Oficina HUE

SERPRO
Soluções digitais que conectam governo e sociedade

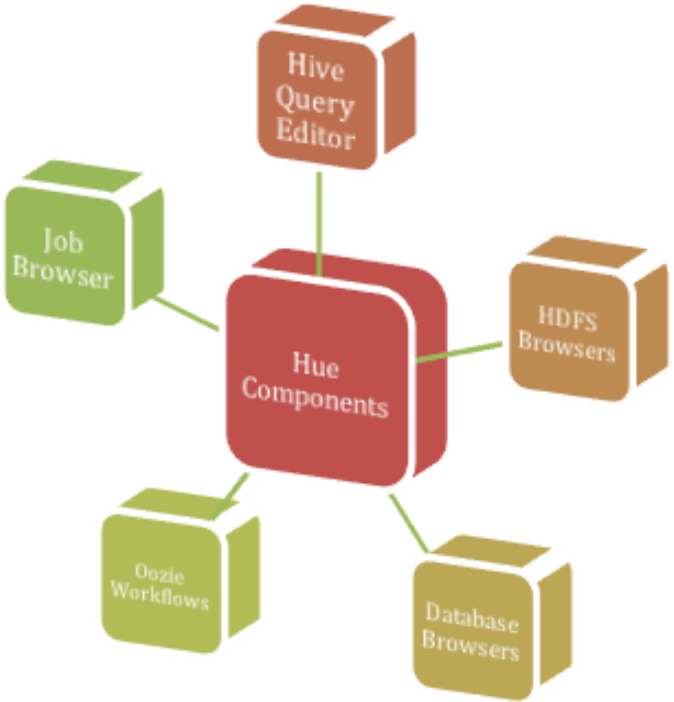
Ementa

I. Conceitos e Definições
II. Componentes
III. Navegação
IV. Job Browser
V. Database Browser
VI. Query Editor
VII. Impala x Hive
VIII. Boas Práticas Impala
IX. Hands On 1
X. Exercícios 1
XI. Workflow Oozie
XII. Hands On 2
XIII. Exercícios 2

Conceitos HUE

Hue é um assistente de SQL para bancos de dados e DW, desenvolvido com o objetivo de acessar via browser os dados / arquivos armazenados em um cluster HDFS.

- HDFS Browser
- Job Browser
- Database Browser
- Query Editor
- Oozie Workflows



Conceitos HUE

- **HDFS Browser** - Interface web cujo principal objetivo é permitir à interação do usuário com os arquivos armazenados no cluster Hadoop, como um sistema de arquivos.

- **Job Browser** - Ferramenta que nos permite acompanhar tarefas executadas no cluster Hadoop.

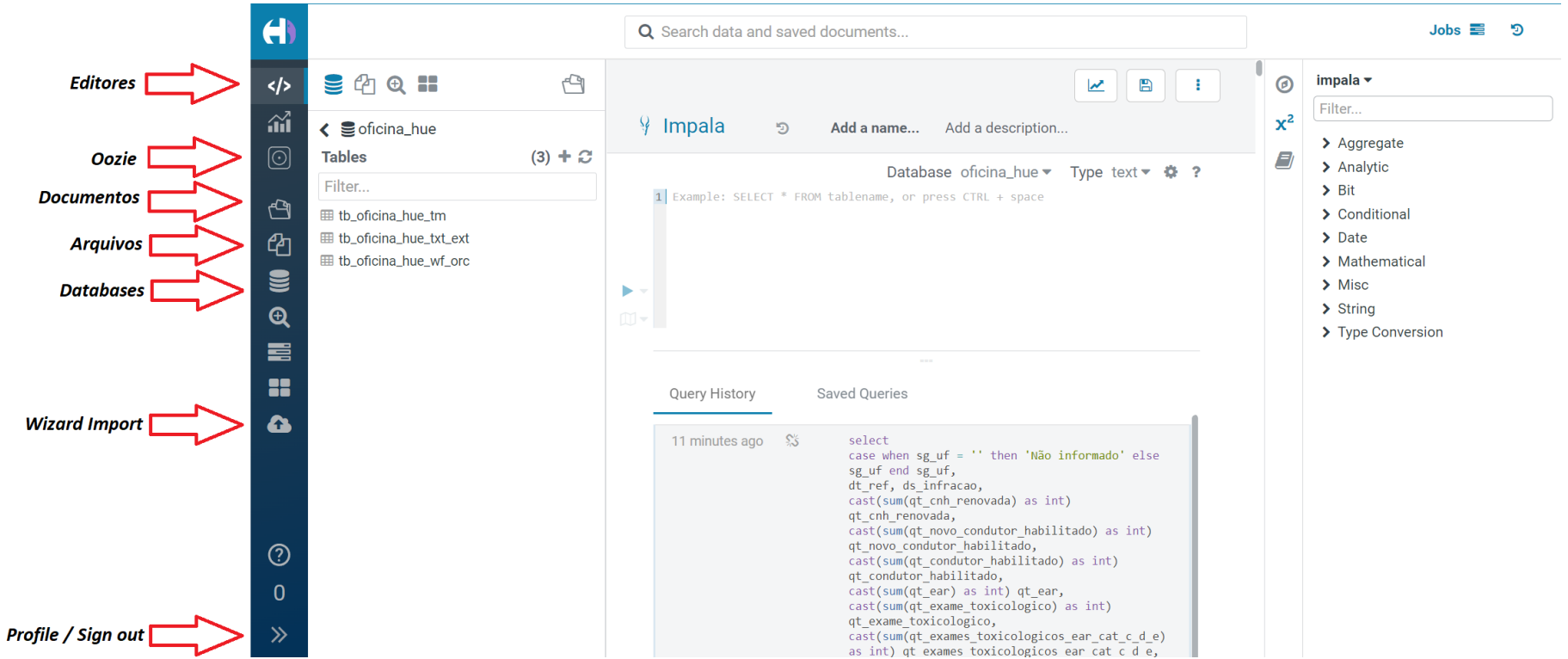
- **Database Browser** - Interface que nos permite visualizar de forma gráfica os “databases” existentes no HDFS, e seus componentes.

- **Query Editor** - Editor de consultas HQL que nos permite utilizar diversas ferramentas de consulta (Hive, Impala, etc) aos dados do cluster Hadoop de forma mais amigável.

- **Oozie Workflows** - Aplicação incorporada ao HUE que nos permite a criação de fluxos de trabalho e agendamento dos mesmos para execução no cluster Hadoop.

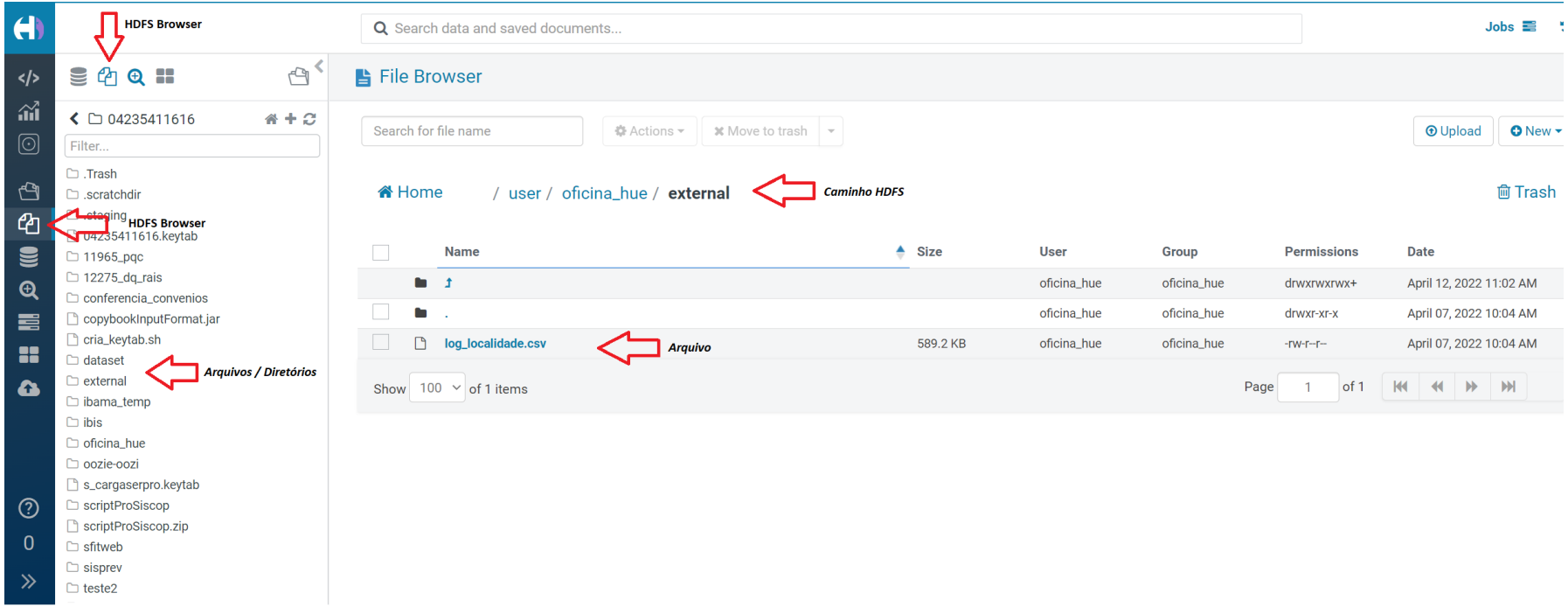
Menu acesso rápido HUE

Barra de atalhos da nova versão do HUE



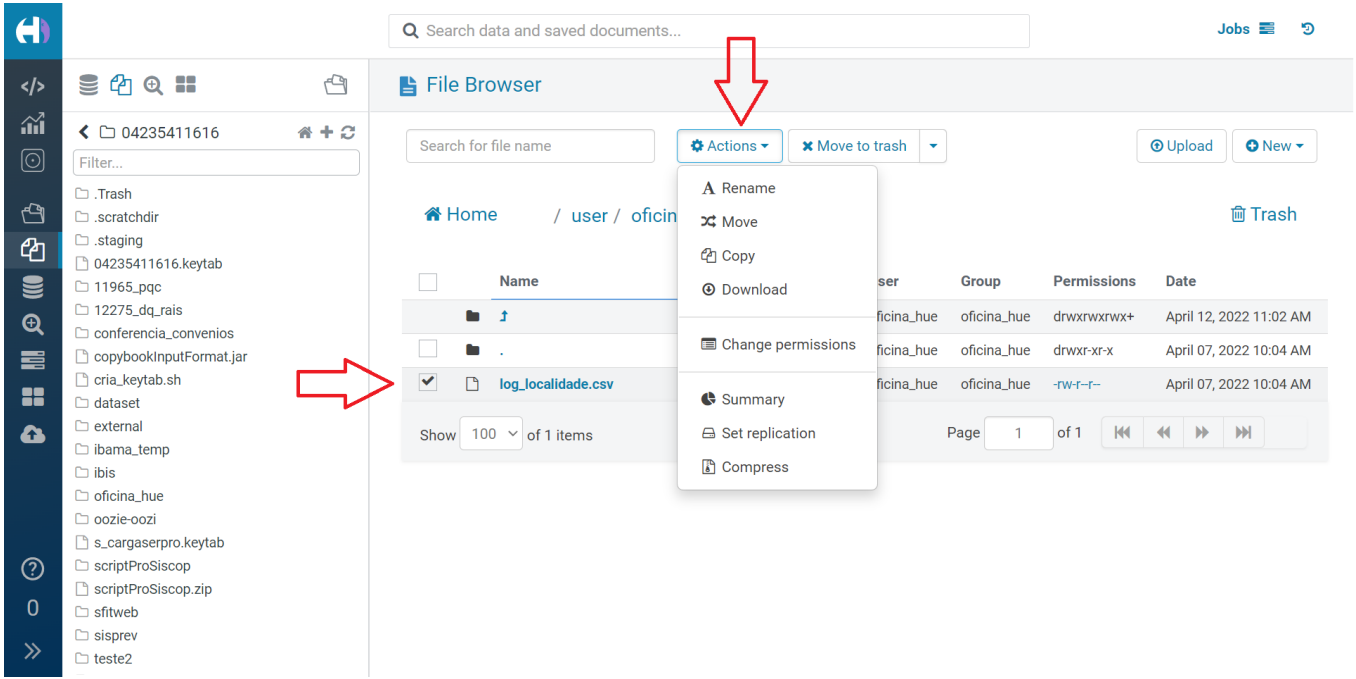
HDFS Browser - Navegação

HDFS Browser, que permite visualizar, apagar, incluir, criar, compartilhar e movimentar arquivos e diretórios no HDFS através da interface web.



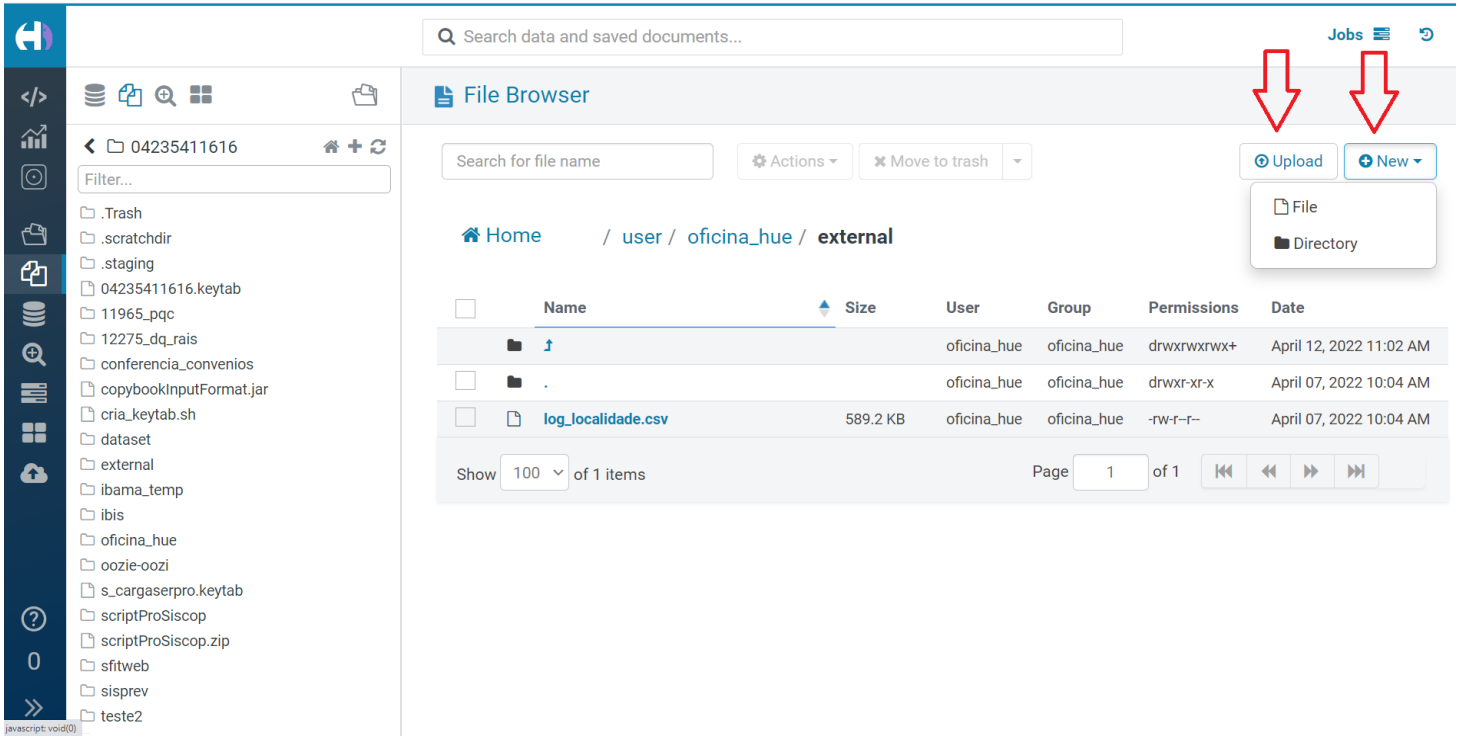
HDFS Browser - Navegação

Por esta janela o usuário poderá, dependendo das permissões nos arquivos, realizar as ações disponíveis na dropbox “*Ações*” (renomear, mover, copiar, baixar, alterar permissões, alterar número de réplicas e compactar) para os arquivos selecionados através das *checkbox*, localizadas à esquerda de cada arquivo / diretório.



