In this book, we prepare you for your journey into big data by firstly introducing you to backgrounds in the big data domain along with the process of setting up and getting familiar with your Hive working environment. Next, the book guides you through discovering and transforming the values of big data with the help of examples. It also hones your skill in using the Hive language in an efficient manner. Towards the end, the book focuses on advanced topics such as performance, security, and extensions in Hive, which will guide you on exciting adventures on this worthwhile big data journey.

By the end of the book, you will be familiar with Hive and able to work efficiently to find solutions to big data problems.

Who this book is written for

If you are a data analyst, developer, or simply someone who wants to use Hive to explore and analyze data in Hadoop, this is the book for you. Whether you are new to big data or an expert, with this book, you will be able to master both the basic and the advanced features of Hive. Since Hive is an SQL-like language, some previous experience with the SQL language and databases is useful to have a better understanding of this book.

What you will learn from this book

- Create and set up the Hive environment
- Discover how to use Hive’s definition language to describe data
- Discover interesting data by joining and filtering datasets in Hive
- Transform data by using Hive sorting, ordering, and functions
- Aggregate and sample data in different ways
- Boost Hive query performance and enhance data security in Hive
- Customize Hive to your needs by using user-defined functions and integrate it with other tools

In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 "Overview of Big Data and Hive"
- A synopsis of the book’s content
- More information on Apache Hive Essentials

About the Author

Dayong Du is a big data practitioner, leader, and developer with expertise in technology consulting, designing, and implementing enterprise big data solutions. With more than 10 years of experience in enterprise data warehouse, business intelligence, and big data and analytics, he has provided his data intelligence expertise in various industries, such as media, travel, telecommunications, and so on. He is currently working with QuickPlay Media in Toronto, Canada, to build enterprise big data intelligence reporting for online media services and content providers. He has a master's degree in computer science from Dalhousie University, and he holds the Cloudera Certified Developer for Apache Hadoop certification.

I would like to sincerely thank my wife, Joice, and daughter, Elaine, for their sacrifices and encouragement during this journey. Also, I would like to thank my parents for their support during the time of writing this book.

I would also like to thank everyone at Packt Publishing and the technical reviewers for their valuable help, guidance, and feedback on my book.
Apache Hive Essentials

With an increasing interest in big data analysis, Hive over Hadoop becomes a cutting-edge data solution for storing, computing, and analyzing big data. The SQL-like syntax makes Hive easier to learn and popularly accepted as a standard for interactive SQL queries over big data. The variety of features available within Hive provides us with the capability of doing complex big data analysis without advanced coding skills. The maturity of Hive lets it gradually merge and share its valuable architecture and functionalities across different computing frameworks beyond Hadoop.

*Apache Hive Essentials* prepares your journey to big data by covering the introduction of backgrounds and concepts in the big data domain along with the process of setting up and getting familiar with your Hive working environment in the first two chapters. In the next four chapters, the book guides you through discovering and transforming the value behind big data by examples and skills of Hive query languages. In the last four chapters, the book highlights well-selected and advanced topics, such as performance, security, and extensions as exciting adventures for this worthwhile big data journey.

**What This Book Covers**

*Chapter 1, Overview of Big Data and Hive*, introduces the evolution of big data, the Hadoop ecosystem, and Hive. You will also learn the Hive architecture and the advantages of using Hive in big data analysis.

*Chapter 2, Setting Up the Hive Environment*, describes the Hive environment setup and configuration. It also covers using Hive through the command line and development tools.

*Chapter 3, Data Definition and Description*, introduces the basic data types and data definition language for tables, partitions, buckets, and views in Hive.

*Chapter 4, Data Selection and Scope*, shows you ways to discover the data by querying, linking, and scopes the data in Hive.

*Chapter 5, Data Manipulation*, describes the process of exchanging, moving, sorting, and transforming the data in Hive.

*Chapter 6, Data Aggregation and Sampling*, explains how to do aggregation and sample using aggregation functions, analytic functions, windowing, and sample clauses.
Chapter 7, Performance Considerations, introduces the best practices of performance considerations in the aspects of design, file format, compression, storage, query, and job.

Chapter 8, Extensibility Considerations, describes how to extend Hive by creating user-defined functions, streaming, serializers, and deserializers.

Chapter 9, Security Considerations, introduces the area of Hive security in terms of authentication, authorization, and encryption.

Chapter 10, Working with Other Tools, discusses how Hive works with other big data tools. It also reviews the key milestones of Hive releases.
1

Overview of Big Data and Hive

This chapter is an overview of big data and Hive, especially in the Hadoop ecosystem. It briefly introduces the evolution of big data so that readers know where they are in the journey of big data and find their preferred areas in future learning. This chapter also covers how Hive has become one of the leading tools in big data warehousing and why Hive is still competitive.

