PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

# SQL para Hadoop: Avaliação Comparativa entre as Ferramentas Hive e Impala

SQL to Hadoop: Comparative evaluation between Hive and Impala tools

Tiago Chaves dos Santos[1]
Tadeu dos Reis Faria[2]

**Resumo**

O termo Big Data vem ganhando cada vez mais espaço no mercado, e com isso o Hadoop também ganha uma força notória, pois se trata de uma plataforma criada para processar alto e variado volume de dados. Considerando este contexto, o presente artigo aborda conceitos e características de Big Data e Hadoop e apresenta um estudo comparativo de desempenho entre as ferramentas Apache Hive e Cloudera Impala, que disponibilizam acesso ao Hadoop via SQL. O estudo foi feito com a definição de um ambiente Hadoop, criação de um Banco de Dados neste ambiente; definição de critérios de avaliação, elaboração e realização de consultas SQL por meio das ferramentas. Com a captura dos tempos de execução e análise dos mesmos, se verificou que, de acordo com os critérios de avaliação definidos e com o ambiente usado para os testes, o Impala apresentou melhor desempenho em todas as consultas realizadas.

**Palavras-chave:** Big Data**,** Hadoop, Interface de acesso.

[1]Graduando em Sistemas de Informação pelo Programa de Graduação em Sistemas de Informação da PUC Minas, Tiagoch25@gmail.com – Brasil

[2]Professor do Programa de Graduação em Sistemas de Informação da PUC Minas, tadeurf7@gmail.com - Brasil

**Abstract**

The term big data has been conquering more space in the market, and with it the Hadoop also gained a notorious force because it is a platform created to process high and various forms of data. Considering this context, this article discusses concepts and characteristics of Big Data and Hadoop and presents a comparative performance study between Apache Hive tools and Cloudera Impala, which provides access to Hadoop via SQL. The study has been done with the definition of a Hadoop environment, creating a database in this environment; definition of evaluation criteria, preparation and execution of SQL queries using the tools. It was verified that, according to defined evaluation criteria and the environment used for testing, the impala performance is the best in all checkings with the capture of execution times and its analysis.

**Keywords:** Big data, Hadoop, Access Interface.

## 1 INTRODUÇÃO

A difusão da internet e o fácil acesso a *smartphones e tablets,* junto com as diversas informações geradas e armazenadas por empresas de diversas áreas, têm contribuído para a geração de uma grande quantidade de dados. Segundo Sobers (2013), essa alta quantidade de dados gerados pelo homem, está compreendido por dados estruturados e não estruturados que chegam a ordem de e*xabytes*.

Para que seja possível gerar valor a partir desse imenso volume de dados é necessário que eles sejam armazenados e analisados, possibilitando assim que as empresas que utilizam os resultados dessas análises, possam se diferenciar de seus concorrentes. Sobers (2013) afirma que a maioria das grandes empresas ainda não possuem ferramentas capazes de explorar todos esses dados, e segundo Rezende (2014), o Apache Hadoop é uma tecnologia que vem ganhando espaço nesse mercado.

Devido à complexidade que envolve o Apache Hadoop, faz-se importante a utilização de algo já totalmente difundido no mercado, como, por exemplo, *Structured Query Language* (SQL). Sem as ferramentas de acesso ao Hadoop via SQL, o acesso aos dados fica mais restrito a pessoas com amplo conhecimento de interfaces de aplicações técnicas, como o *Hadoop Distributed File System* (HDFS) e

o MapReduce (LANS, 2014).

Pensando nisso, desde o início do projeto Apache Hadoop já estava incluído um componente capaz de permitir o uso do SQL. Mas com o passar do tempo e com a difusão dessa tecnologia, diversas empresas criaram suas próprias ferramentas para SQL em Hadoop (LANS, 2014). Porém, com essa diversidade de ferramentas, uma questão central é como escolher aquela que melhor atende a determinadas necessidades. Ainda segundo Lans (2014), a escolha de qual ferramenta utilizar deve ser fundamentada na avalição de aspectos importantes como: dialeto SQL, desempenho na realização de junções entre tabelas com alto volume de dados, capacidade do motor trabalhar com dados não estruturados de maneira a otimizar consultas, suporte a formatos de armazenamento padrão do Hadoop, liberdade para o usuário definir suas próprias funções e distribuição destas em diferentes nós, cargas de trabalho multiusuários e possibilidade de trabalhar com diferentes fontes de dados.

Considerando essas questões sobre como escolher a ferramenta adequada, este artigo tem como objetivo geral realizar um estudo comparativo entre as ferramentas Apache Hive e Cloudera Impala, ferramentas essas que possibilitam o uso de SQL no Apache Hadoop.

Os objetivos específicos são: apresentar um estudo sobre o Apache Hadoop; apresentar os conceitos de MapReduce e HDFS; apresentar as principais características das ferramentas Apache Hive e Cloudera Impala e mostrar os resultados da avaliação comparativa entre essas ferramentas.

A produção deste trabalho foi motivada pelo fato do Hadoop ser uma tecnologia que ainda está se consolidando no mercado em termos de volume de uso e por estar surgindo diversas ferramentas SQL para a mesma.

É importante, então, elaborar trabalhos que mostrem as características dessas ferramentas e comparem o desempenho delas em determinadas situações para orientar decisões sobre qual utilizar.

## 2 REFERENCIAL TEÓRICO

Nesta seção são apresentados os conceitos teóricos dos elementos utilizados para o desenvolvimento deste artigo.

## 2.1 Big Data

A internet, o maior número de computadores e *smartphones* facilitam de maneira imensurável a geração de dados e esse variado volume de dados, se processado, pode gerar valor para as empresas.

Neste contexto é apresentado o termo *Big Data*. Segundo Eaton et al (2012), esse termo é definido como grande quantidade de dados complexos gerados em alta velocidade que necessitam de técnicas apuradas para capturar, armazenar, distribuir, manusear e analisar as informações contidas.

Segundo Alecrim (2015), *Big Data* pode ser definido como um conjunto de dados extremamente amplo que, por este motivo, se faz necessário ferramentas especializadas para tratar grandes volumes, de maneira que as informações contidas neste meio possam ser encontradas, analisadas e aproveitadas em tempo hábil.

Em Dumbill (2012) é definido *Big Data* como dados que excedem a capacidade de processamento dos Sistemas de Gerenciamento de Banco de Dados (SGBDs) convencionais, sendo preciso caminhos alternativos para processá-los, e aponta três grandes pilares, os chamados 3 ‘V’s; Volume devido ao grande montante de dados; Velocidade devido à rapidez que estes dados são gerados e muitas vezes analisados, como por exemplo dados de transações bancárias; Variedade que se relaciona a origem dos dados, pois estes vêm de diversas fontes como, textos e imagem de redes sociais, dados gerados em compras online, em sistemas Enterprise Resource Planning (ERP) das empresas. Assim, *Big Data* pode ser considerado como sendo a combinação entre volume + velocidade + variedade.

