Hadoop*

Quem é o *HBase*?

- **Nativo** do *Hadoop*, apesar da possibilidade do **Cassandra**
- Banco de dados **orientado a coluna**
- **Alta escalabilidade** com **bilhões** de linhas em *sharding*
- Usa *HDFS* e suporta *MapReduce*
- **NÃO**: é *full ACID*, possui **esquema**, **tipagem**, **transações**, *stored procedures* ou *Joins*
- **Acessos**: *Shell*, *REST*, *Java* / *Thrift Api*

Família de Colunas



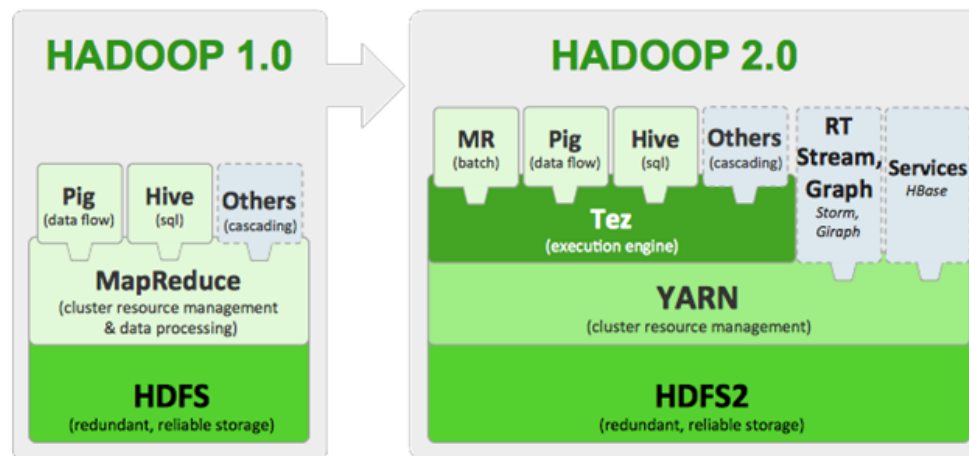
Chave ID	Dados Pessoais		Dados Profissionais	
emp_id	nome	cidade	cargo	salario
1	Penélope	Porto Alegre	Gerente	20000
2	Pepé Legal	Canoas	Eng.	15000
3	Pateta	Viamão	Analista	8000

Tabela Empresa

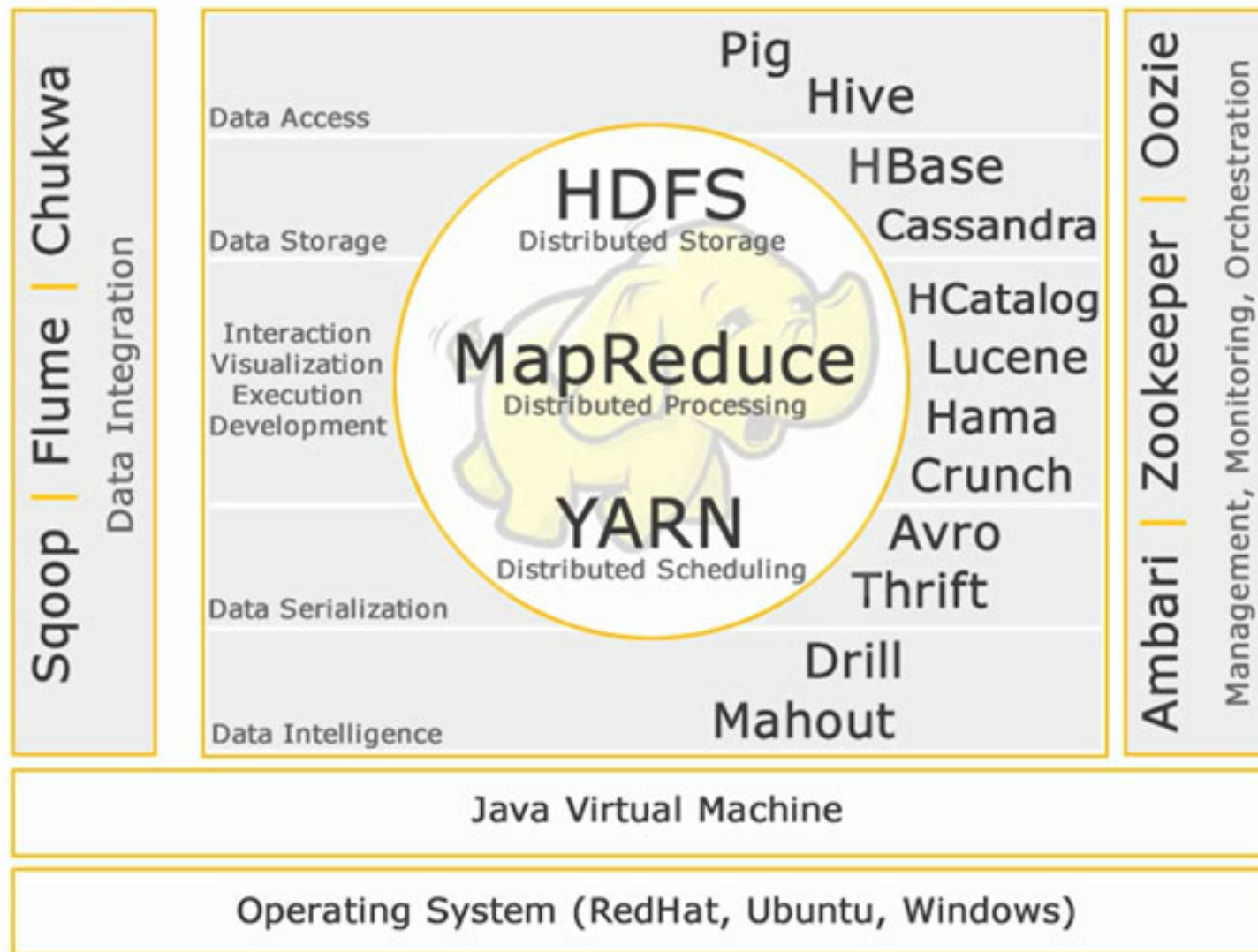
Banco de Dados em Coluna *HBase* e *Hadoop*

