

# BIG DATA



# **APLICAÇÕES DE BIG DATA COM HADOOP**

**Disciplina: Aplicações de Big Data com Hadoop**

**Tema da Aula: HIVE**

## **Coordenação:**

**Prof. Dr. Adolpho Walter  
Pimazzi Canton**

**Profª. Dra. Alessandra de  
Ávila Montini**

**Prof. Bruno Paulinelli**

**Junho de 2016**

## Formação

- Engenharia de Computação – USJT
- Mestrando estatística – USP

## Experiência

- Professor dos cursos de MBA e extensão Big Data na FIA
- Pesquisador do laboratório de análise de dados LABDATA-FIA
- Professor no curso Inteligência na Gestão de Dados na POLI PECE
- Arquiteto de Soluções Big Data no Itaú Unibanco
- Implementação do Hadoop no Itaú Unibanco e pesquisas de novas tecnologias para Big Data (2013-2014)
- Analista de Business Intelligence no Itaú Unibanco (2007-2013)
- Analista de Business Intelligence na Totvs/Microsiga (2004-2007)

## Conteúdo da Aula

- Hive – Introdução
- Exemplo - WordCount
- Comandos – no HDFS e no S.O. pelo Hive
- Carga - Load, put e Insert
- DDL – External table, Manage table e View
- Funções - Split e Explode
- HUE
- Formatos de arquivos - Posicional, delimitado e json
- Extração de dados - Get e insert directory
- DML – Manipulação de dados
- HiveQL
- Performance - Map Join, particionamennto, paralelismo, bucket
- Hive OverHead
- Um pouco sobre Impala

## Hive - Introdução

- O Hive é um framework para soluções de Data Warehousing no ambiente Hadoop.
- Contruído pelo do Facebook em 2007.
- Hive foi criado também visando aproveitar os "skills" de uso do SQL dos analistas e desenvolvedores do Facebook, que não eram na época tão proficientes em Java para usar o MapReduce.
- Atualmene é um projeto da Apache ( <http://hive.apache.org>).
- O Hive utiliza uma linguagem chamada HiveQL (Hive Query Language) , que transforma as sentenças SQL em Jobs MapReduce executados no cluster Hadoop.

## Hive - Introdução

- O Hive não é um banco de dados.
- Interpreta instruções SQL para Job MapReducer.
- Le os dados de arquivos estruturados e semi-estruturados contidos no HDFS, e se baseia em metadados para simular uma tabela de um banco de dados relacional.
- O Hive não foi desenhado para executar queries em real time, com baixa latência.
- Foi desenhado para melhor performance analisando grandes quantidades de dados que se encontram em clusters.

## Hive - Introdução

- O Hive e o Hadoop não possuem um SGBD.
- Escrever query da melhor maneira
- Não existe inferência no plano de execução (por enquanto)
- Maior flexibilidade
- Independência
- Performance



# SQL para Job MapReducer

## Word Count

```
1 package org.myorg;
2
3 import java.io.IOException;
4 import java.util.*;
5
6 import org.apache.hadoop.fs.Path;
7 import org.apache.hadoop.conf.*;
8 import org.apache.hadoop.io.*;
9 import org.apache.hadoop.mapreduce.*;
10 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
11 import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
12 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
13 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
14
15 public class WordCount {
16
17     public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
18         private final static IntWritable one = new IntWritable(1);
19         private Text word = new Text();
20
21         public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
22             String line = value.toString();
23             StringTokenizer tokenizer = new StringTokenizer(line);
24             while (tokenizer.hasMoreTokens()) {
25                 word.set(tokenizer.nextToken());
26                 context.write(word, one);
27             }
28         }
29     }
30
31     public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
32
33         public void reduce(Text key, Iterable<IntWritable> values, Context context)
34             throws IOException, InterruptedException {
35             int sum = 0;
36             for (IntWritable val : values) {
37                 sum += val.get();
38             }
39             context.write(key, new IntWritable(sum));
40         }
41     }
42
43     public static void main(String[] args) throws Exception {
44         Configuration conf = new Configuration();
45
46         Job job = new Job(conf, "wordcount");
47
48         job.setOutputKeyClass(Text.class);
49         job.setOutputValueClass(IntWritable.class);
50
51         job.setMapperClass(Map.class);
52         job.setReducerClass(Reduce.class);
53
54         job.setInputFormatClass(TextInputFormat.class);
55         job.setOutputFormatClass(TextOutputFormat.class);
56
57         FileInputFormat.addInputPath(job, new Path(args[0]));
58         FileOutputFormat.setOutputPath(job, new Path(args[1]));
59
60         job.waitForCompletion(true);
61     }
62
63 }
```





# SQL para Job MapReducer

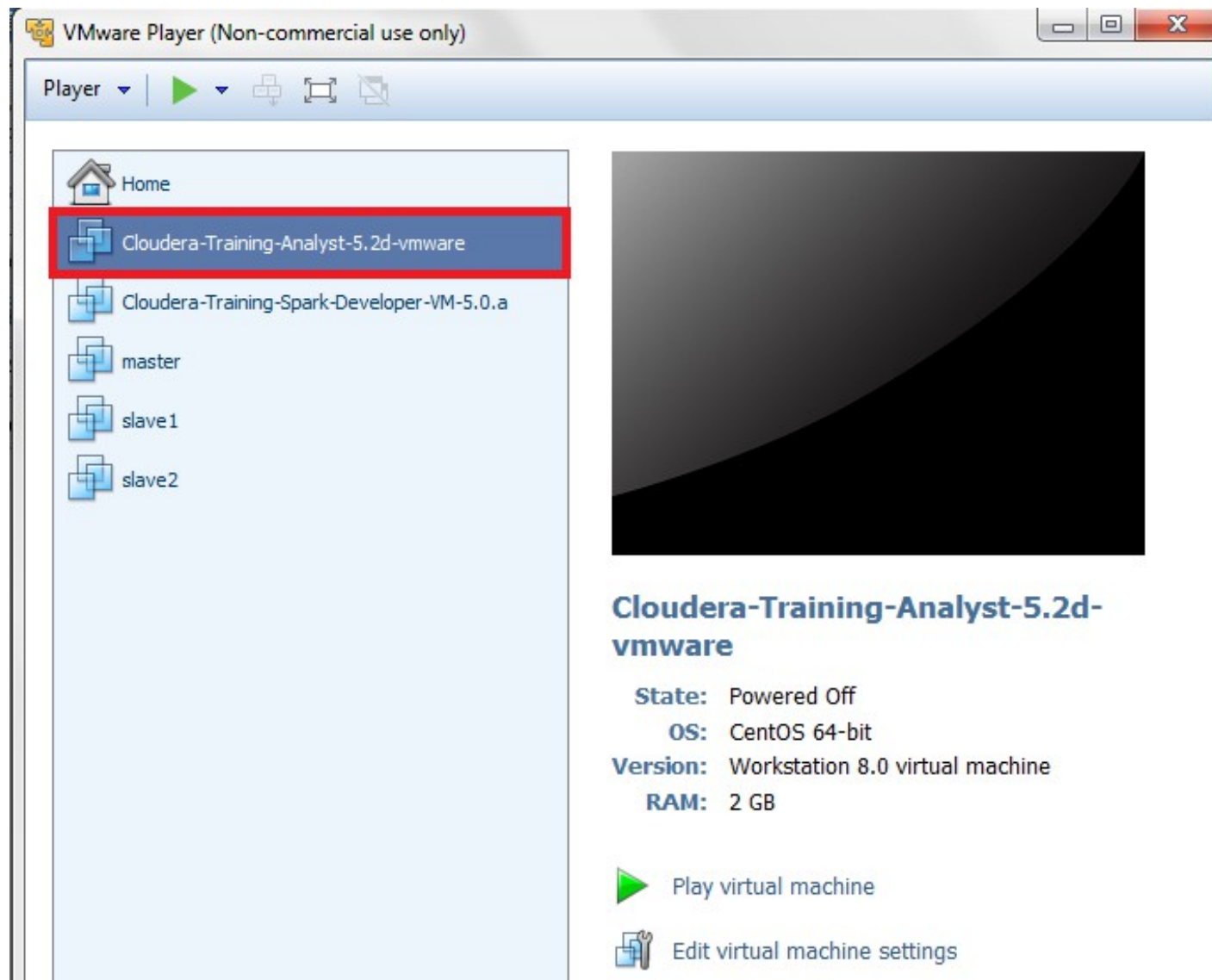
## Word Count

```
1 package org.myorg;
2
3 import java.io.IOException;
4 import java.util.*;
5
6 import org.apache.hadoop.fs.Path;
7 import org.apache.hadoop.conf.*;
8 import org.apache.hadoop.io.*;
9 import org.apache.hadoop.mapreduce.*;
10 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
11 import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
12 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
13 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
14
15 public class WordCount {
16
17     public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
18         private final static IntWritable one = new IntWritable(1);
19         private Text word = new Text();
20
21         public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
22             String line = value.toString();
23             StringTokenizer tokenizer = new StringTokenizer(line);
24             while (tokenizer.hasMoreTokens()) {
25
26
27                 1 select word,count(1) as count
28
29             }
30         }
31     public
32
33     publ
34     th
35
36
37
38
39
40     }
41 }
42
43 public static void main(String[] args) throws Exception {
44     Configuration conf = new Configuration();
45
46     Job job = new Job(conf, "wordcount");
47
48     job.setOutputKeyClass(Text.class);
49     job.setOutputValueClass(IntWritable.class);
50
51     job.setMapperClass(Map.class);
52     job.setReducerClass(Reduce.class);
53
54     job.setInputFormatClass(TextInputFormat.class);
55     job.setOutputFormatClass(TextOutputFormat.class);
56
57     FileInputFormat.addInputPath(job, new Path(args[0]));
58     FileOutputFormat.setOutputPath(job, new Path(args[1]));
59
60     job.waitForCompletion(true);
61 }
62
63 }
```

