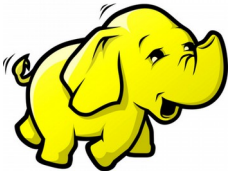


Cloudera Administrator Apache Hadoop

Parte 02-3 Inserindo dados no CDH



Marco Reis
<http://marcoreis.net>

Agenda



- Inserir dados no HDFS
 - Hue
 - Hive
 - Impala
 - Sqoop
- Formatos de arquivo de entrada e saída
- Técnicas de ETL
 - Extração, transformação e carregamento de dados



Inserindo dados

- O ponto inicial da inserção de dados é o HDFS
 - Os dados devem estar, preferencialmente, gravados no HDFS
- O Hadoop permite flexibilidade para importação e exportação de dados
 - Não há restrição para formato de arquivo, banco de dados ou mesmo tecnologia
 - É possível trabalhar com praticamente qualquer fonte de dados
- As estratégias usadas aqui incluem a importação destes tipos de arquivo:
 - CSV: importação mais direta
 - CSV com SerDe: campos gravados com aspas
 - XML com SerDe: conversão de XML para formato tabular
 - JSON com SerDe: conversão de JSON para formato tabular
 - Banco relacional: importação/exportação de tabelas

Importação de CSV



- Mover o arquivo Municipio.csv para o diretório de dados
 - `/user/dataengineer/dados/municipio/Municipio.csv`
- Exemplo:
 - `$ hdfs dfs -mkdir -p /user/dataengineer/dados/municipio`
 - `$ hdfs dfs -mv /user/dataengineer/Municipio.csv /user/dataengineer/dados/municipio/Municipio.csv`
 - `$ hdfs dfs -ls /user/dataengineer/dados/municipio`
- Opção para copiar para o HDFS sem passar pelo gateway é usar o cat. Exemplo:
 - `$ cat Municipio.csv | ssh dataengineer@hadoopclient.lab "hdfs dfs -put - /user/dataengineer/dados/municipio/Municipio.csv"`



Formatos no Hive

- O Hive suporta diversos formatos de arquivos para armazenamento de dados
- Cada formato tem diferenças no tamanho e velocidade de leitura e gravação
- O Impala tem limitações para criação e inserção de dados em alguns formatos
- Os formatos de arquivo do Hive são:
 - Texto
 - SequenceFile
 - RCFile
 - Apache ORC
 - Apache Parquet
 - Apache Avro
 - Apache Kudu

Apache Parquet



- O Parquet é o formato selecionado nos exemplos
- Ele tem boa compressão e performance, além de permitir a criação de tabelas, inserção e carregamento de dados
- A compressão usará a biblioteca Snappy porque ela oferece bom equilíbrio entre velocidade e tamanho dos arquivos



Configurações do ambiente

- A consulta no Hive pode usar o MapReduce ou o Spark como mecanismo de processamento
- Configure o mecanismo de acordo com sua preferência
 - SET `hive.execution.engine=[spark|mr];`
- Testando as duas opções, o Spark tende a ser mais rápido que o MR
- A compressão é indicada para diminuir a utilização do disco, minimizar o tráfego de rede e melhorar a performance do processamento (na maioria dos casos)
- Geralmente utilizamos Snappy, GZIP ou UNCOMPRESSED
 - SET `parquet.compression=[snappy|gzip|uncompressed];`

Hue → Hive



- As operações de importação podem ser realizadas no Hue → Hive ou no cliente Hive (beeline)
- A estratégia mais direta e simples de importação consiste desses passos:
 - Copiar os dados para o HDFS
 - Criar uma tabela externa temporária, com acesso direto no arquivo HDFS (baixa performance mas é prático para testes rápidos)
 - Criar uma tabela no Hive com o formato de arquivo desejado (Parquet)
 - Importar os dados do HDFS na tabela Hive
 - Remover a tabela temporária (opcional)
- Obs 1: vamos remover a tabela municipio para recriá-la usando essas técnicas
 - `drop table datalake.municipio;`
- Obs 2: remover uma tabela externa não exclui seus arquivos no HDFS

Tabela externa municipio_temp

```
use datalake;  
create external table municipio_temp  
(  
  codmunicipio String ,  
  uf string ,  
  nomemunicipio string ,  
  populacaoresidente string )  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY '\t'  
STORED AS TextFile  
LOCATION '/user/dataengineer/dados/municipio'  
TBLPROPERTIES("skip.header.line.count" = "1");
```

