The *Free* Hive Book

by Christian Prokopp, @prokopp, Google+

About This Book

License

Thank you.

About The Author

Christian Prokopp is a Data Scientist at Rangespan, and writes as Blogger and Data Journalist in his spare time. Christian holds a BSc, MCom, PhD and has lived, worked and researched in three continents.

Why this book?

This book was sparked by the need to give some tutorial material to business users at Rangespan. Christian has been working with Hive and Hadoop for the last two years. Giving access to Big Data to business stakeholder through a SQL-like interface has proved extremely valuable and popular. Facebook, a company with one of the largest data sets world-wide for example, heavily relies on Hive and Hadoop for data processing, insight and reporting with thousands of users accessing data spread across thousands of computers.

In that spirit this book provides an example driven introduction to working with Hive. The book helps SQL experienced users to apply their knowledge when working with Hive. At the end of the book are also some more advanced tips for power-users, dev-ops or sysadmins.

Original and Latest Version

The original with the latest updates is available as the The Free Hive Book.

Table of Contents

The book is work in progress and the TOC as well as the actual chapters will evolve.

1. **Introduction**
2. **Getting Started - Setup Hive**
3. **Create, Load, Query, Drop a Table**
   - *Managed Table*
   - *External Table*
4. **Setting up the example table**
5. **Query**
   - *SELECT … WHERE …*
   - *SELECT … ORDER BY …*
   - *SELECT … CLUSTER BY …*
   - *SELECT … SORT BY …*
6. **Normalize Tables**
7. **Joining Tables**
8. **…**
9. **…**
10. **…**
11. **Tuning Tips**
    - *Controlling File and Split Size*
    - *Data Storage Formats*
    - *Compression*
    - *Partitioning*
    - *Bucketing*
    - *Limit*
    - *Parallel Execution*

1. Introduction - What is Hive

(If you know Hive and Hadoop you can safely skip the introduction.)

Apache Hive is a data warehouse system build on top of Hadoop to query *Big Data*. Hive originated at Facebook and was open sourced in August 2008. The challenge Facebook had to address is one faced by many companies since then. Eventually data growth in a company challenges the capabilities of deployed RDBMS or NoSQL systems. Reports and analytics start to take minutes, then hours, and eventually overlap with other queries and the whole system grinds to a halt. Another common scenario companies start processing big data with Hadoop discovers the value of making the data accessible beyond the development team capable of writing complex map-reduce jobs.

What is Big Data

The term Big Data is freely used in this context. A colleague defined Big Data jokingly as anything beyond one million rows - Microsoft Excels row limit. The underlying point is that Big Data is a point of view and can be generalized as the point where simple solutions and deployed technology fail.

The subsequent question is if scaling and investing heavily in it is the most economical solution. Commercial large-scale data warehouse solutions are very expensive. Furthermore, some of the data collected today, e.g. poorly structured or highly denormalized data, can be impractical to manage with these systems. The Hadoop ecosystem regularly is utilized to scale data processing in a feasible manner. Hadoop becomes either a replacement or a batch process addition to the existing infrastructure for data analysis, extraction, loading, transformation, reporting, and machine learning.

Accessing Big Data

The downside, which Facebook encountered, was that data stored in Hadoop is inaccessible to business users. There are higher-level languages like Pig, Cascading, Crunch or Scalding. They are, however, geared towards software developers that want to avoid the verbosity of pure Java map-reduce development. Even Pig, a popular data-flow language, still requires users to learn a completely new skill. Facebook realized that most of their users already had a common skill - they knew SQL. Hive was developed to give access to data stored in Hadoop translating SQL-like statements into complex map-reduce jobs reading and processing data on large distributed scale.

Democratizing Big Data

Hive is a success story. Today, Facebook's largest Hadoop cluster consists of thousands of computers providing a combined storage of 150 Petabytes - roughly 150,000,000 Gigabytes. Hive provides access to literally thousands of employees to the data running together thousands of map-reduce jobs every day using Hive. The additional training for employees knowing SQL to use Hive is minimal. Most statements can be expressed equivalently and full SQL support is coming to Hive very soon.