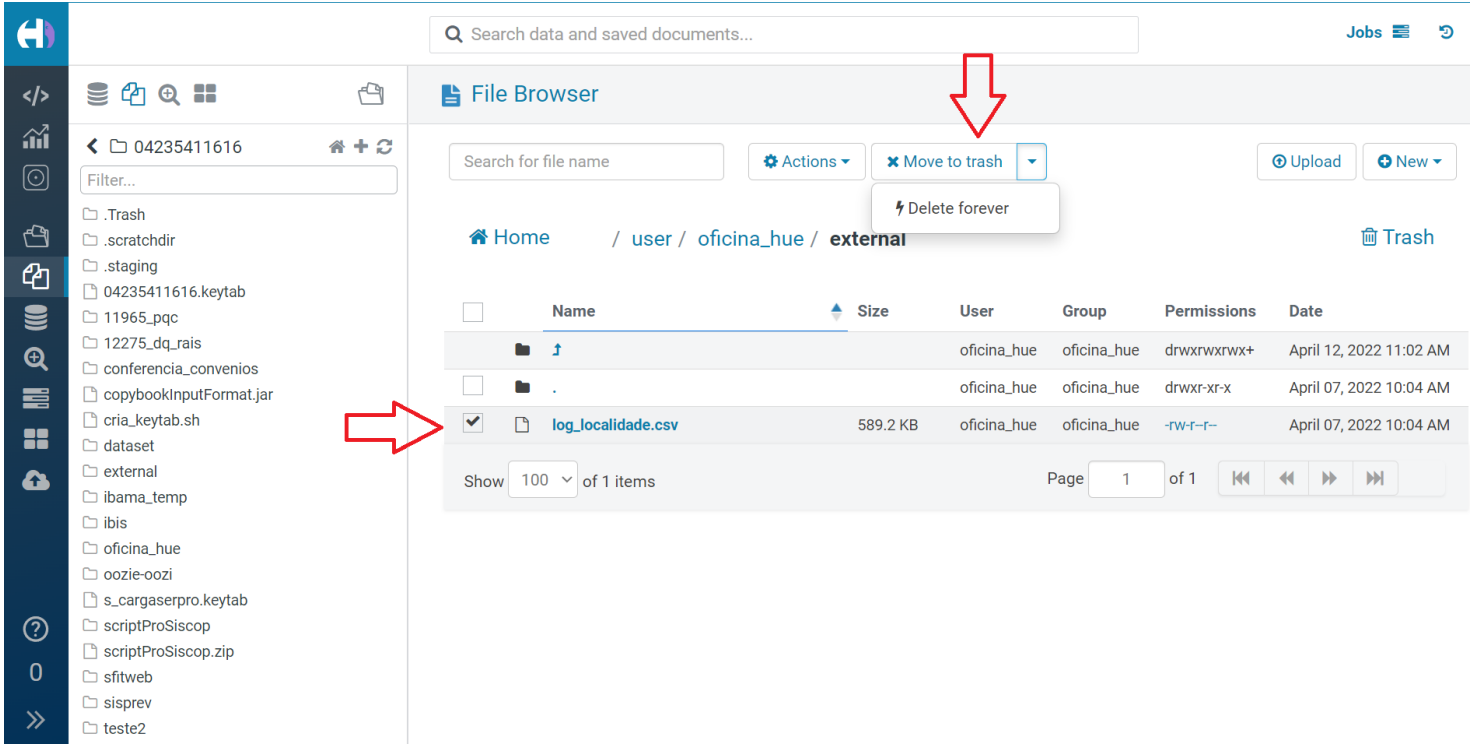
HDFS Browser - Navegação

Poderá também criar diretórios e arquivos através da dropbox “*Novo*” e fazer upload de arquivos no diretório corrente pelo botão carregar.



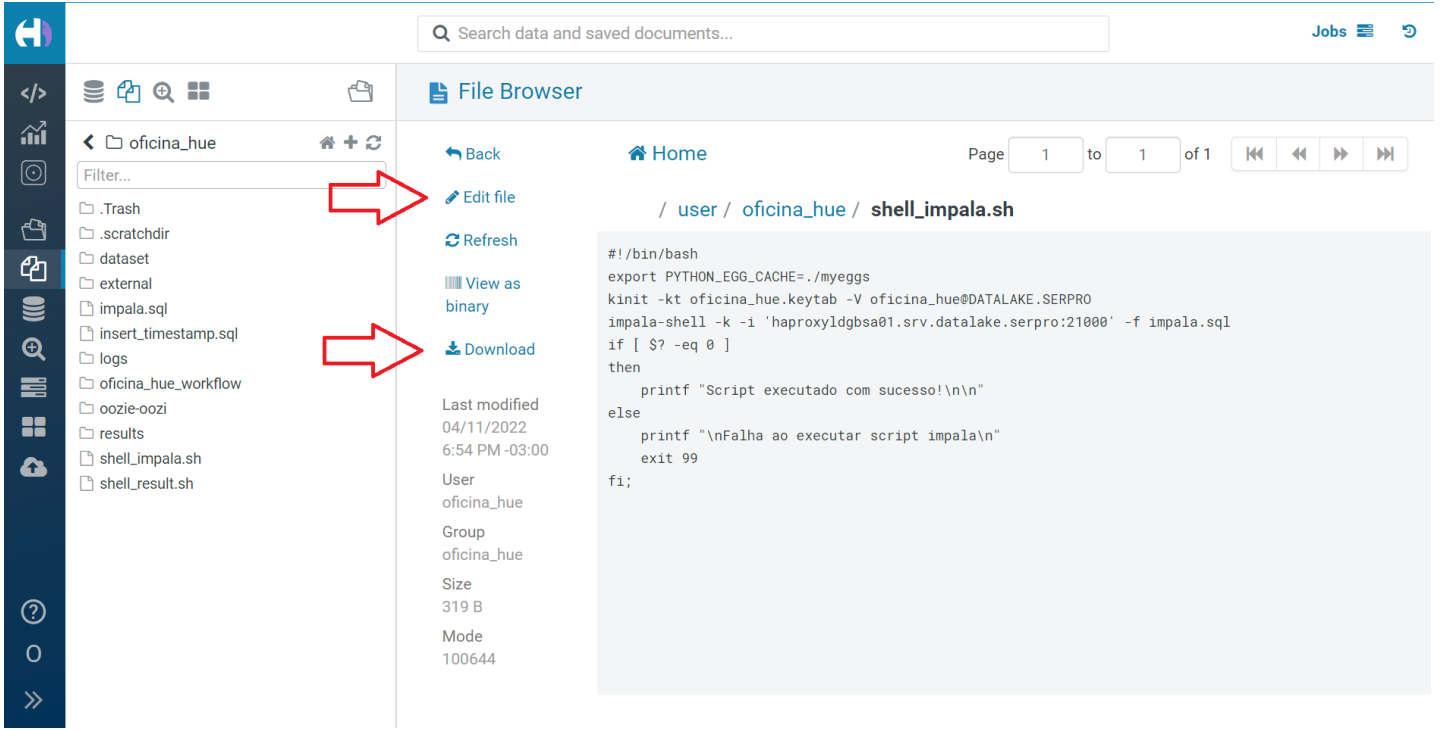
HDFS Browser - Navegação

Para apagar arquivos ou diretórios basta selecionar o referente pela checkbox e utilizar a dropbox “*Move to trash*”.



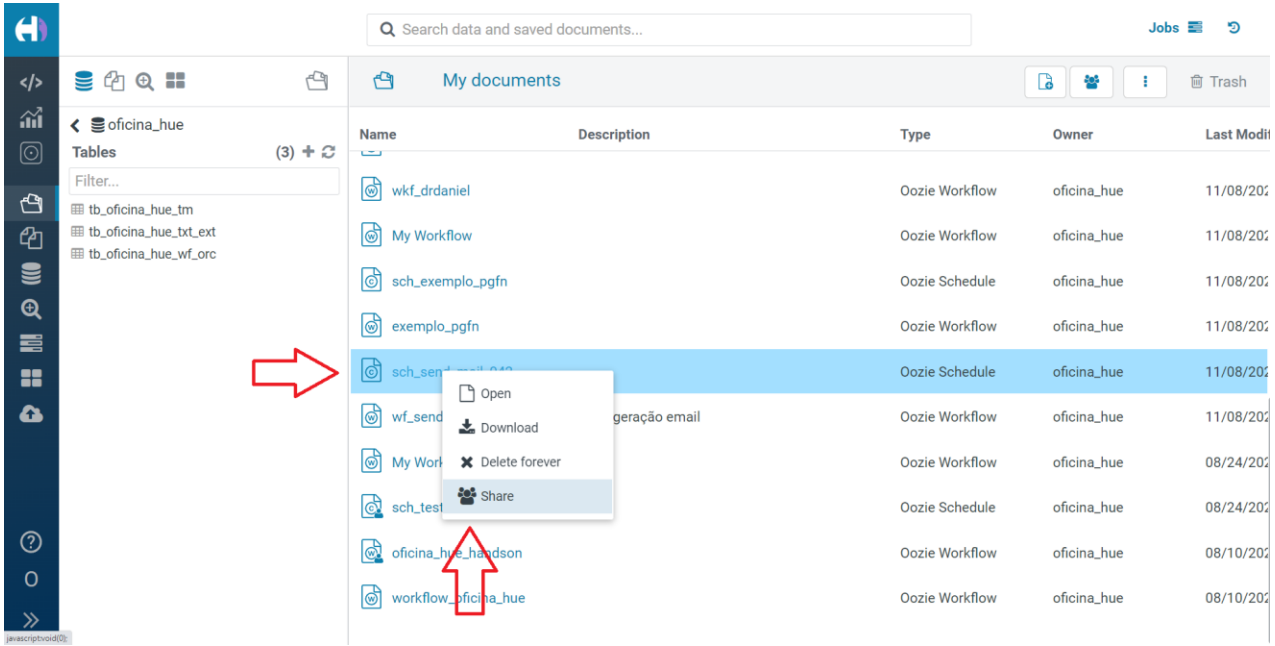
HDFS Browser - Navegação

O HDFS Browser ainda permite que realizemos alterações nos arquivos localizados no HDFS diretamente no Browser e download do mesmo, caso prefira editar em sua máquina.



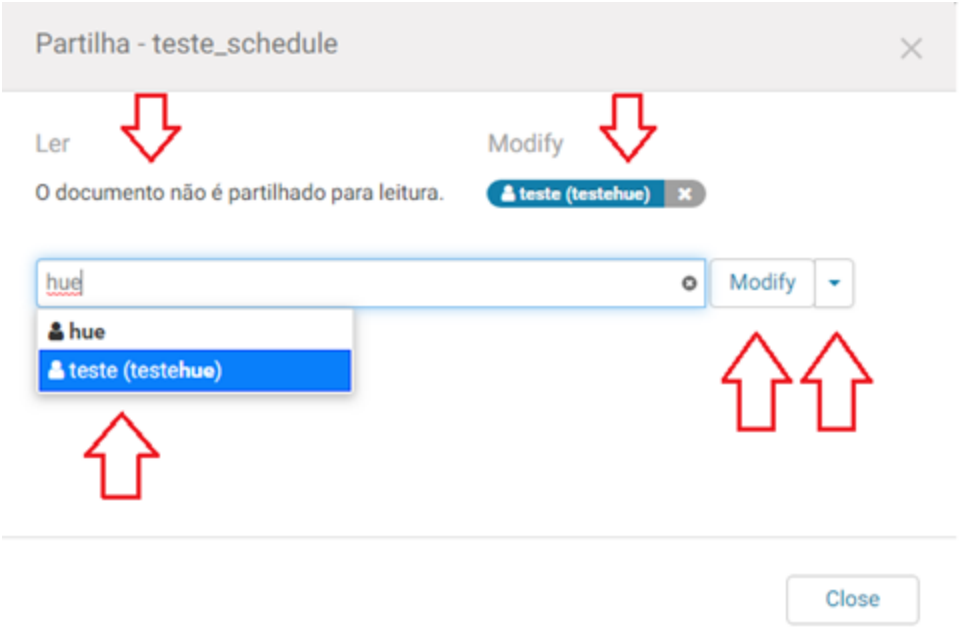
HDFS Browser - Compartilhamento

Podemos também realizar o compartilhamento de queries, workflows, schedules e outros documentos, com outros usuários do cluster. Acessando o menu principal, opção *Documents*, conseguimos visualizar os arquivos cuja edição foram feitas através das ferramentas do HUE (queries salvas, workflows oozie, schedules, diretórios, etc).



HDFS Browser - Compartilhamento

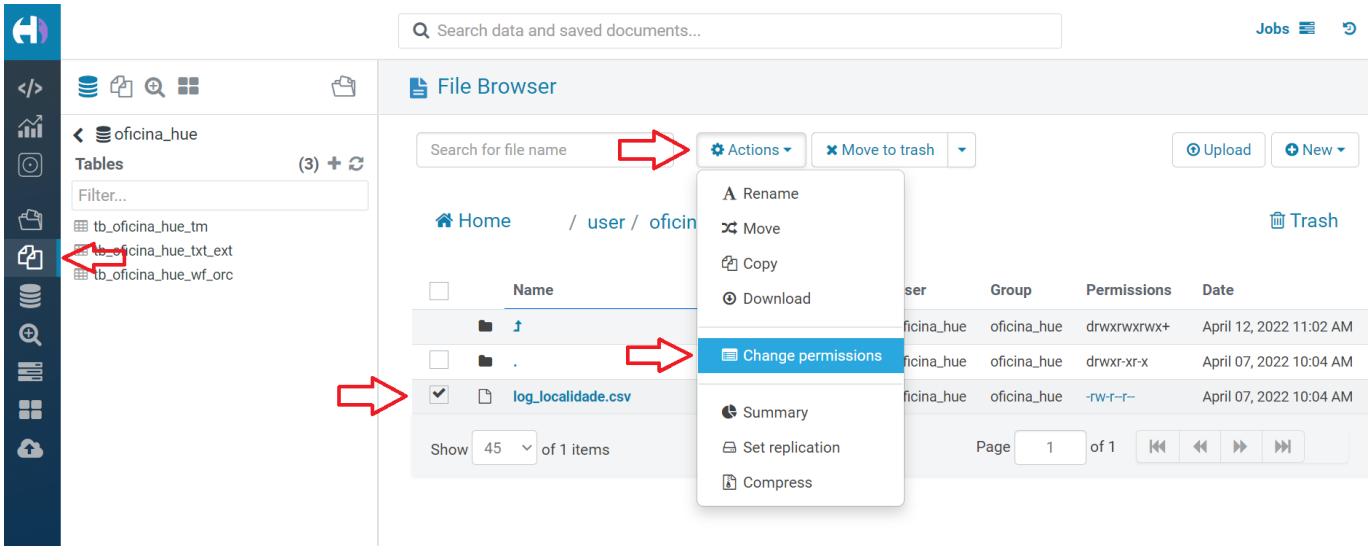
Na janela Partilha, o usuário deve digitar o nome do usuário (ou parte do nome) com quem deseja compartilhar o documento, e selecioná-lo quando mostrado. Para que seja efetivado o compartilhamento deve-se selecionar à direita qual tipo de permissão e clicar na descrição da permissão, para que a mesma apareça na parte superior da janela e seja assim efetivada.



HDFS Browser - Compartilhamento

O compartilhamento dos Documentos, conforme mostramos, pode ser realizado para usuários específicos, já o compartilhamento de arquivos dentro do HDFS só se realiza a nível de grupos ou abertura total do arquivo/diretório.

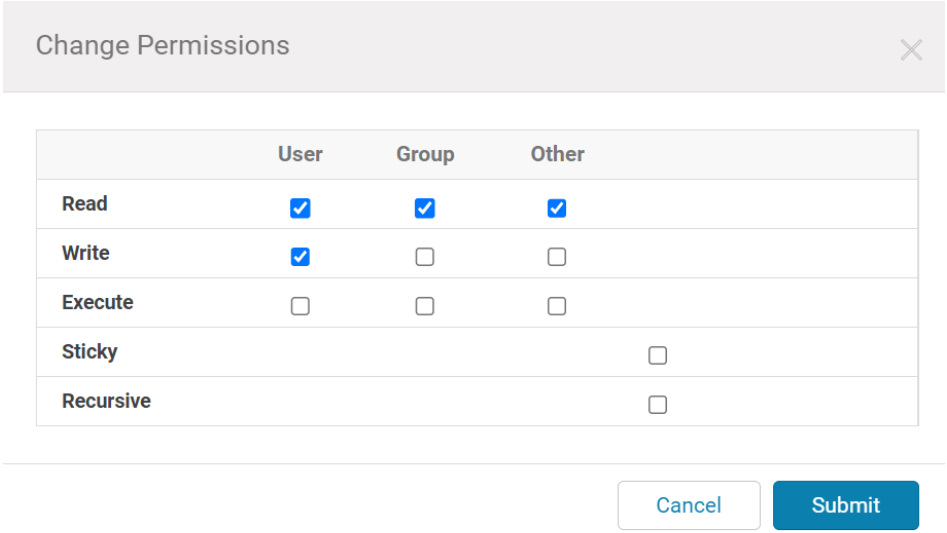
Para o compartilhamento de arquivos no HDFS devemos acessar o menu principal, opção *Files,* que nos levará para o diretório home do usuário. Selecionamos o arquivo a ser compartilhado através do checkbox à esquerda do arquivo e em seguida acessamos a dropdown *Acções (Actions)* selecionando a opção *Alterar Permissões (Change permissions)*.