Quais são os componentes do *Hadoop*?

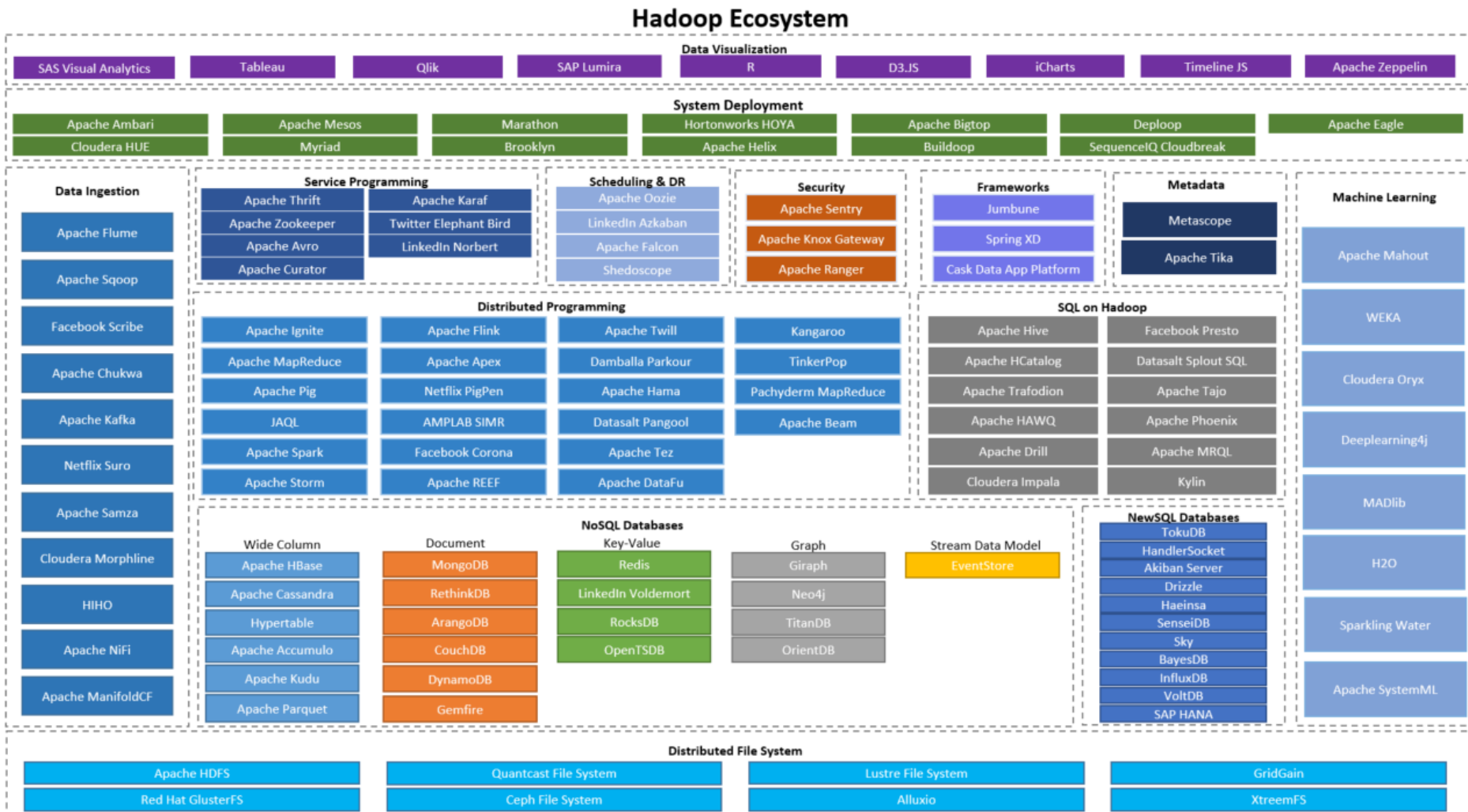
- Os principais componentes são:
 - ✓ *Hadoop Common*: API e utilitários do ecossistema
 - ✓ *Hadoop Distributed File System (HDFS)*: sistema de arquivo distribuído que garante a **escalabilidade** e auto desempenho
 - ✓ *Hadoop YARN*: agendamento de trabalhos e gerenciamento de recursos do *cluster*
 - ✓ *Hadoop MapReduce*: baseado no **YARN** para processamento paralelo de grandes massas de dados