What Hive is not

Hive does not use sophisticated indexes like many RDBMS which are able to answer queries in seconds. Hive queries usually take minutes or hours. However, Hive scales very well and can execute queries across data of the magnitude of Petabytes. The Hadoop ecosystem and Hive are under very active development, however, and speedups are achieved with every new iteration. In 2013 many new developments are due to be introduced, a new Hadoop framework that goes beyond map-reduce and manages resources better, and Hive improvements that will make queries on smaller data faster and interactive.

What this book will teach

You will learn how to access data with Hive.

2. Getting Started - Setup Hive

(You can skip this section if you have access to a Hive CLI (Command Line Interface) or a Hue and Beeswax web interface to Hive.)

Installing Hive for evaluation purposes is best done with a virtual machine provided by one of the popular Hadoop distributions:

- **Hortonworks**
- Cloudera
- MapR

The examples in this book are based on the **Hortonworks Sandbox version 1.2**.

3. Create, Load, Query, Drop a Table

Hive uses a metastore - a database - to store information about the tables it knows. Tables to Hive often are not much more than storing the information where the data is stored and how it is formatted. We do not have to know much about the metastore itself to use Hive though.

Managed Table

Managed tables's data is controlled by Hive. It will create a directory for the data on HDFS and when we drop the table it will delete the data. Later in this chapter we will see that we can manage the data ourselves with an *EXTERNAL* table.

Creating an empty table

Let us create a table, which will be a simple list of all names of all countries. Go to the Beeswax query editor and execute the following query:

```
CREATE TABLE country_list (name STRING);
```

You can now find the table in the Beeswax table list. Once you selected the table you can view the file location on HDFS which Hive automatically selected and created for you. The table and directory are empty. Note: We can define the location of a new table as we will learn later.

Describing a table

We can get all the information we need about a table through a query too. Go back to the query editor and execute:

```
DESCRIBE EXTENDED country_list;
```

The result describes the schema of the table and detailed table information. This includes the location of the table and the information that it uses TextInputFormat which is default setting in the Hortonworks Hive setup.

Loading data into a table

We want to load data to the table so we need to upload it to HDFS. We know the table has single column of country name, uses a simple text format, and Hive always uses new line characters to as row delimiters. All we need to do to add data is upload one or several files into the directory linked with the table. The files have to be formatted to have one country name on each line.

Go back to the table list and select the `country_list` table. Select `View File Location` and you will be viewing the HDFS directory of where the able stores its data. Upload the `coutry_example.tsv` file into the directory using the file upload function on the top right of the interface.

Query a table

Hive is very flexible and checks the table's data location for every query. Adding or removing data on the file system is reflected on the table. After adding the file its content is now automatically retrieved as table data. Try it out. Go back to the Beeswax table list and select the `country_list` again. Click on browse data on the left. You should see four countries which are read from the file. You can also query the table to get the same result. Go tot he query editor and execute:

```
SELECT * FROM country_list;
```

Drop a table

You can drop the table in the table view of Beeswax through the graphic interface. Go to the query editor and execute the equivalent query to drop the table:

```
DROP TABLE country_list;
```

The table and the HDFS directory and file(s) are deleted.

External Table

So far the queries and behavior of Hive has not been very different to SQL systems. Hive's separation of data location and storing the schema in a metastore enables it to create tables pointing to existing data, to read, query, and transform it, and then drop the tables without touching the original data on HDFS.

These unmanaged tables make Hive powerful as a tool that queries outputs from other processes and systems. The terminology used is an extension of the SQL `CREATE TABLE` statement. Let us do the above example again with an external table instead to illustrate the difference.

Create an external table

Create the external table by simply adding `EXTERNAL` to the statement:

```
CREATE EXTERNAL TABLE country_list (name STRING);
```

Everything will appear to be the same as in the previous example, i.e. the table will look the same in the Beeswax interface and behave the same as the previous one. Only the detailed description reveals a difference:

```
DESCRIBE EXTENDED country_list;
```

The table description now mentions `tableType:EXTERNAL_TABLE` which previously was `tableType:MANAGED_TABLE`. The managed table meant that Hive would delete the data on dropping the table - managing it. The external table type means that Hive on dropping a table will remove the schema information only - it disappears from Hive but the data remains on HDFS - Hive considers it external.

Go ahead and try it out. If you repeat our previous example and upload the `country_example.tsv` file to the table's HDFS location the data will appears as table data as before. If you drop the table the data will remain:

```
DROP TABLE country_list;
```

Go to the HDFS directory `/apps/hive/warehouse/country_list` to see the data and directory still intact.