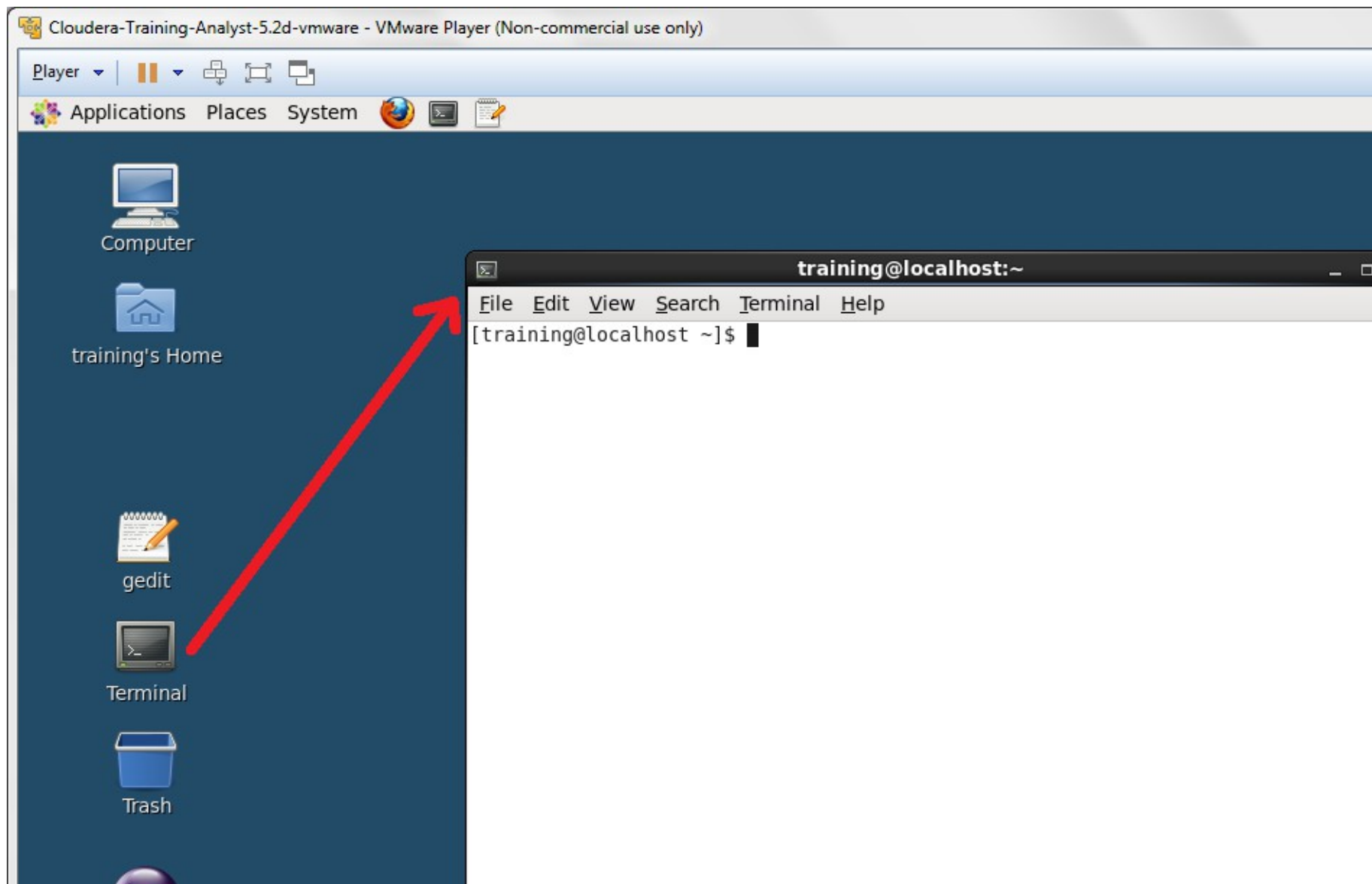
```
1 select word,count(1) as count
2 from (SELECT explode(split(sentence, ' ')) AS word
3      FROM texttable)tempTable
4 group by word
```



# Iniciar a VM



# Abrir terminal



## Iniciar os serviços

Execute o script:

```
training@localhost:~  
File Edit View Search Terminal Help  
[training@localhost ~]$ /home/training/scripts/analyst/da_toggle_services.sh  
* Enabling services required for Data Analyst Training  
* Disabling services not required for Data Analyst Training  
no secondarynamenode to stop  
Stopped Hadoop secondarynamenode: [ OK ]  
Flume agent is not running [ OK ]  
no master to stop because no pid file /var/run/hbase/hbase-hbase-master.pid  
Stopped HBase master daemon: [ OK ]  
no rest to stop because no pid file /var/run/hbase/hbase-hbase-rest.pid  
Stopped HBase rest daemon: [ OK ]  
no thrift to stop because no pid file /var/run/hbase/hbase-hbase-thrift.pid  
Stopped HBase thrift daemon: [ OK ]  
[training@localhost ~]$
```

## Comandos – no HDFS pelo Hive

### Comando **dfs**

Utilizado dentro do Hive para navegar, manipular dados, acessos entre outros, no HDFS.

Entrar no Hive:

```
[training@localhost ~]$ hive
```

Executar comando de navegação:

```
hive> dfs -ls /;  
hive> dfs -mkdir /aula;  
hive> dfs -ls /;
```