In this chapter, we will cover the following topics:

- A short history from database and data warehouse to big data
- Introducing big data
- Relational and NoSQL databases versus Hadoop
- Batch, real-time, and stream processing
- Hadoop ecosystem overview
- Hive overview

A short history

In the 1960s, when computers became a more cost-effective option for businesses, people started to use databases to manage data. Later on, in the 1970s, relational databases became more popular to business needs since they connected physical data to the logical business easily and closely. In the next decade, around the 1980s, Structured Query Language (SQL) became the standard query language for databases. The effectiveness and simplicity of SQL motivated lots of people to use databases and brought databases closer to a wide range of users and developers. Soon, it was observed that people used databases for data application and management and this continued for a long period of time.
Once plenty of data was collected, people started to think about how to deal with the old data. Then, the term data warehousing came up in the 1990s. From that time onwards, people started to discuss how to evaluate the current performance by reviewing the historical data. Various data models and tools were created at that time for helping enterprises to effectively manage, transform, and analyze the historical data. Traditional relational databases also evolved to provide more advanced aggregation and analyzed functions as well as optimizations for data warehousing. The leading query language was still SQL, but it was more intuitive and powerful as compared to the previous versions. The data was still well structured and the model was normalized. As we entered the 2000s, the Internet gradually became the topmost industry for the creation of the majority of data in terms of variety and volume. Newer technologies, such as social media analytics, web mining, and data visualizations, helped lots of businesses and companies deal with massive amounts of data for a better understanding of their customers, products, competition, as well as markets. The data volume grew and the data format changed faster than ever before, which forced people to search for new solutions, especially from the academic and open source areas. As a result, big data became a hot topic and a challenging field for many researchers and companies.

However, in every challenge there lies great opportunity. Hadoop was one of the open source projects earning wide attention due to its open source license and active communities. This was one of the few times that an open source project led to the changes in technology trends before any commercial software products. Soon after, the NoSQL database and real-time and stream computing, as followers, quickly became important components for big data ecosystems. Armed with these big data technologies, companies were able to review the past, evaluate the current, and also predict the future. Around the 2010s, time to market became the key factor for making business competitive and successful. When it comes to big data analysis, people could not wait to see the reports or results. A short delay could make a great difference when making important business decisions. Decision makers wanted to see the reports or results immediately within a few hours, minutes, or even possibly seconds in a few cases. Real-time analytical tools, such as Impala (http://www.cloudera.com/content/cloudera/en/products-and-services/cdh/impala.html), Presto (http://prestodb.io/), Storm (https://storm.apache.org/), and so on, make this possible in different ways.
Introducing big data

Big data is not simply a big volume of data. Here, the word "Big" refers to the big scope of data. A well-known saying in this domain is to describe big data with the help of three words starting with the letter V. They are volume, velocity, and variety. But the analytical and data science world has seen data varying in other dimensions in addition to the fundamental 3 Vs of big data such as veracity, variability, volatility, visualization, and value. The different Vs mentioned so far are explained as follows:

- **Volume**: This refers to the amount of data generated in seconds. 90 percent of the world's data today has been created in the last two years. Since that time, the data in the world doubles every two years. Such big volumes of data is mainly generated by machines, networks, social media, and sensors, including structured, semi-structured, and unstructured data.

- **Velocity**: This refers to the speed in which the data is generated, stored, analyzed, and moved around. With the availability of Internet-connected devices, wireless or wired, machines and sensors can pass on their data immediately as soon as it is created. This leads to real-time streaming and helps businesses to make valuable and fast decisions.

- **Variety**: This refers to the different data formats. Data used to be stored as text, dat, and csv from sources such as filesystems, spreadsheets, and databases. This type of data that resides in a fixed field within a record or file is called structured data. Nowadays, data is not always in the traditional format. The newer semi-structured or unstructured forms of data can be generated using various methods such as e-mails, photos, audio, video, PDFs, SMSes, or even something we have no idea about. These varieties of data formats create problems for storing and analyzing data. This is one of the major challenges we need to overcome in the big data domain.

- **Veracity**: This refers to the quality of data, such as trustworthiness, biases, noise, and abnormality in data. Corrupt data is quite normal. It could originate due to a number of reasons, such as typos, missing or uncommon abbreviation, data reprocessing, system failures, and so on. However, ignoring this malicious data could lead to inaccurate data analysis and eventually a wrong decision. Therefore, making sure the data is correct in terms of data audition and correction is very important for big data analysis.