Banco de Dados em Coluna *HBase* e *Hadoop* E o ecossistema do *Hadoop*?






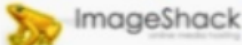













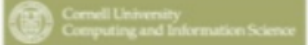



























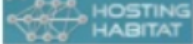





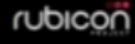












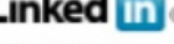
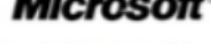






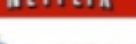



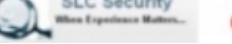





Banco de Dados em Coluna *HBase* e *Hadoop* Ecossistema em mais detalhes



Banco de Dados em Coluna *HBase* e *Hadoop*

Quem usa *Hadoop*?

2007	2008	2009	2010
  	                      	                         	                             

Banco de Dados em Coluna *HBase* e *Hadoop*

Apache *Hadoop* no mundo real

- **Facebook** → logs internos, análise de dados, aprendizado de máquina, **2 clusters**:
 - ✓ **1100** nodos (8 cores, 12 TB storage), 12 PB
 - ✓ **300** nodos (8 cores, 12 TB storage), 3 PB
- **LinkedIn** → 3 clusters:
 - ✓ **800** nodos (2x4 cores, 24 GB RAM, 6x2 TB SATA), 9 PB
 - ✓ **1900** nodes (2x6 cores, 24 GB RAM, 6x2 TB SATA), 22 PB
 - ✓ **1400** nodes (2x6 cores, 32 GB RAM, 6x2 TB SATA), 16 PB
- **Spotify** → geração de conteúdo, agregação de dados, relatórios, análise de dados:
 - ✓ **1650** nodos, 43000 cores, 70 TB RAM, 65 PB, 20000 tarefas diárias
- **Yahoo!** → **40000** nodos com Hadoop, maior cluster:
 - ✓ **4500** nodes (2x4 cores, 16 GB RAM, 4x1 TB storage), 17 PB

Banco de Dados em Coluna *HBase* e *Hadoop*

Quando Usar e Não Usar *Hadoop*?

➤ Não Usar quando:

- ✓ É necessário análise dos dados em **tempo real**
- ✓ **Pequena quantidade** de dados
- ✓ Apenas como **substituto da infraestrutura** existente
- ✓ **Java não é uma opção**

➤ Quando Usar:

- ✓ **Processamento** de dados em **larga escala**
- ✓ **Escalabilidade horizontal** é essencial
- ✓ **Diversidade** no armazenamento dos dados
- ✓ **Necessita integrar** com **muitos frameworks** de *Big Data* (*Mahout*, *R*, *Pentaho*, etc)



Banco de Dados em Coluna *HBase* e *Hadoop*

Onde o *Hadoop* é usado?

Mercado	Caso de Uso
Tecnologia	Pesquisa Recomendações <i>Matching</i>
Bancos	Detecção de Fraudes Gerenciamento de Riscos
Varejo	Análise de Marketing Serviços ao Cliente Recomendações de Produtos
Manufatura	Manutenção preventiva Análise de Restrições (TOC)

Banco de Dados em Coluna *HBase* e *Hadoop*

Como instalar o *HBase* e *Hadoop*

➤ **Instalação *Apache Hadoop*:**

<http://bit.do/GuiaInstalacaoDefinitivoHadoop>

➤ **Docker:**

<https://hub.docker.com/r/sequenceiq/hadoop-docker/> e

<https://hub.docker.com/r/sequenceiq/>

➤ **Distribuições *Hadoop*:**

✓ **HortonWorks:**

<https://br.hortonworks.com/downloads/#sandbox>

✓ **Cloudera:**

<https://www.cloudera.com/downloads.html>

→ Utilizaremos esta distribuição nos nossos tutoriais devido a VM com as principais ferramentas que roda em 4GB e 1CPU

✓ **MapR:**

<http://bit.do/MapRInstall>

Banco de Dados em Coluna *HBase*

Um pouco mais do *Hbase*. Modelo de Dados.