## Comandos – no S.O. pelo Hive

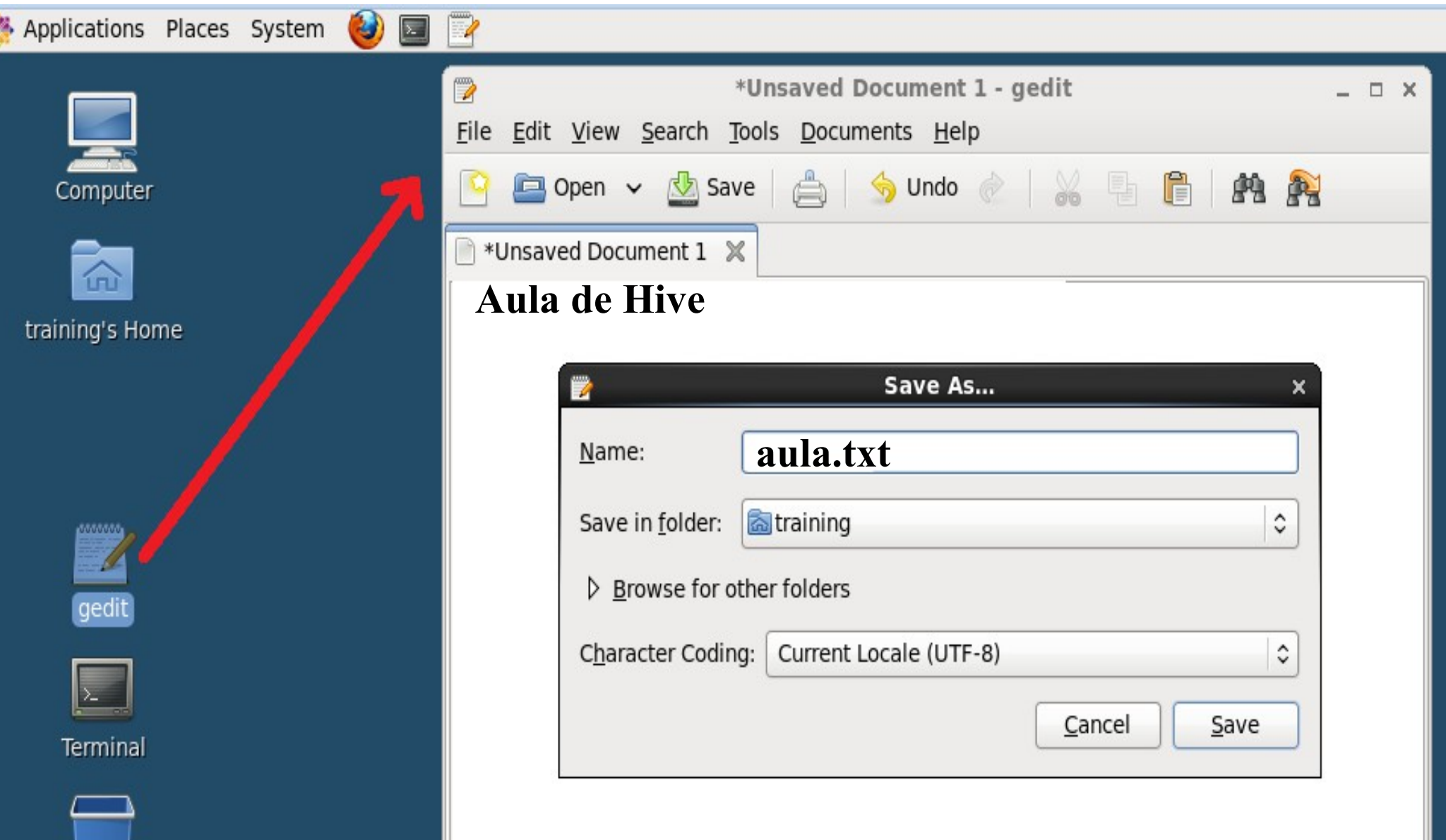
### Comando !

Utilizado dentro do Hive para navegar, manipular dados, acessos entre outros, no Sistema Operacional.

```
hive> ! ls -l /home/training;  
hive> ! ifconfig;  
hive> ! whoami;
```



## Criar um arquivo



## Carga - Load

Utilizando o comando **LOAD**:

```
hive> create table teste_load(  
  > campo string  
  > );  
  > load data local inpath '/home/training/aula.txt'  
  > overwrite into table teste_load;  
  > select * from teste_load;  
  > ! hadoop fs -ls /user/hive/warehouse;  
  >
```

## Carga – Put

Utilizando o comando **PUT**:

```
hive> create table teste_put(  
    >                                     campo string  
    > )  
    > location "/aula";  
    > select * from teste_put;  
    >   
    > ! hadoop fs -put /home/training/aula.txt /aula;  
    >   
    > select * from teste_put;
```

## Carga – Diretórios

- Tabelas criadas sem LOCATION, os dados são armazenados no diretório default do HDFS hive que é:

*/user/hive/warehouse/NOME\_DATABASE.db/NOME\_TABELA*

- Quando é criado no database default a estrutura é:

*/user/hive/warehouse/NOME\_TABELA*

- Tabelas criadas com LOCATION, o arquivo é armazenado neste local definido.

## Hive – Execução Via Sistema Operacional

- Utilizado para consultas direto na linha de comando do shell, em programas ou scripts.

Executando uma query a partir de uma string:

```
[training@localhost ~]$ hive -e "select * from teste_load;"
```

Executando uma query a partir de um arquivo:

```
[training@localhost ~]$ echo 'select * from teste_load;' > /home/training/query.sql
```

```
[training@localhost ~]$ hive -f /home/training/query.sql
```

## DDL – External Table

- External Table, é apenas uma referência a um arquivo, ou seja, é possível que uma ou mais external tables apontem para o mesmo arquivo no HDFS.

```
hive> create external table teste_ext( campo string)
      V
      V location '/aula';
      V
      V
      V select * from teste_ext;
      V
      V
      V drop table teste_ext;
      V
      V
      V ! hadoop fs -ls /aula
      V
```



## DDL – Manage Table

- Manage Table, é proprietária do arquivo, ou seja, caso seja removida, seus arquivos também serão:

```
hive> create table teste_mng( campo string
    )
    location '/aula';

    select * from teste_mng;

    drop table teste_mng;

    ! hadoop fs -ls /aula
```



## Funções

O Hive disponibiliza funções nativas, UDF, UDAF e UDTF, além de permitir o desenvolvimento de funções customizadas.

### **UDF** – Funções de usuário

Uma entrada e uma saída. Ex. SPLIT

### **UDAF** – Funções de agregação

N entradas e uma saída. Ex. SUM

### **UDTF** – Funções de tabelas

Uma entrada e N saídas. Ex. EXPLODE

## Função - SPLIT

Recebe um valor e um delimitador, devolve um array, ou seja, entra um elemento e sai N, no mesmo campo.

```
hive> create table teste_split(  
  > campo string  
  > );  
  > load data local inpath '/home/training/aula.txt'  
  > overwrite into table teste_split;  
  > select * from teste_split;  
  >  
  > select split(campo,' ') from teste_split;
```

Entrada
Aula de Hive

Saida
["Aula","de","Hive"]

## Função - EXPLODE

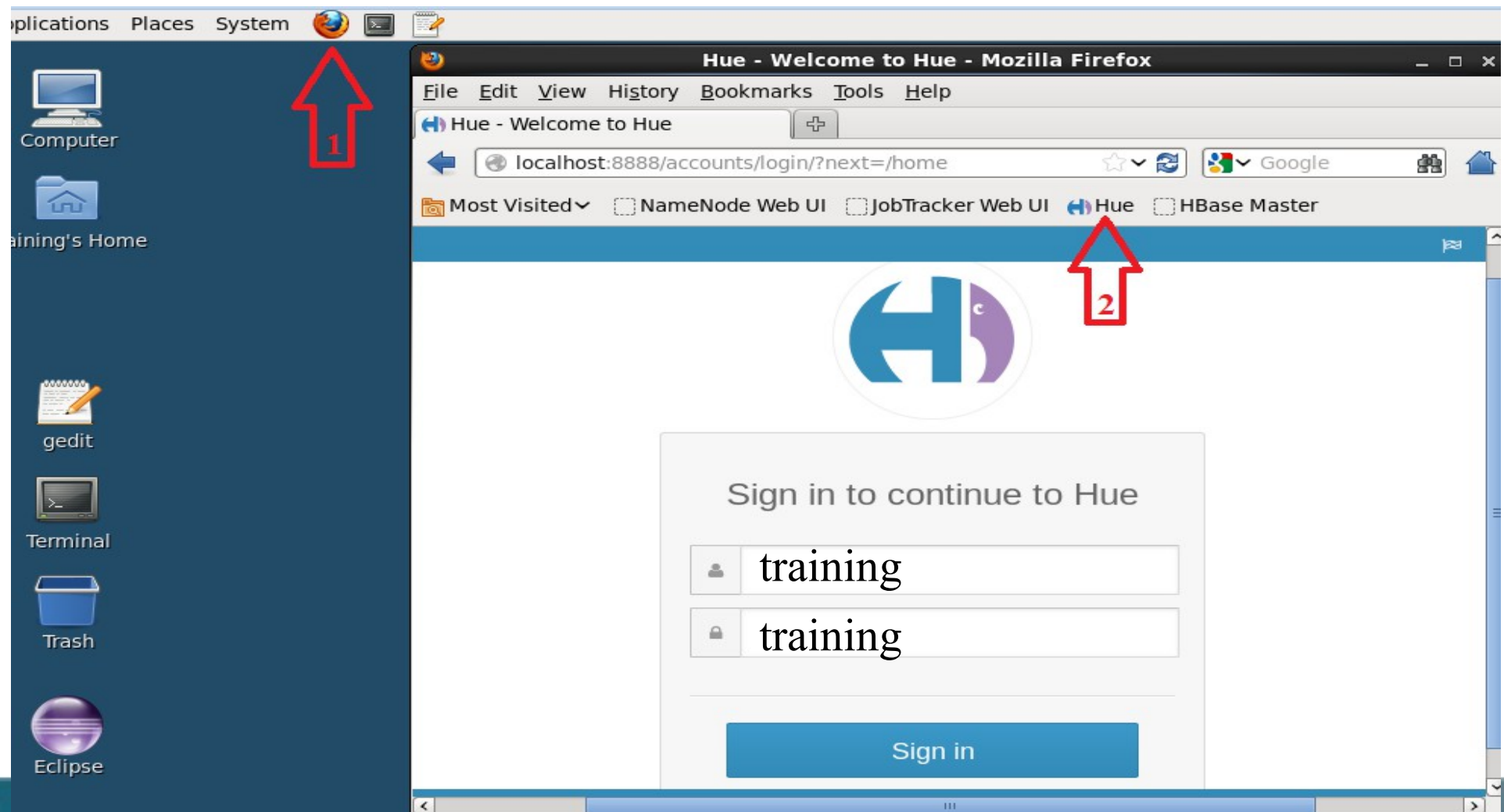
Recebe um array, ou seja, um conjunto de valores delimitados, e devolve em linhas.

```
hive> select explode(split(campo, ' '))  
      v  
      v  
      v  
      v  
      v  
      v  
      v  
      v  
      v  
      v
```