HDFS Browser - Compartilhamento

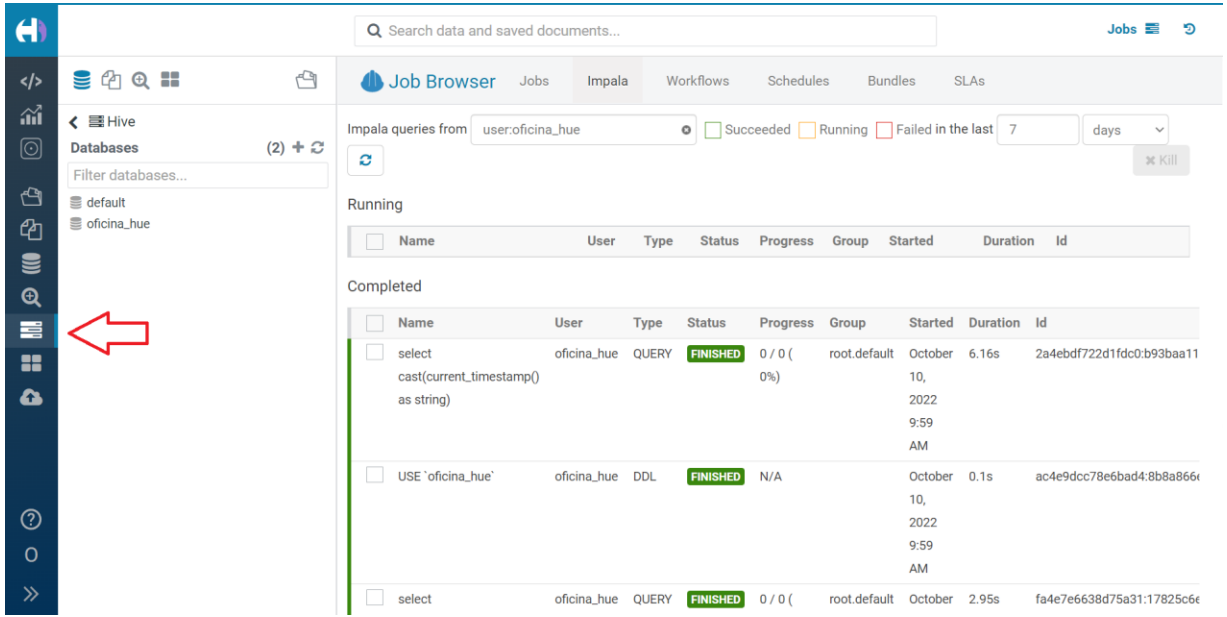
Por default, o usuário proprietário possui todas as permissões no arquivo, e os demais apenas permissão de leitura. Lembramos que mesmo que para leitura, os demais usuários ou o grupo do usuário corrente devem possuir permissão no diretório em que o arquivo se encontra, assim como nos demais diretórios caso ele não esteja no diretório raíz do usuário.

Para garantirmos que a permissão seja dada também aos diretórios de nível inferior ao diretório do arquivo, basta selecionar a opção Recursivo.



Job Browser

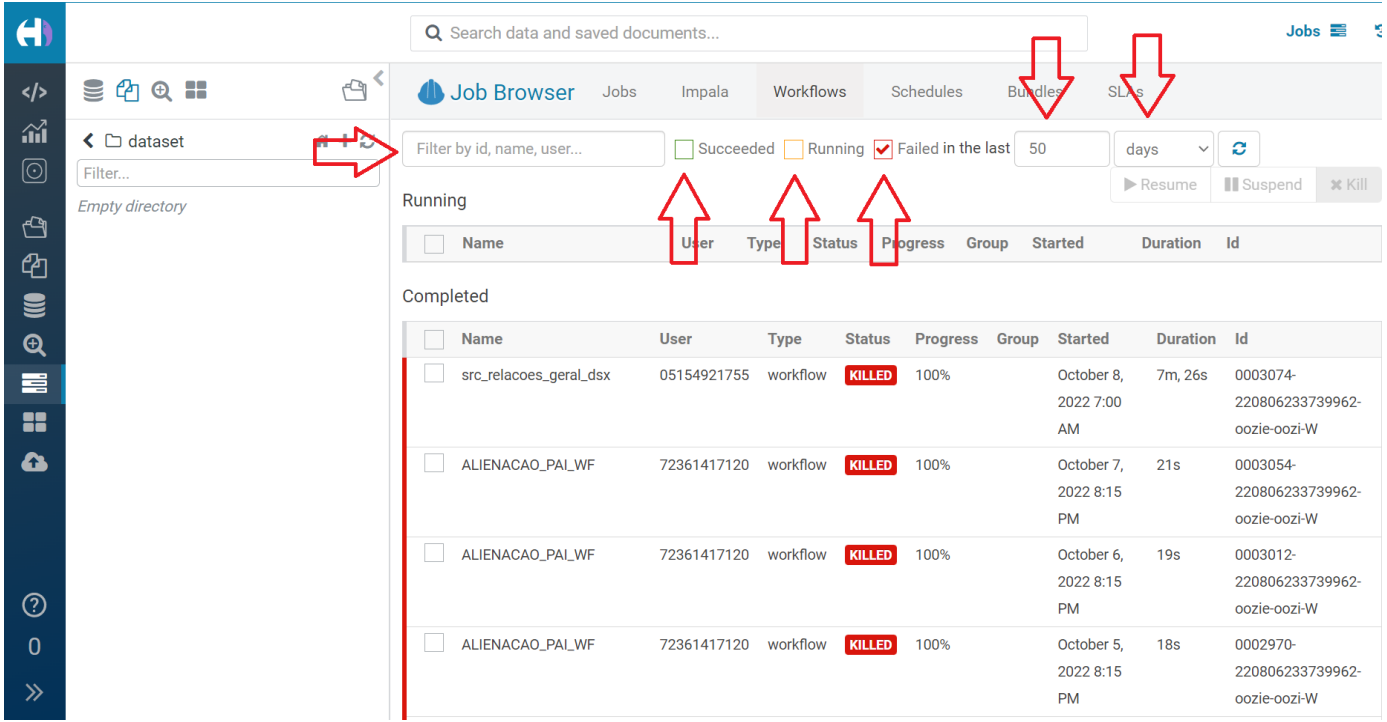
Utilizar o Hadoop implica na criação de muitos jobs e é bastante comum os desenvolvedores terem a necessidade de saber qual job está atualmente em execução no cluster do Hadoop ou qual o resultado de jobs finalizados (se houve sucesso ou falha na execução).



Por meio do navegador de jobs, podemos acessar todas as informações relacionadas ao job. Para isso, existe a opção no Hue que permite listar jobs e seus status.

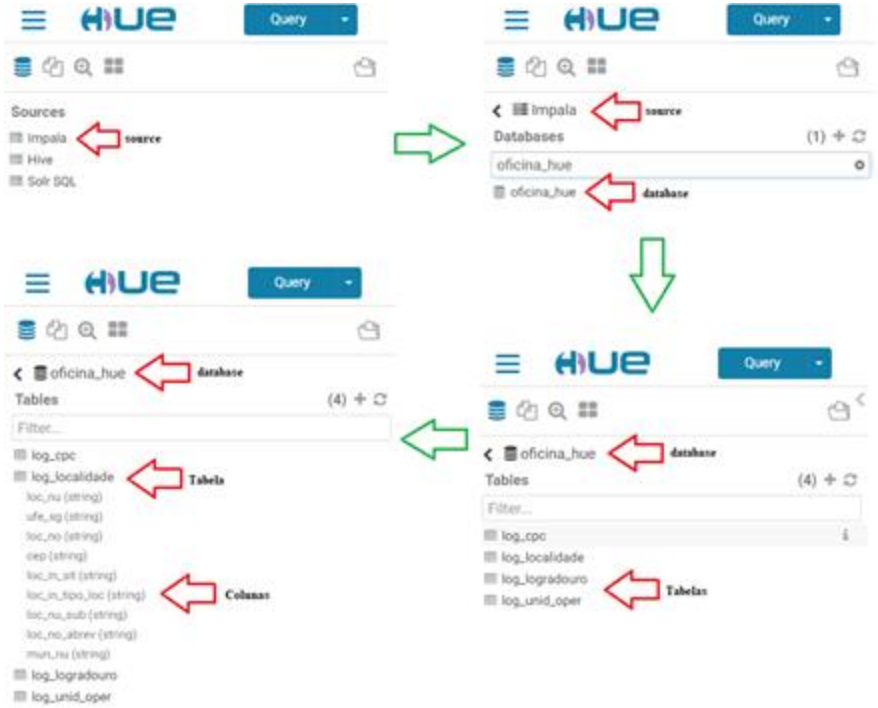
Job Browser

Podemos ainda filtrar os jobs por status, busca textual e período em que foi executado.



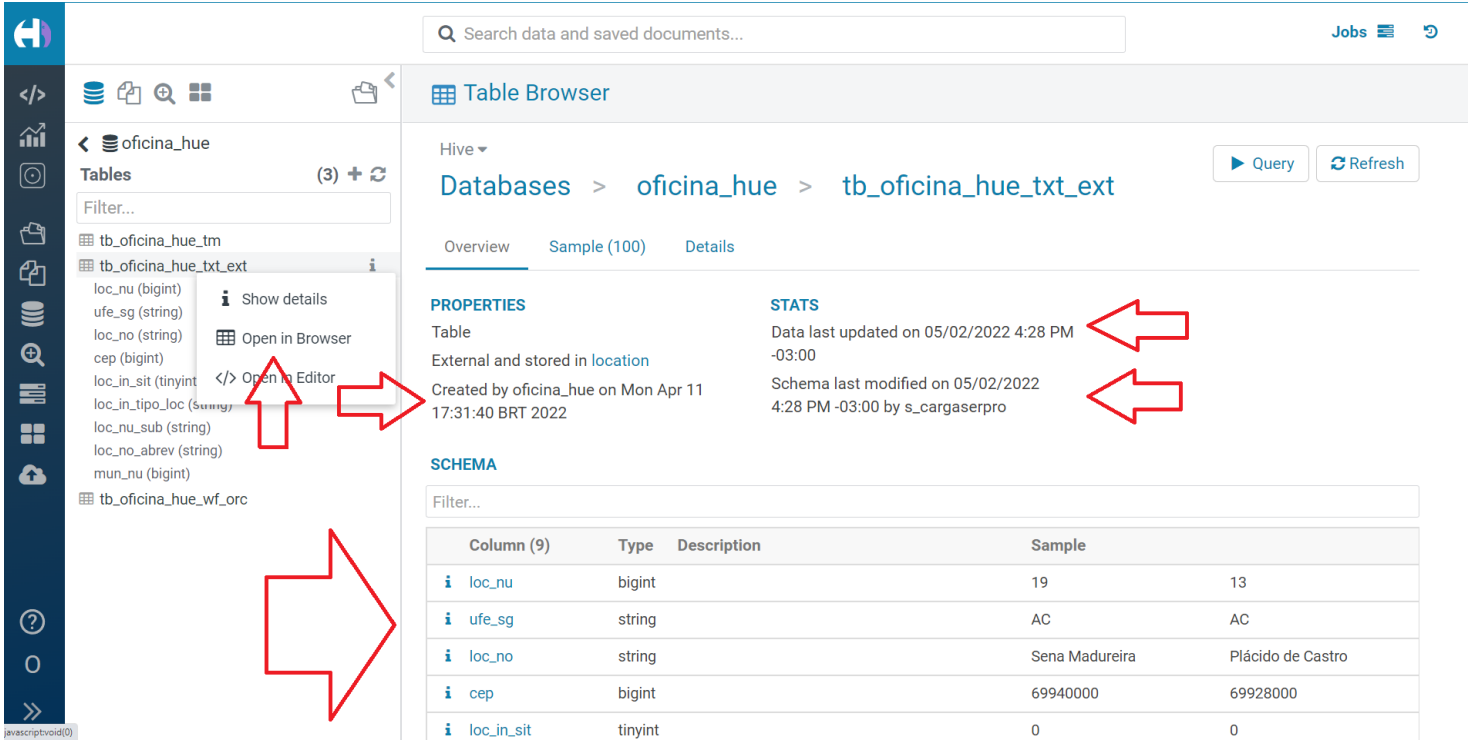
Database Browser

Através do HUE conseguimos visualizar de forma gráfica os “databases” existentes no HDFS, e seus componentes. Tal visualização é separada por fonte de dados capaz de apresentar as informações contidas nos databases (ex: hive, impala, solr, etc) até o nível de coluna.



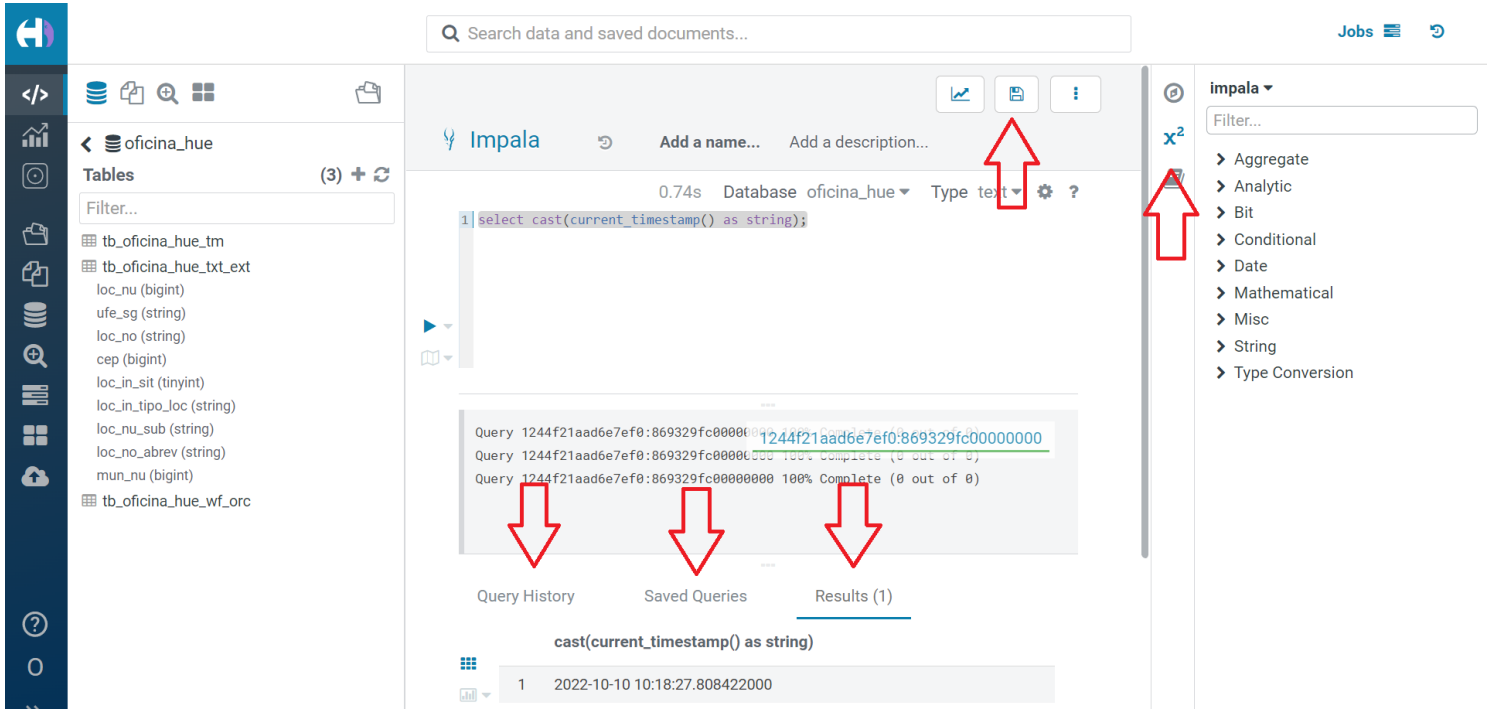
Database Browser

Podemos visualizar também várias informações importantes sobre cada tabela clicando com o botão direito do mouse sobre a tabela desejada e selecionando a opção *Table Browser.*



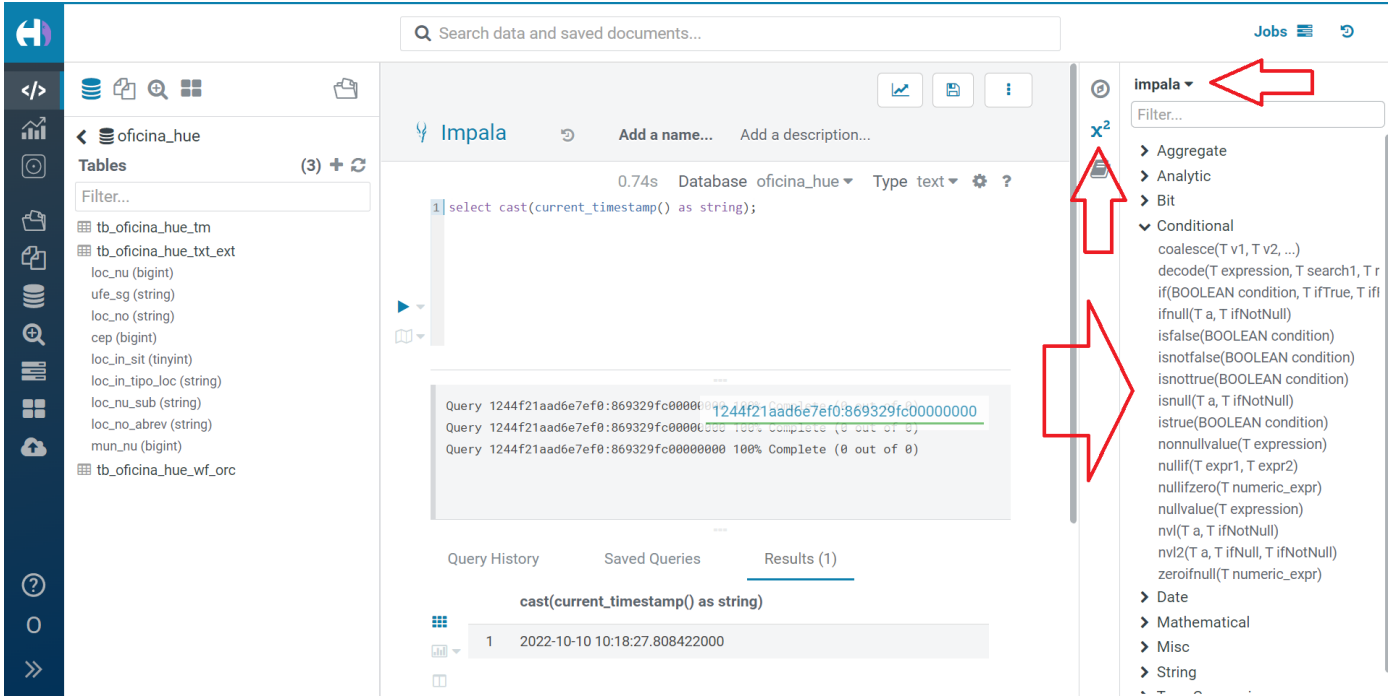
Query Editor

O Query Editor além de permitir a criação de consultas às fontes de dados conectadas através do HUE, permite também que salvemos estas consultas no HDFS e mantém um histórico de todas as consultas executadas pelo usuário.