Mercado	Caso de Uso
Namespace (ns)	Agrupamento lógico de tabelas
Tabela (tb)	Uma tabela (ns:tb) armazena linhas
Linhas	Chave de linha e colunas com valores
Família de Coluna (fc)	Agrupamento de colunas
Colunas (col)	Armazena os valores (fc:col)
Célula	Combinação de linha e coluna + <i>timestamp</i>
Timestamp	versões que mudam após a atualização

- **Importante:** *hbase shell* armazena todos os dados como **String**
- **Ex:** Tabela **Funcionario** com **famílias** de colunas (**pe** = Dados Pessoais e **pf** = Dados Profissionais). **Timestamp** não foi apresentado

emp_id	pe:nome	pe:cidade	pf:cargo	pf:salario
1	Penélope	Porto Alegre	Gerente	20000
2	Pepé Legal	Canoas	Eng.	15000

Banco de Dados em Coluna *HBase*

Principais comandos

Comando	Objetivo
get	retorna os atributos de uma linha
put	adiciona/atualiza coluna(s)
increment	Incrementa valor(es) de coluna(s)
scan	pesquisa/itera sobre linha(s)
delete	remove linha(s), coluna(s) ou família(s) dados são marcados para exclusão e removidos durante compactação
create	cria tabela e especifica a(s) família(s)
alter	altera as propriedades da tabela
describe	propriedades da família da tabela/coluna
list	lista as tabelas
create_namespace	cria um <i>namespace</i>
drop_namespace	remove um <i>namespace</i>

Referência Rápida: http://hbase.apache.org/apache_hbase_reference_guide.pdf

Banco de Dados em Coluna *HBase*

CRUD HBase

0. Para executar os comandos, vá para o terminal e digite
hbase shell

1. Listar todas as tabelas
list

2. Criar uma tabela chamada de aluno com duas famílias 'a' e 'b'
create 'aluno', cf=['a', 'b']

3. Adicionar valores na tabela aluno nas suas respectivas células
put 'aluno', 'IdMonica', 'a:nome', 'Monica'
put 'aluno', 'IdMonica', 'a:idade', 20
put 'aluno', 'IdPepe', 'a:nome', 'Pepe Legal'
put 'aluno', 'IdPepe', 'b:curso', 'Informatica'

4. Listar dados da tabela aluno
scan 'aluno'

Banco de Dados em Coluna *HBase*

CRUD HBase

5. Procurar a chave 'IdPepe' na tabela aluno

get 'aluno', 'IdPepe'

6. Listar, com expressão regular, tabelas que iniciam com aluno

list 'aluno.*'

7. Apresentar o "esquema" da tabela aluno

describe 'aluno'

8. Limitar em 5 versões o armazenamento da tabela 'aluno'

alter 'aluno', **NAME** => 'a', **VERSION** => 5

9. Pesquisar as colunas, limitado a duas linhas e que

conttenham 'Id' no prefixo do chave

scan 'aluno', {**COLUMNS**=>['a:idade','a:nome'], **LIMIT**=>2,
ROWPREFIXFILTER =>'Id'}

Banco de Dados em Coluna *HBase*

CRUD HBase

10. Pesquisar as colunas das linhas que iniciam com 'Id' e terminam em 'IdP'

```
scan 'aluno', { COLUMNS=>['a:idade','a:nome'],  
                STARTROW =>'Id', ENDROW => 'IdP'}  
scan 'aluno', { FILTER => 'KeyOnlyFilter()' }
```

11. Pesquisar aluno com prefixo 'Id' e primeira ocorrência

```
scan 'aluno',{ FILTER => "(PrefixFilter ('Id')) AND  
                          ColumnCountGetFilter(1)" }
```

12. Pesquisar aluno com a substring 'legal' no nome

```
scan 'aluno', { FILTER =>  
    "SingleColumnValueFilter('a','nome',=,'substring:legal')"} 
```

13. Pesquisar aluno com idade maior que 19

```
scan 'aluno',{ COLUMNS=>['a:idade'],  
    FILTER => "SingleColumnValueFilter('a','idade',>,'binary:19')"} 
```