Entrada
["Aula","de","Hive"]
Saida
Aula
de
Hive

# HUE

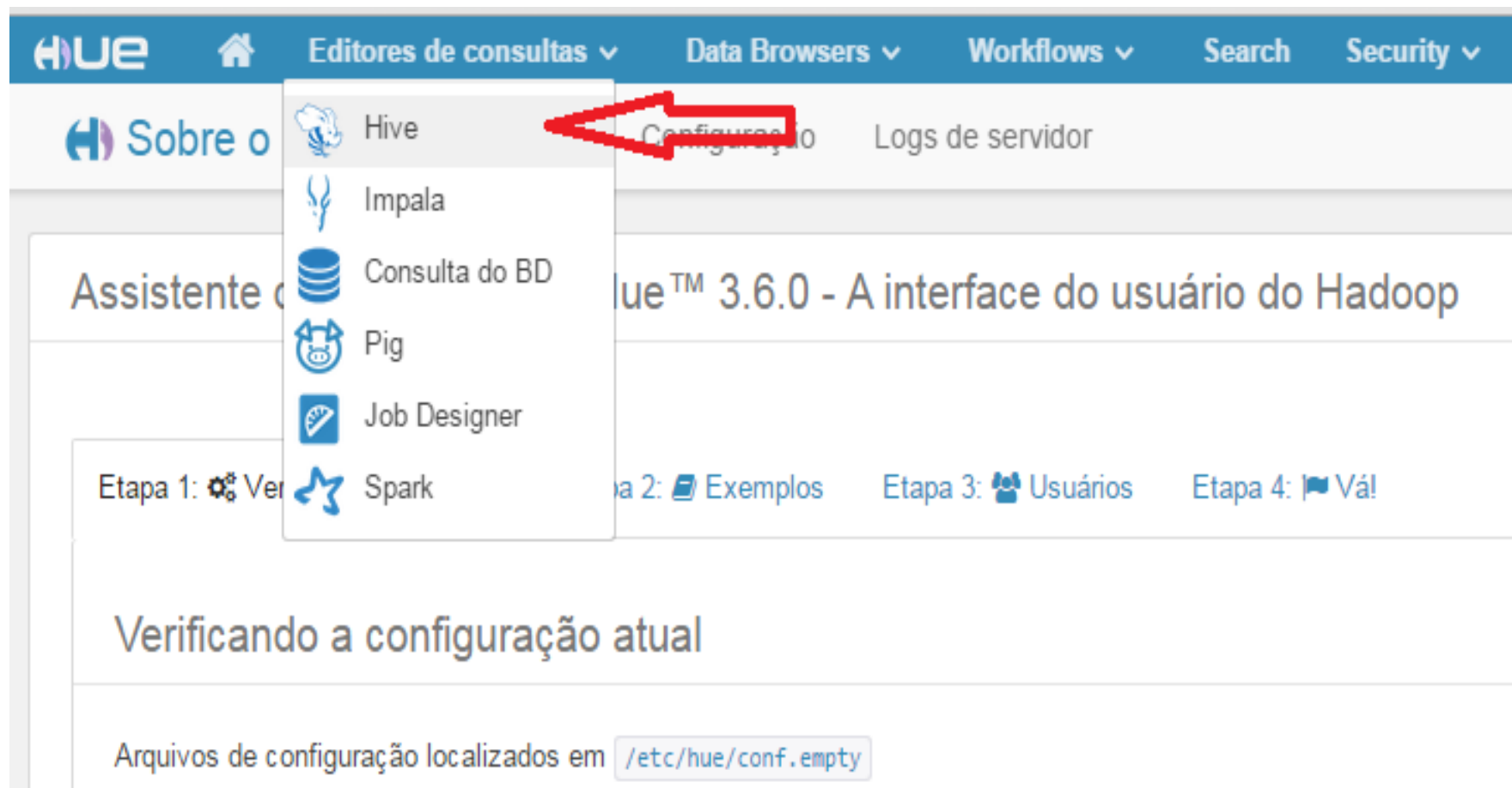
Hue é uma interface web para integração com o hadoop, para executar query no Hive, Impala, acompanhar execuções no Jobtracker, navegar no HDFS e etc...





# HUE

Executando uma query no hive:



# HUE

Executando uma query no hive:



The screenshot shows the HUE Hive Editor interface. The top navigation bar includes 'HUE', a home icon, and tabs for 'Editores de consultas', 'Data Browsers', 'Workflows', 'Search', and 'Security'. Below this, the 'Hive Editor' tab is active, with sub-tabs for 'Editor de consultas', 'Minhas consultas', 'Consultas salvas', and 'Histórico'. On the left sidebar, there are links for 'Assistência' and 'Configurações'. The 'BANCO DE DADOS' section features a dropdown menu currently set to 'default', with a red arrow pointing to it and the text 'Escolher database'. Below the dropdown is a text input field labeled 'Nome da tabela...'. At the bottom of the sidebar, a table icon is next to the text 'customers'. The main query editor area contains a single line of SQL: '1 select \* from teste\_load;', with a red arrow pointing to the word 'Query' above it. Below the query editor are buttons for 'Executar', 'Salvar', 'Salvar como...', 'Explicar', and a link 'ou criar uma' followed by a partially visible button.

HUE

Home

Editores de consultas ▾ Data Browsers ▾ Workflows ▾ Search Security

Hive Editor

Editor de consultas Minhas consultas Consultas salvas Histórico

Assistência

Configurações

BANCO DE DADOS

default

Nome da tabela...