(Re)Creating an external table

This means that we can create a table with the data on HDFS with a single statement:

```
CREATE EXTERNAL TABLE country_list (name STRING);
```

Query the table to see that the data on HDFS has been linked back to the table:

```
SELECT * FROM country_list;
```

4. Setting up the example table

(Table of Contents)

The following examples are based on World Bank data of development indicators of the last four decades. The data is freely available on the World Bank website. The data distributed with the book for the examples has been modified for use with Hive.

Create external table

```
CREATE EXTERNAL TABLE wdi
(
    country_name STRING,
    country_code STRING,
    indicator_name STRING,
    indicator_code STRING,
    `1960` FLOAT, `1961` FLOAT, `1962` FLOAT, `1963` FLOAT, `1964` FLOAT,
    `1965` FLOAT, `1966` FLOAT, `1967` FLOAT, `1968` FLOAT, `1969` FLOAT,
    `1970` FLOAT, `1971` FLOAT, `1972` FLOAT, `1973` FLOAT, `1974` FLOAT,
    `1975` FLOAT, `1976` FLOAT, `1977` FLOAT, `1978` FLOAT, `1979` FLOAT,
    `1980` FLOAT, `1981` FLOAT, `1982` FLOAT, `1983` FLOAT, `1984` FLOAT,
    `1985` FLOAT, `1986` FLOAT, `1987` FLOAT, `1988` FLOAT, `1989` FLOAT,
    `1990` FLOAT, `1991` FLOAT, `1992` FLOAT, `1993` FLOAT, `1994` FLOAT,
    `1995` FLOAT, `1996` FLOAT, `1997` FLOAT, `1998` FLOAT, `1999` FLOAT,
    `2000` FLOAT, `2001` FLOAT, `2002` FLOAT, `2003` FLOAT, `2004` FLOAT,
    `2005` FLOAT, `2006` FLOAT, `2007` FLOAT, `2008` FLOAT, `2009` FLOAT,
    `2010` FLOAT, `2011` FLOAT, `2012` FLOAT
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
LOCATION '/user/sandbox/wdi';
```

The table is empty since we have not loaded any data yet. Hive created a folder at `/user/sandbox/wdi` on HDFS for us and we can copy data there to appear in the tables.

Put the data on HDFS

We first place the data on HDFS for Hive and then create an external table for the data. Upload the `wdi_data.tsv.gz` file to the new HDFS `/user/sandbox/wdi` folder. The file is now located on the distributed HDFS file system and can be read by Hadoop and Hive. The file is Gzip compressed as indicated by the `.gz` postfix. Hive recognizes this format and automatically decompresses the file at query time.

5. Query

(Table of Contents)

We can check if the schema from the create statement aligns with the data we uploaded by either browsing the data from the Beeswax table interface or querying it:

```
SELECT * FROM wdi;
```

The above statement returns all columns and all rows.

SELECT … WHERE …

HiveQL supports `WHERE` constraints on the `SELECT` statement. Let us reduce the selection to a specific indicator. The following query returns all rows for the indicator named `'Trade (% of GDP)'`:

```
SELECT * FROM wdi
WHERE indicator_name = 'Trade (% of GDP)';
```

We can further restrict the result to return only the country name and the indicator result of the year 2011:

```
SELECT `country_name`, `2011` AS trade_2011 FROM wdi
WHERE indicator_name = 'Trade (% of GDP)';
```

We can also exclude empty `NULL` results for the year 2011:

```
SELECT `country_name`, `2011` AS trade_2011 FROM wdi WHERE
  indicator_name = 'Trade (% of GDP)' AND
  `2011` IS NOT NULL;
```

SELECT … ORDER BY …

What are the countries with the greatest and the least percentage of trade to GDP ratio? The result of the above query can be ordered by column(s). This is similarly to SQL's `ORDER BY ... (ASC|DESC)` statement. Postfixing the order statement with `ASC` or `DESC` will order it in ascending or descending order:

```
SELECT `country_name`, `2011` AS trade_2011 FROM wdi WHERE
  indicator_name = 'Trade (% of GDP)' AND
  `2011` IS NOT NULL
  ORDER BY trade_2011 DESC;
```

SELECT … CLUSTER BY …

…

SELECT … SORT BY …

…

[Here be dragons]