Existe uma discussão quanto a inclusão de mais dois ‘V’s: Veracidade, pois o dado precisa ser confiável, e Valor, este sendo gerado para a empresa a partir da combinação dos quatro ‘V’s anteriores. A discussão se dá devido aos dois últimos aspectos fazerem referência ao que para alguns parece ser óbvio, pois acredita-se que já estão implícitos no negócio – “qualquer entidade séria, sabe que precisa de dados consistentes; nenhuma entidade toma decisões e investe se não houver expectativa de retorno” (ALECRIM, 2015).

## 2.2 Apache Hadoop

Com a difusão do conceito *Big Data*, é visível a necessidade de ferramentas e tecnologias capazes de processar grandes volumes de dados em busca de

resultados expressivos para determinado fim. Segundo Rezende (2014), uma dessas tecnologias é o Apache Hadoop.

White (2011) diz que o Hadoop foi criado por Doug Cutting, e tem suas origens no Apache Nutch - um motor de busca na *web* com código aberto. Rezende (2014, p.27) de forma bem simplista, diz que "Hadoop é um *framework* de código aberto para armazenamento e processamento de dados em larga escala através de utilização de *clusters* de *hardware* barato. "

Segundo Apache Hadoop (2014)[3], o Hadoop permite o processamento distribuído de grandes conjuntos de dados em *clusters* de computadores, que utilizam modelos de programação simples, e foi projetado para ampliar a partir de um único servidor para milhares de máquinas, cada uma oferecendo armazenamento e computação local.

Ainda segundo sua documentação são apontados quatro módulos no projeto Hadoop:

- Hadoop Common: Os utilitários comuns que suportam os outros módulos do Hadoop.
- Hadoop Distributed File System (HDFS): Sistema de arquivos distribuídos que fornece acesso de alto rendimento para os dados do aplicativo.
- Hadoop YARN: *Framework* para programação de trabalho e gestão de recursos de *cluster*.
- Hadoop MapReduce: Sistema para processamento paralelo de grandes conjuntos de dados, em larga escala.

Partindo do princípio que computadores são propícios a falhas, Rezende (2014) afirma que todos os módulos do Hadoop são projetados com o propósito fundamental de que as falhas devem ser tratadas automaticamente no *software*.

Segundo Sachin (2014), além dos quatro módulos apresentados, a "plataforma" completa do Apache Hadoop é composta por uma série de projetos relacionados, como por exemplo: Apache Hive, Apache Hbase.

Os dois principais componentes do Apache Hadoop são o Hadoop Distributed.

[3]...Documentação do apache Hadoop no site oficial. Disponível em: <http://Hadoop.apache.org/>. Acesso em: 19 de abril de 2015.

File System - HDFS e o *framework* de processamento paralelo MapReduce. Este é composto por diversos tipos de nós (SACHIN, 2014). Nó no Apache Hadoop é o nome dado a um simples computador que contém dados.

Ainda segundo Rezende (2014), o Hadoop é dividido em nós *Master* (mestre) e nós *Slave* (escravo). Nós mestres consistem em J*obTracker, TaskTracker, NameNode* e *DataNode*, enquanto os nós escravos ou nós de trabalho funcionam como TaskTracker e como DataNode, existindo uma real possibilidade de haver nós escravos somente de cálculos ou somente de dados. Pequenos *clusters* incluem um único mestre e vários escravos e necessita do Java Runtime Environment (JRE) 1.6 ou superior. Para os *scripts* de inicialização e desligamento é necessário o Secure Shell (SSH) a ser estabelecido entre os nós do *cluster*. Em *clusters* maiores, o HDFS é gerenciado por um servidor de *NameNode* dedicado para hospedar os índices do sistema de arquivos e um segundo *NameNode* para garantir a redundância em casos de corrupção de arquivos ou perda de dados. Da mesma maneira pode haver um servidor *JobTracker* autônomo gerenciando agendamento de trabalho.

No Hadoop em modo local, todo processo ocorre dentro de uma única Java Virtual Machine (JVM). Ele não utiliza HDFS, mas sim o sistema de arquivos local, e permite a utilização de todas as ferramentas de desenvolvimento Java. A utilização do sistema de arquivos local permite aplicar rapidamente comandos Unix ou *scripts* simples sobre os dados de entrada e saída, por outro lado, limita-se a comandos fornecidos pela linha de comando Hadoop (LAM, 2011).

Quando um conjunto de dados ultrapassa a capacidade de armazenamento de uma única máquina física, surge a necessidade de particioná-lo em várias máquinas distintas. Sistemas de arquivos que gerenciam o armazenamento através de uma rede de máquinas são chamados de sistemas de arquivos distribuídos (WHITE, 2011).

Segundo Sachin (2014), o HDFS é um sistema de arquivos distribuídos portátil e escalável, escrito em Java para o *framework* Hadoop, derivado do artigo da google, Google File System (GFS).

Segundo Eaton et al (2012), a técnica de MapReduce é o coração do Hadoop, pois é ele que permite a escalabilidade massiva através de centenas ou milhares de servidores em um *cluster* Hadoop. O termo refere-se a duas simples tarefas distintas:

- **Map**: Trabalho de mapear, de maneira parcelada em diferentes nós de *cluster* (nós trabalhadores), convertendo grupo de dados em outro grupo de

dados, onde os elementos individuais são divididos em tuplas (pares de chave e valor).

- **Reduce**: Com o resultado dos nós do processo de mapeamento como entrada, o *reduce* reúne o trabalho, aplica uma função associativa e apresenta o resultado.

## 2.3 Interfaces de acesso

O grande diferencial do Hadoop é a junção de armazenamento distribuído (HDFS) e computação distribuída (MapReduce), permitindo assim alto poder de processamento e escala sobre um grande volume de dados. O uso de SQL permite aplicação de padrões e facilidade no consumo dos dados. Permite também aproveitar as ferramentas de *Business Intelligence* (BI) já existentes e os conhecimentos sobre SQL que a empresa já possui, para pesquisar, analisar e visualizar o seu *Big Data* processado. Isso elimina a necessidade de contratação de especialistas capazes de escrever consultas complexas em diferentes linguagens, como por exemplo Python e Java (QUERYIO, 2013).

A ferramenta pioneira para este fim foi o Hive o qual faz parte do ecossistema Hadoop. Mas com o passar do tempo empresas investiram na criação de outras, como por exemplo, Stinger, Sparck SQL, Presto, Cloudera Impala, Apache Drill, IBM BigSQL, Hadapt (YEGULALP, 2014).

Em Abadi (2013) e feita uma categorização das ferramentas SQL em Hadoop, referenciando seis dessas ferramentas e separando-as pelas seguintes características:

1. SQL traduzido para trabalhos MapReduce sobre um cluster Hadoop.
2. SQL processada por um motor SQL especializado.
3. Processamento de consultas SQL dividido entre MapReduce e armazenamento que utiliza SQL nativo.