Tabela Hive municipio

- A tabela municipio vai ser criada no formato Parquet, usando a estrutura da tabela municipio_tempo
 - `create table municipio like municipio_temp stored as parquet;`
- A importação e transformação dos dados é feita com um comando SQL padrão
- É recomendado o uso de compressão e do Spark para acelerar o processamento
 - `SET parquet.compression=snappy;`
 - `SET hive.execution.engine=spark;`
 - `insert into datalake.municipio select * from datalake.municipio_temp;`
- O Hive cria um programa Spark/MapReduce e converte o diretório de entrada na nova tabela Parquet
- A tabela está pronta para consultas no Hive e no Impala
- Para truncar a tabela:
 - `truncate table datalake.municipio;`

Consulta no Hive

```
> SET hive.execution.engine=spark;  
> select sum(populacaoresidente) as total, uf  
from municipio  
group by uf  
order by total desc;
```

Consulta no Impala

```
> select sum(cast(populacaoresidente as int)) as total, uf  
from municipio  
group by uf  
order by total desc
```

- Obs 1: note que há uma pequena diferença na consulta do Impala
- Obs 2: tabelas criadas no Hive não aparecem automaticamente no Impala. Para atualizar os metadados no Impala use o comando:
 - INVALIDATE METADATA;
 - INVALIDATE METADATA nome-tabela;
- Obs 3: para atualizar tabelas já existentes e que tiveram adição de novos dados:
 - REFRESH nome-tabela;

Salvar a consulta

- Edite o nome da consulta e clique no botão de gravar, tanto no Hive quanto no Impala

Impala

populacao-por-estado-impala Add a description...

dbmarcoreis text ?

```
1 select sum(cast(populacaoresidente as int)) as total, uf
2 from municipio
3 group by uf
4 order by total desc
```

Hive








populacao-por-estado-hive Add a description...

dbmarcoreis text ?

```
1 SET hive.execution.engine=spark;
2
3 select sum(populacaoresidente) as total, uf
4 from municipio
5 group by uf
6 order by total desc
```

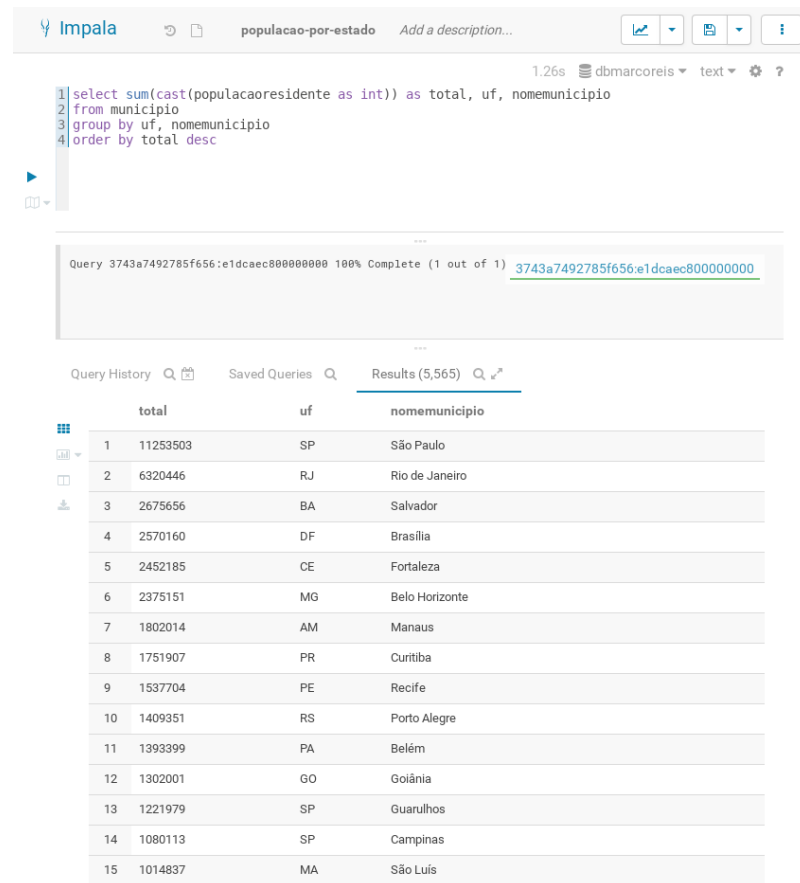
Documentos salvos

- Para acessar as consultas, acesse o menu lateral → Documents

My documents				
All      Trash				
Name	Description	Type	Owner	Last Modified
 populacao-por-estado-hive		Hive Query	marcoreis	08/23/2018 7:29 PM
 populacao-por-estado-impala		Impala Query	marcoreis	08/23/2018 7:28 PM

Consulta por município

- Aproveite para realizar novas pesquisas exploratórias



The screenshot displays the Impala web interface. At the top, the title bar shows 'populacao-por-estado' and 'Add a description...'. The SQL query is entered in the editor:

```
1 select sum(cast(populacaoresidente as int)) as total, uf, nomemunicipio
2 from municipio
3 group by uf, nomemunicipio
4 order by total desc
```