customers

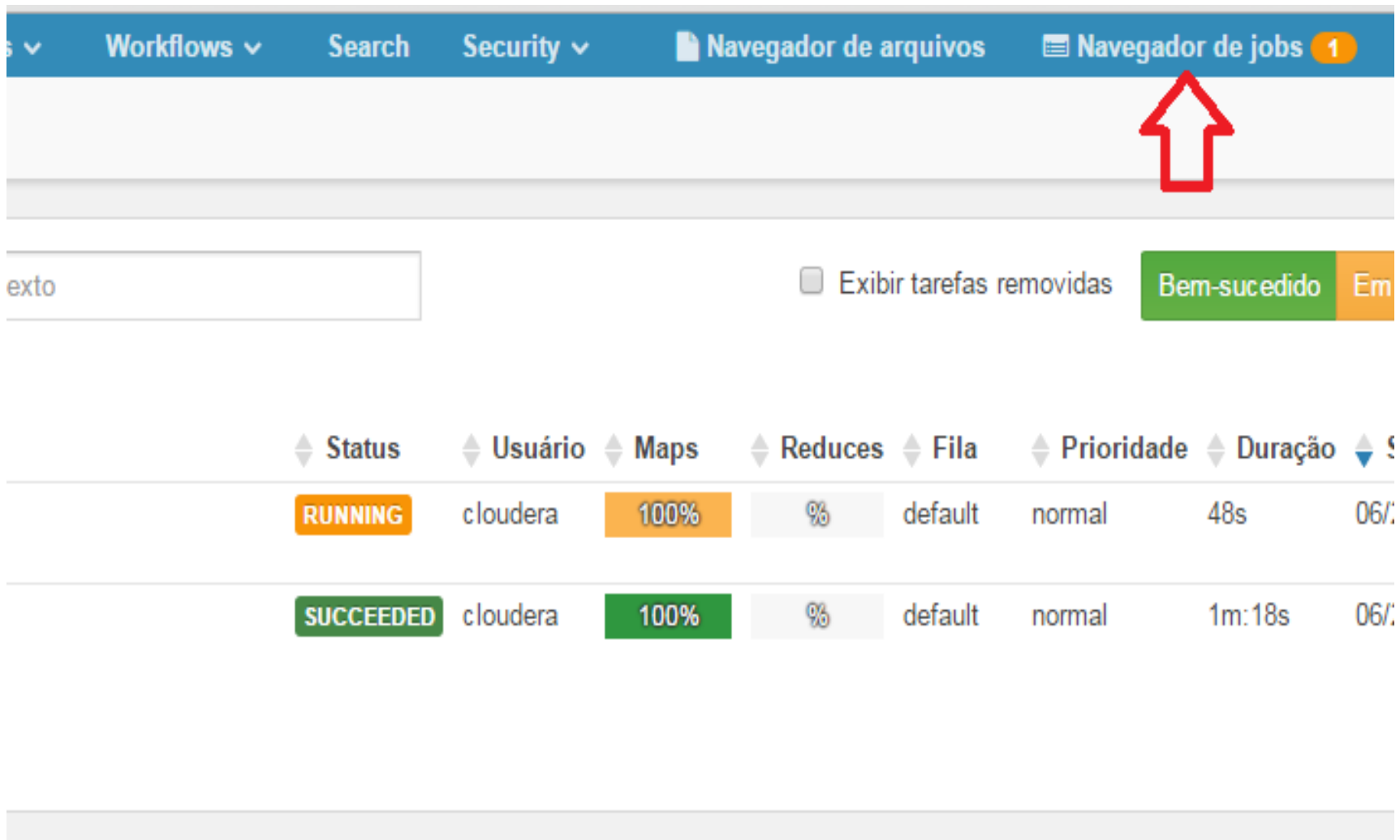
1 select \* from teste\_load;

Query

Escolher database

Executar Salvar Salvar como... Explicar ou criar uma

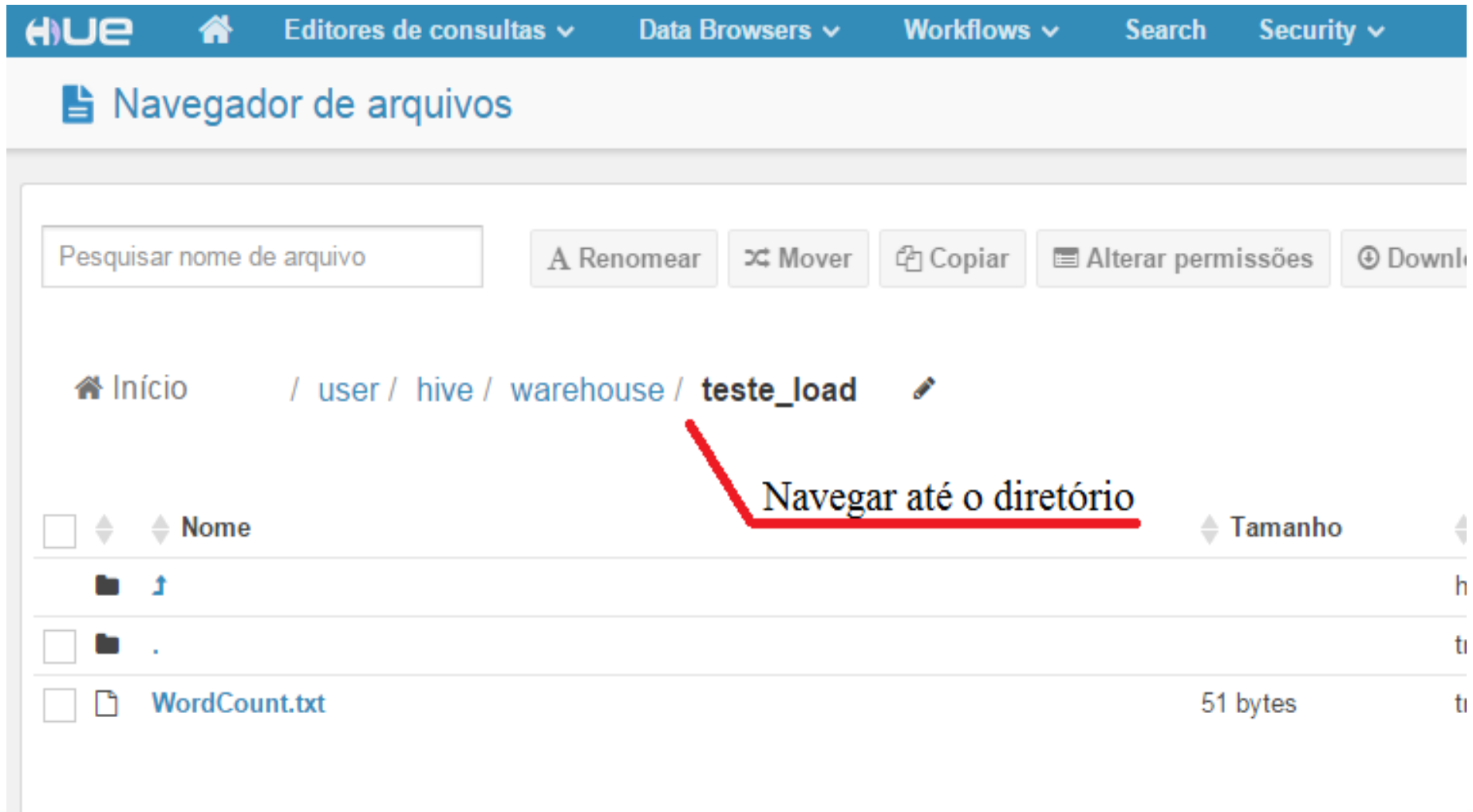
Acompanhar a execução no JobBrowser.



Status	Usuário	Maps	Reduces	Fila	Prioridade	Duração	
RUNNING	cloudera	100%	%	default	normal	48s	06/
SUCCEEDED	cloudera	100%	%	default	normal	1m:18s	06/

# HUE

Navegando no HDFS.



HUE

Editores de consultas ▾ Data Browsers ▾ Workflows ▾ Search Security ▾

Navegador de arquivos

Pesquisar nome de arquivo

Renomear Mover Copiar Alterar permissões Download

Início / user / hive / warehouse / teste\_load

Navegar até o diretório

<input type="checkbox"/>	Nome	Tamanho
<input type="checkbox"/>	↑	
<input type="checkbox"/>	.	
<input type="checkbox"/>	WordCount.txt	51 bytes

## Exercício 1 - Word Count

- Crie uma tabela com o nome WordCount e um campo chamado word, com o tipo string. A tabela deve apontar para /aula (LOCATION).
- Crie um diretório no HDFS e um arquivo, com a estrutura /aula/aula.txt
- Edite o arquivo incluindo o conteúdo:  
aula hive hadoop hive sql hue aula word count hue java sql hue
- Desenvolva uma query que retorne as palavras e quantidades de ocorrência, conforme figura do próximo slide.

## Formatos de arquivos – Posicional

Formato posicional utilizando **Serde**.

Crie um arquivo com o conteúdo abaixo no local do hdfs.