Query Editor

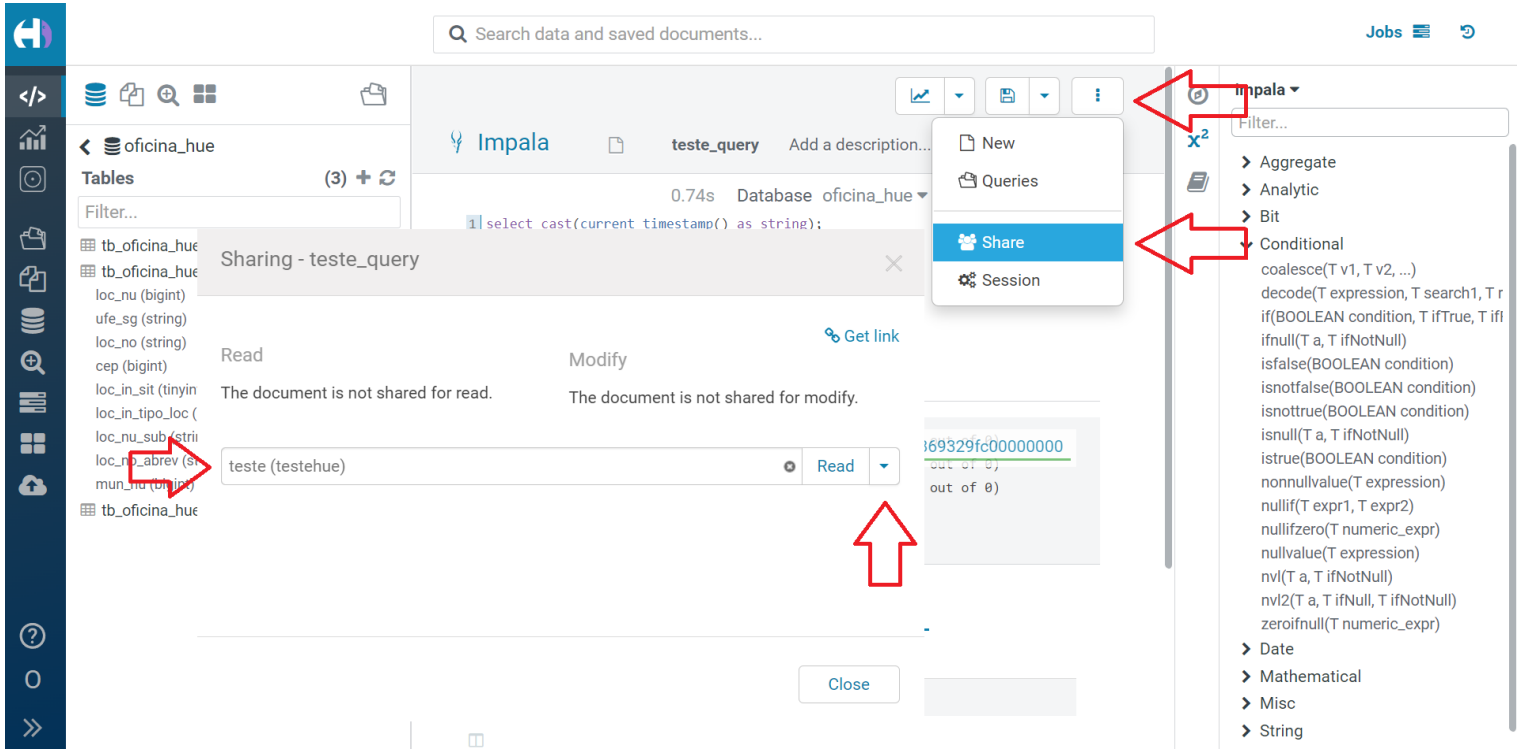
Possui uma aba lateral à direita que nos auxilia na utilização de funções de manipulação de dados, mostrando os nomes das mesmas e a sintaxe a ser utilizada.



Query Editor

Existe ainda a possibilidade de o usuário compartilhar suas consultas salvas com outros usuários do cluster através do menu “*Partilhar*”.

Basta digitar o nome do usuário na caixa de texto e selecionar qual tipo de permissão o usuário terá em sua consulta.



Hive x Impala

Hive x Impala

IMPALA

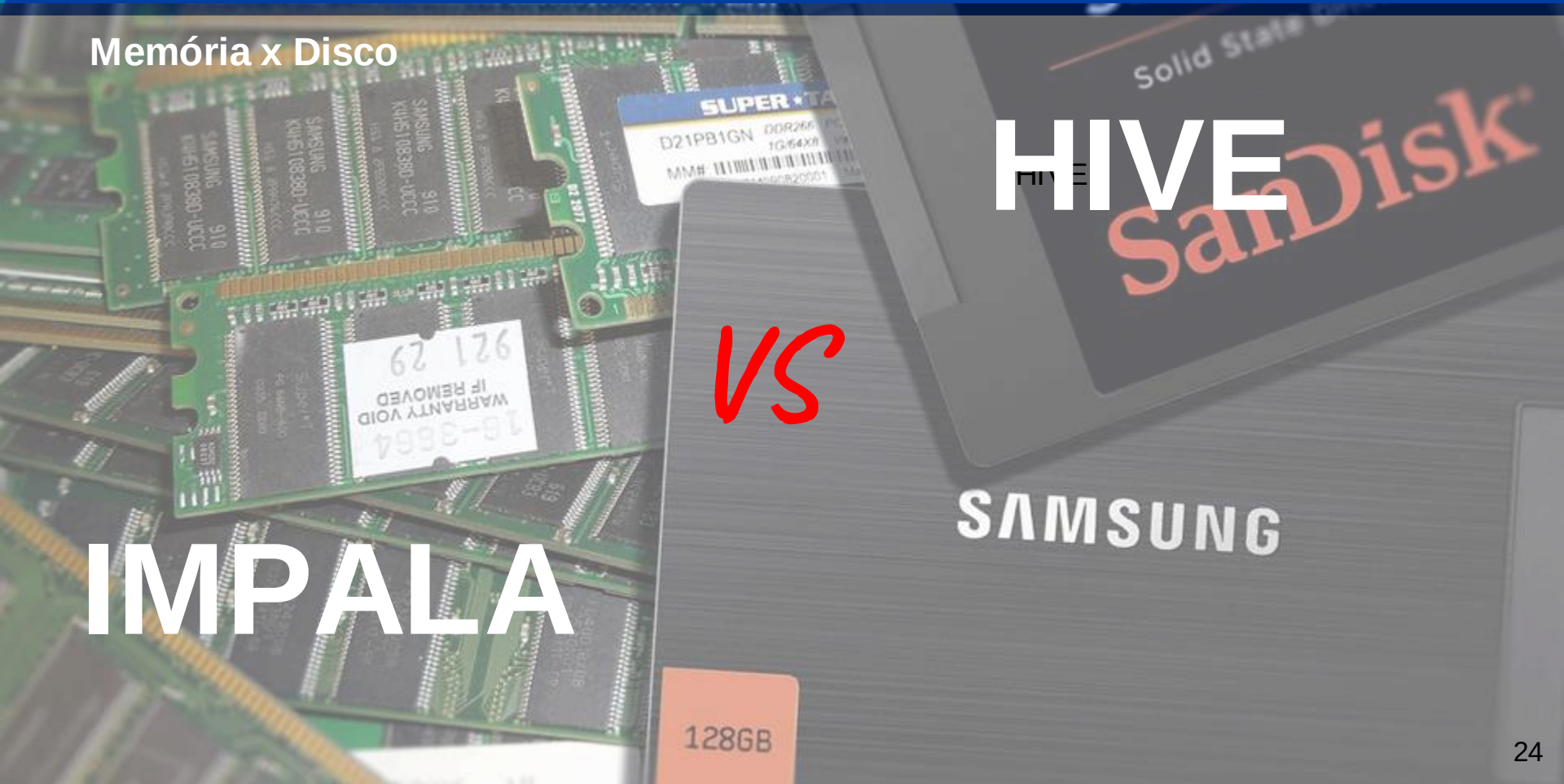
- Escrito em C++
- Data Marts
- Boa escolha para análise interativa e ad-hoc, pela rápida resposta
- Boa escolha para ferramentas de Business Intelligence que permitem aos usuários alterar consultas
- Usa Parquet como o formato de arquivo aconselhado
- Processamento em memória podendo falhar por falta de recurso.
- Utiliza Massively Parallel Processing (MPP)

HIVE

- Escrito em Java
- Data Warehouse
- Boa escolha para consultas de longa duração que requerem transformações pesadas ou múltiplas junções
- Boa escolha para painéis que são predefinidos e não personalizáveis pelo visualizador
- Usa ORC como o formato de arquivo aconselhado
- Funciona melhor com JSON do que Impala
- Se ajusta ao recurso disponível, sendo mais tolerante a falhas.
- Utiliza modelo Map-reduce / Apache Tez

Hive x Impala - Consumo Recursos

Memória x Disco



PARQUET

O Impala permite que você crie, gerencie e consulte tabelas Parquet. Parquet é um formato de arquivo binário, orientado a colunas, altamente eficiente para os tipos de consultas em grande escala. O formato Parquet é extremamente aconselhado para consultas que verificam colunas específicas em uma tabela, por exemplo, para consultar tabelas "largas" com muitas colunas ou para realizar operações de agregação como SUM () e AVG () que precisam processar a maioria ou todos os valores de uma coluna. Cada arquivo de dados Parquet escrito pelo Impala contém os valores de um conjunto de linhas (referido como "grupo de linhas").

Em um arquivo de dados, os valores de cada coluna são organizados de forma que sejam todos adjacentes, permitindo uma boa compactação para os valores dessa coluna. As consultas em uma tabela Parquet podem recuperar e analisar esses valores de qualquer coluna rapidamente e com o mínimo de I/O.

Sales

Product	Customer	Date	Sale
Beer	Thomas	2011-11-25	2 GBP
Beer	Thomas	2011-11-25	2 GBP
Vodka	Thomas	2011-11-25	10 GBP
Whiskey	Christian	2011-11-25	5 GBP
Whiskey	Christian	2011-11-25	5 GBP
Vodka	Alexei	2011-11-25	10 GBP
Vodka	Alexei	2011-11-25	10 GBP

Product

ID	Value
1	Beer
2	Beer
3	Vodka
4	Whiskey
5	Whiskey
6	Vodka
7	Vodka

Customer

ID	Customer
1	Thomas
2	Thomas
3	Thomas
4	Christian
5	Christian
6	Alexei
7	Alexei

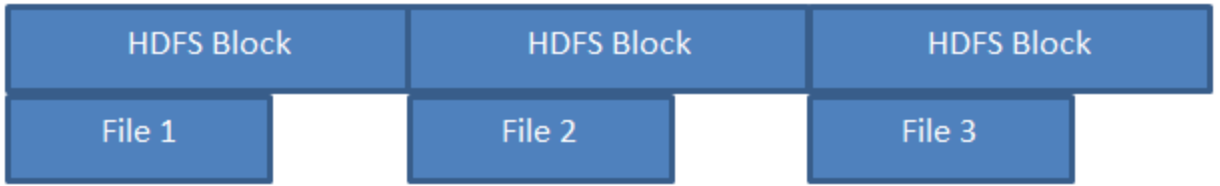
Impala / Hive - Boas Práticas

Formato mais apropriado de arquivo

Normalmente, para grandes volumes de dados (vários gigabytes por tabela ou partição), o tipo de arquivo Parquet tem um desempenho melhor devido a sua combinação de layout de armazenamento em forma de coluna, grandes requisições de I/O, compressão e codificação.

Evite arquivos pequenos

- Arquivos menores que o blocos do HDFS ocuparão o bloco inteiro.
- Recuperar muitos arquivos tem um I/O maior que poucos, já que é lido bloco a bloco.
- Iniciar tarefas (tasks) para cada arquivo, em caso de pequenos arquivos, gerará um número muito maior de tasks, e com isso um custo muito maior de processamento.



Evite arquivos pequenos

- Sempre use INSERT … SELECT para copiar grandes volumes de dados de uma tabela para outra dentro do Impala. Evite INSERT … VALUES para qualquer volume substancial de dados ou tabelas de performance crítica, pois cada cláusula cria um pequeno arquivo de dados separado.
- Caso seu processo de ingestão utilize o conceito de delta, e cada delta ingerido seja com um baixo volume de dados, talvez seja necessário de tempos em tempos gerar novamente sua tabela definitiva de dados, pois a ingestão provavelmente tem gerado pequenos arquivos parquet em cada execução e poderá fazer com que sua performance nas consultas seja prejudicada. Para isso pode utilizar o conceito de INSERT OVERWRITE TABLE.

Impala / Hive - Boas Práticas

Granularidade das partições

- Quando você envia consultas que solicitam um valor específico ou um intervalo de valores para a coluna chave da partição, o Impala pode evitar a leitura de dados irrelevantes, resultando em uma grande economia de I/O em disco.
- Escolha uma estratégia de particionamento que insira pelo menos 256 MB de dados em cada partição para tirar vantagem do bulk de I/O do HDFS e das consultas distribuídas do Impala.
- Se você precisar reduzir o número de partições e aumentar o número de dados em cada partição, primeiro verifique as colunas chaves da partição que raramente são referenciadas ou que não são usadas por consultas críticas. Por exemplo, os dados de log do seu web site são particionados por ano, mês, dia e hora, mas a maioria das consultas são por dia, talvez então você precise apenas particionar por ano, mês e dia.

Números inteiros menores para as colunas chaves das partições e para os joins

- Embora seja tentador usar strings para particionar coluna chave, já que esses valores se tornam nomes de diretórios HDFS, você pode diminuir a utilização de memória usando valores numéricos como campo chave de partição comum tais como ANO, MÊS e DIA.Use o menor número inteiro que possua o intervalo apropriado de dados, geralmente TINYINT para MÊS e DIA, e SMALLINT para ANO.
- Use a função EXTRACT( ) para remover campos de data e hora individualmente de um valor TIMESTAMP, e CAST( ) para retornar valor inteiro. Ao realizar consultas, de preferência para a utilização de colunas numéricas para a realização dos JOINs entre as tabelas, permitindo um cruzamento das informações sem necessidade de grandes varreduras ou utilização de memória.

Minimize resultados se possível

Utilize as seguintes técnicas:

- Agregação: se você quer saber quantas linhas satisfazem uma determinada condição, o valor total de valores de alguma coluna, o menor ou maior valor, chame funções agregadas como COUNT( ), SUM( ), MIN( ) e MAX( ) na consulta para não precisar enviar o resultado para uma aplicação e fazer os cálculos na aplicação.
- Filtro: use os filtros aplicáveis na cláusula WHERE da consulta para eliminar linhas que não são relevantes.
- Cláusula LIMIT: se você precisa visualizar apenas alguns valores de um conjunto de dados, ou os valores superiores ou inferiores de uma consulta usando ORDER BY, inclua a cláusula LIMIT para reduzir o tamanho do retorno da consulta.

Gerenciamento de Memória em Consultas

Em certas situações (queries complexas e/ou que tendem a demandar grandes quantidades de dados em memória) o otimizador do Impala pode tentar reservar um volume de memória maior que o necessário para o processamento da consulta gerando dois tipos de problemas:

- Query não ser executada por não haver recurso de memória suficiente no pool. Neste caso a query sequer será admitida para execução.

- Query é admitida para execução, porém reservará mais recurso de memória que o necessário, o que pode ocasionar a não execução de outras consultas no mesmo pool.

