Learning Hunk

Hunk is the big data analytics platform that lets you rapidly explore, analyze, and visualize data in Hadoop and NoSQL data stores. It provides a single, fluid user experience, designed to show you insights from your big data without the need for specialized skills, fixed schemas, or months of development.

You will begin by learning the Hunk architecture and Hunk Virtual Index before moving on to how to easily analyze and visualize data using Splunk Search Language (SPL). Next, you will meet Hunk Apps, which can easy integrate with NoSQL data stores such as MongoDB or Sqirl. You will also discover Hunk knowledge objects, build a semantic layer on top of Hadoop, and explore data using the friendly userinterface of Hunk Pivot. You will connect MongoDB and explore data in the data store. Finally, you will go through report acceleration techniques and analyze data in the AWS Cloud.

Who this book is written for

If you are Hadoop developers who want to build efficient real-time operational intelligence solutions based on Hadoop deployments or various NoSQL data stores using Hunk, this book is for you. Some familiarity with Splunk is assumed.

What you will learn from this book

- Deploy and configure Hunk on top of Cloudera Hadoop
- Create and configure virtual indexes for datasets
- Make your data presentable using a wide variety of data visualization components and knowledge objects
- Design a data model using Hunk best practices
- Add more flexibility to your analytics solution via extended SDK and custom visualizations
- Discover data using MongoDB as a data source
- Integrate Hunk with AWS Elastic MapReduce to improve scalability

In this package, you will find:

- The authors biography
- A preview chapter from the book, Chapter 1 'Meet Hunk'
- A synopsis of the book’s content
- More information on Learning Hunk
**Dmitry Anoshin** is a data-centric technologist and a recognized expert in building and implementing big data and analytics solutions. He has a successful track record when it comes to implementing business and digital intelligence projects in numerous industries, including retail, finance, marketing, and e-commerce.

Dmitry possesses in-depth knowledge of digital/business intelligence, ETL, data warehousing, and big data technologies. He has extensive experience in the data integration process and is proficient in using various data warehousing methodologies. Dmitry has constantly exceeded project expectations when he has worked for financial, machine tool, and retail industries.

He has completed a number of multinational full BI/DI solution life cycle implementation projects. With expertise in data modeling, Dmitry also has a background and business experience in multiple relation databases, OLAP systems, and NoSQL databases.

In addition, he has reviewed *SAP BusinessObjects Reporting Cookbook, Creating Universes with SAP BusinessObjects, and Learning SAP BusinessObjects Dashboards*, all by Packt Publishing and was the author of *SAP Lumira Essentials, Packt Publishing*. 
Sergey Sheypak started his so-called big data practice in 2010 as a Teradata PS consultant. His was leading the Teradata Master Data Management deployment in Sberbank, Russia (which has 110 billion customers). Later Sergey switched to AsterData and Hadoop practices. Sergey joined the Research and Development team at MegaFon (one of the top three telecom companies in Russia with 70 billion customers) in 2012. While leading the Hadoop team at MegaFon, Sergey built ETL processes from existing Oracle DWH to HDFS. Automated end-to-end tests and acceptance tests were introduced as a mandatory part of the Hadoop development process. Scoring geospatial analysis systems based on specific telecom data were developed and launched. Now, Sergey works as independent consultant in Sweden.
Preface

This book offers a step-by-step approach to learning Hunk, diving into the technical aspects of it first. It will demonstrate the various aspects of big data analytics using the powerful capabilities of Hunk. In addition to this, it provides detailed sections on the deployment and configuration of Hunk on top of Hadoop and the NoSQL data stores. It will also teach you how to create queries using SPL, reports, and dashboards. This book covers security questions and demonstrates how to set up security for big data implementation based on Hadoop and Hunk. Moreover, it will teach you how to use the Hunk SDK and extend its default functionality. Finally, it acts as a guide to deploying Hunk on top of MongoDB and AWS Elastic MapReduce.