The query execution status is shown as 'Query 3743a7492785f656:e1dcaec800000000 100% Complete (1 out of 1)'. Below the query, the results are displayed in a table with 5,565 rows. The table has three columns: 'total', 'uf', and 'nomemunicipio'. The results are ordered by 'total' in descending order.

	total	uf	nomemunicipio
1	11253503	SP	São Paulo
2	6320446	RJ	Rio de Janeiro
3	2675656	BA	Salvador
4	2570160	DF	Brasília
5	2452185	CE	Fortaleza
6	2375151	MG	Belo Horizonte
7	1802014	AM	Manaus
8	1751907	PR	Curitiba
9	1537704	PE	Recife
10	1409351	RS	Porto Alegre
11	1393399	PA	Belém
12	1302001	GO	Goiânia
13	1221979	SP	Guarulhos
14	1080113	SP	Campinas
15	1014837	MA	São Luís

CSV com SerDe



- Vamos importar os dados dos pagamentos do programa bolsa família
- Os arquivos estão em formato CSV, mas os campos estão envolvidos em aspas duplas, ou seja, precisam ser pré-processados antes da importação
- Para isso é usado o SerDe (Serializer and Deserializer), um utilitário para ler e gravar os registros no Hive
- Os arquivos têm outra particularidade, pois usam encoding ISO-8859-1, que é um padrão de codificação bastante usado em português
- O cluster costuma usar UTF-8, um padrão de codificação internacional

Dataset do Bolsa Família



- Os arquivos estão disponíveis em formato CSV compactado no endereço:
 - <http://www.portaldatransparencia.gov.br/download-de-dados/bolsa-familia-pagamentos>
- Para efeito de estudo, vamos usar 2 arquivos:
 - 201801_BolsaFamilia_Pagamentos.zip
 - 201802_BolsaFamilia_Pagamentos.zip
- O procedimento é:
 - Copiar os arquivos ZIP para o hadoopclient
 - Descompactar
 - Converter para UTF-8
 - Copiar para o HDFS
- Exemplo:
 - # Disco local
 - \$ scp 20180*_BolsaFamilia_Pagamentos.zip dataengineer@hadoopclient:/home/dataengineer/
 - \$ unzip 201801_BolsaFamilia_Pagamentos.zip ; unzip 201802_BolsaFamilia_Pagamentos.zip
 - \$ iconv -f ISO-8859-1 -t UTF-8 201801_BolsaFamilia_Pagamentos.csv -o 201801_BolsaFamilia_Pagamentos_utf8.csv
 - \$ iconv -f ISO-8859-1 -t UTF-8 201802_BolsaFamilia_Pagamentos.csv -o 201802_BolsaFamilia_Pagamentos_utf8.csv
 - # HDFS
 - \$ hdfs dfs -mkdir -p /user/dataengineer/dados/bolsafamilia/
 - \$ hdfs dfs -put *utf8.csv /user/dataengineer/dados/bolsafamilia/
 - \$ hdfs dfs -ls -h /user/dataengineer/dados/bolsafamilia

Hue → Hive

- O procedimento no Hive já é conhecido
 - Criar a tabela externa temporária
 - Criar a tabela de destino no formato Parquet
 - Inserir e transformar os dados na tabela de destino
 - Remover a tabela externa (opcional)

Tabela externa bolsafamilia_temp

```
create external table datalake.bolsafamilia_temp
(
  anomesreferencia String ,
  anomescompetencia string ,
  uf string ,
  codmunicipio string,
  nomemunicipio string,
  nisbeneficiario string,
  nomebeneficiario string,
  valorbeneficio string )
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
  "separatorChar" = ";",
  "quoteChar"    = "\"",
  "escapeChar" = "\\"
)
STORED AS TextFile
LOCATION '/user/dataengineer/dados/bolsafamilia'
TBLPROPERTIES("skip.header.line.count" = "1");
```

Tabela Hive bolsafamilia

- Uma forma mais direta de criar a tabela é com o comando “create table as”, uma técnica conhecida como CTAS

- Script:

```
SET parquet.compression=snappy;  
SET hive.execution.engine=spark;  
create table datalake.bolsafamilia  
stored as parquet  
as select anomesreferencia ,  
anomescompetencia ,  
uf ,  
codmunicipio ,  
nomemunicipio ,  
nisbeneficiario ,  
nomebeneficiario ,  
cast(regex_replace(valorbeneficio, ',', '.') as decimal(7,2)) as valorbeneficio  
from bolsafamilia_temp;
```

Consulta no Hive e Impala

Execute a consulta no Hive e no Impala para comparar as performances

```
> select count(*) as total, uf from bolsafamilia  
group by uf  
order by total desc;
```