Banco de Dados em Coluna *HBase*

Filtros no *HBase*

- Os **filtros** são **classes Java** restringindo pesquisas
- **Lista de filtros**: combina vários filtros com **AND** e **OR**
- **Compara** valores de **coluna(s)**: menor, igual, maior, substring, prefixo, ...
- **Compara metadados**: **família** de coluna e **qualificador**
 - ✓ Filtro de **prefixo de qualificador**: retornar (primeiro) colunas correspondentes
 - ✓ Filtro de **faixa de coluna**: retorna uma faixa de colunas (por exemplo, aa-az)
- **Compara nomes de linhas**
 - ✓ **Nota**: é preferível usar o *scan*



Banco de Dados em Coluna *HBase*

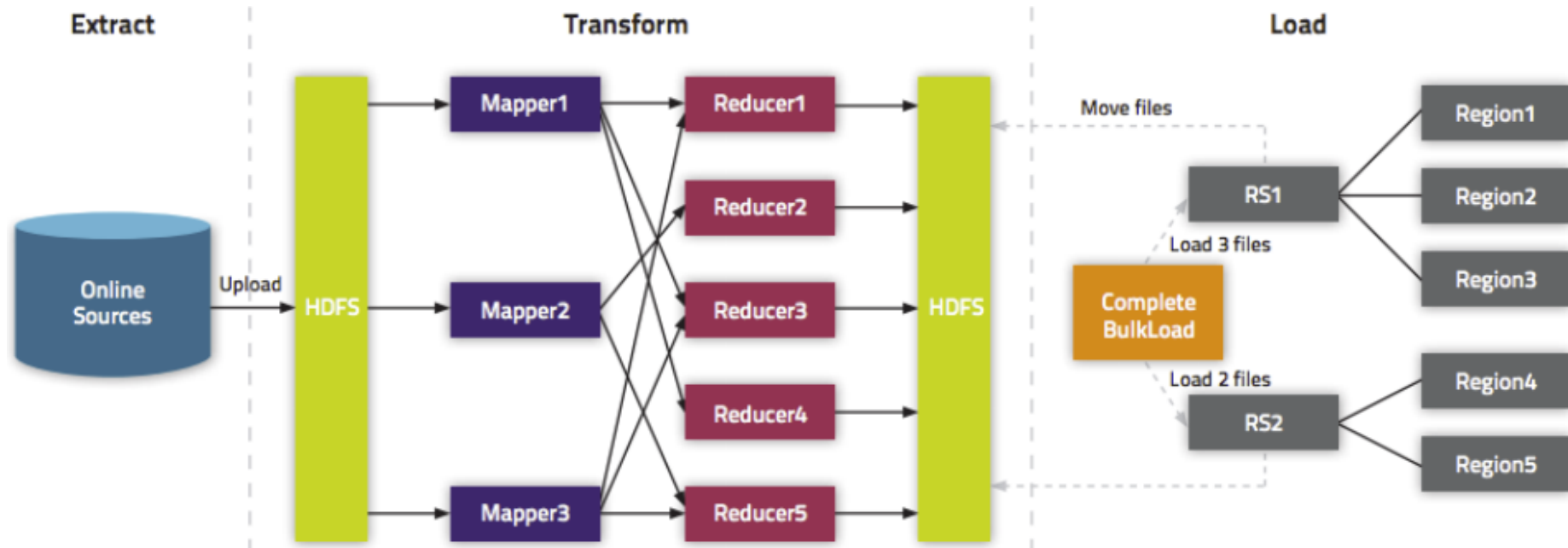
Sobre consistência

- Chaves de linha não podem ser alteradas
- Exclui logicamente até a compactação
- Forte consistência de leitura e escrita
- Todas as operações são atômicas
- As famílias de várias colunas de uma linha podem ser alteradas atômicamente
- Alterações linhas não são atômicas
- Ordem das alterações simultâneas não é definida
- Operações bem sucedidas são duráveis
- A tupla (linha, coluna, versão) especifica a célula
- Normalmente, a versão é o um *timestamp*, mas pode ser alterada
- Qualquer ordem de versões pode ser escrita
- *Get* e *scan* retornam as versões mais recentes mais podem não ser as mais novas devido a replicação

Banco de Dados em Coluna *HBase*

Processo Geral de *ETL* (*Extract*, *Transform* e *Load*)