What this book covers

Chapter 1, Meet Hunk, covers Hunk and its basic features. Hunk is a full-featured platform to rapidly explore, analyze, and visualize data in Hadoop and the NoSQL data stores. You will learn how to install and configure Hunk. Moreover, you will learn about Hunk’s architecture and Hunk Virtual Index. You will also be introduced to loading data into Hadoop in order to aid its discovery by Hunk.

Chapter 2, Explore Hadoop Data with Hunk, talks about how you can easily analyze and visualize data using the Splunk search processing language (SPL). Getting a large amount of data into Hadoop is easy but getting analytics from this data is the challenge. You will learn about the use cases of big data analytics and the security aspect of Hunk.

Chapter 3, Meet Hunk Features, teaches you about Hunk’s knowledge objects. Hunk is a powerful big data analytics platform, which gives us many tools in order to explore, analyze, and visualize big data. You will learn how to build a semantic layer on top of Hadoop and discover data using the friendly user interface of Hunk Pivot.
Preface

Chapter 4, Adding Speed to Reports, covers the techniques related to report acceleration. Hunk is an extremely powerful tool and can handle a vast amount of data. However, business decisions, which depend on fresh data, can’t wait.

Chapter 5, Customizing Hunk, introduces REST API, SDK, and so on. Sometimes, we want to get out of the box or need to meet business expectations and are restricted by the initial functionality. Thus, you will learn how to create customized visualization, and you will also be introduced to the Splunk Web Framework.

Chapter 6, Discovering Hunk Integration Apps, introduces you to Hunk’s apps that can easily integrate with the NoSQL data stores, such as MongoDB or Sqql. Hunk is a universal big data analytics platform, which can easy explore data in Hadoop or the NoSQL data stores. You will learn how to connect MongoDB and explore data in its data store.

Chapter 7, Exploring Data in the Cloud, shows you how to analyze data in AWS Cloud. Some big organizations prefer to store their big data on the cloud because it gives them many benefits.
Meet Hunk

Before getting started with Hunk, let's dive into the problem of modern big data analytics and highlight the main drawbacks and challenges. It is important to understand how Hunk works and what goes on under the hood. In addition, we'll compare Splunk and Enterprise in order to understand their differences and what they have in common as powerful and flexible products. Finally, we'll perform a lot of practical exercises through real-world use cases.

In this chapter you will learn:

- What big data analytics is
- Big data challenges and the disadvantages of modern big data analytics tools
- Hunk's history
- Hunk's architecture
- How to set up Hadoop for Hunk
- Real-world use cases

Big data analytics

We are living in the century of information technology. There are a lot of electronic devices around us that generate lots of data. For example, you can surf the Internet, visit a couple of news portals, order new Nike Air Max shoes from a web store, write a couple of messages to your friends, and chat on Facebook. Every action produces data. And we can multiply the actions by the amount of people who have access to the Internet, or just use a mobile phone, and we get really big data. Of course, you have a question: how big is big data? It probably starts from terabytes or even petabytes now. The volume is not the only issue; we are also struggling with the variety of data. As a result, it is not enough to analyze just the data structure. We should explore unstructured data, such as machine data generated by various machines.
World-famous enterprises try to collect this extremely big data in order to monetize it and find business insights. Big data offers us new opportunities; for example, we can enrich customer data through social networks, using the APIs of Facebook or Twitter. We can build customer profiles and try to predict customer wishes in order to sell our product or improve the customer experience. It is easy to say, but difficult to do. However, organizations try to overcome these challenges and use big data stores, such as Hadoop.

The big problem

Hadoop is a distributed file system and a distributed framework designed to compute large chunks of data. It is relatively easy to get data into Hadoop. There are plenty of tools for getting data into different formats, such as Apache Phoenix. However it is actually extremely difficult to get value out of the data you put into Hadoop.

Let's look at the path from data to value. First we have to start with collecting data. Then we also spend a lot of time preparing it, making sure that this data is available for analysis, and being able to question the data. This process is as follows:

![Diagram of data processing]

Unfortunately, you may not have asked the right questions or the answers are not clear, and you have to repeat this cycle. Maybe you have transformed and formatted your data. In other words, it is a long and challenging process.