XML com SerDe



- Vamos usar o dataset do StackOverflow para trabalhar com arquivos XML
- Os esquemas de dados usados são Users e Comments
- O SerDe é a forma mais simples de importar arquivos complexos no Hive
- Diversos SerDes disponíveis na internet, vamos selecionar as versões mais eficientes para os exemplos
 - Por ser um framework, é possível criar SerDes personalizados

Instalação do SerDe



- O SerDe é uma aplicação Java em formato jar e a versão utilizada nos exemplos está disponível na internet
- Os SerDes serão gravados no diretório de /app (crie o diretório) de cada host do Hive (EdgeNode na nossa arquitetura)
- Exemplo:
 - `$ mkdir /app; cd /app`
 - `$ wget http://central.maven.org/maven2/com/ibm/spss/hive/serde2/xml/hivexmlserde/1.0.5.3/hivexmlserde-1.0.5.3.jar`
- Na configuração do Hive precisamos adicionar esse novo diretório com as bibliotecas externas que serão usadas no cluster
 - A propriedade é `hive.aux.jars.path`, altere com o valor `/app` e reinicie os serviços

Hive Auxiliary JARs
Directory

Hive (Service-Wide) ↻

/app/hivexmlserde-1.0.5.3.jar

Dataset StackOverflow stackoverflow

- Disponível em
 - <https://archive.org/download/stackexchange>
- Estão disponíveis os dados da versão em português

pt.stackoverflow.com.7z	05-Jun-2018 15:38	273.3M
---	-------------------	--------

- E a versão completa em inglês

stackoverflow.com-Badges.7z	05-Jun-2018 15:48	196.7M
stackoverflow.com-Comments.7z	05-Jun-2018 15:58	3.6G
stackoverflow.com-PostHistory.7z	05-Jun-2018 17:05	21.1G
stackoverflow.com-PostLinks.7z	05-Jun-2018 22:05	68.7M
stackoverflow.com-Posts.7z	05-Jun-2018 22:30	12.1G
stackoverflow.com-Tags.7z	05-Jun-2018 21:49	736.7K
stackoverflow.com-Users.7z	05-Jun-2018 21:50	365.7M
stackoverflow.com-Votes.7z	05-Jun-2018 21:52	896.4M

Esquema dos Users



```
<row Id="-1"  
  Reputation="1"  
  CreationDate="2013-11-18T22:54:05.127"  
  DisplayName="Comunidade"  
  LastAccessDate="2013-11-18T22:54:05.127"  
  WebsiteUrl="http://meta.stackexchange.com/"  
  Location="no datacenter"  
  AboutMe="xxxxxxx"  
  Views="0"  
  UpVotes="6276"  
  DownVotes="10567"  
  AccountId="-1" />
```

Esquema dos Comments



```
<row Id="2"  
  PostId="35314"  
  Score="8"  
  Text="xxxxxxxxxxxxxx"  
  CreationDate="2008-09-06T08:09:52.330"  
  UserId="3" />
```

Dataset no HDFS

- Diretório no HDFS para gravar os dados
 - /user/marcoreis/dados/stackoverflow/user/
 - /user/marcoreis/dados/stackoverflow/comment/
- Copiar os arquivos Users.xml e Comments.xml para os respectivos diretórios

Tabela temporária User

```
drop table if exists datalake.userstackoverflow_temp;
create external table datalake.userstackoverflow_temp (
  id bigint,
  reputation bigint,
  creationdate string,
  displayname string,
  lastaccessdate string,
  websiteurl string,
  location string,
  views bigint,
  upvotes bigint,
  downvotes bigint,
  accountid bigint )
ROW FORMAT SERDE 'com.ibm.spss.hive.serde2.xml.XmlSerDe'
WITH SERDEPROPERTIES (
  "column.xpath.id"="/row/@Id/text()",
  "column.xpath.reputation"="/row/@Reputation/text()",
  "column.xpath.creationdate"="/row/@CreationDate/text()",
  "column.xpath.displayname"="/row/@DisplayName/text()",
  "column.xpath.lastaccessdate"="/row/@LastAccessDate/text()",
  "column.xpath.websiteurl"="/row/@WebsiteUrl/text()",
  "column.xpath.location"="/row/@Location/text()",
  "column.xpath.views"="/row/@Views/text()",
  "column.xpath.upvotes"="/row/@UpVotes/text()",
  "column.xpath.downvotes"="/row/@DownVotes/text()",
  "column.xpath.accountid"="/row/@AccountId/text()"
)
STORED AS
INPUTFORMAT 'com.ibm.spss.hive.serde2.xml.XmlInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'
LOCATION '/user/hive/dados/staging'
TBLPROPERTIES (
  "xmlinput.start" = "<row",
  "xmlinput.end" = ">"
);
```