**Arquivo:**

**Estrutura da tabela**

5      10      2

~~~~~

00001cliente\_01PF  
00002cliente\_02PJ  
00003cliente\_03PF

```
CREATE EXTERNAL TABLE tabela_posicional (  
  conta string,  
  nome string,  
  tipo_pessoa string  
)  
ROW FORMAT SERDE  
'org.apache.hadoop.hive.contrib.serde2.RegexSerDe'  
WITH SERDEPROPERTIES ('input.regex'='(.{5})(.{10})(.{2})')  
LOCATION '/tabela_posicional';
```



## Formatos de arquivos - Delimitado

Formato posicional utilizando delimitador pipe.

Crie um arquivo com o conteúdo abaixo no local do hdfs

### Arquivo:

1|cliente\_01|PF

2|cliente\_02|PJ

3|cliente\_03|PF

### Estrutura da tabela

```
CREATE EXTERNAL TABLE tabela_delimitada(  
  conta string,  
  nome string,  
  tipo_pessoa string  
)  
ROW FORMAT delimited  
fields terminated by '|'   
lines terminated by '\n' stored as textfile  
LOCATION '/tabela_delimitada';
```

## Formatos de arquivos – JSON

**JSON** (JavaScript Object Notation) é um modelo para armazenamento de dados com capacidade de estruturar informações de uma forma bem mais compacta do que os arquivos sequenciais ou delimitados.

### Crie a tabela:

```
CREATE EXTERNAL TABLE tabela_json(  
campo string)  
stored as textfile  
LOCATION '/tabela_json';
```

### Crie um arquivo no HDFS /tabela\_json com o conteúdo:

```
{"conta":1,"cliente":{"nome":"cliente_1","tipo_pessoa":"PF"}}  
{"conta":2,"cliente":{"nome":"cliente_2","tipo_pessoa":"PJ"}}  
{"conta":3,"cliente":{"nome":"cliente_3","tipo_pessoa":"PF"}}
```

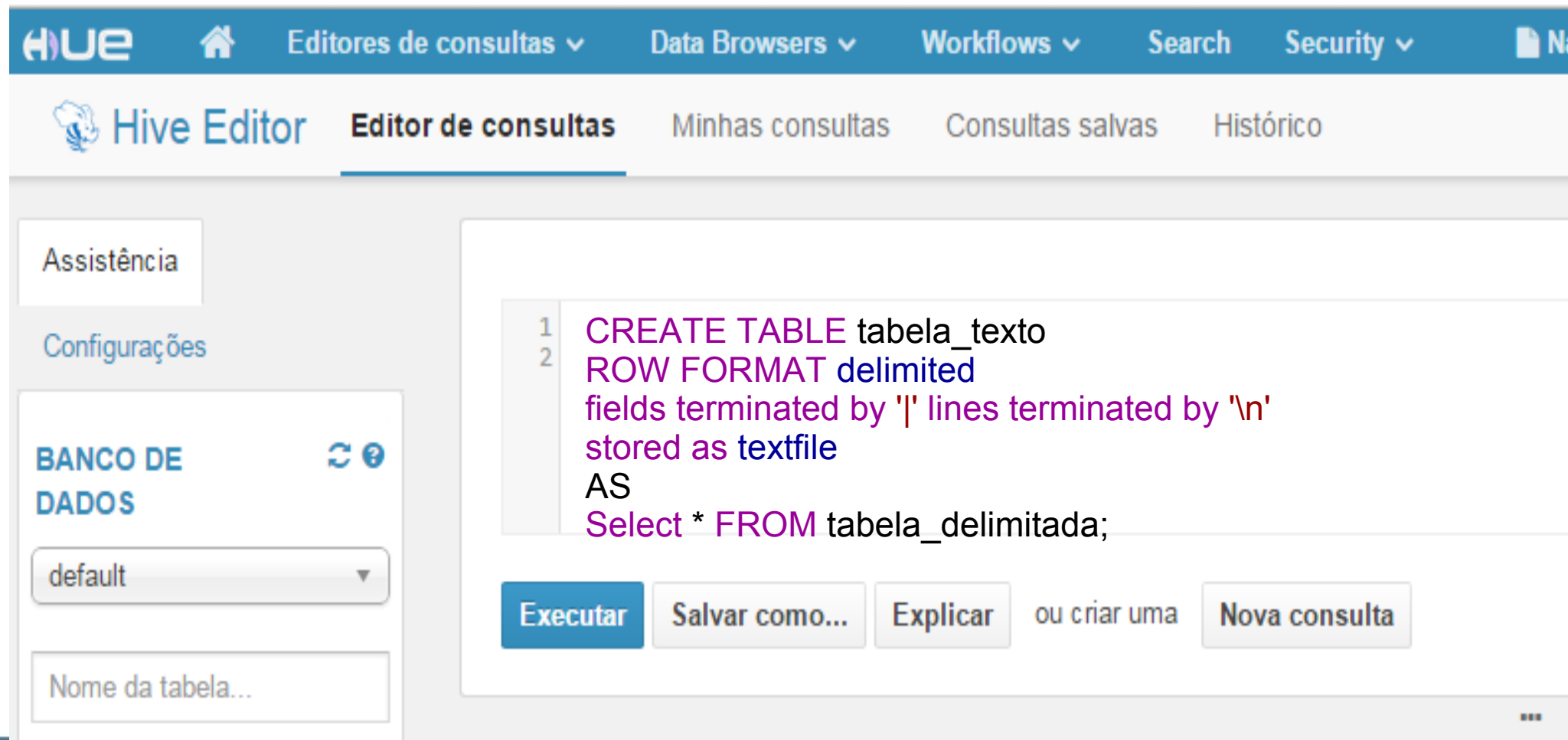
### Execute a query:

```
select get_json_object(campo, "$.conta") as conta,  
       get_json_object(campo, "$.cliente.nome") as nome,  
       get_json_object(campo, "$.cliente.tipo_pessoa") as tipo_pessoa  
from tabela_json;
```



## Extração de Dados

Vamos criar uma tabela no Hive que armazena os dados em formato texto delimitado, e extrair do HDFS.



The screenshot shows the Hue web interface for Hive. The top navigation bar includes links for Home, Query Editors, Data Browsers, Workflows, Search, and Security. The main header identifies the user as 'Hive Editor' and lists tabs for 'Editor de consultas' (active), 'Minhas consultas', 'Consultas salvas', and 'Histórico'.

On the left sidebar, there are sections for 'Assistência', 'Configurações', and 'BANCO DE DADOS'. The 'BANCO DE DADOS' section shows a dropdown menu set to 'default' and a text input field labeled 'Nome da tabela...'. A refresh icon is also present.

The main workspace contains a SQL editor with the following query:

```
1 CREATE TABLE tabela_texto
2 ROW FORMAT delimited
   fields terminated by '|' lines terminated by '\n'
   stored as textfile
   AS
   Select * FROM tabela_delimitada;
```

Below the query editor are several action buttons: 'Executar' (highlighted in blue), 'Salvar como...', 'Explicar', 'ou criar uma', and 'Nova consulta'.

## Extração de Dados - GET

Baixar o arquivo pela linha de comando.

```
[training@localhost ~]$ mkdir ~/saida_get
```

```
[training@localhost ~]$ hadoop fs -get /user/hive/warehouse/tabela_texto/* ~/saida_get
```

```
[training@localhost ~]$ ls -l ~/saida_get
```


```
[training@localhost ~]$ cat ~/saida_get/000000_0
```

Obs.: “~” = “/home/training”

```
[training@localhost ~]$  
1|cliente_01|PF  
2|cliente_02|PJ  
3|cliente_03|PF
```

## Extração - Insert Directory

Insere os dados da saída da query, no diretório indicado no **HDFS**.  
Após a execução, verifique o conteúdo no diretório.



The screenshot shows the HUE web interface for Hive. The top navigation bar includes links for Home, Query Editors, Data Browsers, Workflows, Search, and Security. Below this, the 'Hive Editor' tab is active, showing 'Editor de consultas', 'Minhas consultas', 'Consultas salvas', and 'Histórico'.

On the left sidebar, there are sections for 'Assistência' and 'Configurações'. Under 'Configurações', the 'BANCO DE DADOS' section is visible, showing a dropdown menu set to 'default' and a text input field for 'Nome da tabela...'. There are also refresh and help icons.

The main area displays an SQL query in a text editor:

```
1 INSERT OVERWRITE DIRECTORY '/saida_hdfs'
2 select * from tabela_texto;
```

Below the query editor, there are buttons for 'Executar' (Execute), 'Salvar como...' (Save as...), 'Explicar' (Explain), and 'Nova consulta' (New query), along with the text 'ou criar uma' (or create a).

## Extração - Insert Local Directory

Insere os dados da saída da query, no diretório **LOCAL** indicado, ou seja, no diretório especificado no linux.

Após a execução, verifique o conteúdo no diretório.

Obs.: Execute na linha de comando.