A primeira categorização apresenta vantagem por utilizar os recursos do MapReduce nativo do Hadoop, onde uma consulta SQL enviada a um *cluster* é traduzida em uma série de trabalhos MapReduce no *cluster*, permitindo, assim, a existência de tolerância a falhas e minimizando problemas de desempenho. Porém, apresenta desvantagem na fraca presença de cobertura e padrões SQL, dificultando a integração com ferramentas de terceiros. Nesta categoria são incluídas as ferramentas Hive e Stinger.

Abadi (2013) enquadra as ferramentas Drill e Impala na segunda categoria. Elas foram inspiradas nos projetos, Google Dremel e Google's F1 respectivamente. São motores SQL especializados que se sentem em um *cluster* Hadoop. Ambos motores são armazenados no sistema de arquivos distribuídos (HDFS). Por esse motivo apresentam vantagens quanto ao processamento de dados. Porém, devido terem construído motores de execução de consulta SQL a partir do zero, perdem na cobertura e padrão SQL. Além disso, pelo não uso do MapReduce, perdem em escalabilidade e em tolerância a falhas.

Por fim, na terceira categoria são enquadradas as ferramentas Hadapt e Polybase. Elas tentam obter o que se tem de melhor no MapReduce e SQL nativos, realizando processamento em ambos. Dependendo dos aspectos necessários, quando a consulta exige resposta rápida, o MapReduce é evitado e toda consulta é realizada através de SQL nativo. Mas quando a consulta exige alta escala e alta tolerância a falhas, mais trabalho é destinado ao motor MapReduce.

Neste artigo são analisadas as ferramentas Apache Hive e Cloudera Impala em relação aos aspectos de desempenho.

### 2.3.1 Apache Hive

Segundo White (2011), Hive é uma estrutura para armazenamento de dados sob o Apache Hadoop. Teve seu crescimento a partir da necessidade de gerenciar e aprender com o grande volume de dados produzido pelo *facebook* no dia a dia. Hive foi criado para possibilitar que especialistas em linguagem SQL pudessem consultar e analisar esses dados.

Segundo Apache Hive Wiki (2015)[4], o Hive consiste em um *software* de *Data Warehouse* que facilita consultas e gerenciamento de grandes conjuntos de dados residentes em sistemas de armazenamento distribuído. Construído como um dos projetos do Apache Hadoop, fornece ferramentas que permitem realização do processo de ETL (extração, transformação e carga de dados), acesso aos arquivos armazenados em sistemas de armazenamento de dados e execução de consultas via MapReduce. O Hive é composto por:

[4]...Documentação do apache Hive no site oficial. Disponível em: <https://cwiki.apache.org/confluence/display/Hive/Home>. Acesso em: 05 de março de 2015.

- **HCatalog**: camada que permite o gerenciamento e o armazenamento de dados no Hadoop possibilitando o processamento de dados com diferentes ferramentas incluindo Pig e MapReduce, facilitando leitura e gravação dos dados em *grid,* apresenta uma visão relacional de dados no HDFS e garante que os usuários não precisem se preocupar sobre onde ou em que formato os seus dados são armazenados.
- **WebHCat**: oferece serviço que permite executar Hadoop MapReduce (ou YARN), Pig, Hive Jobs ou realizar operações de metadados Hive usando um *Hypertext Transfer Protocol* (HTTP) por meio de *Representational State Transfer* (REST).

### 2.3.2 Cloudera Impala

Cloudera Impala se trata de mais um projeto de código aberto que implementa a integração de dados do Hadoop com a linguagem SQL. Cloudera (2015)[5] mostra que o Impala combina benefícios de outros *framewoks* Hadoop como, flexibilidade, escalabilidade, usabilidade e funcionalidade SQL, necessários em bancos de dados analíticos empresariais.

Floratou et al (2014) apresentam algumas características do Impala:

- Oferece seus próprios modelos de execução em cada nó do *cluster*, ao invés de confiar em outros quadros do Hadoop.
- Contém uma arquitetura sem compartilhamento de banco de dados paralelos.
- Tem como principal processo o envolvimento entre planejador de comandos, coordenador de consultas e motor de execução das consultas.
- Suporta diferentes tipos de junção.
- Consta camada de I/O de alto desempenho quanto a leitura de dados armazenados em HDFS.
- Explora as instruções Streaming SIMD de Extensão (SSE), disponível na mais recente geração de processadores Intel para analisar e processar dados textuais com eficiência.

Bigdatanerd (2014), Dezyre (2015) e Discovered Intelligence (2014) apresentam algumas diferenças entre as duas ferramentas. Elas são resumidas na Tabela 1.

[5]... Site oficial. Disponível em: <http://www.cloudera.com/content/cloudera/en/products-and-services/cdh/impala.html>. Acesso em: 10 de março de 2015.

**Tabela 1: Comparação *Hive* Vs *Impala***

| Aspectos | Hive | Impala |
|---|---|---|
| **Modo SQL** | Batch | Interativo |
| **Suporte a Integralidade SQL ANSI** | ***Nível*** | |
| ***Consultas Select*** | Médio | Médio |
| ***DDL/DML*** | Médio | Baixo |
| ***Funções UDFs Personalizadas*** | Alto | Baixo |
| **Cliente de Acesso** | Shell, JDBC, ODBC | Shell, JDBC, ODBC |
| **Suporte a formatos de arquivos.** | TXT, CSV, Sequence, RC, ORC, Parquet, Avro, JSON, Compression, Hive SerDe. | TXT, Sequence, RC, Parquet, Avro, Compression. |
| **Fontes de dados** | Arquivo e Hbase. | Arquivo e Hbase. |
| **Tipos de Dados** | Relacional e complexo. | Relacional |
| **Execução de consultas.** | Traduz as consultas em trabalho MapReduce. | Nativa, usa processamento paralelo. |
| **Apresenta melhor desempenho** | -Tarefas ETL pesadas.<br>-Processamento em lote das altas necessidades de big data. | -Consultas de processamento em grandes volumes de dados. |
| **Vantagens** | -Maior tempo no mercado.<br>-Utilização de MapReduce.<br>-Bom suporte a funções definidas pelos usuários.<br>-Facilmente mapeado para Hbase e outros sistemas. | -Processamento de consultas adhoc quase em tempo real.<br>-O cálculo acontecer na memória, reduzindo a quantidade de latência e disco I/O. |
| **Desvantagens** | -Devido utilizar MapReduce, maior tempo de execução e maiores operações de I/O.<br>-Fraco suporte a funções de agregação e ordenação, fazendo com que essas gastem mais tempo.<br>-Lentidão quando comparado aos seus concorrentes. | -Sem tolerância a falhas para a execução de consultas. Se uma consulta falhou em um nó, a consulta tem de ser reeditada, não pode retomar a partir de onde ele falhou. |

**Fonte: Elaborada pelo autor**

## 3 METODOLOGIA

Neste trabalho foram utilizadas as ferramentas Apache Hive e Cloudera Impala. A escolha destas ferramentas foi baseada no fato de Apache Hive ser nativo do Hadoop e o Impala ser de um dos grupos que fazem distribuição de uma plataforma com todos os serviços do Hadoop já pré-instalados, plataformas essas que estão em

plena ascensão no mercado.