What you actually want is to collect the data and spend some time preparing it; then you can ask questions and get answers repetitively. Now, you can spend a lot of time asking multiple questions. In addition, you can iterate with data on those questions to refine the answers that you are looking for. Let's look at the following diagram, in order to find a new approach:
The elegant solution
What if we could take Splunk and put it on top of all the data stored in Hadoop? This is what Splunk actually did. The following shows the names Hunk was derived from:

Hadoop + Splunk = Hunk

Let's discuss some goals that Hunk's inventors were thinking about when they were planning Hunk:

• Splunk can take data from Hadoop via the Splunk Hadoop Connection App. However, it is a bad idea to copy massive amounts of data from Hadoop to Splunk, it is much better to process data in-place, because Hadoop provides both storage and computation; why not take advantage of both of them?

• Splunk has the extremely powerful Splunk Processing Language (SPL) and it has a wide range of analytic functions. That's why it is a good idea to keep SPL in the new product.

• Splunk has a true on-the-fly schema. Data that we store in Hadoop changes constantly. So, Hunk has to be able build a schema on-the-fly independently of the data format.
Meet Hunk

- It is a very good idea to provide the ability to make a preview. As you know, when searching you can get incremental results. It can dramatically reduce outage. For example, we don't want to wait till a map reduce job finishes; we can look at the incremental result and, in the event of a wrong result, we can restart the search query.
- Deployment of Hadoop is not easy, and Splunk tries to make the installation and configuration of Hunk easy for us.

Supporting SPL

Let's discuss more closely the reasons for supporting SPL. You are probably familiar with Splunk and SPL and know how powerful and flexible this language is. These are some of the advantages of SPL:

- Naturally suitable for MapReduce
- Reduces adoption time for people who are already familiar with Splunk

There are some challenges in integrating SPL and Hadoop. Hadoop is written in Java but all SPL code is in C++. Does SPL need to convert to Java or reuse what Splunk has provided? Finally, it was decided to reuse C++ code entirely.

Intermediate results

No one likes to look at a blank screen. A lot of people using other tools such as Pig or Hive have to wait until the query is finished and you have no idea what the query is actually retrieving for you. Maybe you made a mistake, but you didn't know about it; you will have to wait till the job is completed. It is a kind of frustration—running queries and waiting for hours.

That's why the Hunk team gave their users the ability to preview the result. You will be able to play with this in future chapters.

Getting to know Hunk

Before going deeper into Hunk, let's clarify what Hunk does not do:

- Hunk does not replace your Hadoop distribution
- Hunk does not replace or require Splunk Enterprise
- Interactive but no real-time or needle in the haystack searches
- No data ingest management
- No Hadoop operation management
Hunk is a full-featured platform for rapidly exploring, analyzing, and visualizing data in Hadoop and NoSQL data stores. Based on years of experience building big data products deployed to thousands of Splunk customers, Hunk drives dramatic improvements in the speed and simplicity of getting insights from raw, unstructured, or polystructured big data—all without building fixed schemas or moving data to a separate in-memory store. Hunk delivers proven value for security, risk management, product analytics, a 360-degree customer view, and the Internet of Things.

While many big data initiatives take months and have high rates of failure, Hunk offers a unique approach. Hunk provides a single, fluid user experience designed to drive rapid insights from your big data. Hunk empowers self-service analytics for anyone in your department or organization to quickly and easily unlock actionable insights from raw big data, wherever it may reside.