Gerenciamento de Memória em Consultas

As soluções para resolvermos estes tipos de problema seriam:

- Gerenciar manualmente o recurso de memória a ser utilizado pelo Impala.
  Ex: SET mem_limit="1G";

- Adaptar sua consulta para execução no Hive, uma vez que o mesmo é mais robusto por usar map_reduce.

- Particionar sua consulta para utilização de uma quantidade de memória menor.
  Utilizar tabelas intermediárias, realizando assim uma consulta maior em várias etapas, reduzindo a utilização de memória em relação a uma única instrução sql mais complexa.

Impala / Hive - Boas Práticas

Melhorando Performance e Consumo de Recursos Utilizando Estatísticas

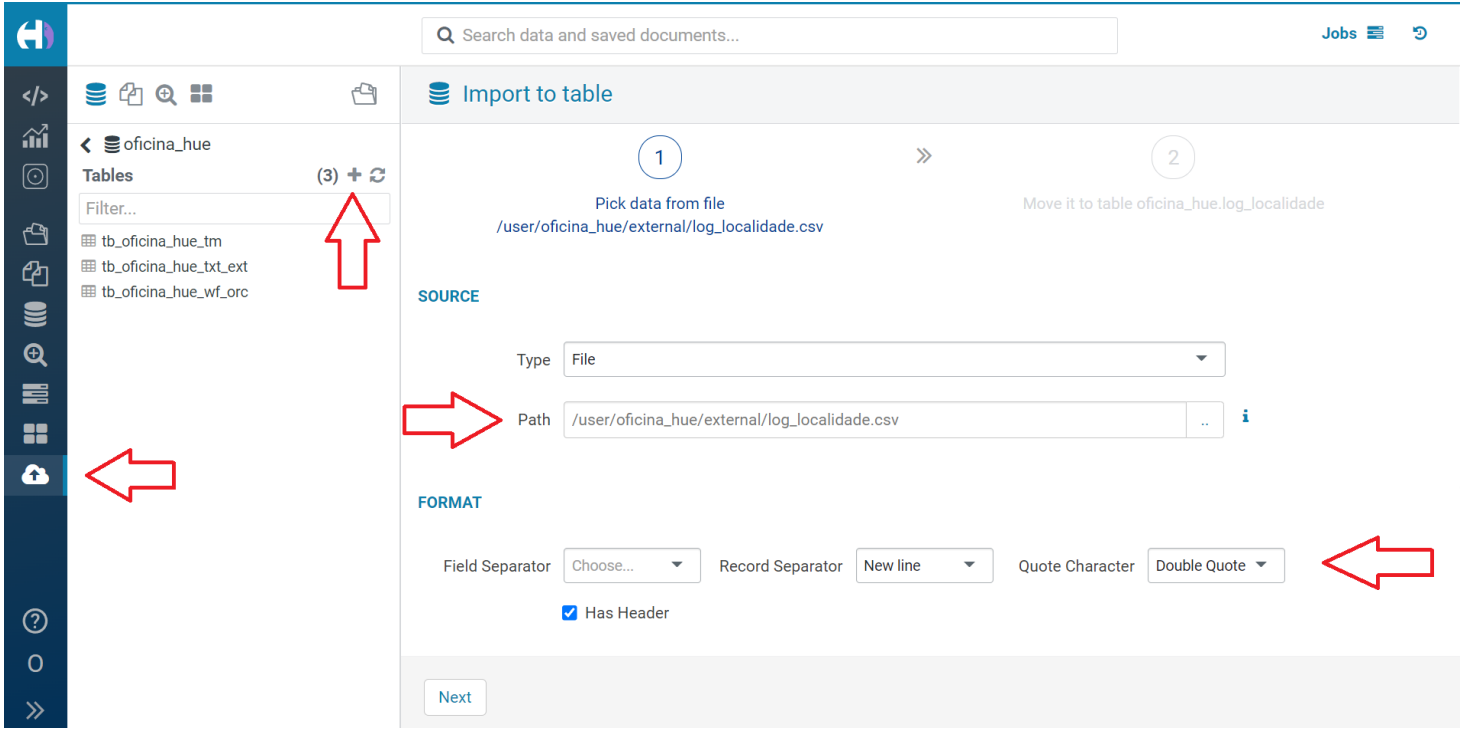
Tanto o Impala quanto o Hive se utilizam das estatísticas geradas em suas tabelas para definição de plano de acesso e execução das consultas no Hadoop. Uma prática essencial para que consiga extrair a melhor performance de suas consultas é sempre manter as estatísticas das tabelas atualizadas. Para tanto devemos nos atentar em executar a atualização das estatísticas sempre que realizarmos alguma modificação nos dados das tabelas a serem consumidas.

Comandos:

Hive: Analyze table nome_da_tabela compute statistics;
Impala: compute stats nome_da_tabela;

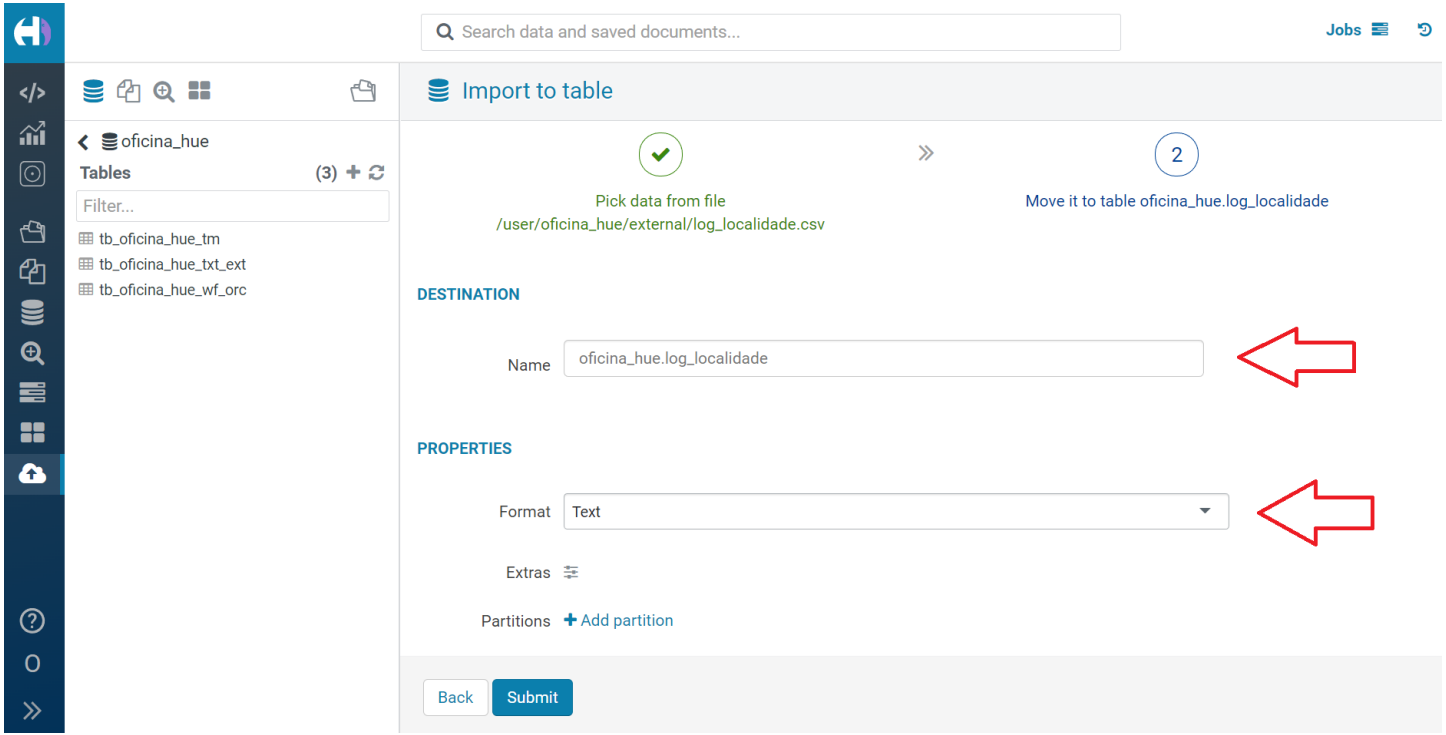
Hands on

Criando estruturas e ingerindo dados



Hands on

Criando estruturas e ingerindo dados



Hands on

Criando external table

```
CREATE EXTERNAL TABLE oficina_hue.tb_log_cpc_ext (cpc_nu bigint, ufe_sg string, loc_nu bigint, cpc_no string, cpc_endereco string,cep bigint)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ';'
LOCATION '/user/oficina_hue/external'
TBLPROPERTIES("skip.header.line.count"="1")
```

External tables funcionam como uma estrutura (esqueleto) de uma tabela que aponta diretamente para o arquivo/local, dentro do HDFS, indicado no momento de sua criação.

Quando dropamos uma external table, os arquivos referenciados por ela continuam no diretório em que se encontram, sendo eliminada apenas a estrutura de acesso.

Hands on

Detalhe importante sobre internal (managed tables) e external tables:

Por definição, hoje no cluster de Brasília todas as tabelas criadas, independente de colocarmos de forma explícita que serão external, serão criadas como tal. Vale salientar que o conceito de external continuará valendo para as que explicitamente forem declaradas desta forma, logo se dropar uma tabela criada com o comando CREATE EXTERNAL TABLE, os dados continuarão no diretório cujo apontamento foi definido na criação da tabela, mas as tabelas external criadas sem explicitamente declaradas forem, quando dropadas, apagarão tanto a estrutura da mesma quanto os dados.

A exceção se dá para tabelas ACID (Atomicidade, Consistência, Isolamento e Durabilidade), estas serão criadas como tabelas internas gerenciadas pelo Hive (TBLPROPERTIES('transactional'='true')).

Hands on

Ingerindo/criando tabela linha de comando

```
CREATE TABLE oficina_hue.tb_log_cpc_txt( cpc_nu bigint, ufe_sg string, loc_nu bigint, cpc_no string, cpc_endereco string, cep bigint)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ';'
STORED AS TEXTFILE
TBLPROPERTIES("skip.header.line.count"="1");

LOAD DATA INPATH '/user/oficina_hue/dataset' into table oficina_hue.tb_log_cpc_txt;
```

Quando utilizamos o comando *LOAD DATA* para inserírmos dados de um dado arquivo/diretório do HDFS para uma tabela específica, os arquivos são movidos para o diretório da tabela em questão e quando dropada a tabela, os arquivos serão literalmente apagados do HDFS.

Hands on

Criando tabela Parquet utilizando uma tabela texto.

*CREATE TABLE oficina_hue.tb_log_cpc_pq STORED AS PARQUET as select * from oficina_hue.tb_log_cpc_txt;*

Podemos realizar a criação de tabelas tipo parquet, tanto pelo Impala (preferencialmente) quanto pelo Hive, através da consulta via HQL em uma tabela já existente.

Views

Hands on

Criando Views

CREATE VIEW vw_log_cpc as
*SELECT * FROM log_cpc*
WHERE ufe_sg = 'AL';

Views podem ser criadas para armazenamento de queries pré-definidas como de costume, mas podem representar também uma restrição para um grupo de usuários com menos privilégio.
Obs: para se conseguir criar uma view, o usuário necessita ter acesso à todos os dados cuja view faz referência.

Hands on

Views Hive x Impala (incompatibilidades)

Ao criar uma view, é importante observarmos em que editor (hive ou impala) a view está sendo criada, devido ao comportamento das definições desta view relativos tanto às estruturas de dados, quanto às funções nativas de cada editor. Isto vale tanto para views quanto para tabelas.

Por exemplo, quanto ao uso de tipos e/ou estruturas de dados que funcionam exclusivamente em um dos editores. O impala, não conseguirá ler uma view que possua colunas do tipo binary ou utilizem funções específicas do Hive.

Views

Hands on

Views / Tabelas Hive x Impala (incompatibilidades)

No Hive:

```
CREATE TABLE oficina_hue.teste_date(id_teste int, dt_teste date);

INSERT INTO oficina_hue.teste_date VALUES (1,'2021-01-01');

CREATE VIEW oficina_hue.vw_teste_date as
SELECT id_teste, date_format(dt_teste,'y') as year FROM oficina_hue.teste_date;
```

No Impala:

```
SELECT * FROM oficina_hue.vw_teste_date;
```

Hands on

Materialized views

Materialized Views não eram suportadas na versão do Hive que utilizávamos (Hive 2.1.1 - CDH 6.3.4), mas com a migração do cluster, já temos a versão Hive 3.1.3 - CDP 7.1.7 em produção, estando disponível à mesma. Para que a mesma funcione, deve-se criar a tabela cuja Materialized View irá realizar a consulta, utilizando o parâmetro 'transactional'='true' e um formato que permita que a tabela seja transacionada (Ex: ORC).

```
CREATE TABLE oficina_hue.teste_date(id_teste int, dt_teste date) stored as orc tblproperties('transactional'='true');
```

```
CREATE materialized VIEW oficina_hue.vw_teste_date as
select id_teste, dt_teste from oficina_hue.teste_date;
```

Hands on

Materialized views

Para atualização dos dados das materialized views devemos usar o comando do exemplo abaixo:

ALTER MATERIALIZED VIEW oficina_hue.vw_teste_date REBUILD;

As tabelas ACID no hive suportam tanto insert, quanto update e delete, mas devemos salientar um detalhe. As materialized views criadas com base em tabelas que suportam as três operaçoes de modificação de dados, não poderão ser lidas pelo Impala, o mesmo só aceita leitura de materialized views criadas em tabelas cujo ACID suportado seja apenas insert (tblproperties 'transactional_properties'='insert_only')

Tipos de Dados

Considerando a versão Cloudera do condomínio atual, estes são os tipos de dados suportados, respectivamente, pelo hive e impala.

Hive	Impala
ARRAY*	ARRAY*
BIGINT	BIGINT
BOOLEAN	BOOLEAN
CHAR	CHAR
DATE	DATE
DECIMAL	DECIMAL
DOUBLE	DOUBLE
FLOAT	FLOAT
INT	INT
MAP*	MAP*
REAL	REAL
SMALLINT	SMALLINT
STRING	STRING
STRUCT*	STRUCT*
TIMESTAMP	TIMESTAMP
TINYINT	TINYINT
VARCHAR	VARCHAR
BINARY	-

Complex types

Também conhecidos como tipos aninhados (nested types), esses tipos de dados permitem a representação de vários valores de dados em uma única posição de linha/coluna. Eles diferem dos tipos de coluna conhecidas, como BIGINT e STRING, conhecidos como tipos escalares ou tipos primitivos, que representam um único valor de dados em uma determinada posição de linha/coluna.

Tipos de Dados

Tipos dados Complex types

- **Array:** Um tipo complex type que pode representar um número arbitrário de elementos ordenados. Os elementos podem ser escalares ou outro tipo complexo (ARRAY, STRUCT ou MAP)
- **Map:** representa um conjunto arbitrário de pares de valores-chave. A parte chave deve ser, necessariamente, um tipo escalar, enquanto a parte do valor pode ser um escalar ou outro tipo complexo. Um exemplo simples de pensarmos em um uso para o tipo map, seria representarmos um conjunto de métricas para uma tabela de municípios, onde poderíamos ter informações como o código e nome do município como campos escalares e um campo do tipo map, contendo uma chave com o nome da métrica e um valor correspondente para o município (exemplos: 'Area:432244', 'Populacao':213685, 'PibPerCapta:R$ 19.184,77')
- **Struct:** este tipo é útil para combinar duas tabelas relacionadas, de modo a minimizar a repetição. A maneira mais comum de representar esses dados é como um ARRAY de elementos STRUCT. O tipo STRUCT costuma ser mais útil como um item de um ARRAY ou a parte do valor do par de valores-chave em um MAP.

Manipulando complex types - Impala x Hive

O Impala usa a notação de ponto para se referir a nomes de elementos ou elementos dentro de tipos complexos e notação de join para fazer referência cruzada de colunas escalares com os elementos de tipos complexos na mesma linha, em vez da cláusula LATERAL VIEW e da função EXPLODE () do HiveQL.

Uma consulta como SELECT * FROM banco.tabela feita pelo hive sobre uma tabela que contenha colunas do tipo complex type é aceita normalmente. Já no impala, apesar de não termos erro de execução, o retorno será apenas dos dados referentes às colunas escalares.

A versão atual do Impala não suporta a criação de novos dados com colunas do tipo complex type. Assim, o caminho que demonstraremos aqui é criarmos uma tabela do tipo parquet e alimentá-la com o hive, para então realizarmos consultas via impala.