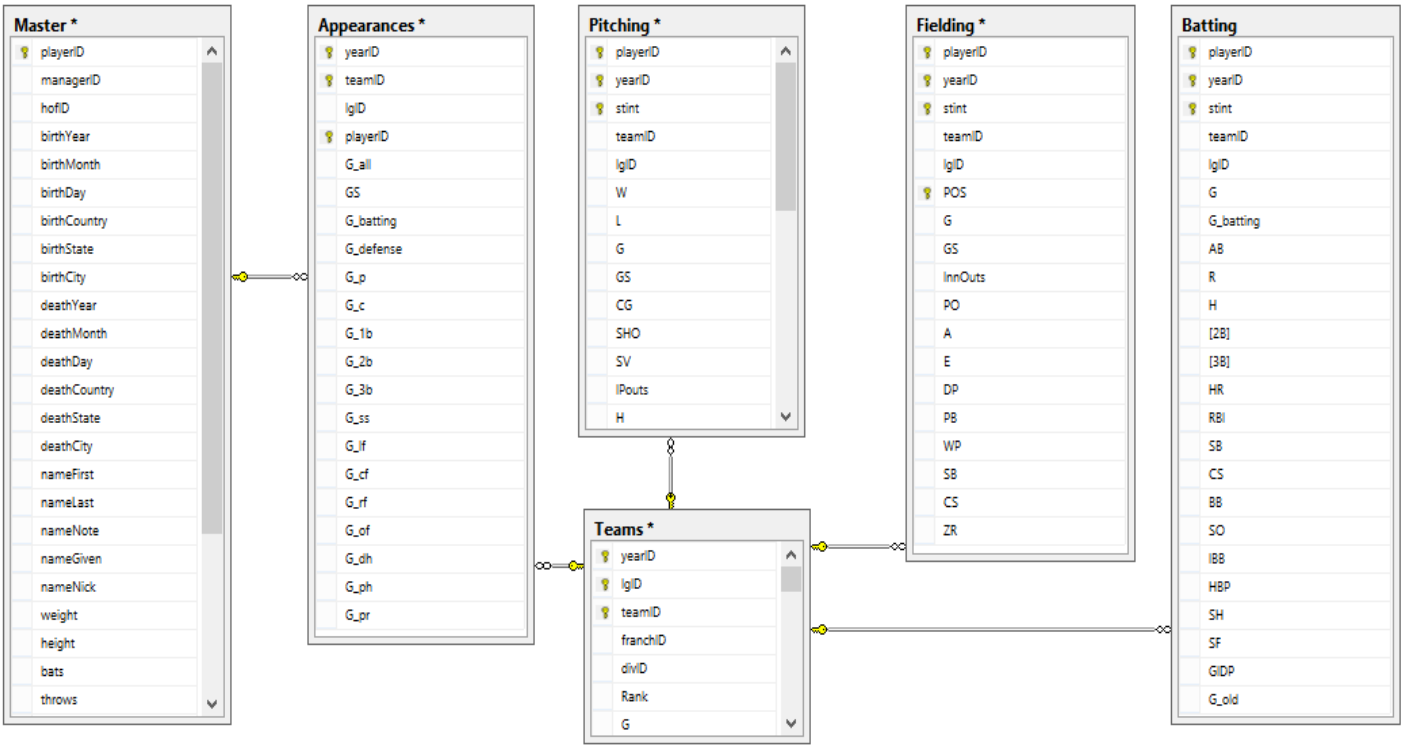
O foco dos testes é comparar desempenho através da medição dos tempos de execução da injeção de consultas SQL no apache Hadoop.

Para a realização dos testes, o Hadoop foi executado em apenas um cluster com as demais aplicações inicializadas:

1) Hue 3.7 – Aplicação responsável por disponibilizar uma interface gráfica agradável para gerenciamento e utilização do *Hadoop*.
2) Hive e Impala – Editores de *Query* (ferramentas usadas para a comparação).
3) Metastore Table Manager – Cria uma interface de gerenciamento das tabelas.

A base de dados foi extraída de Seanlahman (2015), onde são disponibilizados, em diversos formatos, dados referentes às ligas de Baseball de 1871 até 2014. Os dados das tabelas correspondem a lançamentos, rebatidas e diversos dados estatísticos das ligas. A base é composta por 24 tabelas. O formato escolhido para esse trabalho foi arquivo compactado contendo todas as tabelas em formato *comma-separated values* (CSV). Foram selecionadas seis tabelas para os testes. O esquema relacional dessas tabelas é mostrado na Figura 1.

**Figura 1: Esquema Relacional do Banco de Dados**




**Fonte: Elaborada pelo autor.**

As tabelas relacionais carregadas no Hadoop e usadas nos testes estão apresentadas na Tabela 2:

**Tabela 2: Apresentação das tabelas importadas para o Hadoop**

| **Nome** | **Sobre** | **Quantidade de registros.** |
|---|---|---|
| ***Appearances*** | Possui dados sobre as aparições dos jogadores nas ligas. | 99.466 |
| **Batting** | Possui dados estatísticos referentes a rebatidas | 99.846 |
| ***Fielding*** | Possui dados sobre os jogos. | 167.938 |
| ***Master*** | Possui dados sobre os jogadores. | 18.589 |
| ***Pitching*** | Possui dados estatísticos sobre arremessos. | 43.330 |
| ***Teams*** | Possui dados dos times. | 2.775 |

**Fonte: Elaborada pelo autor**

A carga dos dados no Hadoop foi feita pelo Metastore Table Manager. Os dados foram armazenados de forma tabular no HDFS.

As consultas utilizadas para os testes seguiram os seguintes padrões:

1. *Select* na tabela mais populada *(Fielding);*
2. *Select* na tabela mais populada com cláusula *where (Fielding);*
3. *Select* com *joins (Batting, Teams e Master);*
4. *Select* com *joins e duas* cláusulas *where (Fielding, Teams e Master);*
5. *Select* com *joins* cláusula *where, group by* e *Order by (Teams, Appearances e Master);*

Cada consulta foi executada três vezes em cada ferramenta. Com os tempos obtidos foi calculado o valor médio, lembrando que o processo de distribuição aconteceu apenas em um *cluster* Hadoop.