These are the main capabilities of Hunk:

- Full-featured, integrated analytics
- Fast to deploy and drive value
- Interactive search
- Supported data formats
- Report acceleration
- Results preview
- Drag-and-drop analytics
- Rich developer environment
- Custom dashboards and views
- Secure access
- Hunk apps
- Hunk on the AWS cloud
Meet Hunk

Splunk versus Hunk
Let's compare Splunk and Hunk:

<table>
<thead>
<tr>
<th>Features</th>
<th>Splunk Enterprise</th>
<th>Hunk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indexing</td>
<td>Native</td>
<td>Virtual</td>
</tr>
<tr>
<td>Where data is stored and read</td>
<td>Splunk Buckets on Local or SAN Disks</td>
<td>Any Hadoop-compatible file system (HDFS, MapR, Amazon S3) and NoSQL, or other data stores via streaming resource libraries</td>
</tr>
<tr>
<td>A 60-day free trial license</td>
<td>500 MB/day</td>
<td>Unlimited</td>
</tr>
<tr>
<td>Pricing model</td>
<td>Data invested per day</td>
<td>Number of task trackers (compute nodes in YARN)</td>
</tr>
<tr>
<td>Real-time streaming events</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Data model</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Pivot</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Rich developer environment</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Event breaking, timestamp</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>extraction, source typing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rare term search</td>
<td>Index time</td>
<td>Search time</td>
</tr>
<tr>
<td>Report acceleration</td>
<td>Fast: Uses index and bloom filters</td>
<td>Slow: Requires full data scan within partitions</td>
</tr>
<tr>
<td>Access control and single sign-on</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Universal forwarder</td>
<td>+</td>
<td>NA</td>
</tr>
<tr>
<td>Forwarder management</td>
<td>+</td>
<td>NA</td>
</tr>
<tr>
<td>Splunk apps</td>
<td>+</td>
<td>Limited</td>
</tr>
<tr>
<td>Premium apps</td>
<td>+</td>
<td>N/A</td>
</tr>
<tr>
<td>Standard support</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Enterprise support</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

In the preceding table + means product support mentioned feature and NA means feature is not supported in a product.

As you saw, there are some differences. But Hunk is designed for another purpose; it is a kind face in the complex world of big data. Throughout this book, we will introduce the various features of Hunk and you will definitely learn this amazing tool.

Let's look closely at Hunk and try to understand how it works.
Hunk architecture

Let's explore how Hunk looks. From the end user perspective, Hunk looks like Splunk. You use same interface, you can write searches, visualize big data, and create reports, dashboards, and alerts. In other words, Hunk can do everything Splunk can do. In the following screenshot, you can see a schematic of Hunk’s architecture:

Hunk has the same interface and command lines as well. The only change is that Splunk works with data stored in native indexes but the Hunk SPL acts with external data; that's why they call virtual indexes.

Connecting to Hadoop

Hunk is designed to connect to Hadoop via the Hadoop interface. The following screenshot demonstrates that Hunk can connect a Hadoop cluster via Hadoop client libraries and Java:
Meet Hunk

Moreover, Hunk can work with multiple Hadoop clusters:

In addition, you can use Splunk and Hunk together. You can connect Splunk Enterprise if you have Splunk and Hadoop in your environment. As a result, it is possible to correlate Hunk searches through Hadoop and Splunk Enterprise via the same search head.

**Advance Hunk deployment**

Sometimes, organizations have really big data. They have thousands of instances of Hadoop. It is a real challenge to get business insight from this extremely large data. However, Hunk can easily handle this titanic task. Of course this isn't as easy as it sounds, but it is possible because you can scale Hunk deployments. Let's look at the following example:
There are hundreds or thousands of users who put their business questions to big data. Business users send their queries, they go through the **Load Balancer (LB)**, LB sends them to Hunk, and Hunk makes distributive work to Hadoop.

**Native versus virtual indexes**

Before we start to compare native and virtual indexes, let's use our previous Splunk experience and see how SPL actually works.

For example, we have a query:

```splunk
Index=main | stats count by status | rename count AS qty
```

As you may remember, every step in Splunk is divided by pipes. You can read expression from left to right and follow expression execution sequence.

Splunk development was motivated by Unix Shell pipes.

In our example, we:

1. Get all data from `Index=main`.
2. Count all rows for every status.
3. Rename the `count` field as `qty`.
4. Retrieve the final result.

It is interesting to know that SPL uses a MapReduce algorithm. In other words, it has a map phase when performing retrieve operations and reducing step, and when performing count operations.