- **Variability**: This refers to the changing of data. It means that the same data could have different meanings in different contexts. This is particularly important when carrying out sentiment analysis. The analysis algorithms are able to understand the context and discover the exact meaning and values of data in that context.
Overview of Big Data and Hive

- **Volatility**: This refers to how long the data is valid and stored. This is particularly important for real-time analysis. It requires a target scope of data to be determined so that analysts can focus on particular questions and gain good performance out of the analysis.

- **Visualization**: This refers to the way of making data well understood. Visualization does not mean ordinary graphs or pie charts. It makes vast amounts of data comprehensible in a multidimensional view that is easy to understand. Visualization is an innovative way to show changes in data. It requires lots of interaction, conversations, and joint efforts between big data analysts and business domain experts to make the visualization meaningful.

- **Value**: This refers to the knowledge gained from data analysis on big data. The value of big data is how organizations turn themselves into big data-driven companies and use the insight from big data analysis for their decision making.

In summary, big data is not just about lots of data, it is a practice to discover new insight from existing data and guide the analysis for future data. A big-data-driven business will be more agile and competitive to overcome challenges and win competitions.

**Relational and NoSQL database versus Hadoop**

Let's compare different data solutions with the ways of traveling. You will be surprised to find that they have many similarities. When people travel, they either take cars or airplanes depending on the travel distance and cost. For example, when you travel to Vancouver from Toronto, an airplane is always the first choice in terms of the travel time versus cost. When you travel to Niagara Falls from Toronto, a car is always a good choice. When you travel to Montreal from Toronto, some people may prefer taking a car to an airplane. The distance and cost here is like the big data volume and investment. The traditional relational database is like the car in this example. The Hadoop big data tool is like the airplane in this example. When you deal with a small amount of data (short distance), a relational database (like the car) is always the best choice since it is more fast and agile to deal with a small or moderate size of data. When you deal with a big amount of data (long distance), Hadoop (like the airplane) is the best choice since it is more linear, fast, and stable to deal with the big size of data. On the contrary, you can drive from Toronto to Vancouver, but it takes too much time. You can also take an airplane from Toronto to Niagara, but it could take more time and cost way more than if you travel by a car. In addition, you may have a choice to either take a ship or a train. This is like a NoSQL database, which offers characters from both a relational database and Hadoop in terms of good performance and various data format support for big data.
Batch, real-time, and stream processing

**Batch processing** is used to process data in batches and it reads data input, processes it, and writes it to the output. Apache Hadoop is the most well-known and popular open source implementation of batch processing and a distributed system using the MapReduce paradigm. The data is stored in a shared and distributed filesystem called Hadoop Distributed File System (HDFS), divided into splits, which are the logical data divisions for MapReduce processing. To process these splits using the MapReduce paradigm, the map task reads the splits and passes all of its key/value pairs to a map function and writes the results to intermediate files. After the map phase is completed, the reducer reads intermediate files and passes it to the reduce function. Finally, the reduce task writes results to the final output files. The advantages of the MapReduce model include making distributed programming easier, near-linear speed up, good scalability, as well as fault tolerance. The disadvantage of this batch processing model is being unable to execute recursive or iterative jobs. In addition, the obvious batch behavior is that all inputs must be ready by map before the reduce job starts, which makes MapReduce unsuitable for online and stream processing use cases.

**Real-time processing** is to process data and get the result almost immediately. This concept in the area of real-time ad hoc queries over big data was first implemented in Dremel by Google. It uses a novel columnar storage format for nested structures with fast index and scalable aggregation algorithms for computing query results in parallel instead of batch sequences. These two techniques are the major characters for real-time processing and are used by similar implementations, such as Cloudera Impala, Facebook Presto, Apache Drill, and Hive on Tez powered by Stinger whose effort is to make a 100x performance improvement over Apache Hive. On the other hand, in-memory computing no doubt offers other solutions for real-time processing. In-memory computing offers very high bandwidth, which is more than 10 gigabytes/second, compared to hard disks' 200 megabytes/second. Also, the latency is comparatively lower, nanoseconds versus milliseconds, compared to hard disks. With the price of RAM going lower and lower each day, in-memory computing is more affordable as real-time solutions, such as Apache Spark, which is a popular open source implementation of in-memory computing. Spark can be easily integrated with Hadoop and the resilient distributed dataset can be generated from data sources such as HDFS and HBase for efficient caching.