# 4 EXPERIMENTOS E RESULTADOS

## 4.1 Ambiente de testes

A instalação da máquina virtual (VM) e a realização dos testes foram feitos em um computador com as seguintes especificações: Processador: Intel Core i7 3517U @ 1.90 GHz; 8GB de RAM; 500 GB de HD. O sistema Operacional Windows 8.1 Pro. 64 bits.

A imagem virtual com o Hadoop pré-instalado foi extraída de Cloudera (2015). A máquina virtual utilizou a versão CentOS 6.4 do Linux. A versão da máquina foi a Cloudera – Quick Start 5.3. Para a virtualização da máquina foi utilizado o Oracle VirtualBox versão 5.0.

Para carregar a base no Hadoop, foi necessário o *framework* Metastore Table. Os arquivos no formato CSV foram integrados ao *File Browser* possibilitando a carga no Hadoop. Durante a carga foi possível definir tipo de dados, número e nome das colunas. Todo o gerenciamento das aplicações do Hadoop foi feito através do Hue 3.7 facilitando assim a interação. As consultas foram executadas no terminal.

## 4.2 Experimentos Realizados

Nesta seção são mostradas as consultas em SQL realizadas para o experimento. Cada consulta foi executada 3 vezes em cada uma das ferramentas através do terminal. Os tempos apresentados correspondem aos tempos de execução de cada consulta mais o tempo médio das três execuções. Eles foram calculados em segundos. Os resultados das consultas executadas no Hive são mostrados na própria tela do terminal, ao final da apresentação dos dados conforme apresentado na Figura 2. Já os tempos do Impala foram baseados na página de *logs*, onde são apresentados os tempos de *SCAN*, conforme Figura 3. Estes foram convertidos em segundos e somados. A escolha dos tempos de *SCAN* foi devido o Impala mostrar um tempo após a exibição dos dados considerando o tempo gasto para a impressão e no *log* considerar o tempo de avaliação como parte da execução.

**Figura 2: Tempo de execução da consulta no Hive**

```
Time taken: 34.456 seconds, Fetched: 1289 row(s)
hive>
```

**Fonte: Elaborada pelo autor.**

**Figura 3: Log usado para cálculo do tempo de execução no Impala**

Query ba4c1ec326743828:8519b80929006c9f

*Status:* OK

☑ **Auto-refresh on** Last updated: Tue Nov 03 2015 17:05:17 GMT-0800 (PST)

```
select * from fielding where yearid = 2014 and lgid = 'AL'
```

Exec Summary

```
Operator       #Hosts   Avg Time   Max Time  #Rows  Est. #Rows  Peak Mem  Est. Peak Mem  Detail
------------------------------------------------------------------------------------------------
01:EXCHANGE         1   33.660us   33.660us  1.29K           0         0       -1.00 B  UNPARTITIONED
00:SCAN HDFS        1  209.669ms  209.669ms  1.29K           0   8.21 MB       32.00 MB  default.fielding
```

**Fonte: Elaborada pelo autor.**

### 4.2.1 Consulta 1 – *Select* na tabela com maior número de registros

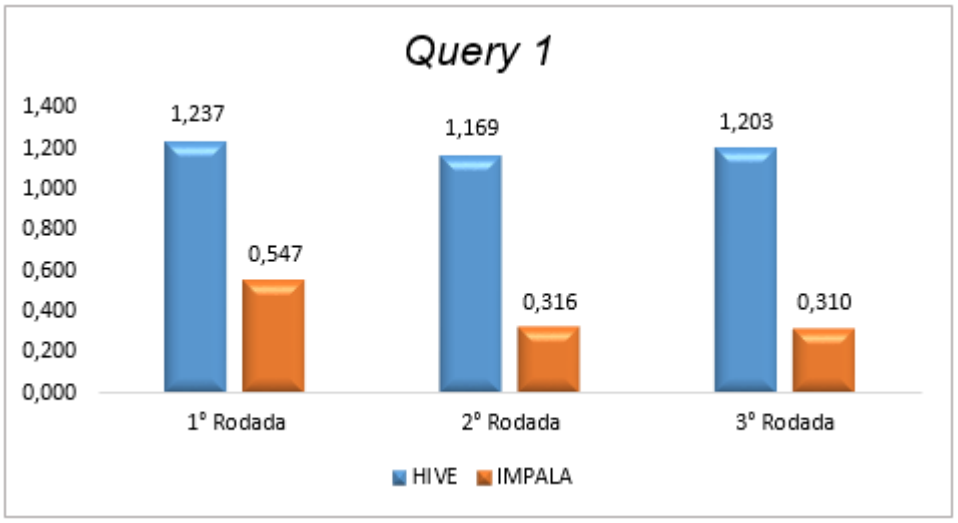
Na primeira consulta foi feito um simples *select* em todos os dados da tabela Fielding.

a) Consulta

*select * from Fielding*

b) Resultados: Como apresentado no Gráfico 1**,** o Impala teve tempo que, após calcular a média, ficou em terço da média dos tempos do Hive.

**Gráfico 1: Tempos Consulta 1**




**Fonte: Elaborada pelo autor.**

### 4.2.2 Consulta 2 – *select* na tabela com maior número de registros com clausula *where*.

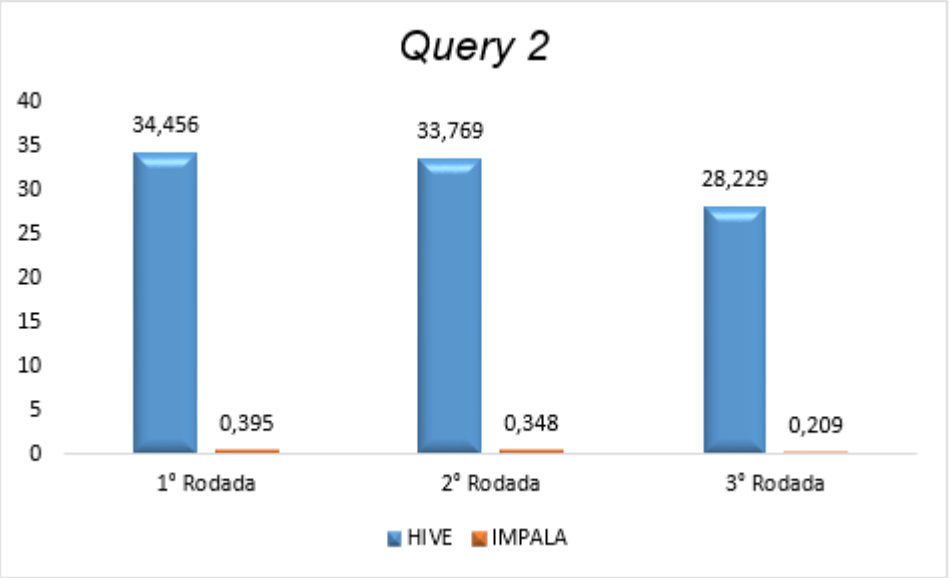
Para a consulta 2 foi utilizada a base da consulta 1 com o acréscimo de uma cláusula *where*, contendo duas condições, onde o resultado representa a seleção de todos os dados da tabela Fielding que pertencem a liga principal (‘AL’) na temporada de 2014.

a) Consulta

*Select * from fielding*
*where yearId = 2014 and lgId = 'AL'*

b) Resultados: O Impala manteve um bom desempenho com tempos abaixo de 0,5 segundos enquanto o Hive sofreu uma significativa alteração levando a uma média de quase 0,5 minutos, como pode ser observado no Gráfico 2.