The rule is that the first search command is always responsible for retrieving events.

**Native indexes**

Before Hunk was created, there were only the native indexes of Splunk Enterprise. The data was ingested by Splunk and access to it was via the Splunk interface.

A native index is basically a data store or collection of data. We can put web logs, syslogs, or other machine data in Splunk. We have access controls and the ability to give permissions to users to access data on specific indexes. In addition, Splunk gives us the opportunity to optimize popular and heavy searches. As a result, business users will get their dashboards very quickly.
**Virtual index**

Virtual indexes lack some features of native indexes. A virtual index is a data container with access controls. Hunk can only read data. Data gets into Hadoop somehow and Hunk can use this data as a container. The inventors of Hunk decided to not build indexes on top of Hadoop data and to not optimize Hunk to perform needle-in-the-haystack searches. However, if data layout is properly designed in Hadoop (for example, there is a hierarchical structure or data is organized based on the timestamp, year, month, or date), this can really improve search performance.

Let’s compare both indexes in one table:

<table>
<thead>
<tr>
<th>Native Indexes</th>
<th>Virtual Indexes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serve as data containers</td>
<td>Serve as data containers</td>
</tr>
<tr>
<td>Access control</td>
<td>Access control</td>
</tr>
<tr>
<td>Reads/writes</td>
<td>Read only</td>
</tr>
<tr>
<td>Data retention policies</td>
<td>N/A</td>
</tr>
<tr>
<td>Optimized for keyword search</td>
<td>N/A</td>
</tr>
<tr>
<td>Optimized for time range search</td>
<td>Available via regex/pruning</td>
</tr>
</tbody>
</table>

You can learn more about virtual indexes on the Splunk website: [http://docs.splunk.com/Documentation/Hunk/latest/Hunk/Virtualindexes](http://docs.splunk.com/Documentation/Hunk/latest/Hunk/Virtualindexes).

**External result provider**

The core technology of Hunk is a virtual index and **External Result Provider (ERP)**. We have already encountered virtual indexes. The term ERP is sometime known as **resource provider**.

The ERP is basically a helper process. It goes out and deals with the details of the external systems that are going to interact with Hadoop or another data store. In other words, it takes searches that users perform in Hunk and somehow translates or interprets them in mrjob. That’s how it pushes computation.

There are a few other implementations of ERP that Splunk’s partners developed in order to integrate Hunk with Mongo DB, Apache Accumulo, and Cassandra. There are just different implementations of the same interface that helps Hunk to interact with external systems and use any type of data via virtual indexes.
The following diagram demonstrates how ERP looks:

For each Hadoop cluster (or external system) the search process spawns an ERP process that is responsible for executing the (remote part of the) search on that system.

You can learn more about ERP on the Splunk web site: http://docs.splunk.com/Documentation/Hunk/latest/Hunk/Externalresultsproviders.

**Computation models**

Previously, we considered some challenges in big data analytics and found out powerful solutions via Hunk. Now we can go deeper in order to understand some of the core advantages of Hunk. Let's start with an easy question: *how do we provide interactivity?*

There are at least two computational models.

**Data streaming**

In this approach, data moves from HDFS to the search head. In other words, data is processed in a streaming fashion. As a result users can immediately start to work with data, slice and dice it, or visualize when the first bytes of data will start to appear. But there is a problem with this process. It is a huge volume of data to move and process.

There is one primary benefit that you will probably get; there is a very low response time. In addition, we get low throughput that is not very positive for us.
Meet Hunk

Data reporting
The second mode is moving computation to data. The way to do this is to create and start a MapReduce job to do the processing, monitor the MapReduce job, and, finally, collect the results. Then, merge the results and visualize the data. There is another problem here – late feedback, because the MapReduce job might take a long time. As a result, this approach has high latency and high throughput.

Mixed mode
Both modes have their pros and cons, but the most important are low latency, because it gives interactivity, and high throughput, because it gives us the opportunity to process larger datasets. These are all benefits and Hunk takes the best from both computational modes.