**Stream processing** is to continuously process and act on the live stream data to get a result. In stream processing, there are two popular frameworks: Storm (https://storm.apache.org/) from Twitter and S4 (http://incubator.apache.org/s4/) from Yahoo!. Both the frameworks run on the Java Virtual Machine (JVM) and both process keyed streams. In terms of the programming model, S4 is a program defined as a graph of Processing Elements (PE), small subprograms, and S4 instantiates a PE per key. In short, Storm gives you the basic tools to build a framework, while S4 gives you a well-defined framework.
Overview of the Hadoop ecosystem

Hadoop was first released by Apache in 2011 as version 1.0.0. It only contained HDFS and MapReduce. Hadoop was designed as both a computing (MapReduce) and storage (HDFS) platform from the very beginning. With the increasing need for big data analysis, Hadoop attracts lots of other software to resolve big data questions together and merges to a Hadoop-centric big data ecosystem. The following diagram gives a brief introduction to the Hadoop ecosystem and the core software or components in the ecosystems:

In the current Hadoop ecosystem, **HDFS** is still the major storage option. On top of it, snappy, RCFile, Parquet, and ORCFile could be used for storage optimization. Core Hadoop MapReduce released a version 2.0 called **Yarn** for better performance and scalability. **Spark** and **Tez** as solutions for real-time processing are able to run on the Yarn to work with Hadoop closely. **HBase** is a leading NoSQL database, especially when there is a NoSQL database request on the deployed Hadoop clusters. **Sqoop** is still one of the leading and matured tools for exchanging data between Hadoop and relational databases. **Flume** is a matured distributed and reliable log-collecting tool to move or collect data to HDFS. **Impala** and **Presto** query directly against the data on HDFS for better performance. However, Hortonworks focuses on Stringer initiatives to make Hive 100 times faster. In addition, Hive over Spark and Hive over Tez offer a choice for users to run Hive on other computing frameworks rather than MapReduce. As a result, Hive is playing more important roles in the ecosystem than ever.
Hive overview

Hive is a standard for SQL queries over petabytes of data in Hadoop. It provides SQL-like access for data in HDFS making Hadoop to be used like a warehouse structure. The **Hive Query Language (HQL)** has similar semantics and functions as standard SQL in the relational database so that experienced database analysts can easily get their hands on it. Hive's query language can run on different computing frameworks, such as MapReduce, Tez, and Spark for better performance.

Hive's data model provides a high-level, table-like structure on top of HDFS. It supports three data structures: tables, partitions, and buckets, where tables correspond to HDFS directories and can be divided into partitions, which in turn can be divided into buckets. Hive supports a majority of primitive data formats such as **TIMESTAMP**, **STRING**, **FLOAT**, **BOOLEAN**, **DECIMAL**, **DOUBLE**, **INT**, **SMALLINT**, **BIGINT**, and complex data types, such as **UNION**, **STRUCT**, **MAP**, and **ARRAY**.

The following diagram is the architecture seen inside the view of Hive in the Hadoop ecosystem. The Hive metadata store (or called metastore) can use either embedded, local, or remote databases. Hive servers are built on Apache Thrift Server technology. Since Hive has released 0.11, Hive Server 2 is available to handle multiple concurrent clients, which support Kerberos, LDAP, and custom pluggable authentication, providing better options for JDBC and ODBC clients, especially for metadata access.
Here are some highlights of Hive that we can keep in mind moving forward:

- Hive provides a simpler query model with less coding than MapReduce
- HQL and SQL have similar syntax
- Hive provides lots of functions that lead to easier analytics usage
- The response time is typically much faster than other types of queries on the same type of huge datasets
- Hive supports running on different computing frameworks
- Hive supports ad hoc querying data on HDFS
- Hive supports user-defined functions, scripts, and a customized I/O format to extend its functionality
- Hive is scalable and extensible to various types of data and bigger datasets
- Matured JDBC and ODBC drivers allow many applications to pull Hive data for seamless reporting
- Hive allows users to read data in arbitrary formats, using SerDes and Input/Output formats
- Hive has a well-defined architecture for metadata management, authentication, and query optimizations
- There is a big community of practitioners and developers working on and using Hive

**Summary**

After going through this chapter, we are now able to understand why and when to use big data instead of a traditional relational database. We also understand the difference between batch processing, real-time processing, and stream processing. We got familiar with the Hadoop ecosystem, especially Hive. We have also gone back in time and brushed through the history of database and warehouse to big data along with some big data terms, the Hadoop ecosystem, Hive architecture, and the advantage of using Hive. In the next chapter, we will practice setting up Hive and all the tools needed to get started using Hive in the command line.
Where to buy this book
You can buy Apache Hive Essentials from the Packt Publishing website.
Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.
Click here for ordering and shipping details.