Mapeamento do XML

- O SerDe usa a estrutura do Hive para criar a tabela e seus campos, por exemplo o id
 - id bigint
- O registro do SerDe, indicando o formato do arquivo XML usa a sintaxe:
 - ROW FORMAT SERDE 'com.ibm.spss.hive.serde2.xml.XmlSerDe'
- O mapeamento do campo id da tabela com o campo Id do XML usa a sintaxe:
 - "column.xpath.id"="/row/@Id"

CTAS do User

- O CTAS é a forma mais simples de criar a tabela definitiva e dar a carga automaticamente
- Para criar a tabela User use o exemplo:
 - SET parquet.compression=snappy;
 - SET hive.execution.engine=spark;
 - create table datalake.userstackoverflow
 - stored as parquet
 - as select * from datalake.userstackoverflow_temp;
- Opcionalmente, podemos remover a anterior:
 - drop table datalake.userstackoverflow_temp;

Tabela externa commentstackoverflow_temp

```
create external table datalake.commentstackoverflow_temp (  
  id bigint,  
  postid bigint,  
  text string,  
  creationdate string,  
  userid string )  
ROW FORMAT SERDE 'com.ibm.spss.hive.serde2.xml.XmlSerDe'  
WITH SERDEPROPERTIES (  
  "column.xpath.id"="/row/@Id/text()",  
  "column.xpath.postid"="/row/@PostId/text()",  
  "column.xpath.score"="/row/@Score/text()",  
  "column.xpath.text"="/row/@Text/text()",  
  "column.xpath.creationdate"="/row/@CreationDate/text()",  
  "column.xpath.userid"="/row/@UserId/text()" )  
STORED AS  
INPUTFORMAT 'com.ibm.spss.hive.serde2.xml.XmlInputFormat'  
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'  
LOCATION '/user/dataengineer/dados/stackoverflow/comment'  
TBLPROPERTIES (  
  "xmlinput.start" = "<row",  
  "xmlinput.end" = ">" )  
);
```

Tabela Hive commentstackoverflow

```
SET parquet.compression=snappy;  
SET hive.execution.engine=spark;  
create table datalake.commentstackoverflow  
stored as parquet  
as select * from datalake.commentstackoverflow_temp;
```


JSON com SerDe



- Há vários SerDes disponíveis para JSON e dois merecem destaque:
 - `org.apache.hive.hcatalog.data.JsonSerDe`: faz parte do Hive, é mais prático
 - `org.openx.data.jsonserde.JsonSerDe`: é bastante popular
- Download dos arquivos do SerDe OpenX no diretório `/app` (já configurado anteriormente)
 - `$ cd /app`
 - `$ wget http://www.congiu.net/hive-json-serde/1.3.8/cdh5/json-serde-1.3.8-jar-with-dependencies.jar`
 - `$ wget http://www.congiu.net/hive-json-serde/1.3.8/cdh5/json-udf-1.3.8-jar-with-dependencies.jar`

Tabela movies

- O arquivo JSON deve estar bem formado, entretanto ele contém caracteres inválidos para o SerDe
- O formato original do movies.json é:
 - [{"title":"After Dark in Central Park","year":1900,"director":null,"cast":null,"genre":null,"notes":null},{ "title":"Boarding School Girls' Pajama Parade","year":1900,"director":null,"cast":null,"genre":null,"notes":null}]
- Alguns problemas devem ser resolvidos, por exemplo, com expressão regular e sed
 - Retirar os colchetes: []
 - Retirar as vírgulas
 - Cada registro deve estar em uma linha
- Exemplo:
 - sed -i 's/[\|\\]/g' movies.json
 - sed -i 's/},{/}\n{/g' movies.json
- Exemplo de JSON bem formado
 - {"title":"Boarding School Girls' Pajama Parade","year":1900,"director":null,"cast":null,"genre":null,"notes":null}
 - {"title":"Buffalo Bill's Wild West Parad","year":1900,"director":null,"cast":null,"genre":null,"notes":null}

Tabela externa movies_temp

```
create external table movies_temp (  
  title string,  
  year string,  
  director string,  
  `cast` string,  
  genre string,  
  notes string )  
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'  
STORED AS TEXTFILE  
LOCATION '/user/dataengineer/dados/movies';
```

Tabela externa movies_hcat_temp

- O SerDe próprio do Hive não precisa de registro e pode ser usado diretamente:

```
create external table datalake.movies_hcat_temp (  
  title string,  
  year string,  
  director string,  
  `cast` string,  
  genre string,  
  notes string )  
ROW FORMAT serde 'org.apache.hive.hcatalog.data.JsonSerDe'  
WITH SERDEPROPERTIES ( 'paths'='title, year, director, cast, genre, notes' )  
STORED AS TEXTFILE  
LOCATION '/user/dataengineer/dados/movies';
```