Let’s visualize both modes in order to better understand how they work:

In addition, we consolidate all the modes in the following table, in order to make things clearer:

<table>
<thead>
<tr>
<th>Streaming</th>
<th>Reporting</th>
<th>Mixed Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pull data from HDFS to search head for processing</td>
<td>Push compute down to data and consume results</td>
<td>Start both streaming and reporting models. Show streaming results until reporting starts to complete</td>
</tr>
<tr>
<td>Low latency</td>
<td>High latency</td>
<td>Low latency</td>
</tr>
<tr>
<td>Low throughput</td>
<td>High throughput</td>
<td>High throughput</td>
</tr>
</tbody>
</table>
Hunk security

With version 6.1, Hunk became more secure. By default Hunk has superusers with full access. However, very often organizations want to apply a security model to their corporate data, in order to keep their data safe. Hunk can use pass-through authentication. It gives the opportunity to control how MapReduce jobs can submit users and what HDFS files they can access. In addition, it is possible to specify the queue MapReduce jobs should use.

Pass-through authentication gives us the capability to make the Hunk superuser a proxy for any number of configured Hunk users. As a result, Hunk users can act as Hadoop users to own the associated jobs, tasks, and files in Hadoop (and it can limit access to files in HDFS). Let's look at the following diagram:

Let's explore some common use cases that can help us understand how it works.

One Hunk user to one Hadoop user

For example, say we want our Hunk user to act as a Hadoop user associated with a specific queue or data set. Then we just map the Hunk user to a specific user in Hadoop. For example, in Hunk the user name is Hemal, but in Hadoop it is HemalDesai and the queue is Books.

Many Hunk users to one Hadoop user

For example, say we have many Hunk users and want them to act as a Hadoop user. In Hunk, we can have several users such as Dmitry, Hemal, and Sergey, but in Hadoop they will all execute as an Executive user and will be assigned to the Books queue.
Meet Hunk

Hunk user(s) to the same Hadoop user with different queues
For example, say you have many Hunk users and the same Hadoop users; it is possible to assign them different queues.

Security will be discussed more closely in Chapter 3, Meet Hunk Features.

Setting up Hadoop
Before starting to play with Hadoop and Hunk, we are going to download and run a VM. You'll get a short description on how to get everything up and running and put in some data for processing later.

Starting and using a virtual machine with CDH5
We have decided to take the default Cloudera CDH 5.3.1 VM from the Cloudera site and fine-tune it for our needs. Please open this link to prepare a VM: http://www.bigdatapath.com/2015/08/learning-hunk-links-to-vm-with-all-stuff-you-need/.

This post may have been be updated by the time you're reading this book.

SSH user
You can run the terminal application by clicking the special icon on the top bar:

Your user is cloudera. sudo is passwordless:

```
[cloudera@quickstart ~]$ whoami
cloudera
[cloudera@quickstart ~]$ sudo su
[root@quickstart cloudera]$
```

```
MySQL

MySQL is used as an example of the data ingestion process. The user name is `dwhuser`, the password is `dwhuser`. You can get root access by using the `root` username and the `cloudera` password:

```
[cloudera@quickstart ~] $ mysql -u root -p
```

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| cdrdb              |
| cm                 |
| firehose           |
| hue                |
| metastore          |
| mysql              |
| oozie              |
| retail_db          |
| sentry             |
+--------------------+
10 rows in set (0.00 sec)
```

We import data from MySQL to Hadoop from the database named `cdrdb`. There are some other databases. They are used by Cloudera Manager services and Hadoop features such as Hive Metastore, Oozie, and so on.

**Hive Metastore** is a service designed to centralize metadata management. It's a kind of Teradata DBC.Table, DBC.Columns, or IBM DB2 syscat.Columns, syscat.Tables. The idea is to create a strict schema description over the bytes stored in Hadoop and then get access to this data using SQL.