Hands on - Carregando dados Complex Type

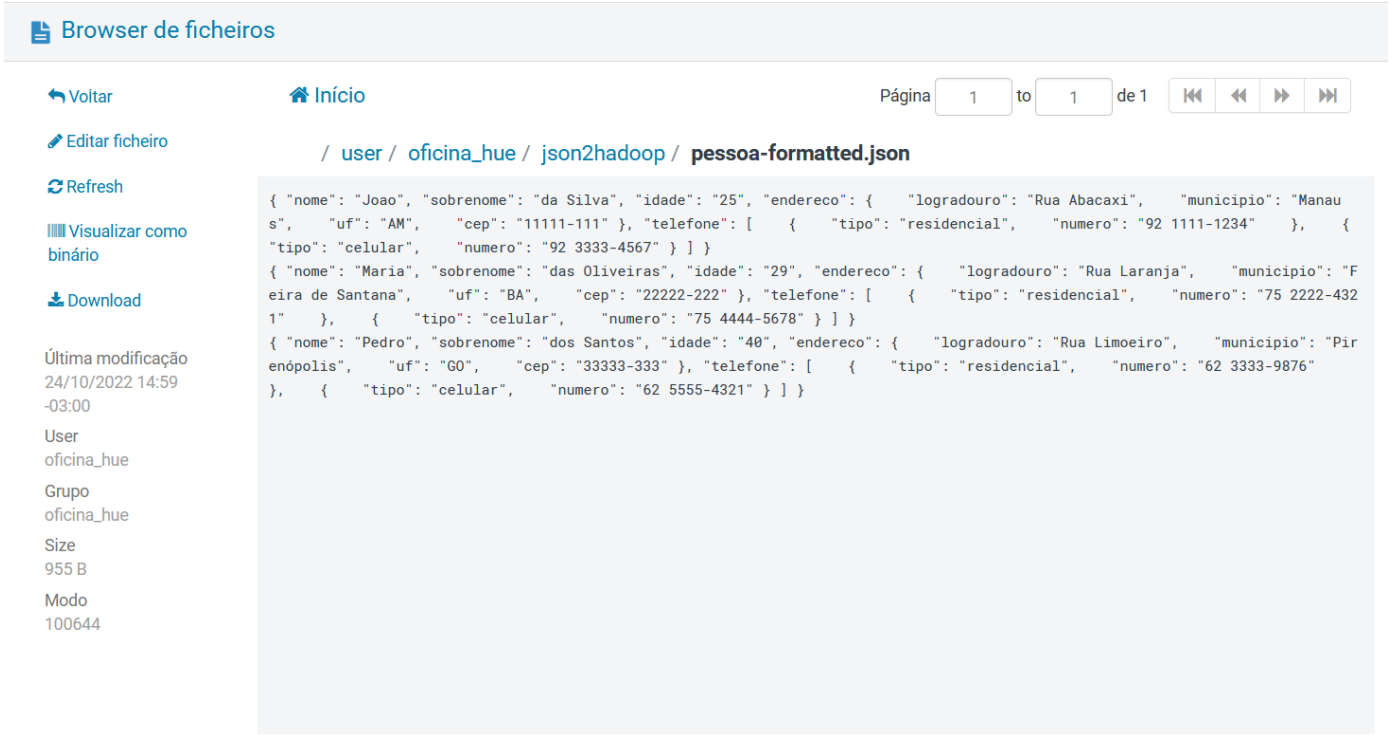
```
[
 {
  "nome": "Joao",
  "sobrenome": "da Silva",
  "idade": "25",
  "endereco": {
   "logradouro": "Rua Abacaxi",
   "municipio": "Manaus",
   "uf": "AM",
   "cep": "11111-111"
  },
  "telefone": [
   {
    "tipo": "residencial",
    "numero": "92 1111-1234"
   },
   {
    "tipo": "celular",
    "numero": "92 3333-4567"
   }
  ]
 },
 {
  "nome": "Maria",
  "sobrenome": "das Oliveiras",
  "idade": "29",
  "endereco": {
   "logradouro": "Rua Laranja",
   "municipio": "Feira de Santana",
   "uf": "BA",
   "cep": "22222-222"
  },
  "telefone": [
   {
    "tipo": "residencial",
    "numero": "75 2222-4321"
   },
   {
    "tipo": "celular",
    "numero": "75 4444-5678"
   }
  ]
 },
 {
  "nome": "Pedro",
  "sobrenome": "dos Santos",
  "idade": "40",
  "endereco": {
   "logradouro": "Rua Limoeiro",
   "municipio": "Pirenópolis",
   "uf": "GO",
   "cep": "33333-333"
  },
  "telefone": [
   {
    "tipo": "residencial",
    "numero": "62 3333-9876"
   },
   {
    "tipo": "celular",
    "numero": "62 5555-4321"
   }
  ]
 }
]
```

Hands on - Carregando dados Complex Type

{ "nome": "Joao", "sobrenome": "da Silva", "idade": "25", "endereco": { "logradouro": "Rua Abacaxi", "municipio": "Manaus", "uf": "AM", "cep": "11111-111" }, "telefone": [{ "tipo": "residencial", "numero": "92 1111-1234" }, { "tipo": "celular", "numero": "92 3333-4567" }] }
{ "nome": "Maria", "sobrenome": "das Oliveiras", "idade": "29", "endereco": { "logradouro": "Rua Laranja", "municipio": "Feira de Santana", "uf": "BA", "cep": "22222-222" }, "telefone": [{ "tipo": "residencial", "numero": "75 2222-4321" }, { "tipo": "celular", "numero": "75 4444-5678" }] }
{ "nome": "Pedro", "sobrenome": "dos Santos", "idade": "40", "endereco": { "logradouro": "Rua Limoeiro", "municipio": "Pirenópolis", "uf": "GO", "cep": "33333-333" }, "telefone": [{ "tipo": "residencial", "numero": "62 3333-9876" }, { "tipo": "celular", "numero": "62 5555-4321" }] }

Tipos de Dados

Hands on - Carregando dados Complex Type



Hands on - Carregando dados Complex Type

```
DROP TABLE IF EXISTS oficina_hue.pessoa_stage;
CREATE EXTERNAL TABLE oficina_hue.pessoa_stage (
  nome string
, sobrenome string
, idade string
, endereco struct<logradouro:string
, municipio:string
, uf:string
, cep:string>
, telefone array<struct<tipo:string
, numero:string>>
)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 'hdfs:///ldg-bsa-01-hdfs/user/oficina_hue/json2hadoop/';

DROP TABLE IF EXISTS oficina_hue.pessoa;
CREATE TABLE pessoa STORED AS PARQUET AS SELECT * FROM oficina_hue.pessoa_stage;


DROP TABLE IF EXISTS oficina_hue.pessoa_stage PURGE;
```

Obs.: Lembre-se que o impala pode levar um tempo para reconhecer os metadados criados pelo Hive. Assim, para vermos as mudanças imediatamente, devemos utilizar o comando INVALIDATE METADATA.
E para maior performance nas consultas, é sempre recomendado que geremos as estatísticas com o comando COMPUTE STATS.

Tipos de Dados

Hands on - Carregando dados Complex Type

Experimente realizar a consulta abaixo no Hive e no Impala. Em seguida, compare os resultados.

```
select * from oficina_hue.pessoa;
```

Hive:

pessoa.nome	pessoa.sobrenome	pessoa.idade	pessoa.endereco	pessoa.telefone
Joao	da Silva	25	{"logradouro":"Rua Abacaxi","municipio":"Manaus","uf":"AM","cep":"11111-111"}	[{"tipo":"residencial","numero":"92 1111-1234"},{"tipo":"celular","numero":"92 3333-4567"}]
Maria	das Oliveiras	29	{"logradouro":"Rua Laranja","municipio":"Feira de Santana","uf":"BA","cep":"22222-222"}	[{"tipo":"residencial","numero":"75 2222-4321"},{"tipo":"celular","numero":"75 4444-5678"}]
Pedro	dos Santos	40	{"logradouro":"Rua Limoeiro","municipio":"Pirenópolis","uf":"GO","cep":"33333-333"}	[{"tipo":"residencial","numero":"62 3333-9876"},{"tipo":"celular","numero":"62 5555-4321"}]

Impala:

nome	sobrenome	idade
Joao	da Silva	25
Maria	das Oliveiras	29
Pedro	dos Santos	40

Hands on - Consultando dados Complex Type

Hive:

```
SELECT
    nome
   ,sobrenome
   ,idade
   ,endereco.logradouro
   ,endereco.municipio
   ,endereco.uf
   ,endereco.cep
   ,telefones.tipo as tipo_telefone
   ,telefones.numero as numero_telefone
FROM
   pessoa
   LATERAL VIEW explode(telefone) telefone as
telefones;
```

Impala:

```
INVALIDATE METADATA oficina_hue.pessoa;
COMPUTE STATS oficina_hue.pessoa;


SELECT
    p.nome
   ,p.sobrenome
   ,p.idade
   ,p.endereco.logradouro
   ,p.endereco.municipio
   ,p.endereco.uf
   ,p.endereco.cep
   ,t.tipo
   ,t.numero
FROM
    pessoa p
   ,p.telefone t ;
```

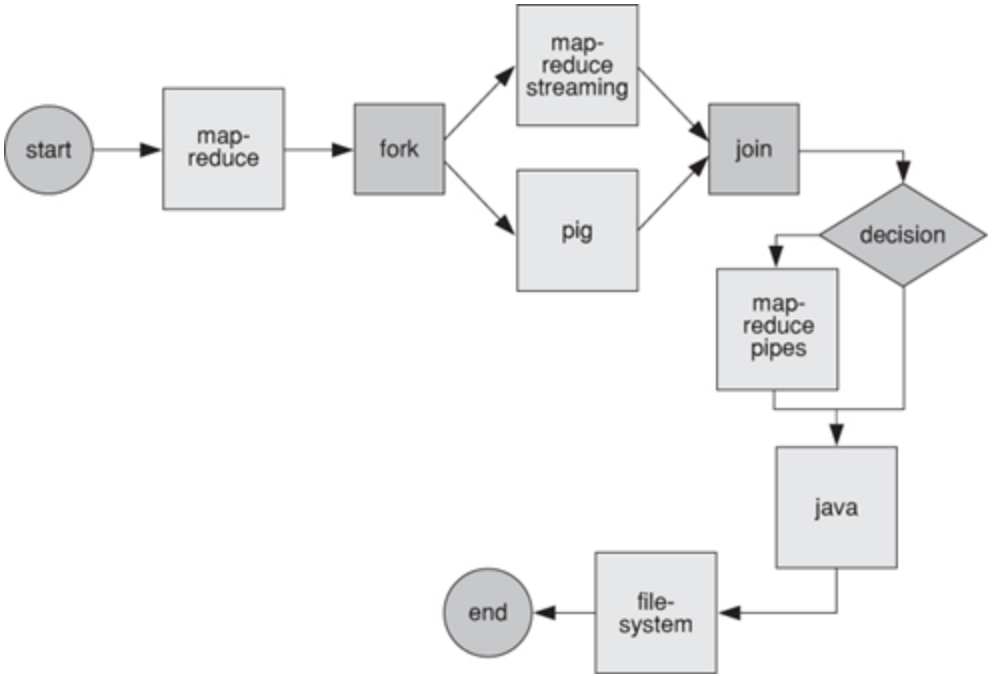
Exercícios 1 a 6

Realizar atividades localizadas no documento **Exercícios,** subtítulo **Ingestão de Dados,** enviado por email no início da Oficina.

Workflows Oozie

O Oozie é um mecanismo de workflow, baseado em servidor, especializado na execução de jobs com ações que executam tarefas em um cluster Hadoop.

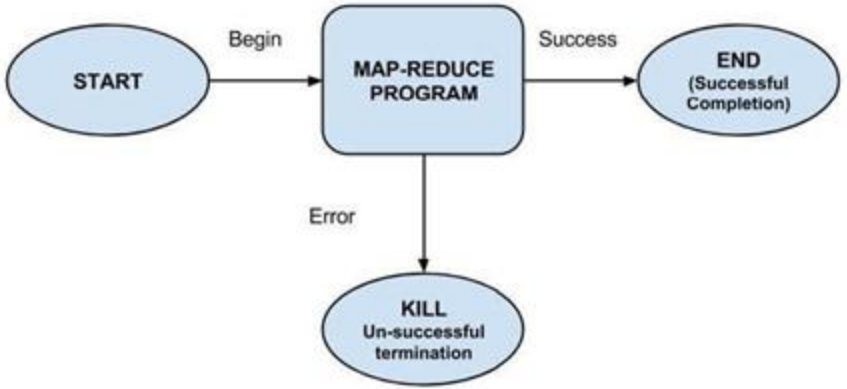
Para fins do Oozie, um Workflow é uma coleção de ações organizadas em um DAG (Direct Acyclic Graph) de controle de dependência.



Workflows Oozie

As definições de workflows do Oozie são escritas em hPDL (uma linguagem de definição de processo XML semelhante ao JBOSS JBPM jPDL).

As ações dos workflows do Oozie iniciam trabalhos em sistemas remotos do Hadoop. Após a conclusão da ação, os sistemas remotos chamam o Oozie de volta para notificar a conclusão da ação, neste ponto o Oozie prossegue para a próxima ação no workflow.



Workflows Oozie

Os workflows do Oozie contém nós de fluxo de controle e nós de ação.

Os nós de fluxo de controle definem o início e o fim de um workflow (nós de início, término e falha) e fornecem um mecanismo para controlar o caminho de execução do workflow (nós de decisão, bifurcação e junção).

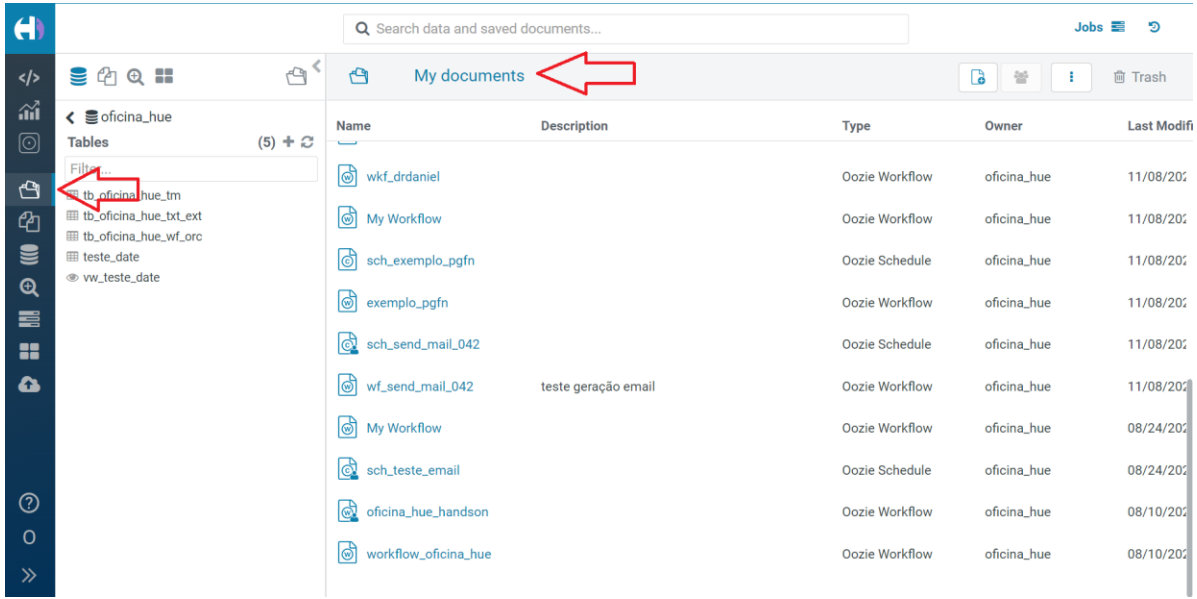
Os nós de ação são o mecanismo pelo qual um fluxo de trabalho dispara a execução de uma tarefa de computação / processamento. O Oozie fornece suporte para diferentes tipos de ações: mapreduce do Hadoop, sistema de arquivos do Hadoop, SSH, HTTP, e-mail e subfluxo do Oozie. O Oozie pode ser estendido para oferecer suporte a tipos adicionais de ações.

Hands on

Workflow na prática

Por definição, tudo que criamos e salvamos com a interface do Hue é definido como Documento (queries, workflows, schedules, scripts, etc)

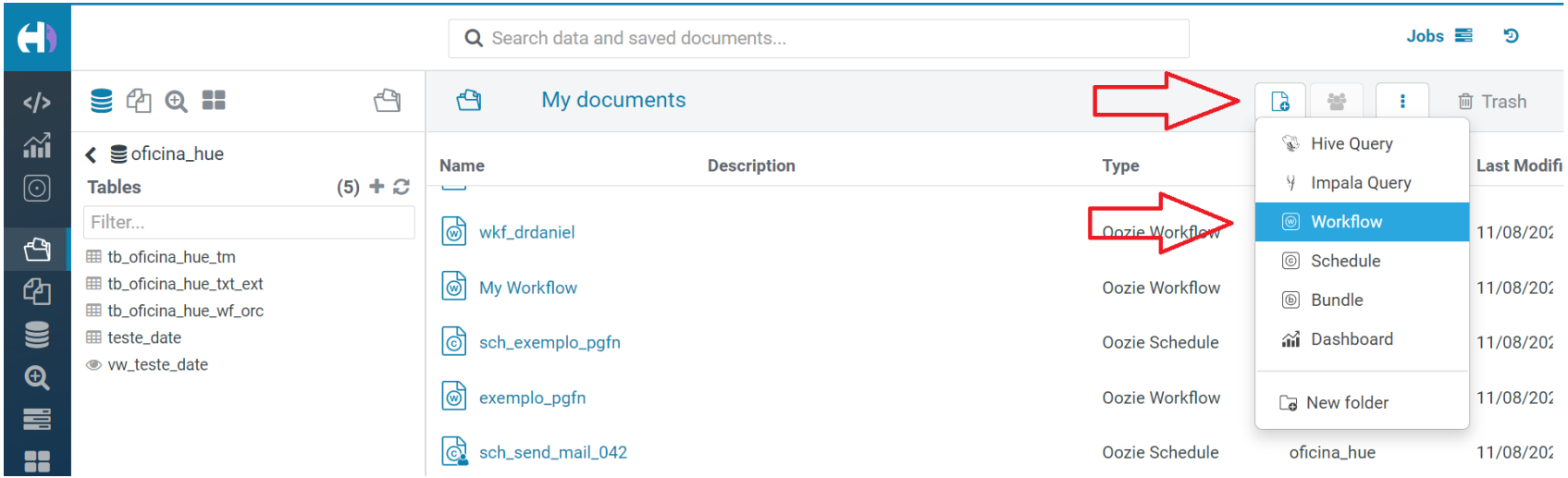
Para acessarmos a interface de construção de workflows do Oozie, bem como os fluxos já criados, devemos acionar o menu Documents, localizado no menu principal do Hue.