Tabela Hive movies

- Para criar a tabela no Hive adotamos o CTAS
- Podemos criar a partir da tabela movies_temp ou movies_hcat_temp
- Exemplo:

```
SET parquet.compression=snappy;  
SET hive.execution.engine=spark;  
create table datalake.movies  
stored as parquet  
as select * from datalake.movies_temp;
```
- Uma sugestão de consulta seria verificar quais os diretores com mais filmes na lista

```
select count(1) as total, director from movies  
group by director  
order by total desc
```

Sugestão de atividade: tabela wine

- O arquivo winemag-data-130k-v2.json contém uma seleção de vinhos e avaliações
- O formato do arquivo não está de acordo com o SerDe. Veja:
 - [{"points": "87", "title": "Nicosia 2013 Vulk\u00e0 Bianco (Etna)", "description": "Aromas include tropical fruit, broom, brimstone and dried herb. The palate isn't overly expressive, offering unripened apple, citrus and dried sage alongside brisk acidity.", "taster_name": "Kerin O\u00d9Keefe", "taster_twitter_handle": "@kerinokeefe", "price": null, "designation": "Vulk\u00e0 Bianco", "variety": "White Blend", "region_1": "Etna", "region_2": null, "province": "Sicily & Sardinia", "country": "Italy", "winery": "Nicosia"}, {"points": "87", "title": "Quinta dos Avidagos 2011 Avidagos Red (Douro)", "description": "This is ripe and fruity, a wine that is smooth while still structured. Firm tannins are filled out with juicy red berry fruits and freshened with acidity. It's already drinkable, although it will certainly be better from 2016.", "taster_name": "Roger Voss", "taster_twitter_handle": "@vossroger", "price": 15, "designation": "Avidagos", "variety": "Portuguese Red", "region_1": null, "region_2": null, "province": "Douro", "country": "Portugal", "winery": "Quinta dos Avidagos"}]
- Expressões regulares para correção
 - sed -i 's/[\\|]//g' winemag-data-130k-v2.json
 - sed -i 's/}, {/}\\n{/g' winemag-data-130k-v2.json
- Criar as tabelas externas e Hive para o arquivo

Sqoop



- O Sqoop é um utilitário para importação e exportação de dados em lote
- É altamente configurável, como pode ser visto nas referências no final da apostila
- Ele cria um job MapReduce e executa a operação no cluster Hadoop, usando os recursos distribuídos para máxima performance
- Roteiro para exportação
 - Criar a tabela de destino no SGBDR (MariaDB) para receber os dados
 - Selecionar o diretório de origem no HDFS, pode ser um diretório existente ou uma tabela Hive
 - Exportar dados com o Sqoop
- Roteiro para importação
 - Selecionar as tabelas de origem no SGBDR para importação
 - Selecionar o diretório de destino no HDFS ou a tabela no Hive
 - Importar os dados com o Sqoop

Exportação com Sqoop

- Criar a tabela de destino no SGBDR

```
create database datalake;  
use datalake;  
create table datalake.populacao_uf (  
  uf varchar(2),  
  populaaoresidente varchar(20));
```

- Diretório no HDFS

```
create table datalake.populacao_uf  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY '#'  
STORED AS TextFile  
LOCATION '/user/dataengineer/saida/populacao_uf/'  
as select uf, cast(sum(populaaoresidente) as  
decimal(20)) as populaaoresidente  
from datalake.municipio  
group by uf;
```

- Exportação com Sqoop

```
url=jdbc:mysql://mariadbserver/datalake  
user=root  
pwd=root  
tabela=populacao_uf  
sqoop \
```

```
export \
```

```
--connect $url \
```

```
--username $user \
```

```
--password $pwd \
```

```
--verbose \
```

```
--table $tabela \
```

```
--input-fields-terminated-by "#" \
```

```
--export-dir /user/dataengineer/saida/populacao_uf
```

- Consulte os dados no MariaDB

```
- select * from datalake.populacao_uf;
```




Observações sobre a exportação

- A exportação geralmente é usada para dados consolidados
 - Resultados de processamento do Hadoop/Hive, como relatórios com poucas linhas
- Podemos exportar apenas texto, ou seja, não é possível exportar tabelas em formatos como Parquet/Avro
 - Antes, crie uma tabela Hive em formato TextFile, com um separador como # (ou outro qualquer)
- No exemplo, vamos criar um diretório HDFS ('/user/dataengineer/saida/populacao_uf/') para exportar para o SGBDR
- Por fim, criamos uma consulta com dados consolidados por UF
 - `select uf, cast(sum(populacaoresidente) as decimal(20)) as populacaoresidente from datalake.municipio group by uf;`