**Oozie** is a kind of Hadoop CRON without a Single Point of Failure (SPOF). Think it through; is it easy to create a distributed reliable CRON with failover functionality? Oozie uses RDBMS to persist metadata about planned, running, and finished tasks. This VM doesn't provide an Oozie HA configuration.
Meet Hunk

Starting the VM and cluster in VirtualBox

Perform the following steps:

1. Install VirtualBox: https://www.virtualbox.org/.
2. Follow the link and download the VM ZIP.
3. Extract the ZIP to your local drive.
4. Open VirtualBox.
5. Go to menu Machine | Add and point to the extracted VBOX file.
6. You should see the imported VM:

![Oracle VM VirtualBox Manager](image)

7. Run the imported VM.

Big data use case

We are going to use several data sources for our applications. The first will use RDBMS interaction since it's one of the most popular use cases. We will show you how to integrate classical data with a strict schema from the RDBMS small dictionary stored in HDFS. Data from RDBMS has big volume in real life (we just have a small subset of it now) and the dictionary keeps the dimensions for the RDBMS value. We will enrich the big data with the dimensions before displaying data on the map.

Importing data from RDBMS to Hadoop using Sqoop

There are many ways to import data into Hadoop; we could easily publish a book called "1,001 methods to put your data into Hadoop." We are not going to focus on these specialities and will use very simple cases. Why so simple? Because you will meet many problems in a production environment and we can't cover these in a book.
An example from real life: you will definitely need to import data from your existing DWH into Hadoop. And you will have to use Sqoop in conjunction with special Teradata/Oracle connectors to do it quickly without DWH performance penalties. You will spend some time tuning your DB storage schema and connection properties to achieve a reasonable result. That is why we decided to keep all this tricky stuff out of the book; our goal is to use Hunk on top of Hadoop.

Here is a short explanation of the import process. We’ve split the diagram into three parts:

- MySQL, a database that stores data
- Oozie, responsible for triggering the job import process
- Hadoop, responsible for getting data from MySQL DB and storing it in an HDFS catalog
Meet Hunk

Initially, the data is stored in MySQL DB. We want to copy the data to HDFS for later processing using Hunk. We will work with the telco dataset in Chapter 6, Discovering Hunk Integration Apps, related to custom application development. We write an Oozie coordinator, started by the Oozie server each night. Oozie is a kind of Hadoop CRON and has some additional features that help us work with data. Oozie can do many useful things, but right now we are using its basic functionality: just running the workflow each day. The coordinator code is here: https://github.com/seregashyypak/learning-hunk/blob/master/import-milano-cdr/src/main/resources/oozie/import-milano-cdr-coord.xml.

Next is the workflow. The coordinator is responsible for scheduling the workflow. The workflow is responsible for doing business logic. The workflow code is here: https://github.com/seregashyypak/learning-hunk/blob/master/import-milano-cdr/src/main/resources/oozie/workflows/import-milano-cdr-wflow.xml.

The workflow has one Sqoop action.

Next is the Sqoop action. This action declares the way the job should read data from RDBMS and store it to HDFS.

The third section is the MapReduce job that reads data from RDBMS and writes it to HDFS. The Sqoop action internally runs a MapReduce job that is responsible for getting table rows out of MySQL. The whole process sounds pretty complex, but you don't have to worry. We've created the code to implement it.

Telecommunications – SMS, Call, and Internet dataset from dandelion.eu

We are going to use several open datasets from https://dandelion.eu. One weekly dataset was uploaded to MySQL and contains information about the telecommunication activity in the city of Milano. Later, you will use an Oozie coordinator with the Sqoop action to create a daily partitioned dataset.


Milano grid map

Milano is divided into equal squares. Each square has a unique ID and four longitude and latitude coordinates.
A mapping between the logical square mesh and spatial area would be helpful for us during geospatial analysis. We will demonstrate how Hunk can deal with geospatial visualizations out-of-the-box.

**CDR aggregated data import process**

We've prepared an Oozie coordinator to import data from MySQL to HDFS. Generally, it looks like a production-ready process. Real-life processes are organized in pretty much the same way. The following describes the idea behind the import process:

We have the potentially huge table `2013_12_milan_cdr` with time-series data. We are not going to import the whole table in one go; we will partition data using a time field named `time_interval`.