Hands on

Workflow na prática

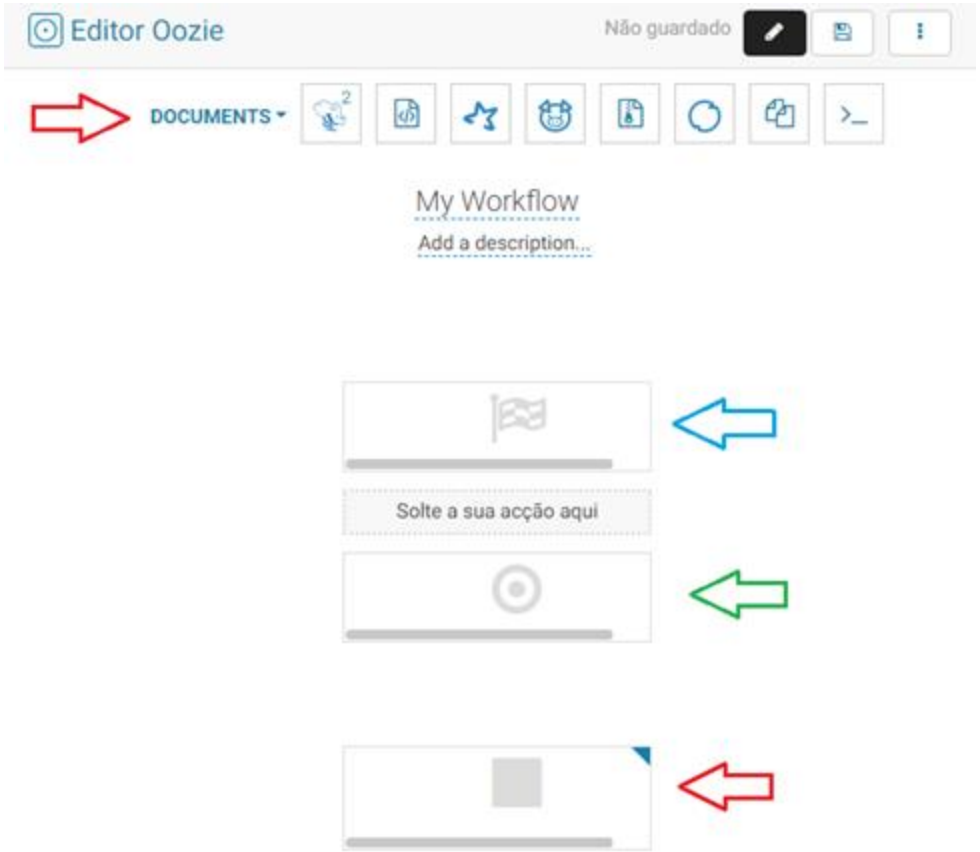
Para criação de um novo workflow, devemos acessar o botão *Novo Documento* localizado na parte superior direita da janela do Hue, e selecionar a opção *Workflow*.



Hands on

Workflow na prática

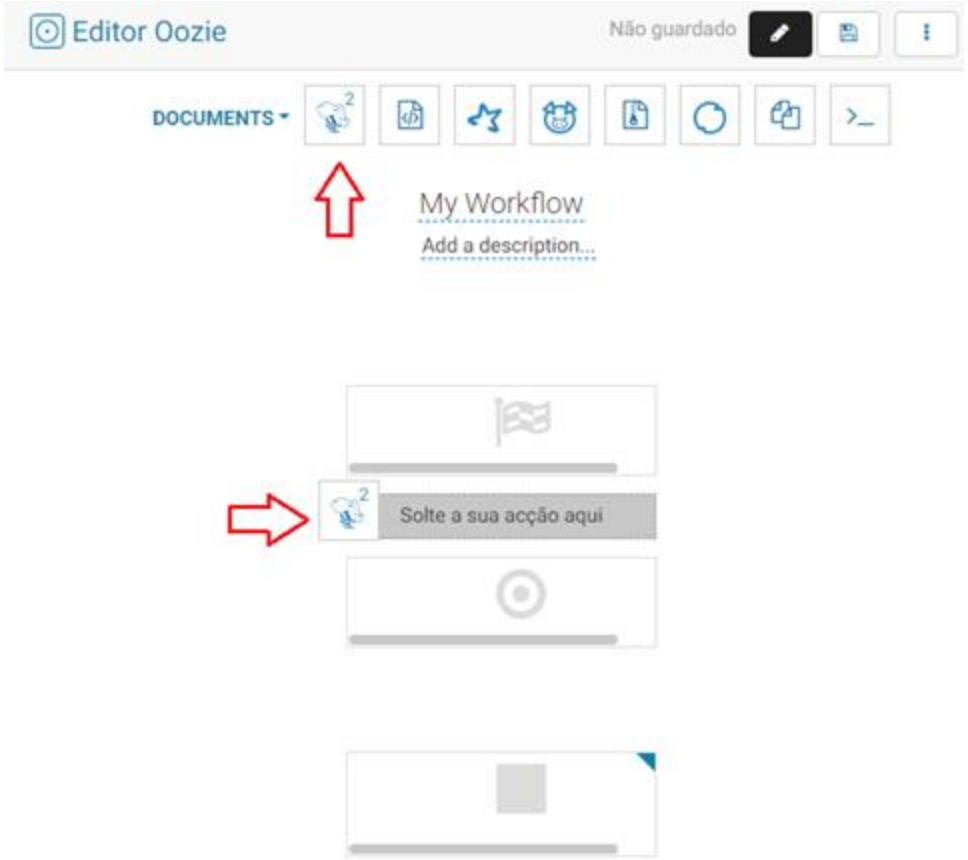
O Hue já trará uma estrutura básica de workflow com o ponto de início do mesmo, o ponto de finalização em caso de sucesso e o ponto de parada em caso de erro. Uma lista de documentos/ações que podemos implementar no fluxo também nos é apresentada.



Hands on

Workflow na prática

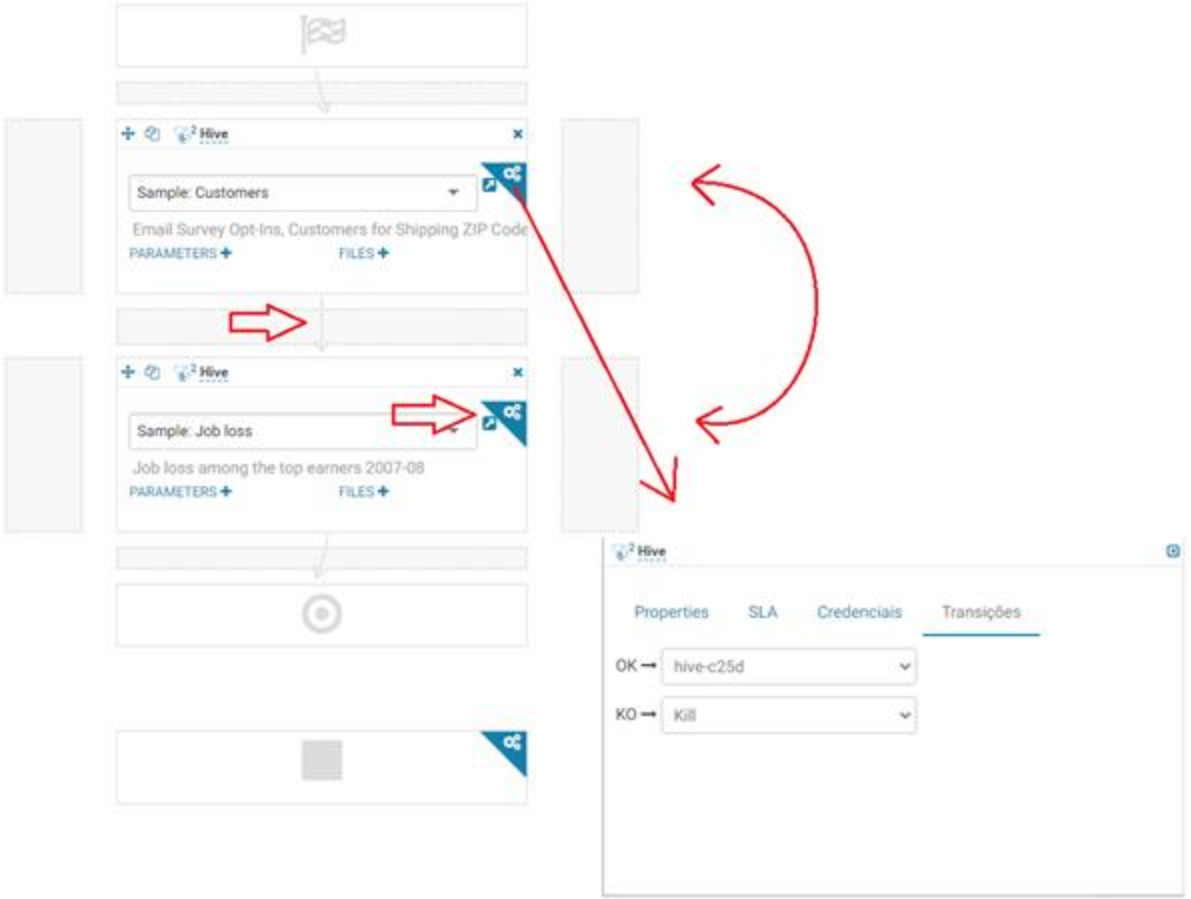
O framework trabalha com o esquema drag & drop. Você seleciona a ação que deseja e a arrasta para o local do workflow em que ela deve ser executada.



Hands on

Workflow na prática

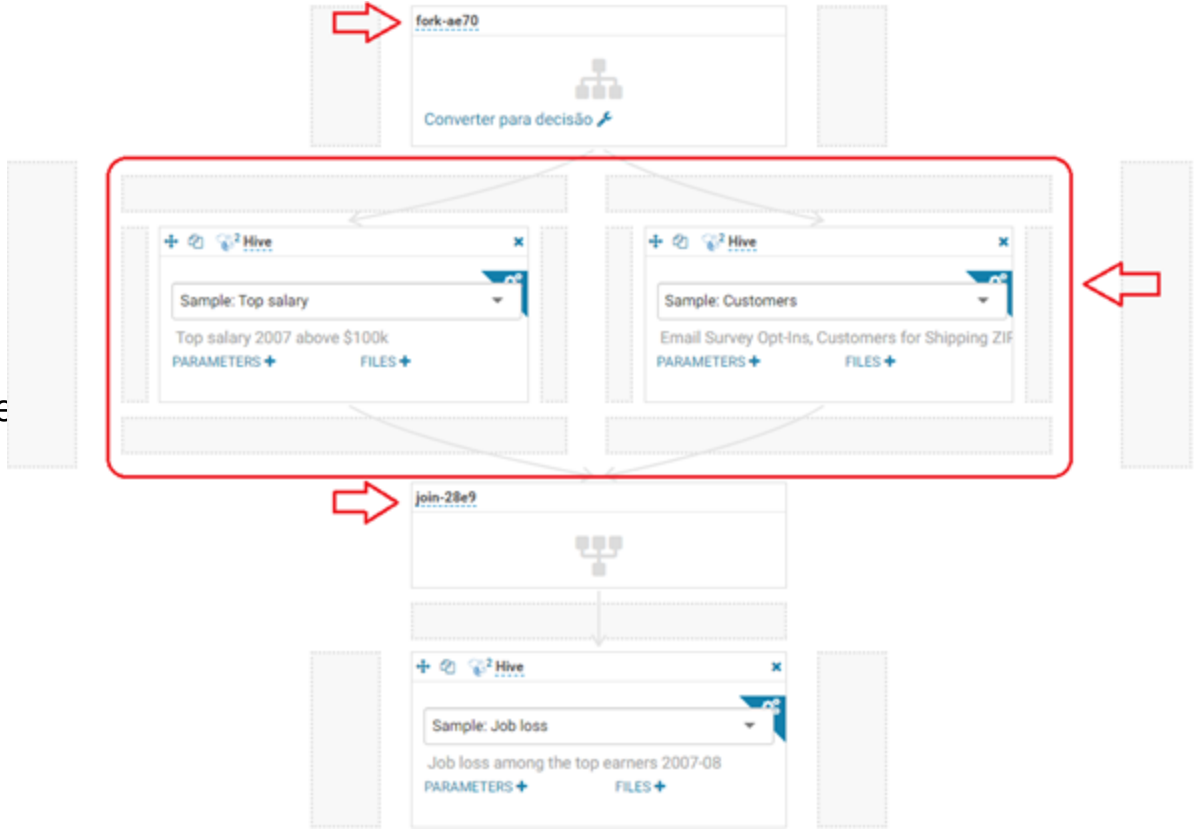
As dependências são criadas de acordo com o local em que se “solta” a ação que será executada, e pode ser vista clicando no ícone de engrenagens de cada etapa do fluxo, na aba *Transições.*



Hands on

Workflow na prática

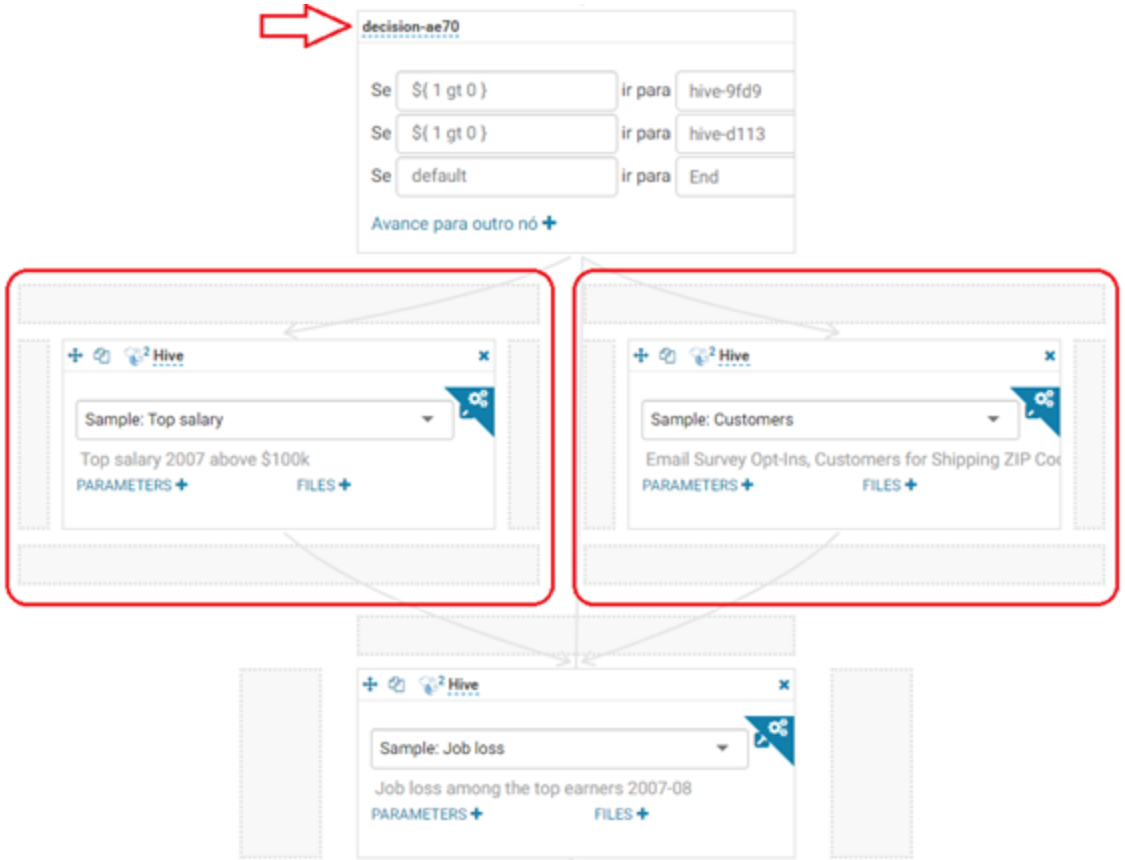
Quando houver necessidade de se executar duas ou mais ações em paralelo, ou que necessite de uma ação de decisão, deve-se arrastar a ação a se executada para caixa lateral à ação já incluída no workflow. Isso fará com que sejam geradas duas novas etapas no fluxo, no caso de paralelismo serão geradas etapas de *fork* e *join*, unicamente para controle do fluxo.