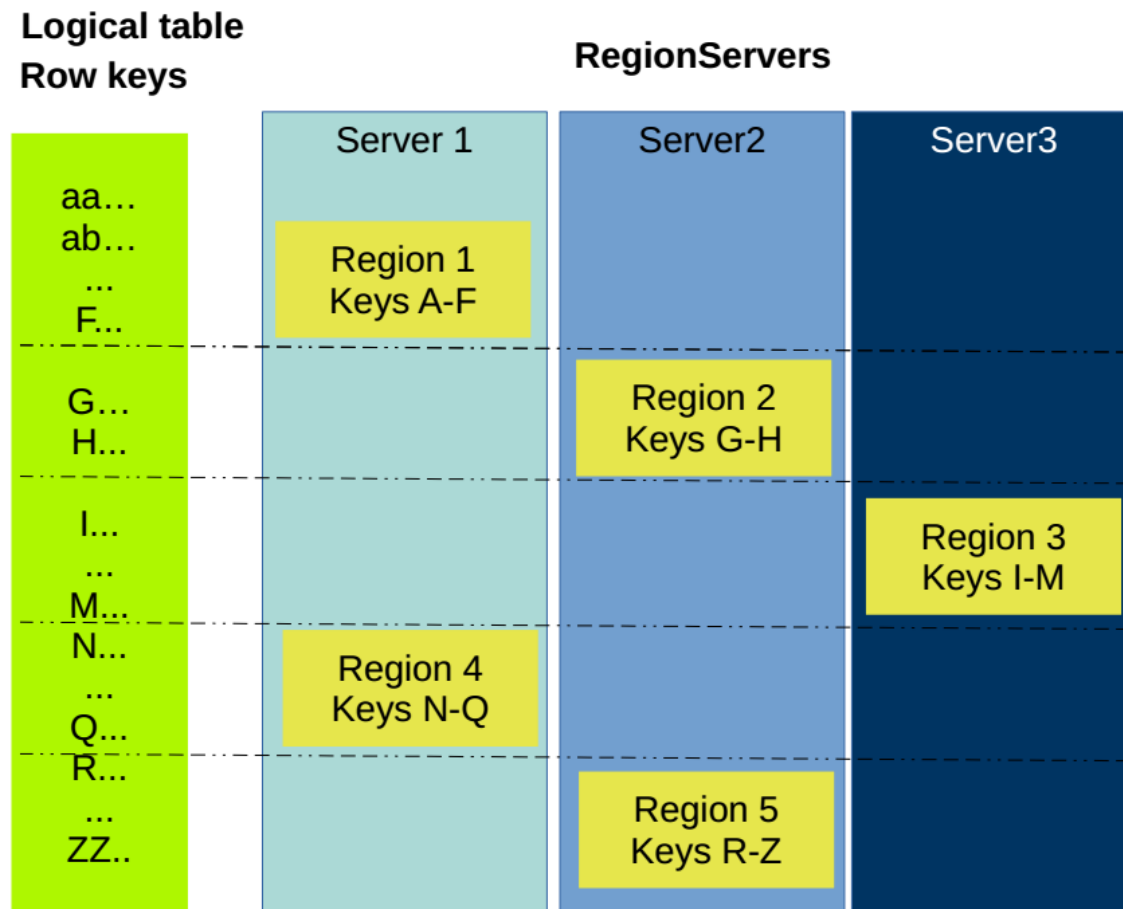
1. Extrai (***Extract***) dados e geralmente importa para *HDFS*
2. Transforma (***Transform***) os dados em HFiles usando MapReduce
3. Carrega (***Load***) os arquivos no HBase informando o RegionServer



Banco de Dados em Coluna *HBase*

E o *Sharding*, como funciona?

- O *sharding* ocorre pela distribuição das chaves nas regiões dos servidores, desta forma é importante criar uma chave que indique a região no prefixo