The idea is to split data by equal time periods and import it to Hadoop. It's just a projection of the RDBMS partitioning/sharing techniques to Hadoop. You'll see seven folders named from `/masterdata/stream/2013/12/01` to `/masterdata/stream/2013/12/07`.


The general idea is to:

- Run the workflow each day
- Import the data for the whole day

We've applied a dummy MySQL date function; in real life, you would use the OraOop connector, Teradata connector, or some other tricks to play with the partition properties.
**Periodical data import from MySQL using Sqoop and Oozie**

To run the import process, you have to open the console application inside a VM window, go to the catalog with the coordinator configuration, and submit it to Oozie:

```bash
cd /home/devops/oozie-configs
sudo -u hdfs oozie job -oozie http://localhost/oozie -config import-milano-cdr-coord.properties -run
```

The console output will be:

```
job: 0000000-150302102722384-oozie-oozi-C
```


Here is the running coordinator. It took two minutes to import one day. There are seven days in total. We used a powerful PC (32 GB memory and an 8-core AMD CPU) to accomplish this task.

---

**Downloading the example code**

The following screenshot shows how the successful result should look:

![Screenshot showing coordinator coordinator import-milano-cdr-coord](image)

We can see that the coordinator did produce seven actions for each day starting from December 1st till December 7th.

You can use the open console application to execute this command:

```bash
hadoop fs -du -h /masterdata/stream/milano_cdr/2013/12
```

The output should be:

<table>
<thead>
<tr>
<th>Size</th>
<th>Size</th>
<th>Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>74.1 M</td>
<td>222.4 M</td>
<td>/masterdata/stream/milano_cdr/2013/12/01</td>
</tr>
<tr>
<td>97.0 M</td>
<td>291.1 M</td>
<td>/masterdata/stream/milano_cdr/2013/12/02</td>
</tr>
<tr>
<td>100.4 M</td>
<td>301.3 M</td>
<td>/masterdata/stream/milano_cdr/2013/12/03</td>
</tr>
<tr>
<td>100.4 M</td>
<td>301.1 M</td>
<td>/masterdata/stream/milano_cdr/2013/12/04</td>
</tr>
<tr>
<td>100.6 M</td>
<td>301.8 M</td>
<td>/masterdata/stream/milano_cdr/2013/12/05</td>
</tr>
<tr>
<td>100.6 M</td>
<td>301.7 M</td>
<td>/masterdata/stream/milano_cdr/2013/12/06</td>
</tr>
<tr>
<td>89.2 M</td>
<td>267.6 M</td>
<td>/masterdata/stream/milano_cdr/2013/12/07</td>
</tr>
</tbody>
</table>
Meet Hunk

The target format is **Avro** with snappy compression. We'll see later how Hunk works with popular storage formats and compression codecs. Avro is a reasonable choice: it has wide support across Hadoop tools and has a schema.

It's possible to skip the import process; you can move data to the target destination using a command. Open the console application and execute:

```
sudo -u hdfs hadoop fs -mkdir -p /masterdata/stream/milano_cdr/2013
sudo -u hdfs hadoop fs -mv /backup/milano_cdr/2013/12/masterdata/stream/
  milano_cdr/2013/
```

### Problems to solve

We have a week-length dataset with 10-minute time intervals for various subscriber activities. We are going to draw using dynamics in dimensions: part of the day, city area, and type of activity. We will build a subscriber dynamics map using imported data.

### Summary

In this chapter we met big data analytics and discovered challenges that can be overcome with Hunk. In addition, we learned about Hunk's history and its internal modes. Moreover, we explored Hunk's architecture and learned about virtual indexes and ERPs. In addition, we touched on some topics related to security. Finally, we started to prepare for technical exercises by setting up Hadoop and discovering real-world use cases.

In the next chapter, we will start to explore big data in Hadoop. We will work with SPL and create amazing visualizations.
Where to buy this book

You can buy Learning Hunk 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.