Hands on

Workflow na prática

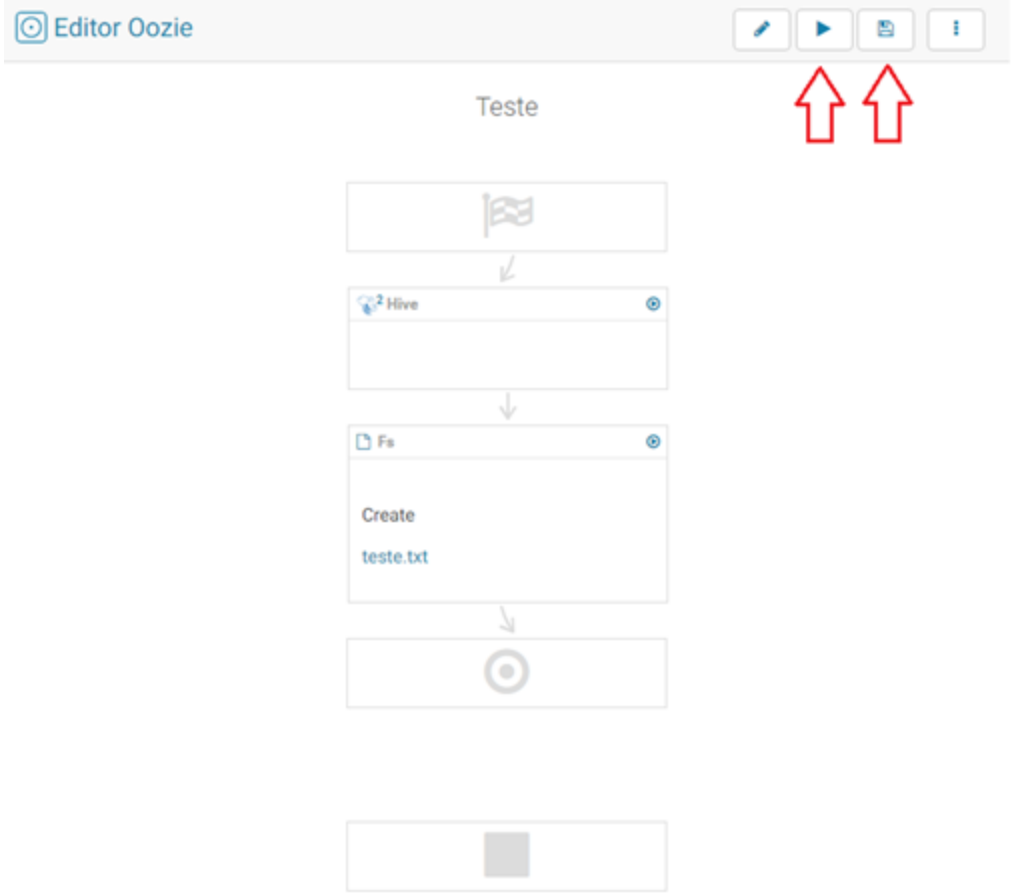
No caso de condicional, é gerada uma etapa *decision*, onde o usuário pode definir em quais situações o fluxo deve seguir determinado caminho.



Hands on

Workflow na prática

Para executar pontualmente o fluxo criado, deve-se primeiramente salvá-lo caso haja alterações recentes e clicar no botão play.



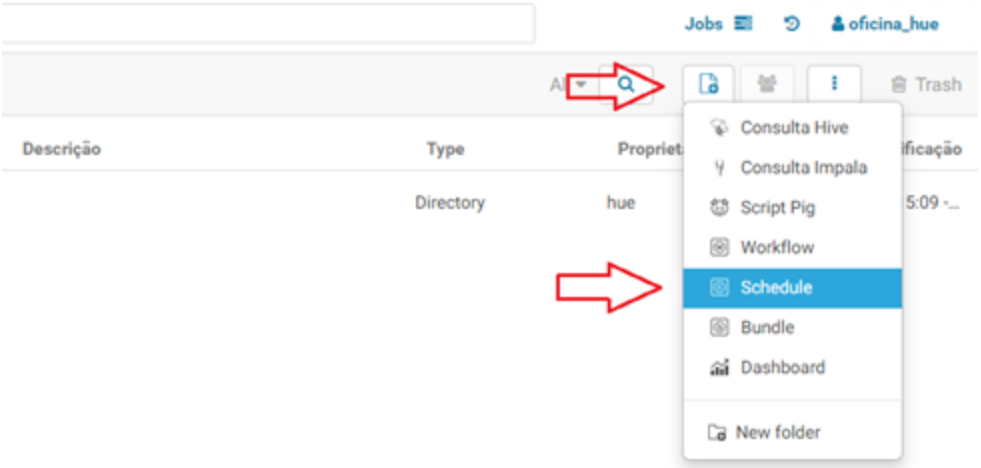
Hands on

Scheduler

Após a criação de seus Workflows oozie, você pode ainda realizar agendamento de suas execuções utilizando o programa Schedule (Oozie Coordinator).

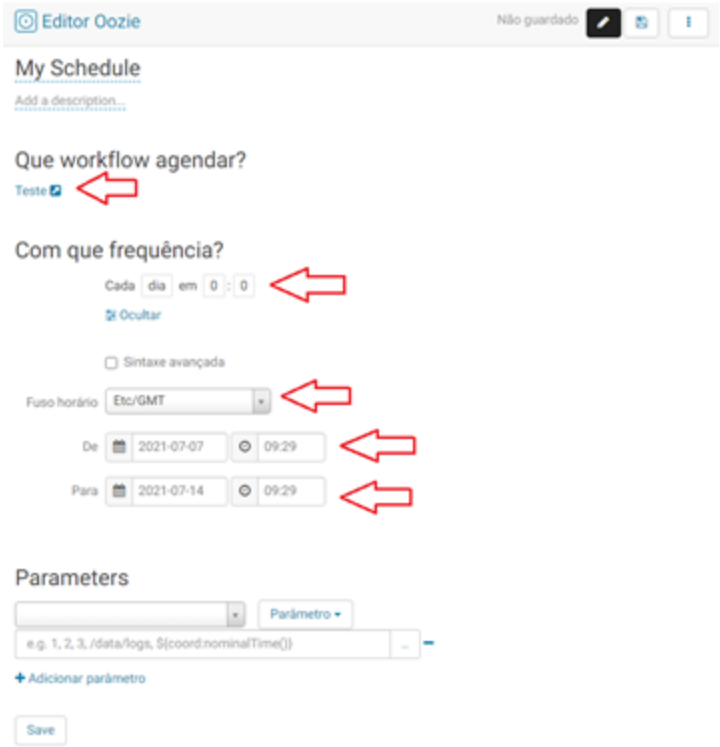
Através do Schedule poderá agendar a execução de seus fluxos com base em tempo, eventos ou dados.

Para criação de um novo fluxo, deve-se acessar o Menu principal, Browser Documents.



No canto superior direito irá encontrar um botão para criação de novos documentos, e nesta lista a opção schedule.

Hands on



Scheduler

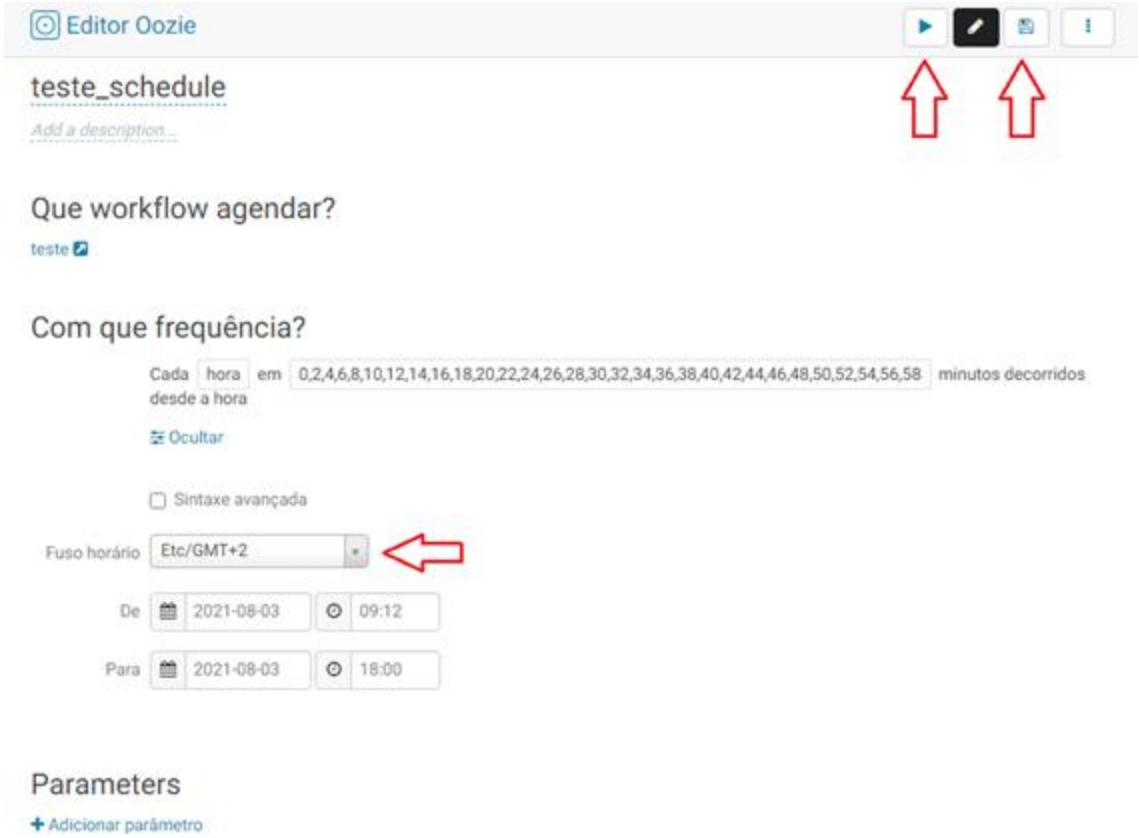
Os dados básicos a serem fornecidos para criação do agendamento são:

- Workflow a ser agendado
- Frequência de execuções: Definição de qual período será utilizado para definição das execuções. ex: Fluxo deve ser executado diariamente às 12 e as 00 horas.
- Timezone− Timezone do aplicativo schedule. Fuso horário que deverá ser utilizado como parâmetro para as execuções do fluxo.
- Período de existência do schedule: Por qual período (data/hora início e fim) o schedule deverá existir. ex: Schedule de 01/01/2021 10:00 até 21/12/2021 10:00.

Hands on

Scheduler

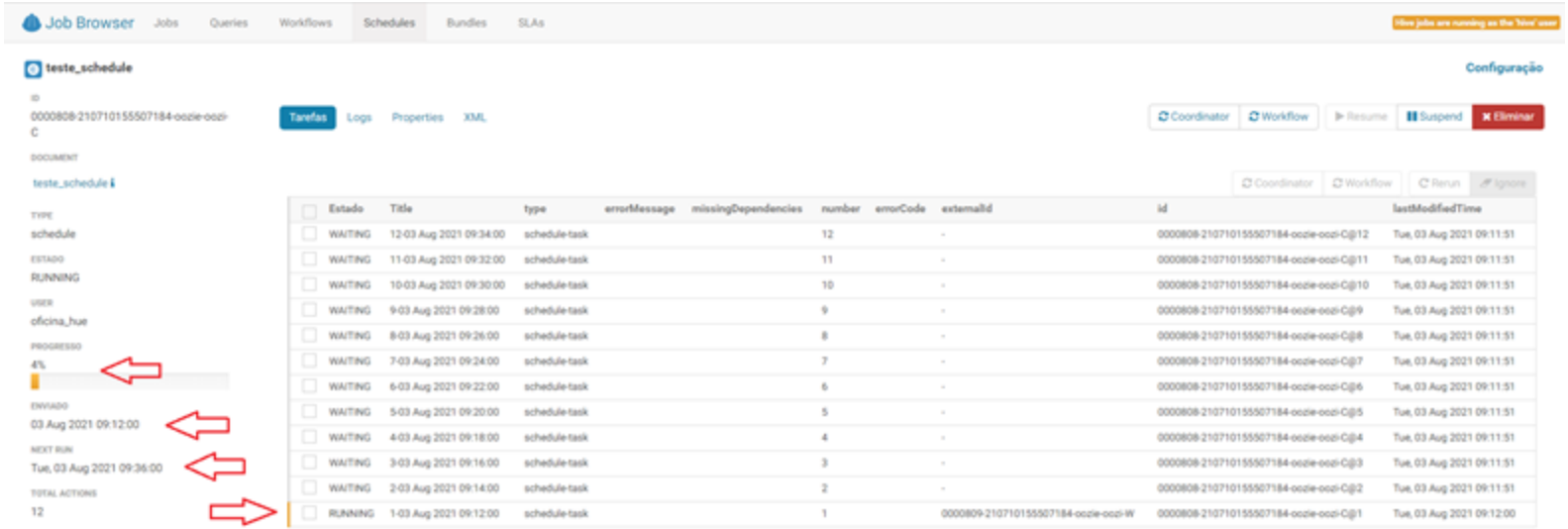
Após salvar o schedule, para que o mesmo seja executado deve-se clicar no botão Play, localizado no canto superior direito da janela. Dessa forma o scheduler irá criar as instâncias de execução de acordo com as definições feitas pelo usuário no momento da criação do scheduler.



Hands on

Scheduler

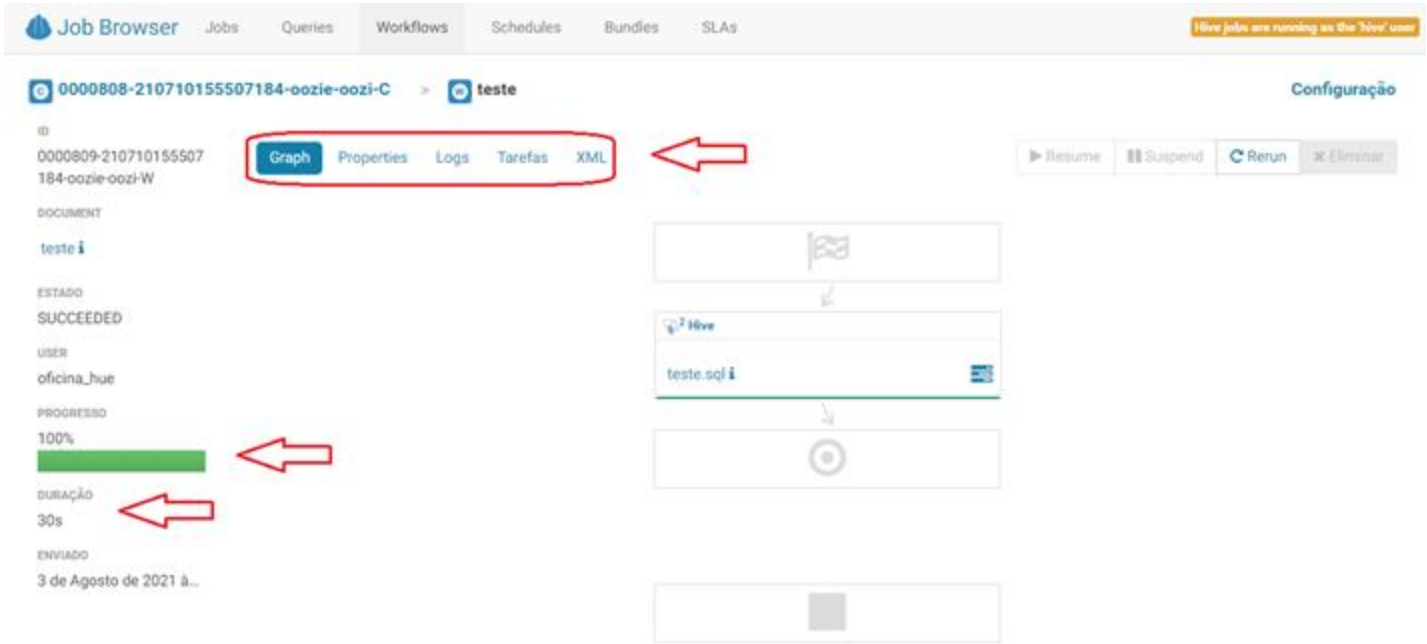
Podemos ver na imagem seguinte o scheduler já iniciado, mostrando o percentual de execução do mesmo, quando foi enviado para execução e o status de execução de cada instância gerada. Podemos observar que foram geradas neste caso 12 instâncias de execução, mas reparem que ele já coloca como próxima execução o primeiro horário subsequente à última instância gerada.



Hands on

Scheduler

Durante ou mesmo após a execução o usuário pode clicar em uma instância para verificar maiores detalhes da execução.



Exercícios 2

Realizar atividades localizadas no documento **Exercícios,** subtítulo **Criação de Workflow e Schedule,** enviado por email no início da Oficina**.**

Dúvidas ?

Instrutores

Bruno Mariz
bruno.oliveira@serpro.gov.br

Anderson Ceo
anderson.ceo@serpro.gov.br

SUPAI - Janeiro/2023

/serprobrasil
@serprobrasil
@serpro
/serpro
serpro.gov.br

SERPRO
Soluções digitais que conectam governo e sociedade