**Gráfico 2: Tempos Consulta 2**




**Fonte: Elaborada pelo autor**

### 4.2.3 *Consulta 3 - Select* contendo dois *joins*

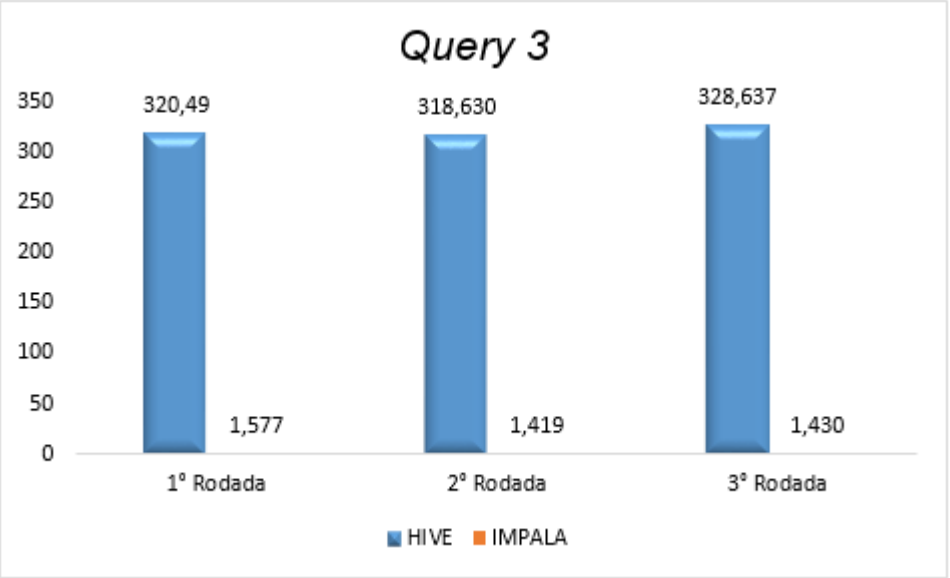
A consulta 3 foi elaborada focando junções entre tabelas, onde os dados apresentados correspondem a seleção de dados da tabela Batting, que representam dados estatísticos de arremessos fazendo *join* com as tabelas Teams e Master, trazendo descrição dos nomes de time e jogador.

a) Consulta

```
Select
        t.name      as Nome_Time,
        m.nameFirst as Nome_jogador,
        b.*
from [dbo].[Batting] as b
inner join [dbo].[Teams] as t on b.teamID = t.teamID
inner join [dbo].[Master] as m on m.playerID = b.playerID
```

b) Resultados: Como se pode observar no Gráfico 3 os tempos da consulta apresentam uma variação muito grande de uma ferramenta para a outra, tanto que nem aparece as barras referentes aos tempos do Impala, pois ele apresentou uma média de menos de 2 segundos enquanto o Hive média de mais de 5 minutos.

**Gráfico 3: Tempos Consulta 3**




**Fonte: Elaborada pelo autor.**

### *4.2.4* Consulta 4 – *Select* contendo dois *joins* com clausulas *where*

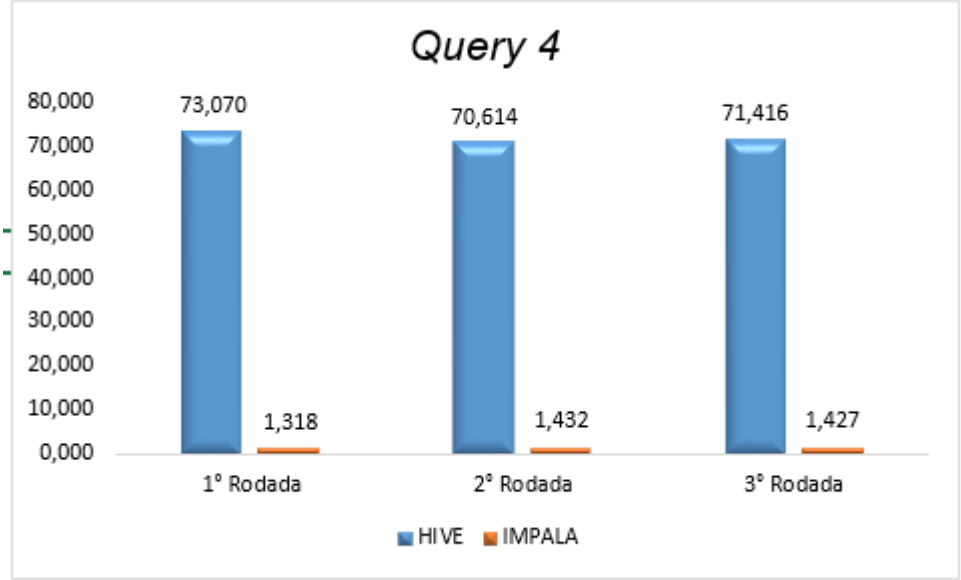
A consulta 4 foi criada focando junções com clausula *where*, são apresentados dados da tabela Fielding fazendo *joins* com a Master e a Teams para trazer descrição dos nomes dos times e dos jogadores, e a cláusula *where* filtra apenas dados da Liga principal (‘AL’) e da temporada de 2014.

a) Consulta

```
Select
        t.name   as Nome_Time,
        m.nameFirst as Nome_jogador,
        f.*
from [dbo].[Fielding] as f
inner join [dbo].[Teams] as t on f.teamID = t.teamID
inner join [dbo].[Master] as m on m.playerID = f.playerID
where f.yearID = 2014 and f.lgID = 'AL'
```

b) Resultados: Para esta consulta o Impala também apresentou melhor desempenho com sua média abaixo de 1,5 segundos enquanto o Hive teve seus tempos todos superiores a 1 minuto, como é mostrado no Gráfico 4.