```
hive> INSERT OVERWRITE LOCAL DIRECTORY '/home/training/saida_local'
> row format delimited
> FIELDS TERMINATED BY ','
> select * from tabela_delimitada;
>
> ! ls /home/training/saida_local;
> ! cat /home/training/saida_local/000000_0;
>
>
>
```

## DML – Manipulação de dados

Antes da versão 0.14, o Hive não disponibilizava cláusulas de inserção de valores, exclusão ou alteração.

O conceito é baseado em arquivos, com isto, a inserção seria incluir um arquivo no HDFS, a exclusão seria remover e substituição para alteração.

Existe a cláusula **INSERT**, que insere dados em “tabelas”, diretório do HDFS e no diretório local, como vimos anteriormente, mas sempre partindo do resultado de uma query.

Com isto conseguimos excluir, alterar e inserir utilizando resultados de consulta, ou a manipulação direta do arquivo.

# HiveQL

## SEMI JOIN

Antes do hive 0.13 não tínhamos a opção de fazer subquery no WHERE para filtrar uma lista contida ou não (in / not in) em outra tabela.

Segue comparação para simularmos um filtro de lista:

Subquery:

```
select * from tabela_texto  
where conta in (select * from tabela_delimitada);
```

Semi Join:

```
select * from tabela_texto a  
left semi join tabela_delimitada b  
on a.conta=b.conta;
```



## Diferença entre **SORT BY** e **ORDER BY**.

O SORT ordena isoladamente por bloco do reducer, enquanto o ORDER, garante a ordem total dos registros mas para isto, executa a ordenação em um único reducer.

Exemplo:

| Entrada |
|---------|
| 3       |
| 6       |
| 4       |
| 5       |
| 1       |
| 2       |

**SORT**

| Bloco 1 |
|---------|
| 4       |
| 5       |
| 6       |

| Bloco 2 |
|---------|
| 1       |
| 2       |
| 3       |

**ORDER**

| Bloco único |
|-------------|
| 1           |
| 2           |
| 3           |
| 4           |
| 5           |
| 6           |

## Performance - MAP JOIN

O Hive suporta **MAP JOIN**, que permite subir tabelas “pequenas” para a memória, aumentando a performance por não ler arquivos em disco.

O MAP JOIN pode ser forçado na escrita da query ou configurado o AUTO MAP JOIN, onde o tamanho da tabela é comparado a um parâmetro.

MAP JOIN forçado

```
select /*+ MAPJOIN(tabela_delimitada) */  
* from tabela_texto a  
left semi join tabela_delimitada b  
on a.conta=b.conta;
```

MAP JOIN Parametrizado

```
set hive.auto.convert.join=true;  
select * from tabela_texto a  
left semi join tabela_delimitada b  
on a.conta=b.conta;
```

**Obs.:** Existe outros parâmetros de configuração do MAP JOIN, onde é definido o tamanho máximo da tabela em memória por exemplo.

## Performance – Tabela Particionada

O particionamento por diretório dos dados armazenado no HDFS, é uma maneira muito eficiente de ganhar performance no processamento de grandes volumes, evitando a leitura da base inteira.

Exemplo de partição por data de referência:

| Diretórios        | Dados |
|-------------------|-------|
| /dados/ref=201501 | 1     |
|                   | 2     |
|                   | 3     |
| /dados/ref=201502 | 4     |
|                   | 5     |
|                   | 6     |
| /dados/ref=201503 | 7     |
|                   | 8     |
|                   | 9     |

```
CREATE EXTERNAL TABLE tabela_particionada(
  conta string)
```

```
PARTITIONED BY (ref int)
ROW FORMAT delimited
fields terminated by '|'
lines terminated by '\n' stored as textfile;
```

```
ALTER TABLE tabela_particionada ADD PARTITION (ref=201601)
location '/tabela_particionada/ref=201601';
```

```
ALTER TABLE tabela_particionada ADD PARTITION (ref=201602)
location '/tabela_particionada/ref=201602';
```

```
ALTER TABLE tabela_particionada ADD PARTITION (ref=201603)
location '/tabela_particionada/ref=201603';
```

## Performance – Tabela Particionada

Seleção em tabela particionada.

Para utilizar o recurso de particionamento, ao submeter uma query, no filtro deve ser colocado o nome da partição a ser selecionada e o valor exato, sem funções, ou seja, o filtro com o valor de referência, fará com que seja lido somente o diretório em questão, já um filtro com função, por exemplo uma adição, fará com que o hive leia todas as partições (full table scan).

CORRETO:

```
select * from tabela_particionada  
where ref=201601;
```

FULL TABLE SCAN:

```
select * from tabela_particionada  
where ref=201601+1;
```

## Performance - Paralelismo

Paralelismo e Subquery no join.

Outro método para eficiência no processamento no Hive, é utilizar subquery no lugar da tabela de join e ligar o processamento paralelo do Hive.

Exemplo Query não indicada:

```
select * from tabela_1 a
```

```
inner join tabela_2 b  
on a.chave=b.chave
```

```
inner join tabela_3 c  
on a.chave=c.chave
```

```
where b.chave=2 and  
c.chave=1;
```

Seleção  
tabela\_1

Junção  
tabela\_2

Junção  
tabela\_3

Filtros

Leitura e junção de TODOS os dados, e por fim o filtro.

## Performance - Paralelismo

Paralelismo e Subquery no join.

Outro método para eficiência no processamento no Hive, é utilizar subquery no lugar da tabela de join e ligar o processamento paralelo do Hive.

Exemplo Subquery no join:

```
select * from tabela_1 a
```

```
inner join (select * from tabela_2 where chave=2) b  
on a.chave=b.chave
```

```
inner join (select * from tabela_3 where chave=1) c  
on a.chave=c.chave;
```

Filtro  
tabela\_2

Filtro  
tabela\_3

Junção  
das 3 tabelas

Filtro das tabelas 2 e 3, e por fim uma junção com registros já reduzidos, menor tempo de processamento e menor área temporária consumida.

## Performance - Paralelismo

Paralelismo e Subquery no join.

Outro método para eficiência no processamento no Hive, é utilizar subquery no lugar da tabela de join e ligar o processamento paralelo do Hive.

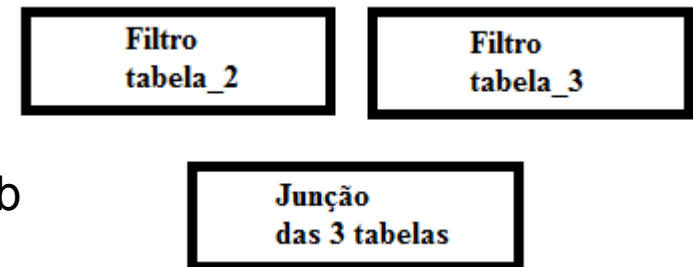
Exemplo Subquery no join + parallel ligado:

