**You will also enjoy...**

Spring Boot vs Node.js

TypeScript or Babel?

Node.js Performance Tips

The V8 JavaScript Engine

JavaScript Performance Tips

Optimize JavaScript for the V8 Engine

Introduction to JavaScript Linters

Vue vs React

An Introduction to Browserify

JavaScript ES6 Classes

# Hive Partitioning

*Last modified: July 11, 2017*

Hive uses partitions to logically separate and query data. Partitioning reduces the time it takes to run queries on larger tables. In this article, we explore what partitioning is and how to implement it with Hive. We also examine the differences between static and dynamic partitioning and provide a few examples for adding partitions to Hive tables.

## Preface: What is Hive?

Apache Hive brings a SQL-like interface to HDFS. Using Hive, you can run MapReduce jobs with a SQL-like syntax called HiveQL. Hive allows you to perform analytics and queries on a Hadoop cluster without all the complicated Java MapReduce code.

It's important to remember that Hive is simply an abstraction of MapReduce. Hive is not a database. All of the data you query through Hive comes from HDFS. While Hive stores schema information related to the Hive tables you create, it does not store any of the underlying data you query.

For more information on Hive, check out this discussion on Hive.

## What is partitioning?

Partitioning is the logical separation of data into directories based on the values of specific columns. Tables are typically partitioned by date/time or a column with an appropriate level of cardinality (uniqueness of data values). For example, say you have the following dataset:

```
Name    | Email         | Phone     | Country
Sam     | sam@email.com | 555123231 | US
```

**You will also enjoy...**

Spring Boot vs Node.js

TypeScript or Babel?

Node.js Performance Tips

The V8 JavaScript Engine

JavaScript Performance Tips

Optimize JavaScript for the V8 Engine

Introduction to JavaScript Linters

Vue vs React

An Introduction to Browserify

JavaScript ES6 Classes

This dataset could be partitioned by country so that all entries having the same country code are stored in the same data directory:

```
/users/country='US'/

/users/country='IN'/
```

## Why use partitioning?

Partitioning provides a faster way to query data. By logically separating data by partitioned columns, Hive can easily identify and query only the data it needs to.

When you perform queries on non-partitioned tables, Hive must query the entire data set (even with filters like `WHERE`). By partitioning data based on column values, Hive can query HDFS a lot faster with partitioned tables.

Take our previous country code data set as an example. If we run a Hive query `WHERE COUNTRY='US'`, Hive can ignore all of the data directories that don't match the `country='US'` partition. Since the data is already partitioned by country code, Hive can easily isolate the data it needs for the query.

## Creating a Hive table with partitions

Creating a table with partitions is easy. You simply add the `PARTITIONED BY` clause when you create the table:

```
CREATE TABLE CUSTOMERS (email STRING, phone STRING) PARTITIONED BY (country String);
```

Remember that Hive sits on top of HDFS. For the partitioning to work, the underlying HDFS data directories need to be structured like `/path/to/users/country='US'`. It should also be noted that the partition column `country` need not be separately defined with the other columns `email` and `phone`.

**You will also enjoy...**

Spring Boot vs Node.js

TypeScript or Babel?

Node.js Performance Tips

The V8 JavaScript Engine

JavaScript Performance Tips

Optimize JavaScript for the V8 Engine

Introduction to JavaScript Linters

Vue vs React

An Introduction to Browserify

JavaScript ES6 Classes

partitions via `PARTITIONED BY` during its creation). Take the following table we created for our customers:

```
CREATE TABLE CUSTOMERS (email STRING, phone STRING) PARTITIONED BY (country String);
```

Since we defined a partition column when we created this table, we can easily add a partition like so:

```
ALTER TABLE CUSTOMERS ADD PARTITION (country="IN");
```

Notice how this adds a partition to the already defined `country` partitioned column If we wanted to add a new partition column, we would have to create a new Hive table specifying any additional partitioned columns.

# Load data into a partitioned Hive table

## Dynamic partitioning

Once you have partitions defined for a Hive table, you can `dynamically partition` the table via:

```
INSERT OVERWRITE TABLE CUSTOMERS PARTITION (country)

SELECT NAME, EMAIL, PHONE, COUNTRY FROM SOURCE_CUSTOMERS;
```

This utilizes an `INSERT...SELECT` clause to populate a partitioned `CUSTOMERS` table from a non-partitioned `SOURCE_CUSTOMERS` table. This dynamically adds the partitions for our partitioned column `country`. Please note that this method only works if the source table includes the columns we want to partition by (in this case `country`). Also be sure to list the partition column last in the `SELECT` clause.

## Static partitioning

**You will also enjoy...**

Spring Boot vs Node.js

TypeScript or Babel?

Node.js Performance Tips

The V8 JavaScript Engine

JavaScript Performance Tips

Optimize JavaScript for the V8 Engine

Introduction to JavaScript Linters

Vue vs React

An Introduction to Browserify

JavaScript ES6 Classes

We use the `LOAD` clause for loading data from a theoretical `/testdata.txt` file. Notice how we explicitly set `country = 'US'`. This is known as `static partitioning` as we explicitly define a value to partition by (in this case `'US'`).

### Difference between static and dynamic partitioning in Hive

Static partitioning involves manually adding a partition to a table and moving files into the partition of that table. With static partitioning, you have more control and can alter the partitions since you explicitely define them. Loading data is faster with static partitioning.

Dynamic partitioning determines which partitions should be created for you. It can take longer to initially load tables, but does so automatically. Dynamically partitioned tables don't rely on you knowing the partitions yourself and can populate based on existing values already in the table. This is why dynamic partitioning requires that partition column values be present in the data being loaded.

## Hive partition external table

You can partition external tables the same way you partition internal tables. External tables simply define an existing location rather than create a new one like internal tables do.

This leads to a lot of confusion since external tables are based on existing HDFS locations. Remember that the HDFS file structure must reflect the partitions you wish to add. If the table you create doesn't have a directory structure like `/path/country='US'`, then you can't add new partitions. An alternative may be to create a new external table and load it with data from the original non-partitioned location.

## Showing partitions in Hive

**You will also enjoy...**

Spring Boot vs Node.js

TypeScript or Babel?

Node.js Performance Tips

The V8 JavaScript Engine

JavaScript Performance Tips

Optimize JavaScript for the V8 Engine

Introduction to JavaScript Linters

Vue vs React

An Introduction to Browserify

JavaScript ES6 Classes

You can also run:

```
DESCRIBE FORMATTED <TABLE_NAME>;
```

## Conclusion

Partitions make Hive queries faster. Remember that Hive works on top of HDFS, so partitions are largely dependent on the underlying HDFS file structure. Once a Hive table is defined with partition columns, you can either statically or dynamically add partitions to the table. Using existing tables, you can load data into partitioned tables with Hive.

Write a response...

## Responses:

stackchief
October 24, 2017

MSCK REPAIR <TABLE NAME>

this will automatically add partitions that are in the underlying HDFS structure but not included in Hive Metastore yet...

Reply

## You might also like:

Spring Boot vs Node.js

TypeScript or Babel?

JavaScript Performance Tips

Optimize JavaScript for the V8 Engine

Introduction to JavaScript Linters

Vue vs React

An Introduction to Browserify

JavaScript ES6 Classes

**You will also enjoy...**

Spring Boot vs Node.js

TypeScript or Babel?

Node.js Performance Tips

The V8 JavaScript Engine

JavaScript Performance Tips

Optimize JavaScript for the V8 Engine

Introduction to JavaScript Linters

Vue vs React

An Introduction to Browserify

JavaScript ES6 Classes