**Gráfico 4: Tempos Consulta 4**




**Fonte: Elaborada pelo autor.**

### *4.2.5* Consulta 5 – *Select* contendo dois *joins* clausulas *where, group by e order by*

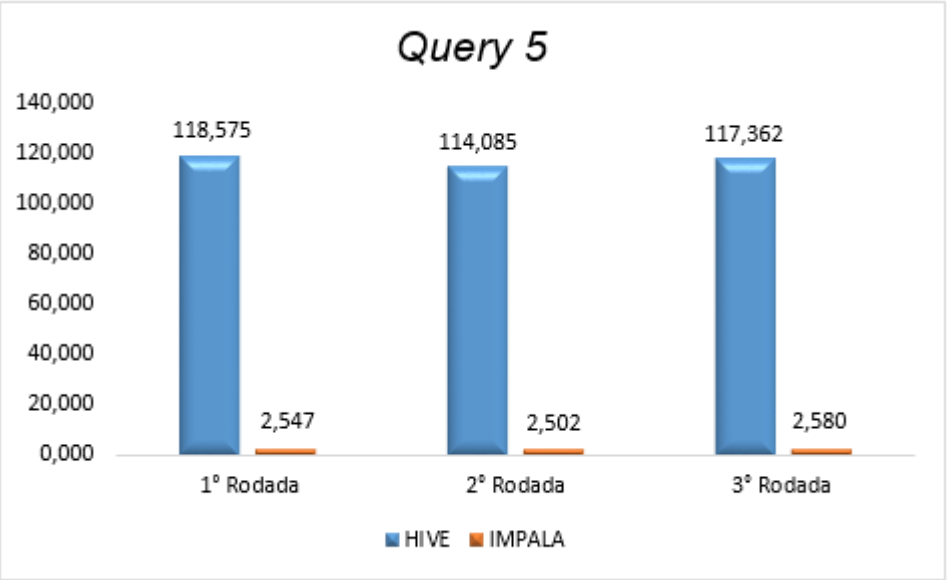
Por fim a consulta 5, possui junção de tabelas, cláusula *where*, funções de agregação e ordenação, traz dados que informam em quantos jogos cada jogador apareceu considerando jogos a partir da temporada do ano 2000, agrupando por liga, time e jogador, ordenados por total de jogos na ordem crescente.

a) Consulta

```
select a.lgId, t.name, m.playerId, m.namefirst,
    count (a.G_all) as jogos
from teams as t
inner join appearances as a on a.teamId = t.teamId
inner join master as m on m.playerId = a.playerId
where a.yearID > 2000
group by
a.lgID,
t.name,
m.playerId,
m.namefirst
order by jogos desc
```

b) Resultados: Para esta consulta como é possível perceber através do Gráfico 5, novamente o desempenho do Hive foi inferior ao do Impala. Enquanto a média do Hive se aproxima de 2 minutos o Impala tem média de 2,5 segundos.

**Gráfico 5: Tempos Consulta 5**




**Fonte: Elaborada pelo autor.**

## 4.3 Avaliação Geral

A Tabela 3 e o Gráfico 6 mostram um consolidado dos testes. Fica novamente

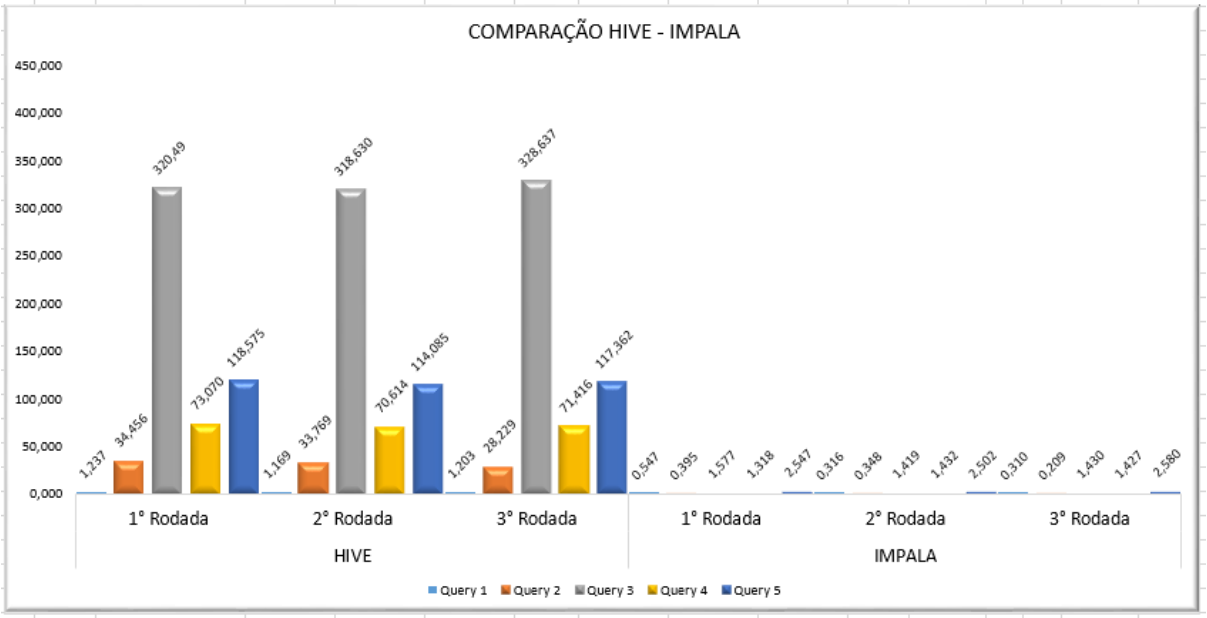
evidenciado que o Impala apresentou melhor desempenho para todas as consultas, mas com uma diferença menor na consulta 1.

**Tabela 3: Tempos de execução das consultas**

| | | HIVE | | | | IMPALA | | | |
|---|---|---|---|---|---|---|---|---|---|
| *Query* | *Linhas retornadas* | *1° Rodada* | *2° Rodada* | *3° Rodada* | *Media* | *1° Rodada* | *2° Rodada* | *3° Rodada* | *Media* |
| 1 | 167.938 | 1,237 | 1,169 | 1,203 | 1,203 | 0,547 | 0,316 | 0,310 | 0,391 |
| 2 | 1.289 | 34,456 | 33,769 | 28,229 | 32,151 | 0,395 | 0,348 | 0,209 | 0,317 |
| 3 | 199.368 | 320,49 | 318,630 | 328,637 | 322,586 | 1,577 | 1,419 | 1,430 | 1,475 |
| 4 | 84.741 | 73,070 | 70,614 | 71,416 | 71,700 | 1,318 | 1,432 | 1,427 | 1,392 |
| 5 | 15.238 | 118,575 | 114,085 | 117,362 | 116,674 | 2,547 | 2,502 | 2,580 | 2,543 |

**Fonte: Elaborada pelo autor**

**Gráfico 6: Resultado Geral**




**Fonte: Elaborada pelo autor.**

A partir da consulta 2 onde não era mais uma simples leitura de dados de uma única tabela, os tempos do Impala passaram a ser muito inferiores aos do Hive. Isso se deve às características das ferramentas, pois o Hive começa a dividir os trabalhos no MapReduce, onde cada *Job* fica responsável por uma parte da execução das consultas. A consulta que mais apresentou diferenças foi a 3 por ela conter junções, cláusulas *where* e trazer um número maior de registros.