Importação com Sqoop

- Este é a operação mais comum do Sqoop, quando os dados do SGBDR são copiados para o cluster (HDFS, Hive, HBase)
- Há opção para importar tabelas, consultas ou a base de dados inteira
 - Para importações recorrentes, pode ser usado o recurso da importação incremental, ou seja, inclusão ou atualização dos novos registros na tabela
- A seguir veremos diversos scripts para importação

Listar tabelas da base de dados

```
$ url=jdbc:mysql://mariadbserver/datalake
```

```
$ user=root
```

```
$ pwd=root
```

```
$ sqoop \
```

```
list-tables \
```

```
--connect $url \
```

```
--username $user \
```

```
--password $pwd \
```

```
--verbose
```

Importação para HDFS com 1 mapper

- Importação a partir de uma consulta SQL
- Utiliza um único mapper, ou seja, um único processo sequencial
- Adequado para tabelas pequenas/médias, de até poucos milhões de registros
- Veja que é necessário o uso da expressão \$CONDITIONS

```
$ sqoop \  
import \  
--query 'select * from populacao_uf where $CONDITIONS' \  
--connect $url \  
--username $user \  
--password $pwd \  
--verbose \  
--target-dir /user/dataengineer/dados/populacao_uf_hdfs_1_mapper/ \  
--num-mappers 1
```

```
$ hdfs dfs -cat /user/dataengineer/dados/populacao_uf_hdfs_1_mapper/part-m-00000
```

```
AC,733559  
AL,3120494  
AM,3483985  
AP,669526  
BA,14016906  
CE,8452381  
DF,2570160  
ES,3514952  
GO,6003788  
MA,6574789  
MG,19597330  
MS,2449024  
MT,3035122  
PA,7581051  
PB,3766528  
PE,8796448  
PI,3118360  
PR,10444526  
RJ,15989929  
RN,3168027  
RO,1562409  
RR,450479  
RS,10693929  
SC,6248436  
SE,2068017  
SP,41262199  
TO,1383445
```

Importação com 4 mappers

- O padrão da importação são 4 mappers, ou seja, 4 processos paralelos
- Os registros do SGBDR são gravados em 4 arquivos no HDFS
- O paralelismo é definido de acordo com a coluna informada no parâmetro split-by
- O número de mappers pode ser adaptado de acordo com a carga, mas considere que pode ser oneroso para o banco (memória e processamento)
- Exemplo:

```
$ sqoop \  
import \  
--table populacao_uf \  
--connect $url \  
--username $user \  
--password $pwd \  
--verbose \  
--target-dir /user/dataengineer/dados/populacao_uf_hdfs_split/ \  
--split-by uf
```

```
$ hdfs dfs -cat /user/dataengineer/dados/populacao_uf_hdfs_split/part-m-00003
```

```
dataengineer@hadoopclient:~$ hdfs dfs -ls /user/dataengineer/dados/populacao_uf_hdfs_split  
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".  
SLF4J: Defaulting to no-operation (NOP) logger implementation  
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.  
Found 5 items  
-rw-r--r-- 1 dataengineer dataengineer 0 2018-08-29 20:09 /user/dataengineer/dados/populacao_uf_hdfs_split/_SUCCESS  
-rw-r--r-- 1 dataengineer dataengineer 87 2018-08-29 20:09 /user/dataengineer/dados/populacao_uf_hdfs_split/part-m-00000  
-rw-r--r-- 1 dataengineer dataengineer 11 2018-08-29 20:09 /user/dataengineer/dados/populacao_uf_hdfs_split/part-m-00001  
-rw-r--r-- 1 dataengineer dataengineer 45 2018-08-29 20:09 /user/dataengineer/dados/populacao_uf_hdfs_split/part-m-00002  
-rw-r--r-- 1 dataengineer dataengineer 157 2018-08-29 20:09 /user/dataengineer/dados/populacao_uf_hdfs_split/part-m-00003
```

Tabelas externas no Hive

- Podemos analisar os dados diretamente no HDFS ou podemos criar tabelas externas para análises rápidas
- Exemplo a ser executado no cliente Hive:

```
create external table datalake.populacao_uf_split (  
  uf string,  
  populacaoresidente string )  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY ','  
location '/user/dataengineer/dados/populacao_uf_hdfs_split/';
```

```
create external table datalake.populacao_uf_1_mapper (  
  uf string,  
  populacaoresidente string )  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY ','  
location '/user/dataengineer/dados/populacao_uf_hdfs_1_mapper/';
```