Banco de Dados em Coluna *HBase*

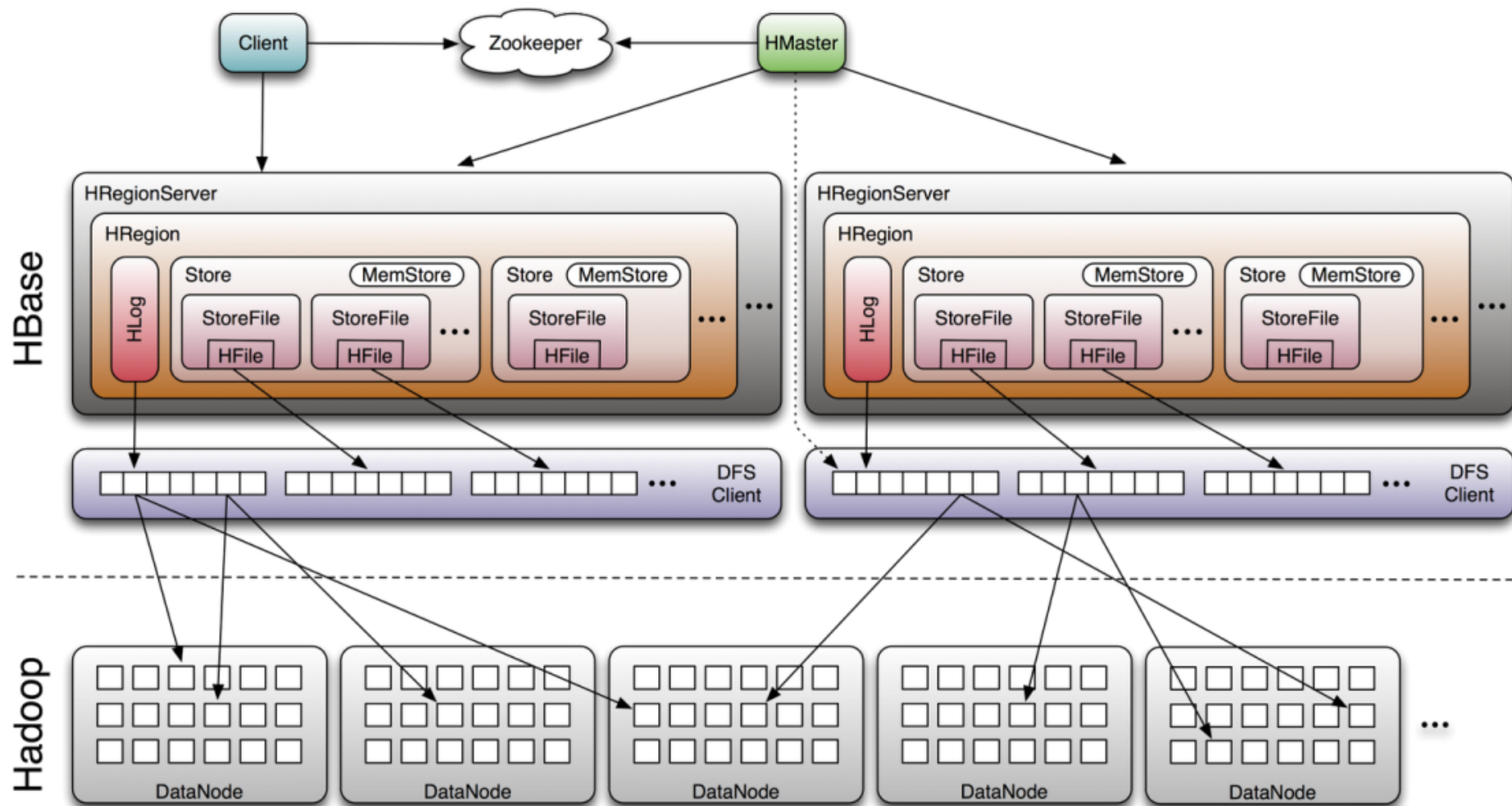
Formato de armazenamento do *HFile*

- Os dados da célula são mantidos em arquivos no *HDFS*
- Índice de várias camadas com filtros e suporte a *snapshot*
- Ordenada pela chave
- Somente inclui, a exclusão marca logicamente a linha
- O processo de compactação combina vários arquivos

Tamanho da Linha short	Chave da Linha byte[]	Tamania da Família byte	Família da Coluna byte[]	Nome da coluna byte[]	<i>Timestamp</i> long	Tipo de Chave byte
----------------------------------	---------------------------------	-----------------------------------	------------------------------------	---------------------------------	---------------------------------	------------------------------