Uma das dificuldades da comparação foi a forma que cada ferramenta trata e exibe os dados referentes aos tempos. Pela interface, o Hive apresenta hora de início e de fim de execução da *query*, e pelo terminal apresenta o tempo em segundos ao

final da apresentação dos dados sem contar o tempo gasto para a impressão na tela. Por outro lado, o Impala tem uma porta específica onde fica registrado os *logs* das consultas executadas seja pela interface ou pelo terminal, mas esse *log* não deixa claro qual é o tempo apenas de execução, e para o tempo geral considera até mesmo o chamado tempo de análise de dados, que é o período em que os dados ficam apresentados na tela, para a análise do resultado da consulta, e no terminal o tempo informado na tela corresponde ao tempo de execução mais tempo de apresentação dos dados na tela.

## 5 CONCLUSÃO

Este trabalho apresentou um estudo comparativo quanto ao desempenho entre as ferramentas Hive e Impala. Ambas permitem acesso via SQL em Hadoop. Foi criado um banco de dados no ambiente Hadoop com um único nó, e submetido algumas consultas SQL, onde os tempos de execução foram usados para a comparação.

Os tempos extraídos apresentaram uma menor diferença quando na leitura de dados de uma única tabela sem a realização de filtros, mas à medida que a complexidade das *queries* foi aumentando, com a inclusão de cláusulas *where*, *joins* e funções de agregação e ordenação, a diferença de tempo de uma ferramenta para a outra foi só aumentando, com o Impala mostrando melhor desempenho em todas as consultas realizadas. O maior tempo apresentado pelo Hive justifica-se devido à ferramenta utilizar os recursos do MapReduce, dividindo a execução da consulta em pequenas partes, e distribuindo em *Jobs* distintos, enquanto o Impala não faz nenhuma divisão de tarefas.

No decorrer da realização do trabalho as maiores dificuldades apareceram no momento de configuração do Hadoop, pois surgiram vários erros, e quanto a extração dos tempos de execução do Impala, tendo que a ferramenta gera um extenso *log* repleto de informações levando a dúvidas quanto a qual tempo coletar.

Como sugestões para trabalhos futuros fica: um estudo aprofundado do *log* gerado pelo Impala; alteração do ambiente para distribuído em diversos *clusters* seja local ou em rede; teste de outras ferramentas para o mesmo fim como por exemplo Stinger, Presto, Apacahe Drill, IBM BigSQL, Hadapt, Hawq, ou ainda, testar o desempenho de ferramentas em diferentes plataformas Hadoop como as disponibilizadas pela TeraData, Hortonworks, Pivotal.

## REFERÊNCIAS

ABADI, Daniel. **Classifying the SQL-on-Hadoop Solutions**. 2013. Disponível em: <http://hadapt.com/blog/2013/10/02/classifying-the-sql-on-hadoop-solutions/>. Acesso em: 07 de maio de 2015.

ALECRIM, Emerson. **O que é Big Data?** .2015. Disponível em: <http://www.infowester.com/big-data.php>. Acesso em: 16 de março de 2015.

**APACHE HADOOP**. 2014. Disponível em: <http://Hadoop.apache.org/>. Acesso em: 19 de abril de 2015.

**APACHE HIVE WIKI**. 2015. Disponível em: <https://cwiki.apache.org/confluence/display/Hive/Home>. Acesso em: 05 de março de 2015.

BIGDATANERD, **Hive, Impala and Presto – The War on SQL over Hadoop.** 2014. Disponível em <https://bigdatanerd.wordpress.com/2013/11/19/war-on-sql-over-hadoop/>. Acesso em 03 novembro de 2015.

**CLOUDERA**. 2015.Disponível em: <http://www.cloudera.com/content/cloudera/en/products-and-services/cdh/impala.html>. Acesso em: 10 de março de 2015.

DEZYRE, **Hive vs Impala – SQL War in the Hadoop Ecosystem.** 2015. Disponível em <http://www.dezyre.com/article/-hive-vs-impala-sql-war-in-the-hadoop-ecosystem/148>. Acesso em 03 novembro de 2015.

DISCOVERED INTELLIGENCE, **SQL on Hadoop – A Common Tool Comparison**. 2014. Disponível em <http://discoveredintelligence.ca/sql-on-hadoop-tool-comparison/>. Acesso em 03 novembro de 2015.

DUMBILL, Edd. **What is Big Data?** An introduction to the big data landscape. 2012. Disponível em: <http://radar.oreilly.com/2012/01/what-is-big-data.html>. Acesso em: 14 de março de 2015.

EATON, Chris et al. **Understanding Big Data**: Analytics for Enterprise Class Hadoop and Streaming Data. 1. ed. United States of America: The McGraw-Hill Companies, 2012.

FLORATOU, Avrilia et al. **SQL on Hadoop**: Full Circle Back to Shared - Nothing Database Architectures. 2014. Disponível em: <http://pages.cs.wisc.edu/~floratou/SQLOnHadoop.pdf >. Acesso em 10 de março de 2015.

LAM, Chuck. **Hadoop in Action**, 1° ed. United States of America: Manning Publications Co, 2011.

LANS, Rick Van Der. **Selecting the right SQL-on-Hadoop engine to access big data**. 2014. Disponível em < http://searchbusinessanalytics.techtarget.com/feature/Selecting-the-right-SQL-on-Hadoop-engine-to-access-big-data >. Acesso em 09 de março de 2015.

**QUERYIO**. 2013. Disponível em: <http://queryio.com/product/hadoop-sql.html >. Acesso em: 07 de maio de 2015.

REZENDE, Ricardo. **Big Data e Hadoop: Descubra o que é**: desvendando o Hadoop. Revista SQL Magazine, Grajaú – RJ, n. 120, p. 26 - 30, 2014.

SACHIN, P. Bappalige. **An introduction to Apache Hadoop for big data**. 2014. Disponível em: <http://opensource.com/life/14/8/intro-apache-hadoop-big-data>. Acesso em: 04 de abril de 2015.

**SEANLAHMAN.** Disponível em: < http://www.seanlahman.com/baseball-archive/statistics/>. Acesso em: 25 de agosto de 2015.

SOBERS, Rob. **Gerenciamento de TI: Análise de dados.** 2013.Disponivel em: https://technet.microsoft.com/pt-br/magazine/jj884505.aspx. Acesso em: 21 de abril de 2015.

WHITE, Tom. **Hadoop**: The Definitive Guide. 2. ed. United States of America: O'Reilly Media, 2011.

YEGULALP, Serdar. **10 ways to query Hadoop with SQL**.2014. Disponivel em < http://www.infoworld.com/article/2683729/hadoop/10-ways-to-query-hadoop-with-sql.html>. Acesso em: 21 de abril de 2015.