Importação para Hive/Parquet

- O Sqoop do CDH tem a opção de importar os dados direto com o uma tabela Hive em formato Parquet
 - Não disponível no Sqoop da Apache
- Exemplo

```
$ sqoop \  
import \  
--table populacao_uf \  
--connect $url \  
--username $user \  
--password $pwd \  
--verbose \  
--compression-codec snappy \  
--hive-import \  
--create-hive-table \  
--hive-database datalake \  
--hive-table populacao_uf_parquet \  
--as-parquetfile \  
--split-by uf
```

Flume





Flume

- O Apache Flume é um serviço para coleta de logs de alta performance utilizado no Hadoop para ingestão de dados em streaming (fluxo constante), um tipo de tecnologia adequado para aplicações de análise de dados em tempo real
- As configurações são feitas por meio de arquivos, o que permite grande flexibilidade por meio de diversas combinações
- O Flume é composto por 3 componentes:
 - Source: origem dos dados (Avro, JMS, Diretórios etc)
 - Channel: canal onde os dados são processados (memória, JDBC, Kafka, arquivo)
 - Sinks: destino onde os dados serão gravados (HDFS, Hive, Logger, Avro, HBase etc)
- O Flume pode ser visto como um listener passivo, pois faz apenas transformações muito simples nos dados. A análise deve ser feita depois da gravação no sink



Agente Flume

- Um host pode ter diversos agentes Flume e deve-se considerar que este é um serviço que consome processamento e memória, mas não é necessário muito espaço em disco
- Preferencialmente, deve estar em um host separado dos DataNodes
- Nosso exemplo será o agente_edgenode, como visto na imagem abaixo
- A configuração do source/channel/sink fica em Configuration File
- O agente vai ler o arquivo indicado (source) e cada nova linha será gravada na memória (channel) até ser finalmente gravada no HDFS (sink)
- Verifique o diretório HDFS o que está acontecendo e note que a gravação é em tempo real

Filters

Clear All

SCOPE

Clear

Flume (Service-Wide) 18

Agent 48

CATEGORY

Advanced 9

Flume-NG Solr Sink 3

Logs 4

Main 4

Monitoring 16

Performance 1

Ports and Addresses 1

Resource Management 5

Agent Name

Agent Default Group

agente_edgenode

Configuration File

Agent Default Group

Source
agente_edgenode.sources = fonte_log_servidor
agente_edgenode.sources.fonte_log_servidor.type = TAILDIR
agente_edgenode.sources.fonte_log_servidor.channels = canal_in_memory
agente_edgenode.sources.fonte_log_servidor.filegroups = arquivo_log_namenode
agente_edgenode.sources.fonte_log_servidor.filegroups.arquivo_log_namenode = /

Flume Home Directory

Agent Default Group

/var/lib/flume-ng



Configuração do Flume

```
• # Source
• agente_edgenode.sources = fonte_log_servidor
• agente_edgenode.sources.fonte_log_servidor.type = TAILDIR
• agente_edgenode.sources.fonte_log_servidor.channels = canal_in_memory
• agente_edgenode.sources.fonte_log_servidor.filegroups = arquivo_log_namenode
• agente_edgenode.sources.fonte_log_servidor.filegroups.arquivo_log_namenode = /var/log/hadoop-hdfs/hadoop-cmf-hdfs-NAMENODE-edgenode.lab.log.out
•
• # Channels
• agente_edgenode.channels = canal_in_memory
• agente_edgenode.channels.canal_in_memory.type = memory
• agente_edgenode.channels.canal_in_memory.capacity = 10000
• agente_edgenode.channels.canal_in_memory.transactionCapacity = 2000
•
• # Sink
• agente_edgenode.sinks = destino_log
• agente_edgenode.sinks.destino_log.type = hdfs
• agente_edgenode.sinks.destino_log.channel = canal_in_memory
• agente_edgenode.sinks.destino_log.hdfs.path = /flume_log/edgenode
•
• # Grava arquivos de 65 k
• agente_edgenode.sinks.destino_log.hdfs.rollSize = 65536
• agente_edgenode.sinks.destino_log.hdfs.rollInterval = 0
• agente_edgenode.sinks.destino_log.hdfs.rollCount = 0
• agente_edgenode.sinks.destino_log.hdfs.filePrefix = eventos
• agente_edgenode.sinks.destino_log.hdfs.batchSize = 1000
• agente_edgenode.sinks.destino_log.hdfs.fileType = DataStream
•
• # Kerberos
• agente_edgenode.sinks.destino_log.hdfs.kerberosPrincipal = $KERBEROS_PRINCIPAL
• agente_edgenode.sinks.destino_log.hdfs.kerberosKeytab = $KERBEROS_KEYTAB
```

Dúvidas?

Marco Reis
<http://marcoreis.net>