Banco de Dados em Coluna *HBase*

Perspectiva de armazenamento de auto nível



Banco de Dados em Coluna *HBase*

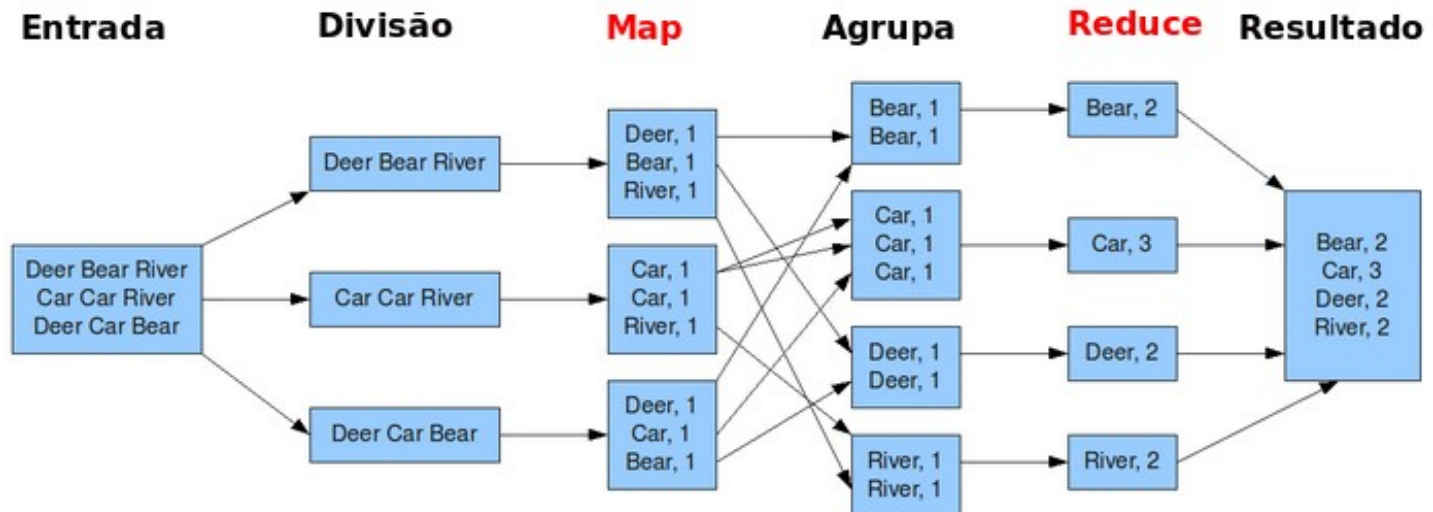
HBase suporta *MapReduce*

➤ Função *Map*:

- ✓ **Entrada:** um par de chave/valor (registro de entrada)
- ✓ **Saída:** um conjunto intermediário de pares chave/valor
- ✓ **(chave, valor) → lista de (chave, valor)**

➤ Função *Reduce*:

- ✓ **Entrada:** uma chave intermediária + um conjunto de (todos) valores desta chave
- ✓ **Saída:** um possível conjunto de valores menor para esta chave
→ a partir do mesmo domínio
- ✓ **(chave, lista de valores) → (chave, lista de valores)**



Banco de Dados em Coluna *HBase*

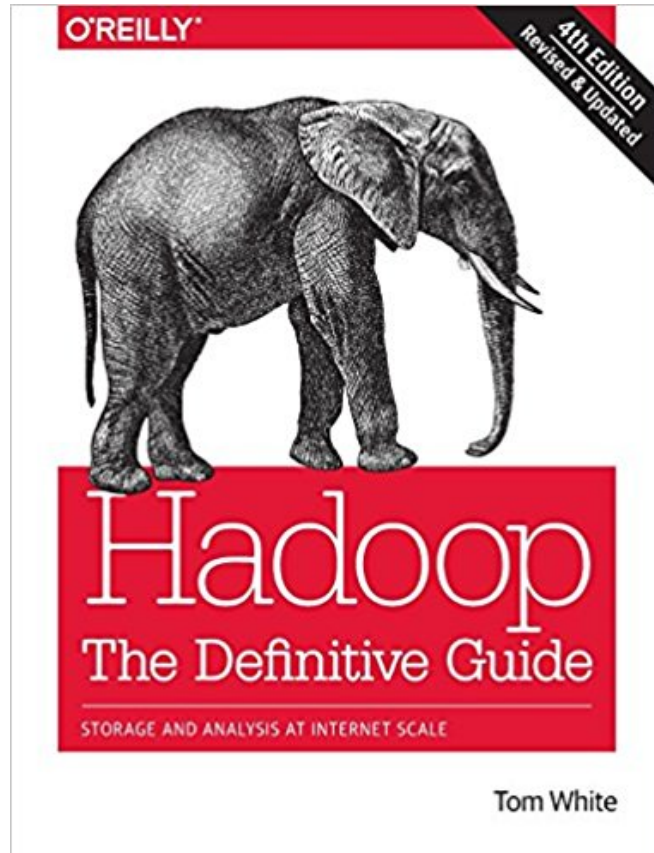
Exemplo: Frequência de Palavra

```
/**
 * Função map
 * @param chave Identificador do Documento
 * @param valor Conteúdo do Documento
 */
map(String chave, String valor) {
    foreach palavra in valor: emit(palavra, 1);
}

/**
 * Função reduce
 * @param chave Palavra em particular
 * @param valores Lista de valores da contagem de palavras
 */
reduce(String chave, Iterator valores) {
    int resultado = 0;
    foreach valor in valores: resultado += valor;
    emit(chave, resultado);
}
```

Banco de Dados em Coluna *HBase* e *Hadoop*

Bibliografias Recomendadas



“ A nova fonte de poder não é o dinheiro nas mãos de poucos, mas informação nas mãos de muitos. ”



John Naisbitt