```
set hive.exec.parallel=true;
```

```
select * from tabela_1 a
```

```
inner join (select * from tabela_2 where chave=2) b  
on a.chave=b.chave
```

```
inner join (select * from tabela_3 where chave=1) c  
on a.chave=c.chave;
```



Filtro das tabelas 2 e 3 em PARALELO, e por fim uma junção com registros já reduzidos, menor tempo de processamento e menor área temporária consumida.

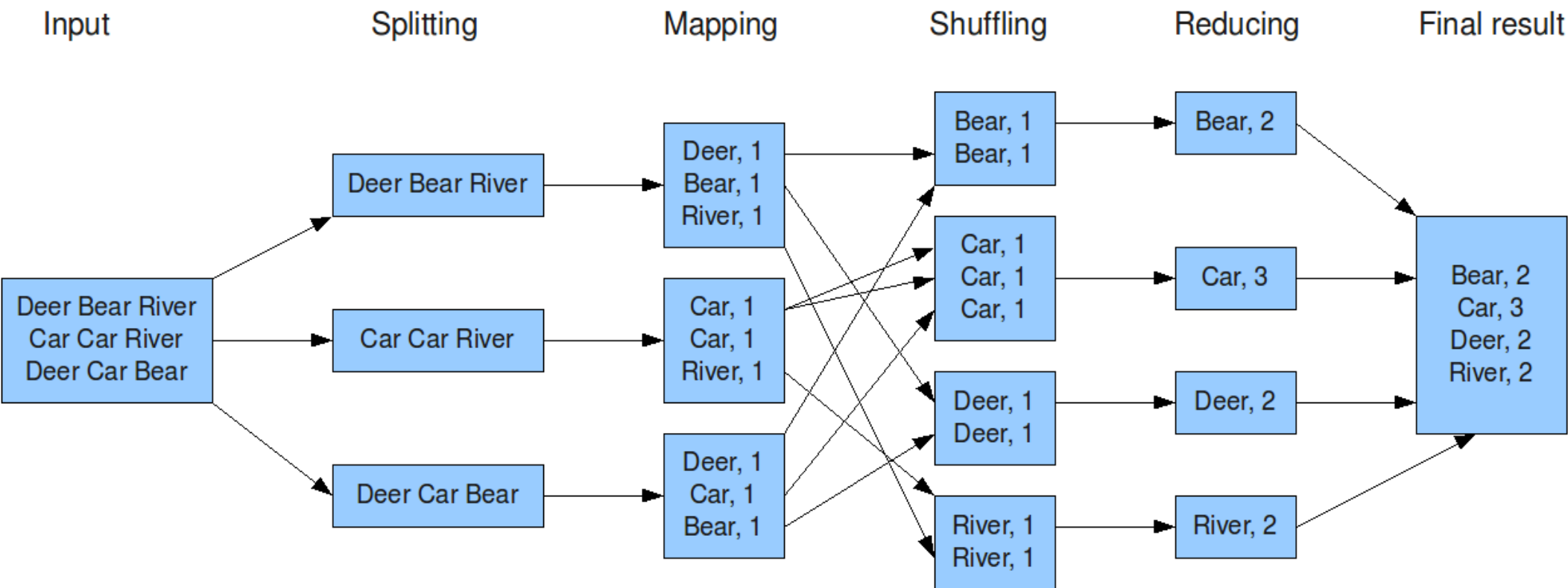
## Performance - Bucket Tables

- Baseia-se em função hash
- Os dados com a mesma chave de hash são armazenados no mesmo arquivo
- O bucket deve ser feito com uma chave com uma boa distribuição
- Dependendo da versão, o número de reducers deve ser atribuído com o mesmo número de bucket
- Esta chave deverá ser utilizada em chaves de agrupamento
- Esta chave deverá ser utilizada em joins com outras tabelas com a mesma chave de bucket



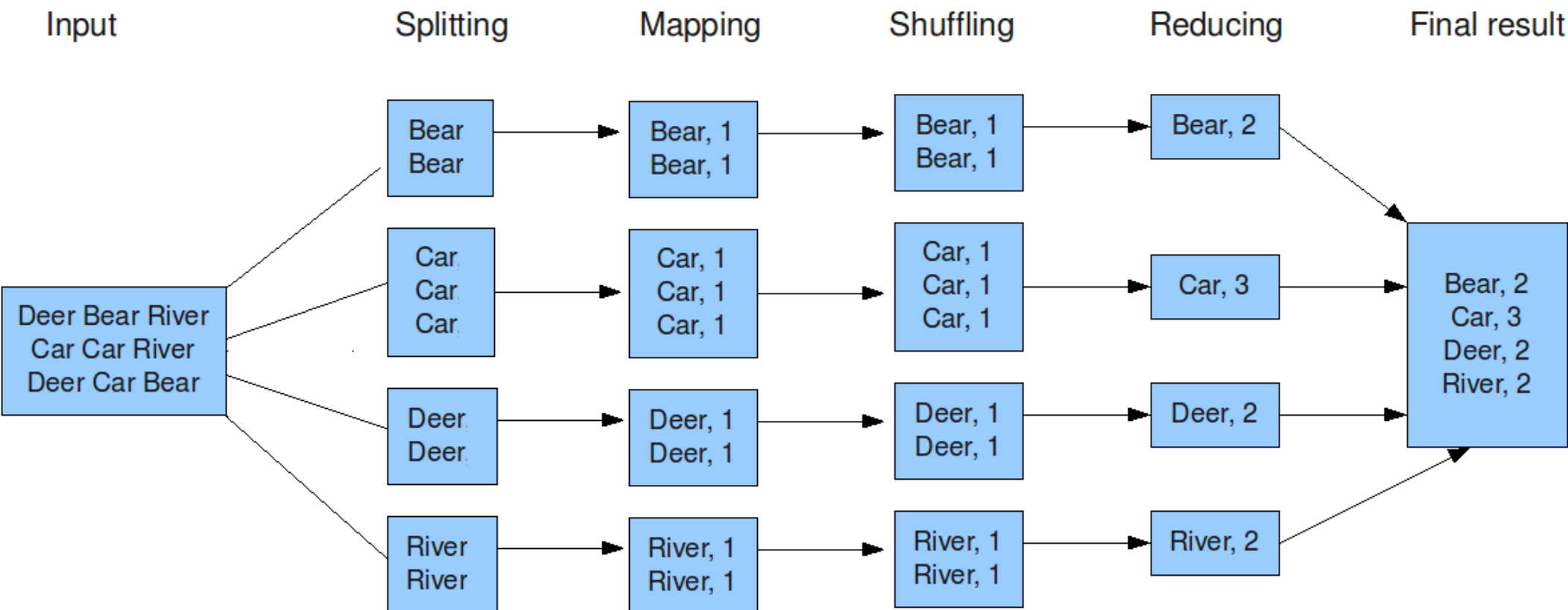
## Performance - Bucket Tables

The overall MapReduce word count process



## Performance - Bucket Tables

The overall MapReduce word count process



# Performance - Bucket Tables

## Exemplo:

```
CREATE TABLE user_info_bucketed(user_id BIGINT, firstname STRING, lastname STRING)
COMMENT 'A bucketed copy of user_info'
PARTITIONED BY(ds STRING)
CLUSTERED BY(user_id) INTO 256 BUCKETS;
```

Note that we specify a column (user\_id) to base the bucketing.

Then we populate the table

```
set hive.enforce.bucketing = true; -- (Note: Not needed in Hive 2.x onward)
FROM user_id
INSERT OVERWRITE TABLE user_info_bucketed
PARTITION (ds='2009-02-25')
SELECT userid, firstname, lastname WHERE ds='2009-02-25';
```

### Version 0.x and 1.x only

The command `set hive.enforce.bucketing = true;` allows the correct number of reducers and the cluster by column to be automatically selected based on the table. Otherwise, you would need to set the number of reducers to be the same as the number of buckets as in `set mapred.reduce.tasks = 256;` and have a `CLUSTER BY ...` clause in the select.

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL+BucketedTables>

## Hive OverHead

- O Hadoop foi feito para “poucos arquivos grandes” e não para “muitos arquivos pequenos”.

Um arquivo pequeno é um que é significativamente menor do que o tamanho do bloco HDFS (64MB padrão).

Se você está armazenando arquivos pequenos, então você provavelmente tem muitos deles (caso contrário, você não se voltaria para Hadoop), e o problema é que o HDFS não pode lidar com lotes de arquivos.

Cada arquivo, diretório e bloco no HDFS é representado como um objeto na memória do namenode, cada uma das quais ocupa 150 bytes, como uma regra de ouro . Assim, 10 milhões de arquivos, cada um usando um bloco, usaria cerca de 3 gigabytes de memória. Intensificação muito além desse nível é um problema com o hardware atual. Certamente um bilhão de arquivos não é viável.

## Hive OverHead

Execute as queries:

```
hive> select * from teste_load;  
✓  
✓  
✓ select campo from teste_load;  
✓  
✓  
✓  
✓  
✓  
✓  
✓  
✓  
✓
```

Por que a primeira é praticamente instantanea, enquanto a segunda demora para iniciar, sendo que o resultado e a origem são os mesmos?

## Hive x Impala

O Hive não foi desenhado para executar queries em real time, com baixa latência. Foi desenhado para melhor performance analisando grandes quantidades de dados que se encontram em clusters.

Cloudera lançou o Impala, permitindo execução de queries com baixa latência para dados armazenados no HDFS e no HBASE, usando a linguagem SQL, permite integração com ferramentas de BI como Tableau, Microstrategy, Pentaho, etc.

Execute a seguinte query no Hive e em seguida no Impala

```
select conta, nome, tipo_pessoa from tabela_texto;
```

## Exercícios

### Exercício 2:

- Criar uma tabela para ser exportada que é populada por uma query, lendo uma outra tabela com arquivo origem em formato json (tabela\_json)
- Exportar o arquivo para o /home/training/exercicio

# Prof. Bruno Paulinelli



<https://br.linkedin.com/pub/bruno-paulinelli/bb/218/154>



[bruno.paulinelli@gmail.com](mailto:bruno.paulinelli@gmail.com)



[b.paulinelli](https://api.whatsapp.com/send?phone=5511999999999)

